QAMyData User Guide

Contents

Introduction

QAMyData

The UK Data Service QAMyData tool is an easy-to-use open source tool/web service that automatically detects some of the most common problems in survey/numeric data and creates a detailed report on the data file submitted (a 'data health check'). Data depositors, users and publishers can act on the results and resubmit the file until a 'clean bill of health'/certificate is produced. The tool offers a number of configurable tests for numeric data files (file, metadata; data integrity and disclosure risk review) whereby users can select thresholds for their own acceptance testing; enabling them to create a unique **Data Quality Profile** that helps them meet FAIR data requirements.

QAMyData is available to download from the UK Data Service GitHub page (https://github.com/ukdataservice/qamd) currently under an MIT Licence. Users should note that the previous version, 0.2.0, is available under a Creative Commons Attribution-NonCommercial 4.0 International Licence (CC BY-NC 4.0)).

About this Guide

This Guide sets out the QAMyData download, installation and running processes on two operating systems: Windows and Mac. The download section is same for both operating systems while the installation and running processes have some minor differences. Many thanks to Cristina Magder for preparing the guide, test data and training materials.

Contact Us

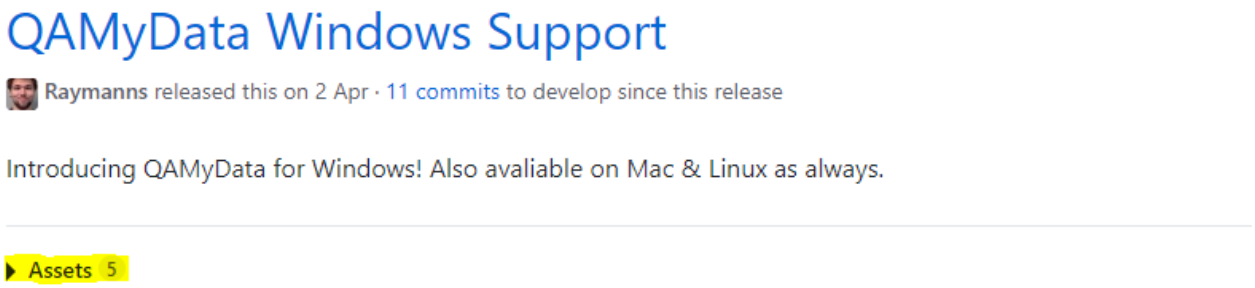
You can contact us using the GitHub page or by e-mailing us at QAMyData@UKDataService.ac.uk.

Downloading QAMyData (Windows and Mac)

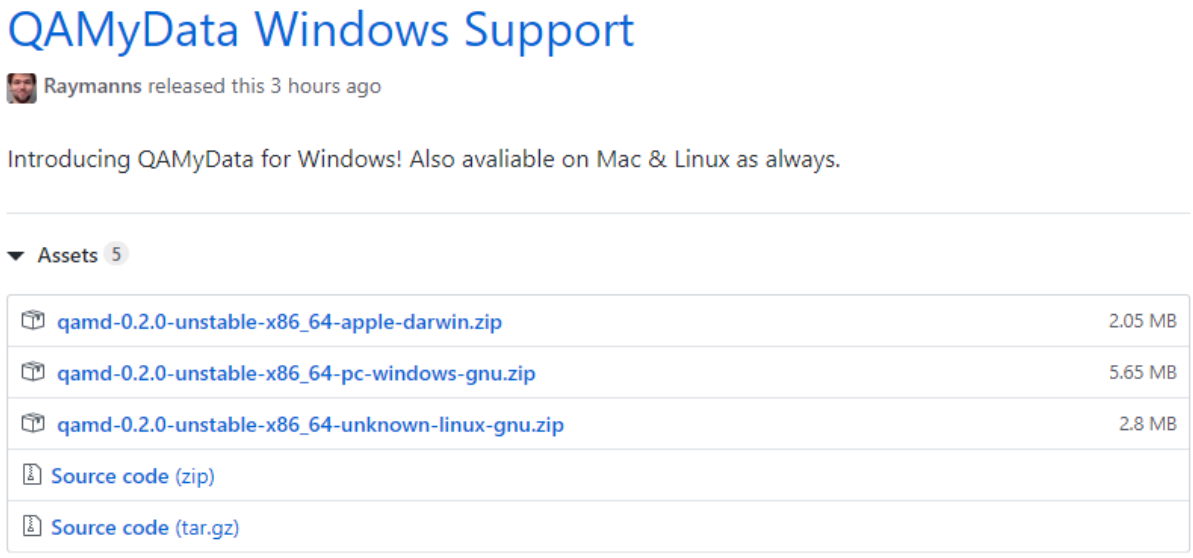
1. Go to the UK Data Service GitHub QAMD (https://github.com/ukdataservice/qamd) page

2. Navigate to the releases tab

3. Click on Assets



4. Download the appropriate for your Operating System (Mac, Windows or Linux) by clicking on the zip's name (text and screenshot need to change with new release)



