**A tutorial on how to use the**

**Frequency-Independent Biological Signal Identification (FIBSI) Program**

By: Max A. Odem (max.odem@uth.tmc.edu; max.neuro.odem@gmail.com) and Ryan M. Cassidy (rmcassidy@pm.me)

This is the program described in the publication: Cassidy RM, Bavencoffe AG, Lopez ER, Cheruvu SS, Walters ET, Uribe RA, Krachler AM, Odem MA. TITLE. *Journal* 2020 XX(XX): XXXX-XXXX. doi:XXXX

The FIBSI source code, readme file, and tutorial data are provided online at <https://github.com/rmcassidy/FIBSI_program>

Table of Contents

[Getting started 1](#_Toc37025253)

[*Prerequisites* 1](#_Toc37025254)

[*Installation* 1](#_Toc37025255)

[Using FIBSI to analyze data collected from a time series 3](#_Toc37025256)

[*Key terms and abbreviations* 3](#_Toc37025257)

[*Formatting the raw data for input* 3](#_Toc37025258)

[*Formatting a command for input* 4](#_Toc37025259)

[*Using the pyplot interactive viewer* 6](#_Toc37025260)

[Workflow examples using real biological data 7](#_Toc37025261)

[*Measuring fluctuations in membrane potential in rodent sensory neurons* 7](#_Toc37025262)

[*Measuring gut motility in zebrafish using fluorescence contrast imaging* 7](#_Toc37025263)

[Methodology 7](#_Toc37025264)

[Processing the tutorial data 8](#_Toc37025265)

[Analyzing and interpreting the processed tutorial data 10](#_Toc37025266)

[Final notes 11](#_Toc37025267)

# **Getting started**

## *Prerequisites*

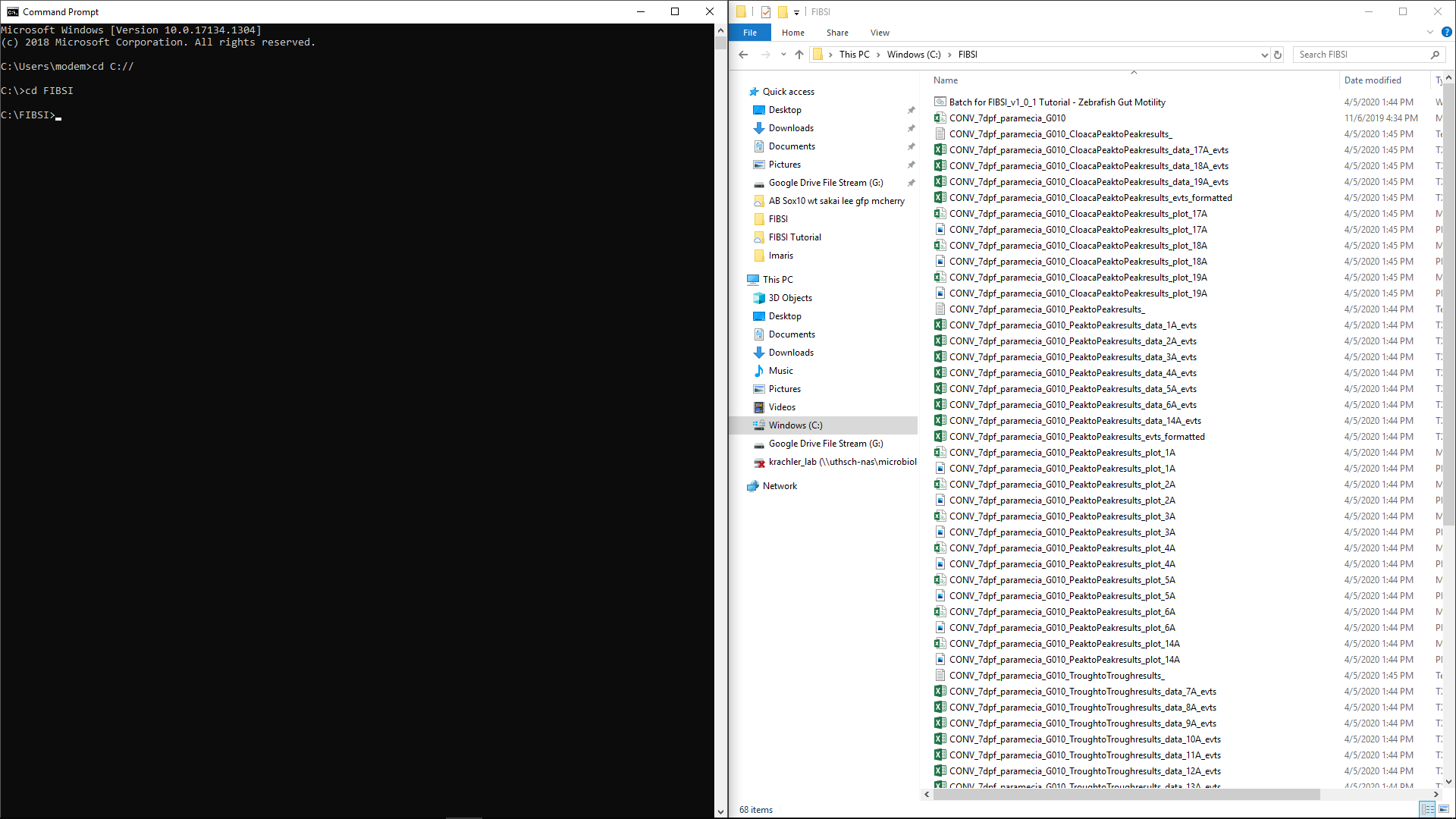
1. **FIBSI.py** – The core analysis program
2. **README.md** – A description of usage, flags, explanations, and background related to FIBSI
3. **ZfishTutorial\_Fed\_G010.csv** – Raw fluorescence data collected from a zebrafish larva fed with paramecia
4. **Batch for FIBSI\_v1\_0\_1 Tutorial – Zfish Fed G010.bat** – Windows batch file with input commands
5. **Python v3.5.2 or higher** distributed by Python Software Foundation, Beaverton, OR
6. **Anaconda v2019.7.0.0 or higher** distributed by Anaconda, Inc., Austin, TX

