AutoDocish: Automated-ish Dataset Documentation

Elizabeth Wickes @elliewix
University of Illinois at Urbana Champaign

The Problem:

- Documentation is usually left to a researcher.
- Who has higher priority research to work on.
- Many think that documentation needs to be extensive.
- So documentation isn't always done.

What does great documentation look like?

What ~~does~~ can great documentation look like?

This isn't about open science

- Well documented and preserved datasets and code are first useful to you and your team.
- You should be able to reuse your own data as part of future work.
- Documentation is almost always necessary for reuse -- even for Future You.

Mimimum viable documentation

- **Enough** information,
- about the project, methods, and materials
- such that the information is **maintainable over time**,
- in an **accessible** format,
- and **valuable for those who need it**.

tl;dr

- Something is better than nothing
- It doesn't need to be a dissertation
- Seriously, just write something
- Seriously.

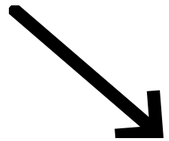
What are the basic pieces of documentation?

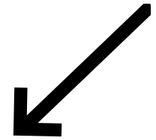
Codebook

- "What do the data values mean?"
- Describes what, if any, coded values mean within the data file.

Data dictionary

- "What does this data file contain?"
- Describes the individual questions or measurements contained within a data file.





readme.txt

- May contain a mix of both, and other important contextual info.

Sometimes it can do all the things

ICPSR

Variable name: Human name

PERSONID: Participant ID

This variable has 32,320 valid cases out of 32,320 total cases.

Location: 6-15 (width: 10; decimal: 0)
Variable Type: character

Descriptive information

R01_AC1002: Ever smoked a cigarette, even one or two puffs

Have you ever smoked a cigarette, even one or two puffs?

(If No (2) or Don't Know (-8) or Refused (-7), Go to Box R01_AE1.)

ASK: All respondents.

Value	Label	Unweighted Frequency	%
1	1 = Yes	25187	77.9 %
2	2 = No	7116	22.0 %
	Total	**32,303**	**99.9%**

Descriptive statistics and codes

http://doi.org/10.3886/ICPSR36498.v1

Sometimes it does a little less, but you can still get along

1. Type of Lawyer
    a. Solicitor: **Total Responses: 23**
        i. Responses: UK-101; UK -102; UK-103; UK-104; UK-109; UK-107; UK-113; UK-115; UK-116; UK-123; UK-124; UK-126; UK-127; UK-132;UK-133; UK-134; UK-135; UK-138; UK-140; UK-143; UK-146; UK-147; UK-151
    b. Barrister: **Total Responses: 28**
        i. Responses: UK-105; UK-106; UK- 108; UK-110; UK-111; UK-112; UK-114; UK-117; UK-118; UK-119; UK-120; UK-121; UK-122; UK-125; UK-128; UK-129;UK-130; UK-131; UK-136; UK-137; UK-139; UK-141; UK-142; UK-144; UK-145; UK-148, UK-149, UK-150.
    c. Barrister doing at least some work for the government: **Total Responses 3**
        i. Responses: UK-119; UK-130; UK-131

http://doi.org/10.3886/E17507V2

Sometimes they are machine readable

Column name	Corresponding survey question	Codes
Q1	Please select the option that best describes your status.	1="I am a US citizen or permanent resident"; 2= "I am an international student"
Q2	Please select your age	1="<18"; 2="18-25"; 3="26-30"; 4="31-35"; 5="36-40"; 6="41-45"; 7="45+"
Q3	What is your gender?	1="Male"; 2="Female", 3="Other", 4="I do not wish to respond"

http://doi.org/10.3886/E43668V1

Or sometimes
there's nothing...

http://www.doi.org/¯_(ツ)_/¯

What are the core elements?

- A little bit of data profiling
- A little bit of data cleaning
- A little bit of human narrative

Easy to improve something...

Perfect > Something > Nothing

Hard to make something from nothing...

AutoDocish

- Automate what you can
- Focus on the elements only you can answer
- Move on with your life

Automate what you can

- Data dictionaries report the headers
- Codebooks report unique values
- Descriptive statistics are...statistics

These are things a computer can do.

What's left for the human?