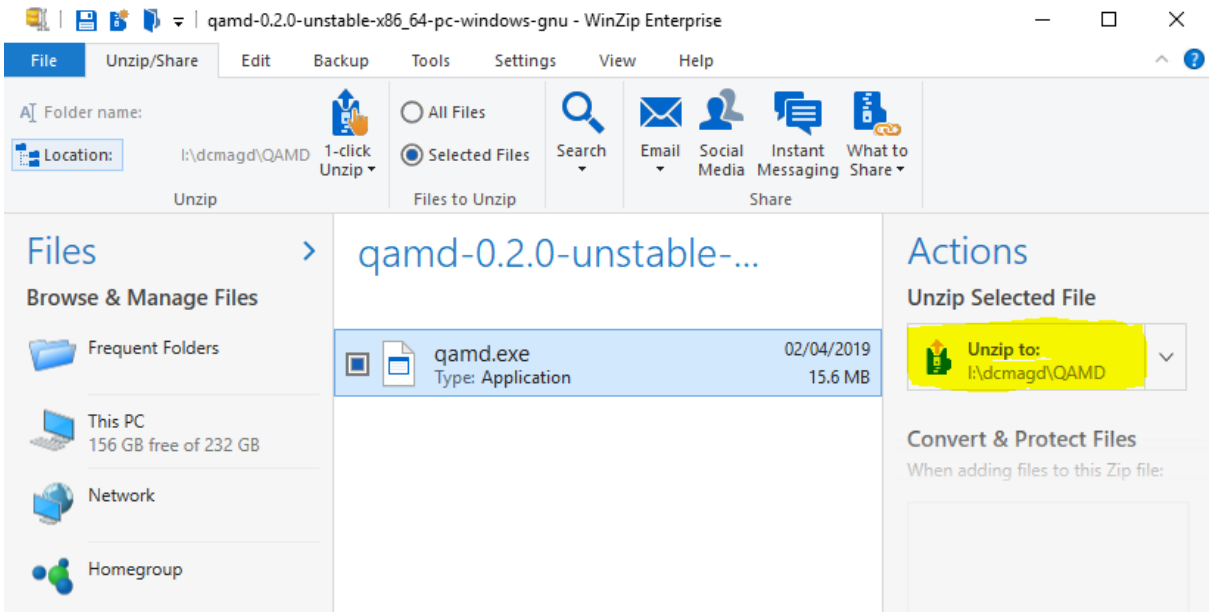
Using QAMyData on Windows

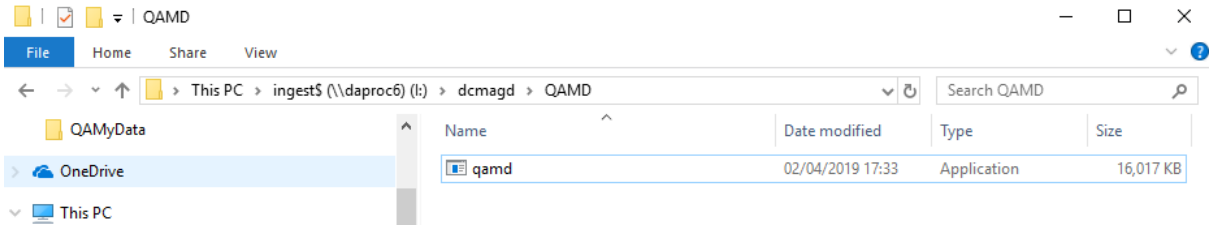
How to Install QAMyData

1. Open File Explorer and create a new folder QAMD in a location of preference (Hold down the Ctrl, Shift, and N keys at the same time to create a new folder and name it QAMD)

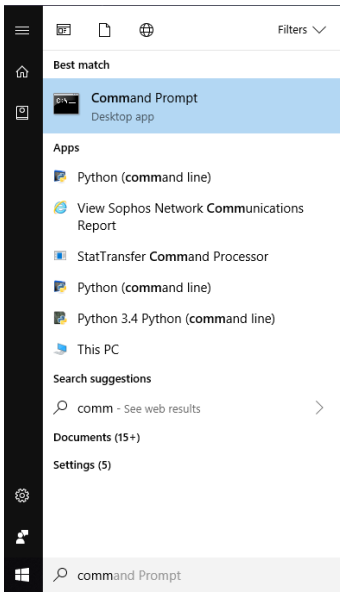
2. Navigate to the where the zip has been downloaded

3. Unzip QAMD by opening the downloaded zip in WinZip or 7zip and selecting Unzip to: (create a QAMD folder in a location of preference for ease of using the tool)

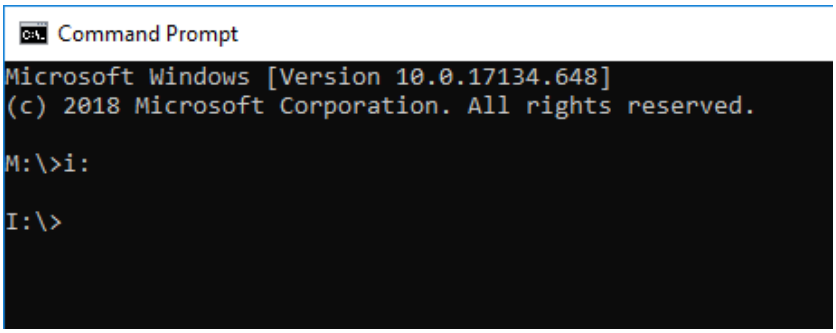




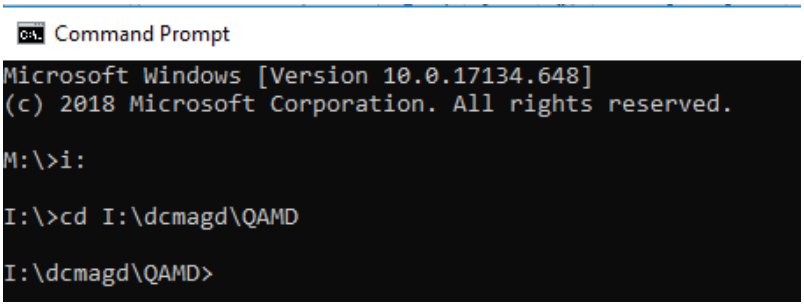
4. Open Command Prompt (type in **comm** in the search taskbar and click on “Command Prompt”)