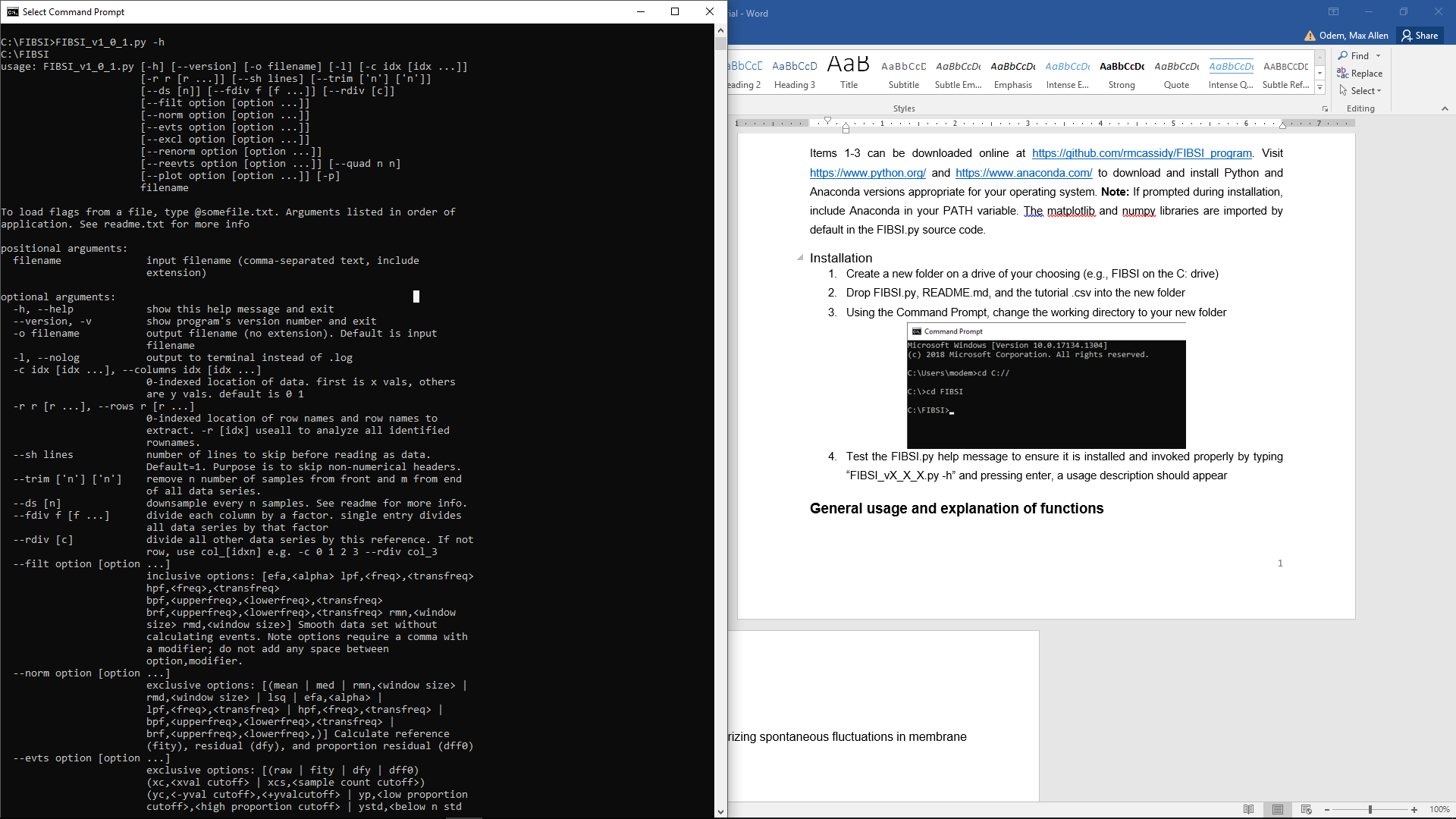
Items 1-4 can be downloaded online at <https://github.com/rmcassidy/FIBSI_program>.

Visit <https://www.python.org/> and <https://www.anaconda.com/> to download and install Python and Anaconda versions appropriate for your operating system. **Note:** If prompted during installation, include Anaconda in your PATH variable.

The matplotlib and numpy libraries are imported by default in the FIBSI.py source code.

## *Installation*

1. Create a new folder on a drive of your choosing (e.g., FIBSI folder on the C: drive)
2. Drop FIBSI.py, README.md, and the tutorial .csv into the new folder
3. Using the Command Prompt, change the working directory to the new folder
4. Test the FIBSI.py help message to ensure it is installed and invoked properly by typing “FIBSI\_vX\_X\_X.py -h” and pressing enter, a usage description with available arguments (i.e., flags) should appear



# **Using FIBSI to analyze data collected from a time series**

Explicit descriptions of the underlying processing and functions used by FIBSI will not be discussed in this tutorial. Only those functions necessary to replicate the analyses performed in our original publication (Cassidy et al., *JOURNAL*, YEAR) will be described. For examples of the processing and flags used to call various functions, background, and conceptual design, please carefully read through the README.md provided at <https://github.com/rmcassidy/FIBSI_program>.

## *Key terms and abbreviations*

There are many key terms and abbreviations used in FIBSI, namely the flags and functions. It is recommended you refer to README.md for a full description. For quick reference, the few used in this tutorial include:

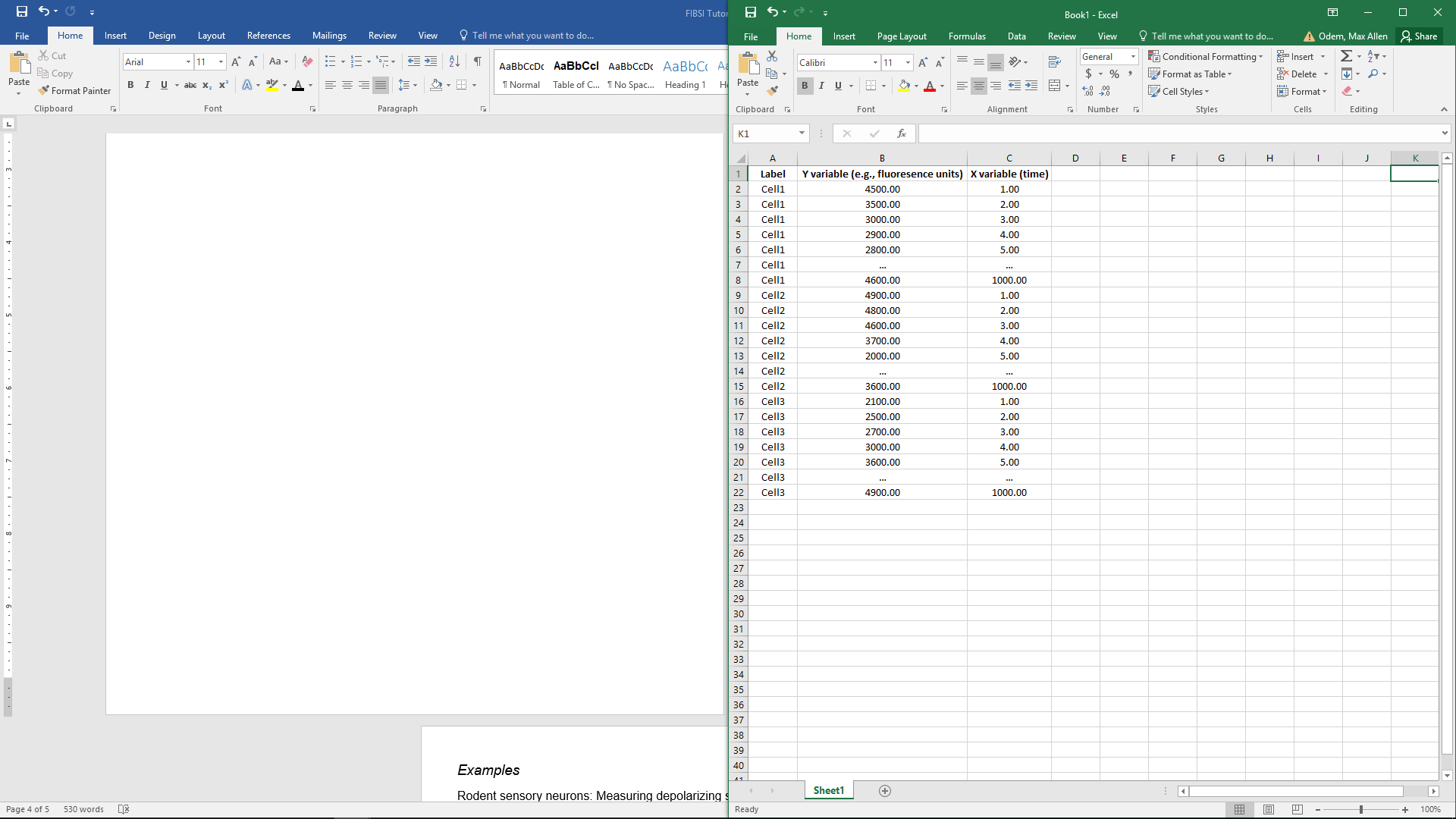
* fity – the reference Y series fit via a user-specific normalization
* rmd – running median, one of multiple normalizations that can be selected (e.g., running mean, least squares regression)
* dfy – the residual Y series ()
* evts – detected events
* ROI – regions of interest

## *Formatting the raw data for input*

FIBSI.py reads X-Y data series in a comma-separated format. The extension must be included with the filename (e.g., FILENAME.csv or FILENAME.txt) when inputting data in FIBSI. No spaces are permitted within the filename or output filename (to be designated by the user).