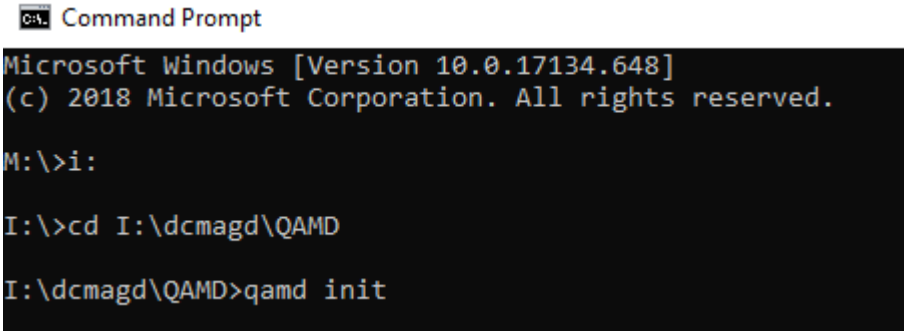
5. Change to the drive to where you created your QAMD folder by typing in, in this case, **I:** (the drive letter followed by a colon); press enter to run the command



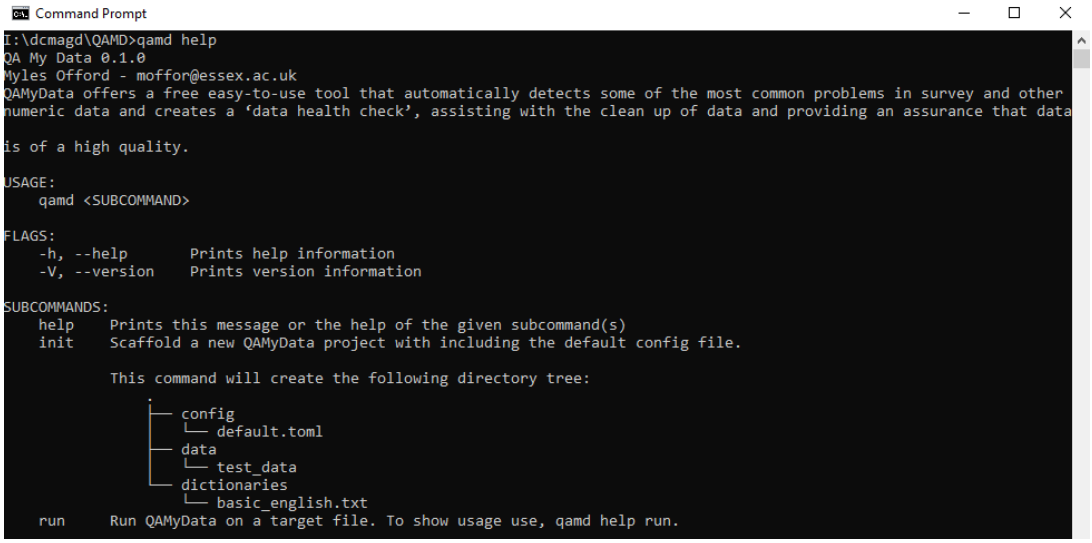
6. Change directories (cd) to the QAMD directory (type in **cd** followed by the path where your QAMD folder is (in this case cd I:\dcmagd\QAMD); press enter to run the command



7. Type in **qamd init** for the tool to create the folder structure and download the default configuration file, a basic English dictionary and the mtcars test data; and press enter to run the command



8. Type in **qamd help** to open the tool's help; and press enter to run the command



QAMyData is now installed and can be run.

How to Run QAMyData

1. In order to run QAMyData, type in **qamd run** followed by the location of your data file and its name (including the extension), the filename for your results, and the configuration file location and name (including the extension)

Such as:
qamd run "I:\dcmagd\QAMD\data\test\mtcars.sav" --output results_mtcars.html --config "I:\dcmagd\QAMD\config\ default.yaml"

Please make sure to run the command all on one line

The command can also take the following not compulsory options:

--metadata-only (if the user does not want locators included in the output file);
--output-format json (if the user would like to create a json format output rather than html);