**Formatting notes:**

1. The X and Y data must be found within distinct columns
2. Sampling interval
   1. Must be consistent for a given X-Y series
   2. Can differ between series if the series are input using separate commands (using either the same source file or separate source files)
   3. Must be consistent if inputting two or more series from the same source file using a single command
3. If multiple series are combined into a single source file, then the rows corresponding to series 1 must be labeled separately from the rows for series 2, and so on



## *Formatting a command for input*

Commands can be written using Command Prompt or run in batches using .bat files (can be edited like text files). Be mindful of proper syntax and spelling as most errors will be due to inputting a command incorrectly and/or misspellings and mislabeling. **Note:** FIBSI uses a 0-index system, so column A in an Excel spreadsheet is column 0, B is column 1, and so on.

For this tutorial, commands will include the following components:

* Invoking FIBSI (required)
* Source filename for input (required)
* Filename for output (-o flag, required)
* Assigning the X and Y data columns (-c flag, required)
* Assigning the series IDs column and series to be analyzed (-r flag, optional if using multiple series in the same source file)
* Assigning a normalization factor (--rdiv flag, optional)
* Assigning the fitting method (--norm flag, required)
* Assigning the event detection method and cutoffs (--evts flag, required)
* Assigning a post-hoc fitting method, event detection method, and cutoffs (--renorm and --reevts flags, all optional)
* Assigning how processed data are visualized using the interactive pyplot viewer (-p and --plot flags, all optional)

FIBSI\_v1\_0\_0.py filename.csv -o filename\_results -c 15 1 -r 0 Cell1 Cell2 Cell3 Cell4 --rdiv Cell4 --norm rmd,50 --evts dfy xc,1 yc,0,0 --renorm above --reevts dfy xc,1 yc,0,0 -p --plot raw fity dfy evts save\_csv save\_png,200

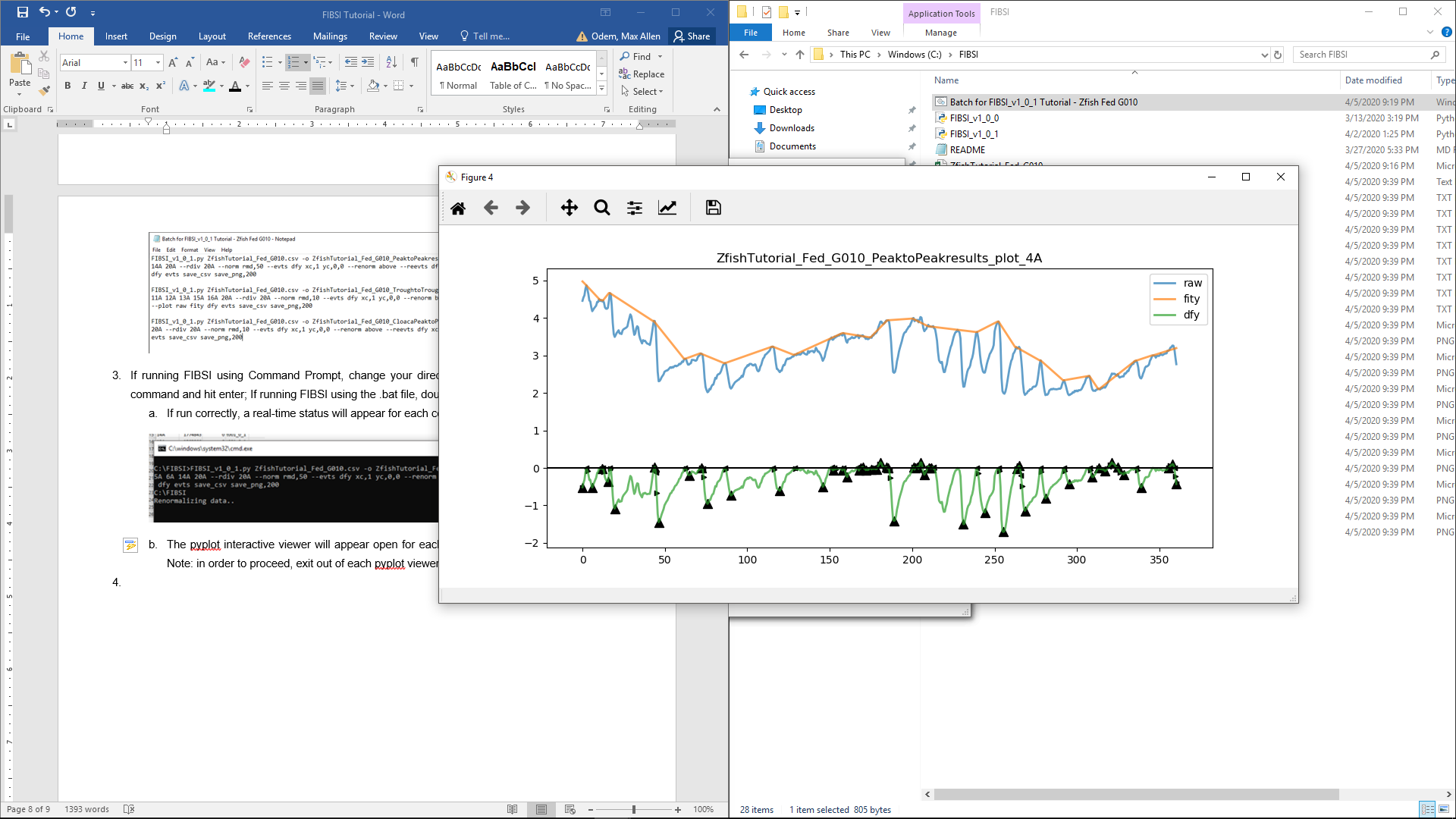
The example command will do the following (flags written in order of processing and color coded):

1. Invoke FIBSI\_v1\_0\_0.py to analyze filename.csv and all results will be dumped into the root folder with the filename\_results moniker
2. The X data are located in column 15 and Y data are in column 1 – give X, then Y
3. There are 4 series in column 0 with row IDs Cell1, Cell2, Cell3, and Cell4 – give column, then row IDs
4. Cells 1-3 will be divided by the Cell4 series – if dividing by a series, then all series must have the same sampling interval
5. The series for Cells 1-3 will be individually fitted and normalized using a running median with a window of 50 X units – this generates the fity line; window uses whatever units X data are collected in
6. Event detection is performed on the residual Y data with an X-cutoff of at least 1 unit and no Y-cutoffs in the positive or negative amplitude – this generates the dfy line; X and Y cutoffs use whatever units data are collected in
7. The dfy line is used to identify peaks (or troughs if “below” is invoked with –renorm flag) and the raw Y series is refitted using a peak-to-peak “above” renormalization – this generates new fity and dfy lines
8. Event detection is performed on the new dfy line with X- and Y-cutoffs
9. pyplot is invoked to plot the raw Y series, fity, and dfy with events labeled; All series data shown in pyplot will be saved as a .csv file in the order input and a .png image of the pyplot viewer will be saved at 200 dpi resolution

**Note:** In pyplot, the fity, dfy, and events will reflect the peak-to-peak renormalization because the --renorm flag was invoked. If --renorm was not invoked, then fity, dfy, and events would reflect the first normalization method (in this example, the running median).

## *Using the pyplot interactive viewer*

FIBSI incorporates the pyplot interactive viewer (dependent on the numpy and matplotlib libraries) as a quick and easy tool for visualizing how the raw data are processed and for comparing the results of event detection to the raw data. Basic functions (in order left to right, red arrows) include reset original view, pan axes, zoom to specified area, change subplot settings, edit axes and line parameters, and save to file. Events are displayed and bounded by isosceles triangles (black) on the dfy line. Rightward point triangles are event start points, leftward are event end points, and upward/downward are peaks/troughs.



# **Workflow examples using real biological data**

Here we will demonstrate the workflow to analyze two different types of biological data. Tutorial data and commands for rodent sensory neurons and zebrafish are provided online. We recommend walking through these examples to familiarize yourself with the workflow so you may begin analyzing your own data.

## *Measuring fluctuations in membrane potential in rodent sensory neurons*

**UNDER CONSTRUCTION**

## *Measuring gut motility in zebrafish using fluorescence contrast imaging*

### Methodology

In Cassidy et al., *JOURNAL*, YEAR, all zebrafish larvae were analyzed using a running median of 100 X units and peak-to-peak renormalization to limit the number of assumptions made. If desired, users can replicate our methods by updating the commands provided in the tutorial .bat file with the appropriate changes to the --norm and --renorm flags.

However, in this tutorial we describe a more thorough workflow for analyzing gut motility in zebrafish to account for differences in contraction dyanmics between regions of the gut. In our fluorescence contrast assay, the total fluorescent intensity either increases or decreases depending on how the gut moves during muscle contractions. We have observed 3 different kinds of movements:

1. Narrowing of the gut = decrease in fluorescence intensity; predominantly observed in the foregut ROIs
2. Widening of the gut = increase in fluorescence intensity; predominantly observed in the midgut/hindgut ROIs, sometimes observed in distal foregut ROIs
3. Undulating motion = generally leads to an increase in fluorescence; predominantly observed in the midgut/hindgut ROIs