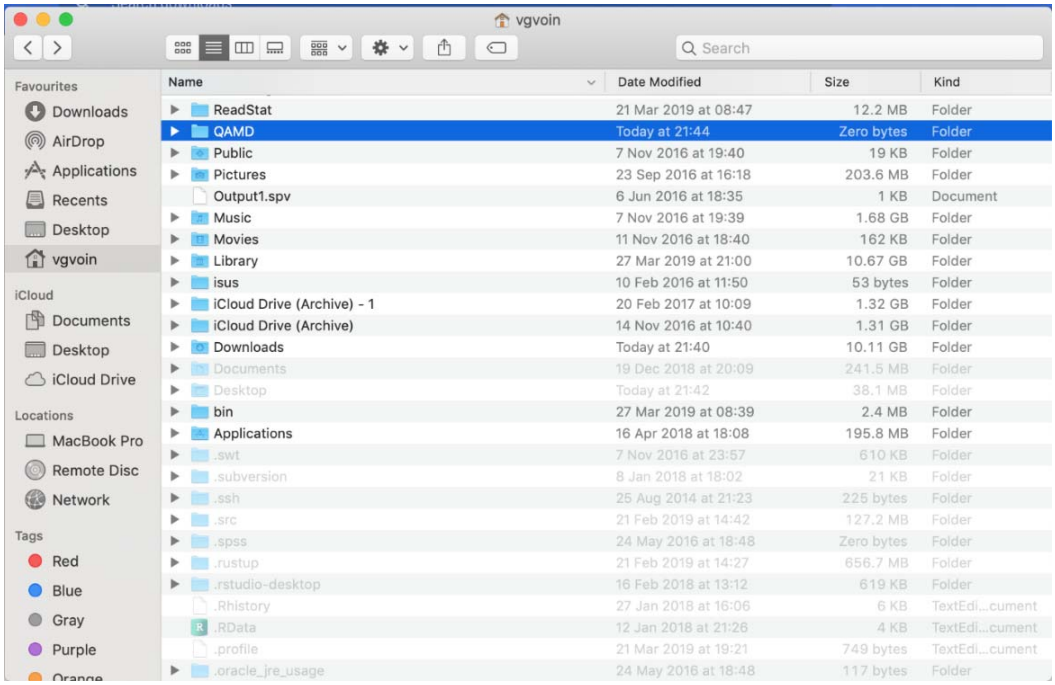
Example:
qamd run "I:\dcmagd\QAMD\data\test\mtcars.sav" --metadata-only --output-format json --output results_mtcars.json --config "I:\dcmagd\QAMD\config\ default.yaml"

Using QAMyData on MacOS

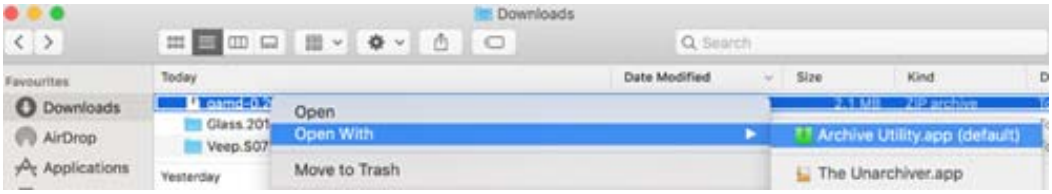
How to Install QAMyData

1. Open **Finder** (from a search or in the dock)

2. Navigate to Home and create a new folder named QAMD

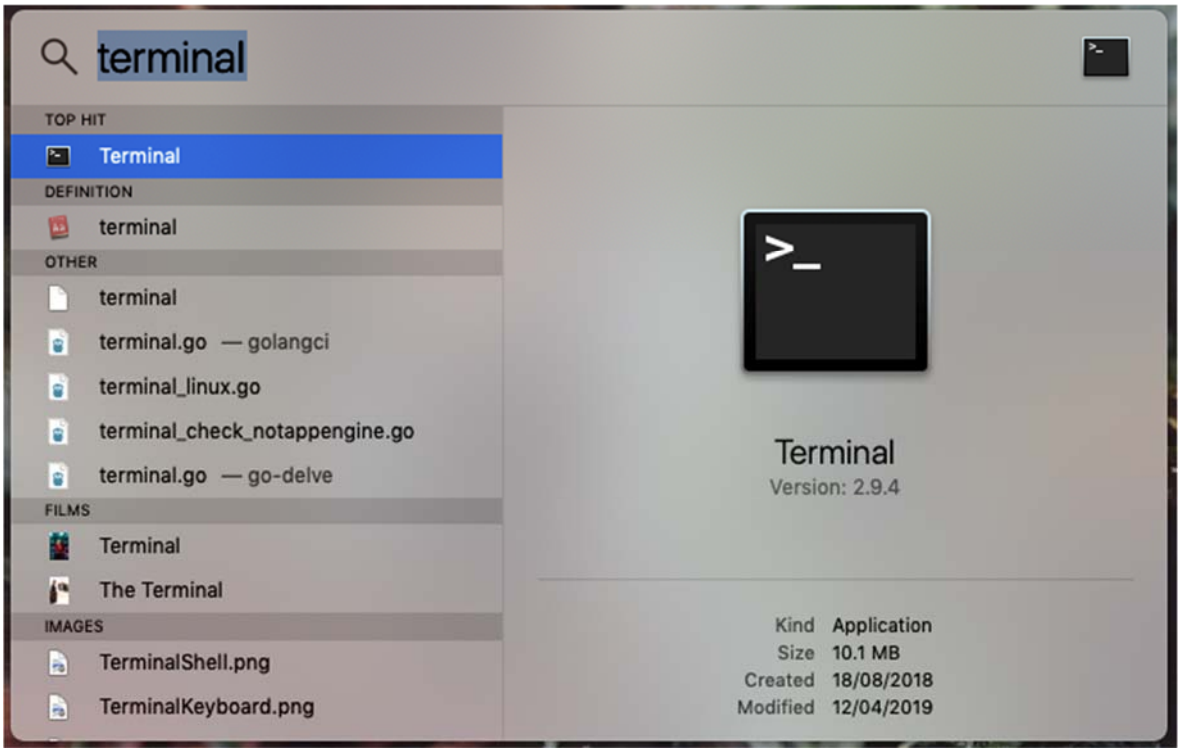


3. Navigate to where you downloaded the QAMD zip file and open the file (with Archive Utility app or any other similar application)

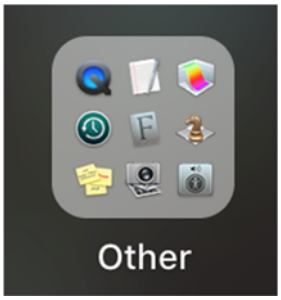


4. Copy the QAMD executable file from the target folder to the QAMD folder created in Step 2

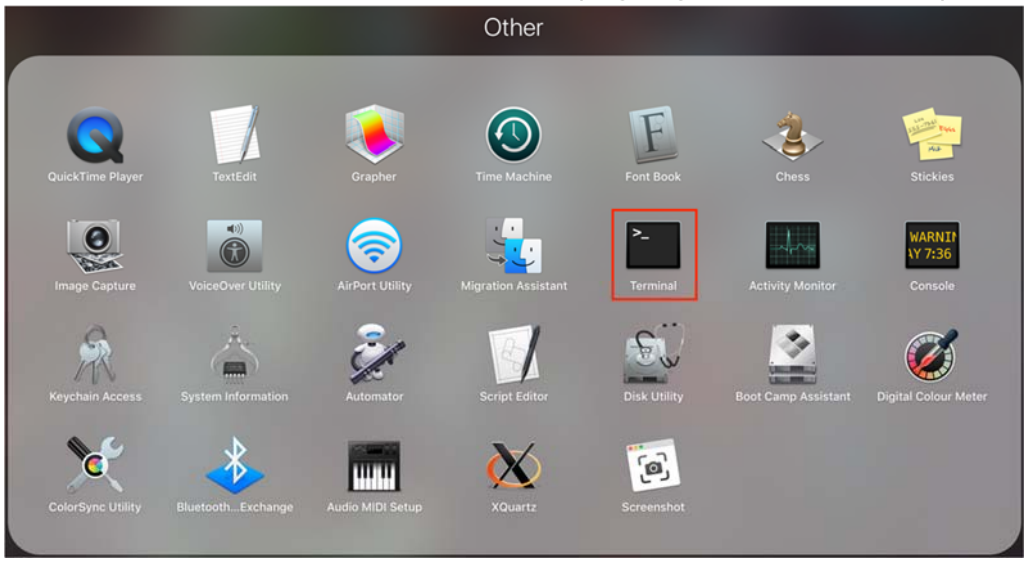
5. Open the **Terminal** application. This is found in the **Applications**>**Other** folder. To locate either:
    a. Enter “Terminal” in the search bar in the top left corner



b. Or open **Launchpad** from the dock and locate the application in the **Other** folder



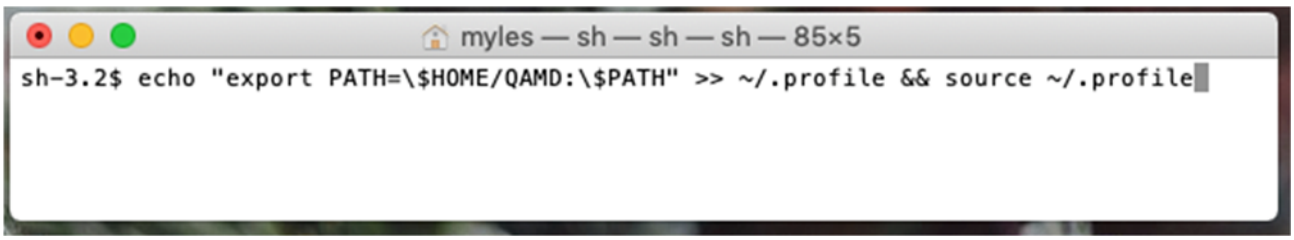
Choose the **Terminal** application (highlighted in red box).



6. In the **Terminal** interface, copy and paste the following to set your PATH variable

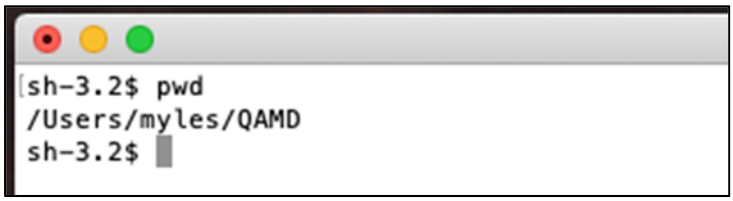
echo "export PATH=\$HOME/QAMD:\$PATH" >> ~/.profile && source ~/.profile

(Press **enter** to run a command in Terminal)