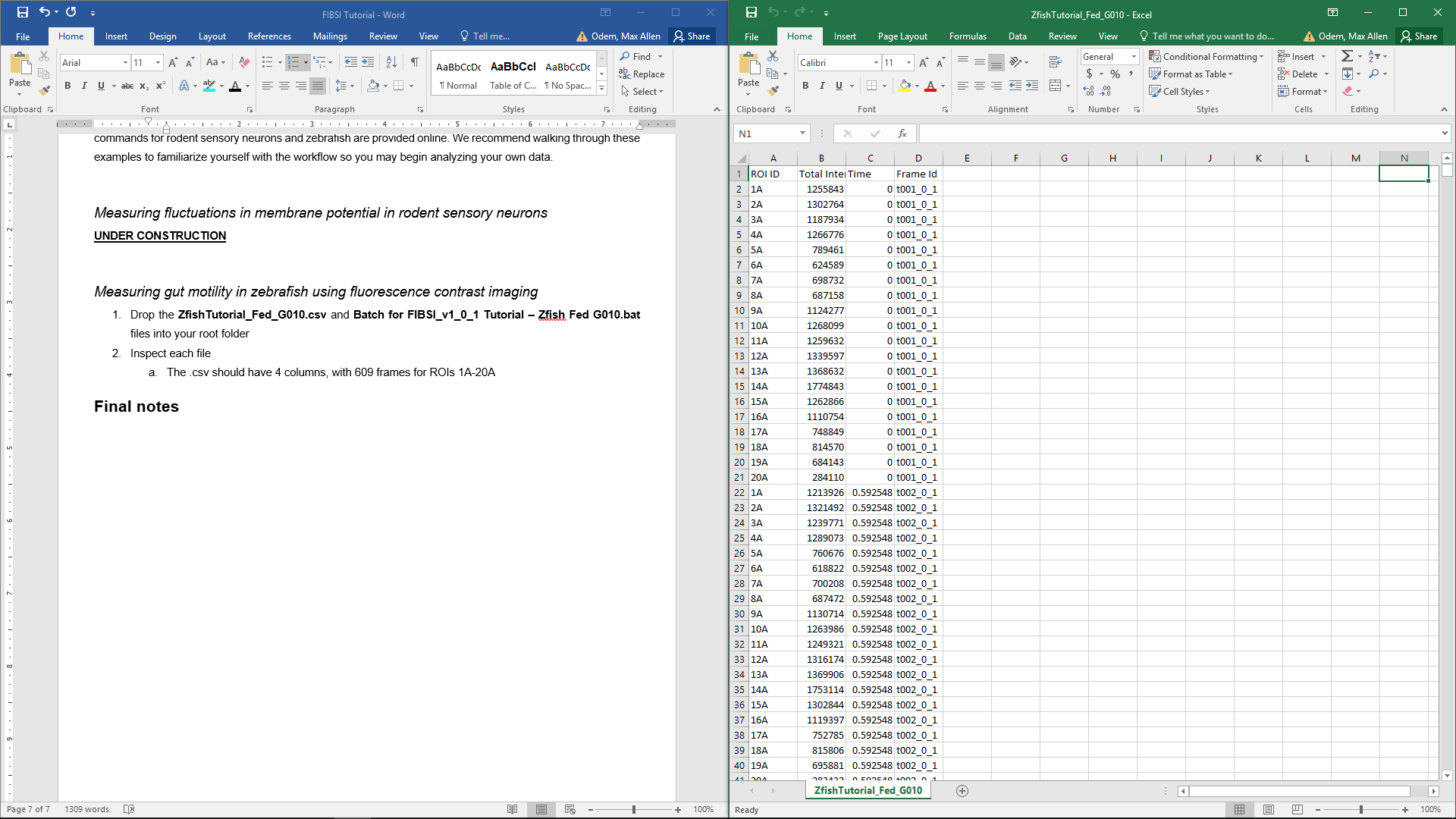
Therefore, the type of movement influences whether the contraction event of interest is detected on the peak or trough of the change in fluorescence intensity. In order to capture appropriate contraction start and end times, the renormalization method used should oppose the directional change in fluorescence intensity (i.e., use the --renorm above method for contractions that decrease fluorescence, and vice versa). Furthermore, setting the window size can influence event detection accuracy. For optimal accuracy, longer windows are better for larger amplitude contractions while smaller windows are better for smaller amplitude contractions. Choosing the best method and window size requires careful observation of videos synced with the fluorescence time series data, some trial and error, and inherently introduces assumptions about best-fit approaches for this kind of stochastic data.

For this tutorial, the following renormalization methods will be used for the **ZfishTutorial\_Fed\_G010.csv** and **Batch for FIBSI\_v1\_0\_1 Tutorial – Zfish Fed G010.bat** files:

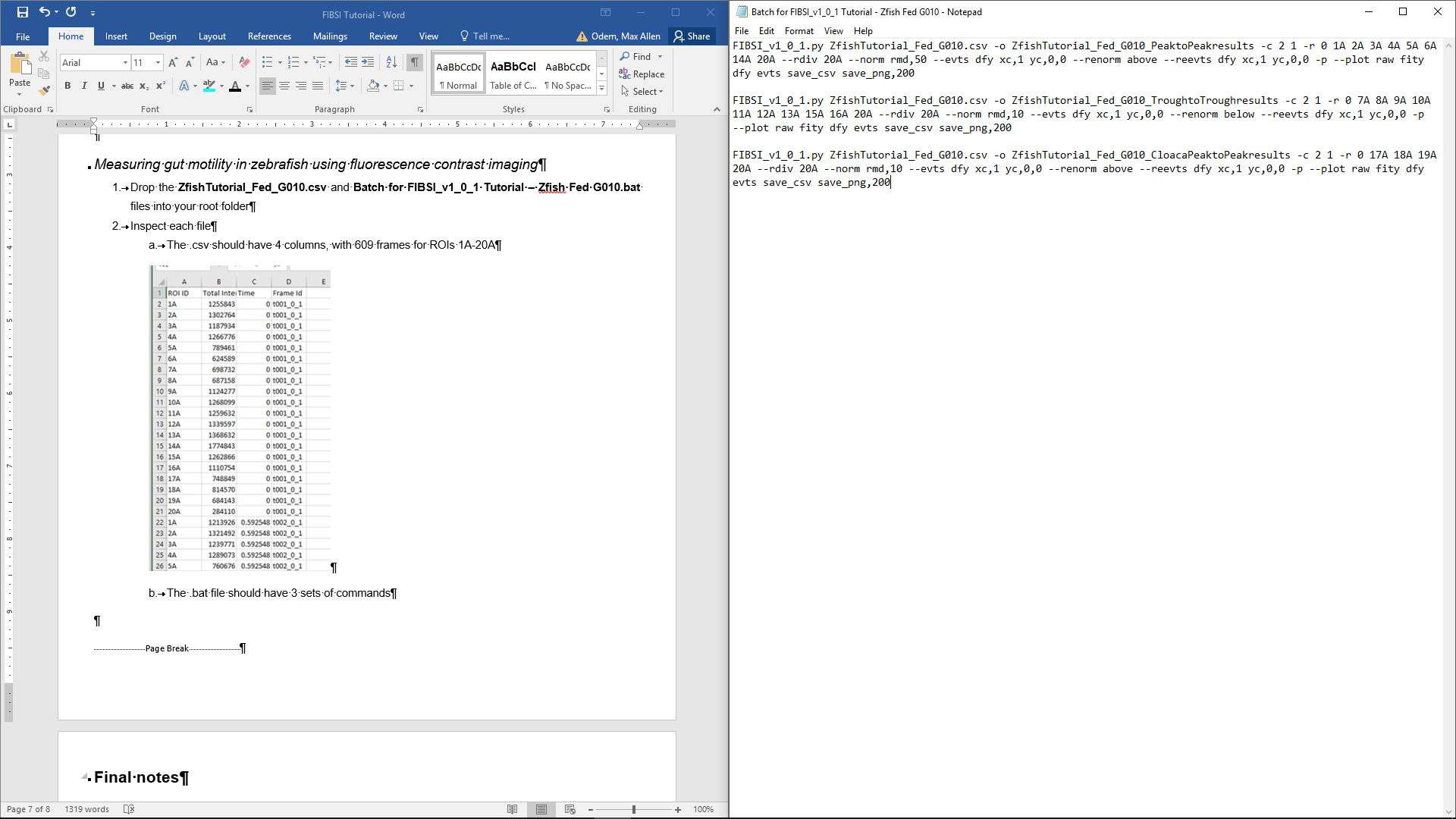
1. Peak-to-peak analysis in ROIs #1-6, 14; window size = 50 X units
2. Trough-to-trough analysis in ROIs #7-13, 15-16; window size = 10 X units
3. Peak-to-peak analysis for the fish cloaca in ROIs #17-19; window size = 10 X units