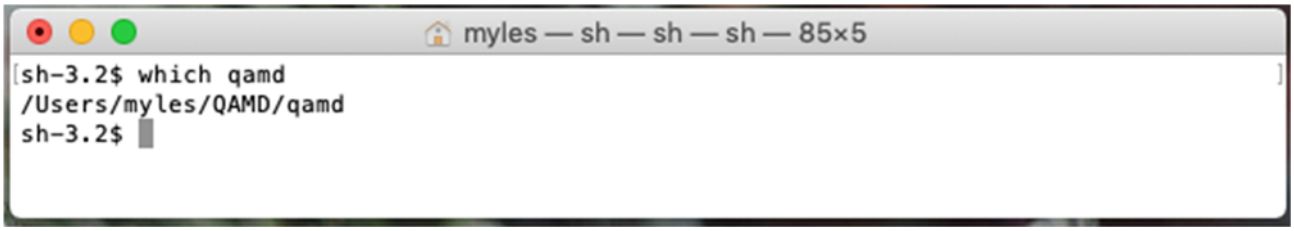
7. Now change the directory by entering cd **$HOME/QAMD**

   You can verify this worked by running **pwd**



8. Enter **qamd init** for the tool to create the folder structure and to download the default configuration file, a basic English dictionary and the test data (mtcars)

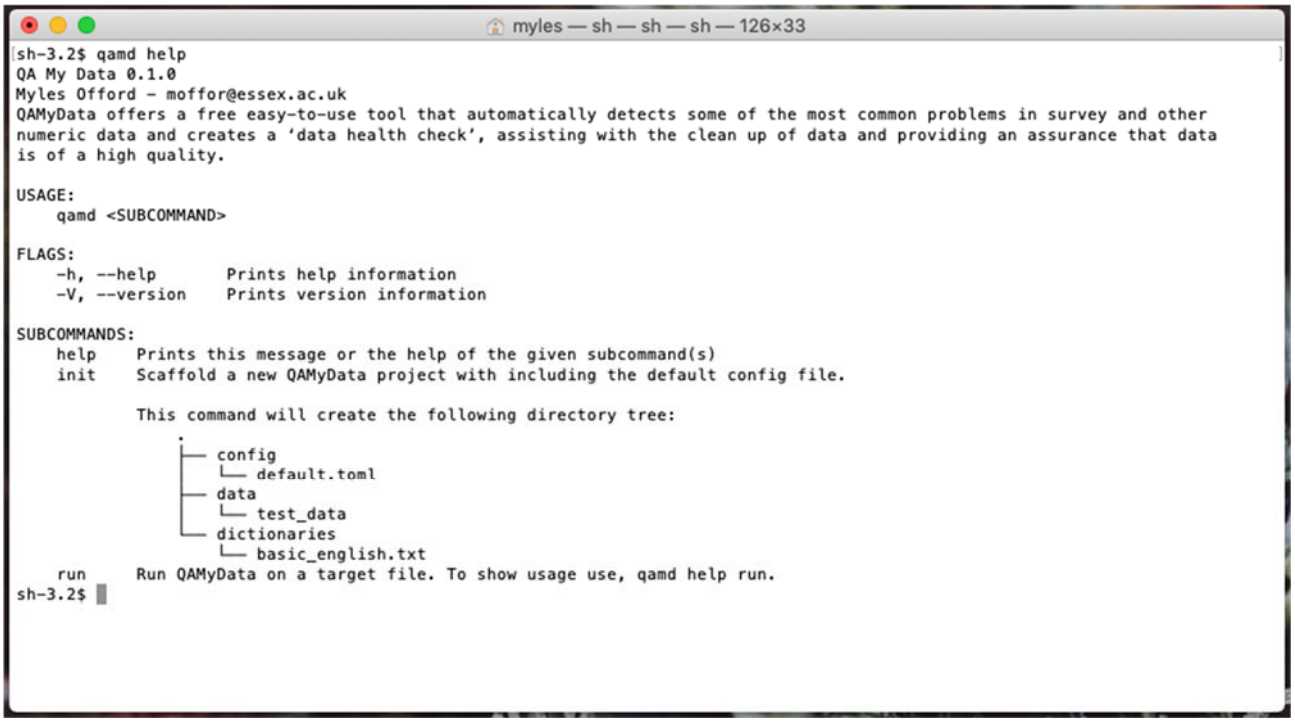
   You can confirm the installation by entering **which qamd**



If the command gives you something like above, QAMyData is fully installed and ready to be run.

If you receive a message “qamd not found” then double check each step in order.

9. Now enter **qamd help** to open the tool's help



How to Run QAMyData

1. In order to run QAMyData on a dataset type in **qamd run** followed by the location of your data file and its name (including the extension), the filename for your results, and the configuration file location and name (including the extension)

Such as:
qamd run ./data/test/mtcars.sav --output results_mtcars.html --config ./config/default.yaml

Please make sure to run the command all on one line

The command can also take the following not compulsory options:

--metadata-only (if the user does not want locators included in the output file);
--output-format json (if the user would like to create a json format output rather than html);

Example:
qamd run ./data/test/mtcars.sav --metadata-only --output-format json --output results_mtcars.json --config ./config/default.yaml

Understanding the Configuration File (config.yaml)