### Processing the tutorial data

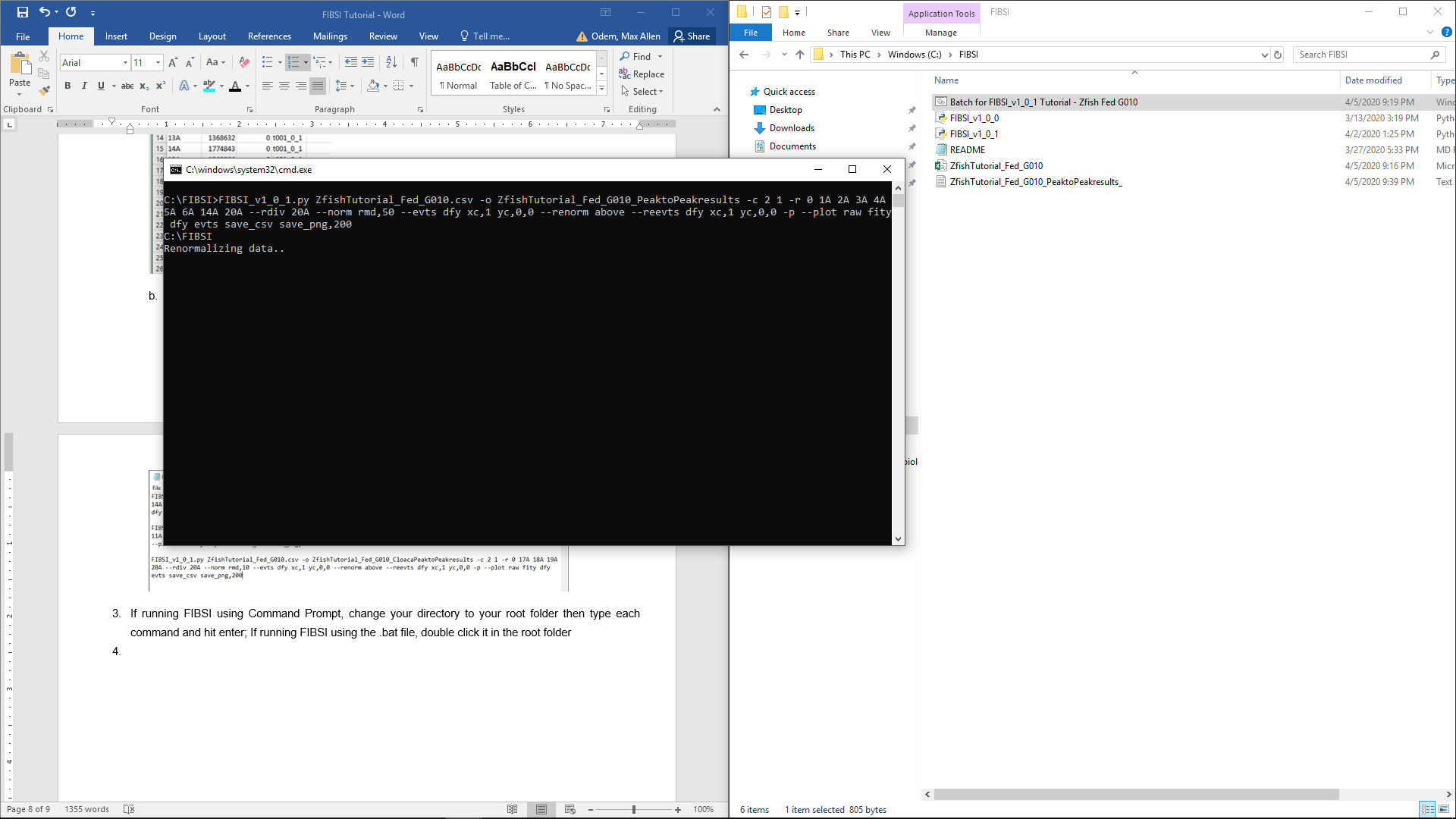
1. Drop the **ZfishTutorial\_Fed\_G010.csv** and **Batch for FIBSI\_v1\_0\_1 Tutorial – Zfish Fed G010.bat** files into your root folder
2. Inspect each file
   1. The .csv should have 4 columns, with 609 frames for ROIs 1A-20A