The configuration is written in YAML, a human-readable data-serialization language that enables very simple and concise configuration files; by opening the configuration file in any available text editor (Notepad++, Pages etc.) the user can easily configure the parameters and change the initial settings for QAMyData.

```yaml
---
##################################################
## QAMYDATA: Health Checks for Your Data Files ##
##################################################

# Welcome to the default configuration (config) file for QAMYDATA.
# The file is written in YAML (YAML Ain't Markup Language), which is a human-readable language commonly used for configuration files.
# The config is divided into 4 types of tests: Basic File Checks, Metadata Checks, Data Integrity Checks and Disclosure Control Checks.
# Lines starting with '#' are comments so they are ignored.


#######################
## Basic File Checks ##
#######################

basic_file_checks:
  # Checks whether the file name contains illegal/odd/non-compliant characters
  bad_filename:
    setting: "^([a-zA-Z0-9]+)\\.([a-zA-Z0-9]+)$"
    desc: "File name should match the user specified pattern"

#####################
## Metadata Checks ##
#####################

metadata:
  # Checks high-level grouping (for example, useful if dataset can be grouped by household)
  primary_variable:
    setting: HouseholdID
    desc: "Counts the unique occurrences for the grouping variable specified"

  # Checks whether any variables do not have labels
  missing_variable_labels:
    setting: true
    desc: "Variables should have a label"

  # Checks whether any user-defined missing values do not have labels (sysmis) - SPSS only
  value_defined_missing_no_label:
    setting: true
    desc: "User-defined missing values should have a label (SPSS only)"
```

The zip bundle downloadable from the GitHub page contains the default configuration file, which is divided into 4 main categories of checks:

- Basic File Checks;
- Metadata Checks;
- Data Integrity Checks;
- Disclosure Control Checks.

Each test is first described in the commented out line(s), followed by the name of the test, the setting the test will check for and also the description that will appear in the output file. Both the setting and the description that will appear in the output file can be changed by the user.

Basic File Checks

Basic file checks contains one configurable test named bad_filename; by default the check tests for the file name to consist only of alphanumeric characters (A-Z and 0-9). If needed the regular expression can be changed to reflect different rules.

```
# Checks whether the file name contains illegal/odd/non-compliant characters
bad_filename:
setting: "^([a-zA-Z0-9]+)\\.([a-zA-Z0-9]+)$"
desc: "File name should match the user specified pattern"
```

Metadata Checks

Metadata checks contains checks for both value and variable level, and the settings can take either a true or false value, an array of user-defined values, or a numeric value as shown in the examples below.

```
#Checks high-level grouping (for example, useful if dataset can be grouped by household)
primary_variable:
setting: HouseholdID
desc: "Counts the unique occurrences for the grouping variable specified
```

The setting for this check can take the variable that would further group your data such as by SchoolID:

```
# Checks high-level grouping (for example, useful if dataset can be grouped by household)
primary_variable:
setting: SchoolID
desc: "Counts the unique occurrences for schools”
```

By default QAMyData will check if all variables have labels, however if you are checking a csv file, or not interested in missing variables labels, you can change the setting to false:

```
# Checks whether any variables do not have labels
missing_variable_labels:
setting: true
desc: "Variables should have a label"
```

```
# Checks whether any variables do not have labels
missing_variable_labels:
setting: false
desc: "Variables should have a label"
```

Or you can comment out the check from your new config file (this is applicable to all tests):

```
# Checks whether any variables do not have labels
# missing_variable_labels:
# setting: true
# desc: "Variables should have a label"
```

```
# Checks whether any user-defined missing values do not have labels (sysmis) - SPSS only
value_defined_missing_no_label:
setting: true
desc: "User-defined missing values should have a label (SPSS only)"
```

The underlying software, *Readstat*, allows to check if any defined system missing values in SPSS don't have a label (such as all -8 and -9 have been defined missing, but the values don't have labels such as "Don't know", "Refused to answer" etc.)

```
#Checks whether any variable names and labels contain
illegal/odd/non-compliant characters
variable_odd_characters:
    setting:
      - "&"
      - "#"
      - "  "
      - "@"
      - "*"
      - "ç"
      - "ô"
      - "ü"
desc: "Variable names and labels should not contain the
specified characters"
```

Several characters can create problems when trying to run a script, or input a data in a data browsing software like Nesstar. This check is to ensure that non-compliant characters are not included in the variable names and labels. The same check exists for value labels as well. All the characters are user-defined based on necessity.

```
#Checks whether any variable names and labels contain
illegal/odd/non-compliant characters
variable_odd_characters:
setting:
- "£"
- "~"
desc: "Variable names and labels should not contain the
specified characters"
```

Stata has a limit of 79 characters per variable label and 39 characters for value labels. By default QAMyData will check whether these parameters are respected; however the user can change the setting to any numeric value that would apply to software they are using:

```
# Checks whether any variable labels exceed user-defined
number of characters, e.g. 79
variable_label_max_length:
setting: 79
desc: "Variable labels should not exceed the defined
number of characters"
```

```
# Checks whether any variable labels exceed user-defined
number of characters, e.g. 79
variable_label_max_length:
setting: 120
desc: "Variable labels should not exceed the defined
number of characters"
```

QAMyData has a built in spellchecker test for both variable and value labels by using a user-defined dictionary (this allows spellchecks for any languages). Depending on the operating system, the user will have to define the path to the dictionary accordingly:

For Mac: - "/usr/share/dict/words"
For Windows: - "C:\\path\\to\\dictonary\\file.txt"

```
# Checks variable labels for spelling errors using a user-
defined dictionary file
# Please remember you must input the correct path to the
dictionary file in order for the check to run on your data
variable_label_spellcheck:
setting:
- "/usr/share/dict/words"
- "C:\\path\\to\\dictonary\\file.txt"
desc: "Variable labels should have correct spelling"
```

An example of the test run on a Windows OS:

```
# Checks variable labels for spelling errors using a user-defined dictionary file
# Please remember you must input the correct path to the dictionary file in order for the check to run on your data
variable_label_spellcheck:
setting:
        - "A:\\dcmagd\\qamd\\dictionaries\\en.txt"
        desc: "Variable labels should have correct spelling"
```

Please remember to specify a dictionary for all spellcheck and stop word tests

Data Integrity Checks

Data Integrity Checks contains tests that verify the integrity of the data file such as system missing values over defined threshold, duplicate values in unique identifiers or non-compliant characters in string values.

```
# Checks the percentage of undefined missing values ('sysmis')
system_missing_value_threshold
setting: 25
desc: "Variable should not exceed the specified
```

Disclosure Control Checks

Disclosure Control Checks are useful for detecting direct identifiers by using RegEx and disclosive outliers by checking for unique values.

```
regex_patterns:
setting:
    - "^([\\w\\.\\-]+)@([\\w\\-]+)((\\.(\\w){2,4})+)$", # checks for
e-mail addresses
desc: "Variable should not contain the user-specified
RegEx pattern" of system missing values"
```

The RegEx check is resource intensive so it has been commented out from the default configuration file (using # on all relevant lines). The user can configure the check with any RegEx they might find useful such as:

```
regex_patterns:
setting:
- "^[A-Za-z]{1,2}[0-9A-Za-z]{1,2}[ ]?[0-9]{0,1}[A-Za-z]{2}$"
desc: Values matching the regex pattern fail (full UK
postcodes found in the data)percentage of system missing
values"
```

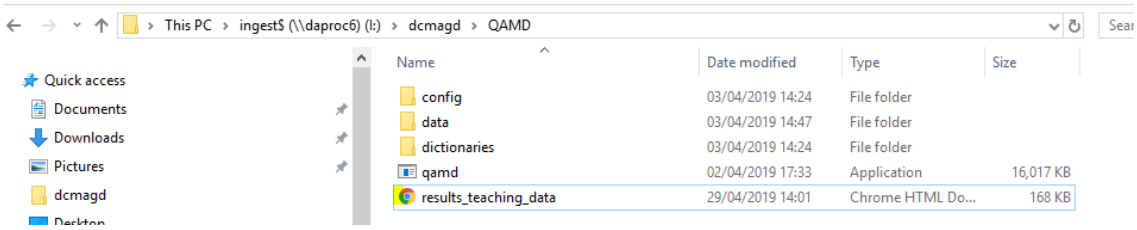
```
regex_patterns:
setting:
- "^([A-HK-PRSVWY][A-HJ-PR-Y])\s?([0][2-9]|[1-9][0-
9])\s?[A-HJ-PR-Z]{3}$"
desc: Values matching the regex pattern fail (UK vehicle
registration numbers (as defined by the DVLA and put into
effect from September 2001) found in the data)
```

The description that will appear in the output file can be changed to match the RegEx used for an easier understanding of the output file especially if several regex checks are used.

If you have any further ideas for useful tests please let us know! QAMyData@UKDataService.ac.uk.

Understanding the Output File

QAMyData save the results of your html output file in the top level of you QAMD directory.



QAMyData save the results of your html output file in the top level of you QAMD directory. Go to your **QAMD** folder in Mac Finder. Open the html file with an internet browser (e.g. Google Chrome, Firefox or Safari) in order to view the results.

QAMyData

teaching-data%set.sav

Raw Case Count: 10210
Aggregated Case Count: 0
Total Variables: 188
Data Type Occurrences: Numeric: 186, String: 2
Created At: 2019-02-18 13:37:39
Last modified at: 2019-02-18 13:37:39
File Label:
File Format Version: 2
File Encoding: WINDOWS-1252
Compression type: Rows

Basic File Checks

Name	Status (N)	Description
Bad file name	failed (1)	File name should match the user specified pattern

Metadata Checks

Name	Status (N)	Description
Missing variable labels	failed (8)	Variables should have a label
Variable odd characters	failed (2)	Variable names and labels should not contain the specified characters ["&", "#", " ", "@", "*", "ç", "ô", "ü"]
Variable label max length	failed (6)	Variable labels should not exceed the defined number of characters (79 characters)

The header of the file contains information on the number of cases and variables, the encoding of the file, and when the file was created and last modified.

All the tests listed that are highlighted in green have passed (there were no issues encountered according to the thresholds set), while the tests in red have failed (QAMyData has identified issues in certain variables/values).

To locate the problems, simply click anywhere on a red test (line) and this will take you to another table underneath, containing the first 1000 issues. For example, to view the results of the failed “Variable odd characters” test, click on the failed test and scroll down to the bottom. QAMyData has identified that variables V137 and OwnTV contain “odd” characters in their label.

Variable odd characters

# (limited to 1000)	Variable	Row number
1	OwnTV	-
2	V137	-