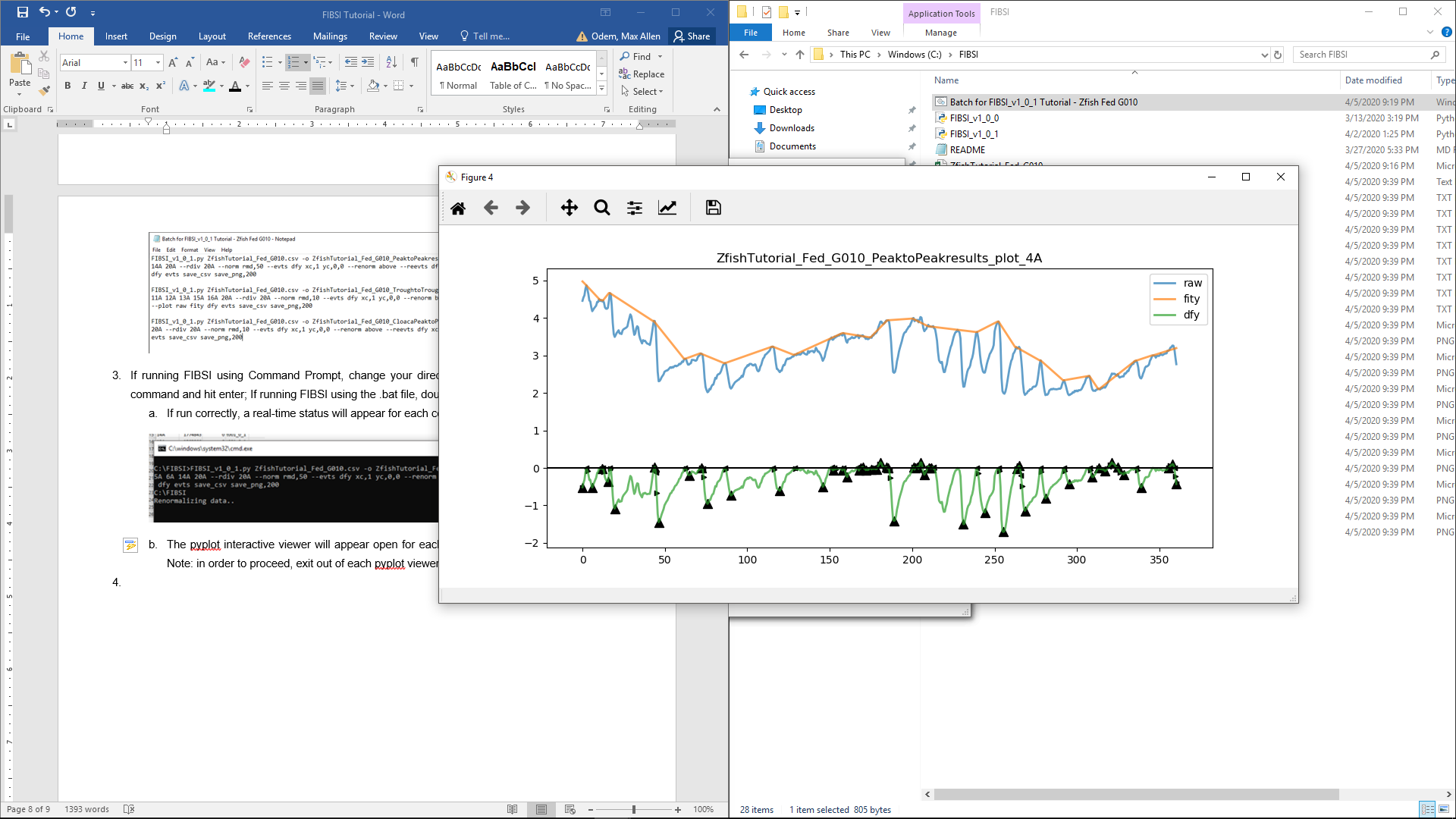
* 1. The .bat file should have 3 commands



1. Run FIBSI
   1. If running FIBSI using Command Prompt, change your directory to your root folder then type each command and hit enter; If running FIBSI using the .bat file, double click the file in the root folder
   2. If run correctly, a real-time status will update as each command is processed



* 1. The pyplot interactive viewer will open for each ROI in order of commands processed; **Note:** in order to proceed, exit out of each pyplot viewer



1. Confirm output files were properly generated and dumped in your root folder
   1. 3 .log files, 1 for each command – tracks the settings used for analysis
   2. 19 .csv files, 1 for each ROI – contains the Y values for the raw, fity, and dfy series
   3. 19 .png files, 1 for each ROI – captures the default view for each pyplot viewer
   4. 19 .txt files ending in \_evts, 1 for each ROI – contains descriptive information for all detected events
   5. 3 .txt files ending in \_evts\_formatted, 1 for each command – combines descriptive information for all detected events
   6. Items a-d should be saved for posterity, items in e will be used for post-processing analyses

### Analyzing and interpreting the processed tutorial data

The evts\_formatted.txt files contain all of the descriptive information necessary for post-processing analyses and interpretation. The 3 files correspond to 3 different commands/normalization methods described above.

# **Final notes**