- Context
- Methods
- Pesky human sentences

Let's take a look

So you've got a CSV...

```
$ python data_profile.py source output missing_code
                         1      2      3
```

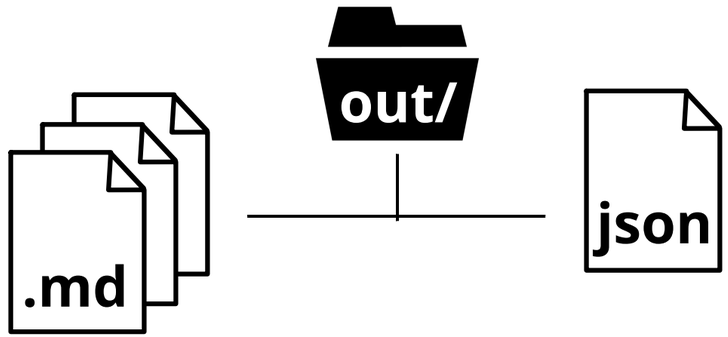
1. The data source, either folder or file

2. Folder name for the profile files to be written

3. Missing code for the data, presumes empty string if not provided.

Outputs

- Inside the output folder:
  - One profile per data file
  - A single JSON file with all profile data



Single file usage

```
$ python data_profile.py vagrants.csv vagrants/ [missing
Generating profile for: vagrants.csv
vagrants/ already exists. Will OVERWRITE.
Profiles written into vagrants/
```

Example data from:

Crymble, Adam et al.. (2015). Vagrant Lives: 14,789 Vagrants Processed by Middlesex County, 1777-1786 (version 1.1). Zenodo. 10.5281/zenodo.31026

Folder usage

```
$ python data_profile.py fakedata fakes ''
Generating profiles for 1000 files
fakes/ created
Profiles written into fakes/
```

File level information

File level information

Given a numerical CSV file

File level information

Data Profile for fakedata/0.csv

Generated on: 2016-Aug-11 20:46:47

Number of columns: 10
Number of rows: 108
Using missing value of: (empty string)

Column info: numerical

yfuncu

Column name

---

* Description of column:
* Collection methods:
* Description of data values and units:
* Reason for missing values:

Questions to fill out

* percent_digit: 100%
* percent_missing: 0%
* min_digit: 1.0
* missing: 0
* unique_value_content: Not reported (More than 10 unique values)
* unique_values: 103 (this includes missing values)
* max_digit: 996.0

Descriptive statistics

File level information

```
Vagrant ID Number,Given Names,Surname,Gender of Le
6625.1.1,Mitchell,Bruce,M,[lead vagrant],1,Solo Ma
6720.1.1,John,Drivee,M,[lead vagrant],1,Solo Male
8352.1.1,Peter,Smith,M,[lead vagrant],1,Solo Male
5750.1.1,Thomas,Herry,M,[lead vagrant],1,Solo Male
5265.1.1,James,Guttery,M,[lead vagrant],1,Solo Mal
5851.1.1,Robert,Ogilby,M,[lead vagrant],1,Solo Mal
460.1.1,Laurence,Least,M,[lead vagrant],1,Solo Mal
1964.1.1,Betty,Bruce,F,[lead vagrant],1,Single Fem
8929.1.1,John,McFarland,M,[lead vagrant],1,Solo Ma
1914.2.2,[Child],Scarlet,[unknown],[Child],2,Depen
1914.1.2,Isabella,Scarlet,F,[lead vagrant],2,Group
9315.1.1,Christiana,Gray,F,[lead vagrant],1,Single
4208.1.1,Donald,Ross,M,[lead vagrant],1,Solo Male
10228.2.2,[Child],McKenzie,[unknown],[Child],2,Dep
10228.1.2,Mary,McKenzie,F,[lead vagrant],2,Group
2769.2.2,[Wife],Frazier,F,[Wife],2,Dependent,City
2769.1.2,David,Frazier,M,[lead vagrant],2,Group Le
```

Given a file of text codes

Column info: text

Gender of Lead Vagrant
---
* Description of column:
* Collection methods:
* Description of data values and units:
* Reason for missing values:
* percent_digit: 0%
* percent_missing: 0%
* min_digit: no digits
* missing: 0
* unique_value_content: The values are:
  * [unknown]
  * M
  * F
* unique_values: 3 (this includes missing values)
* max_digit: no digits

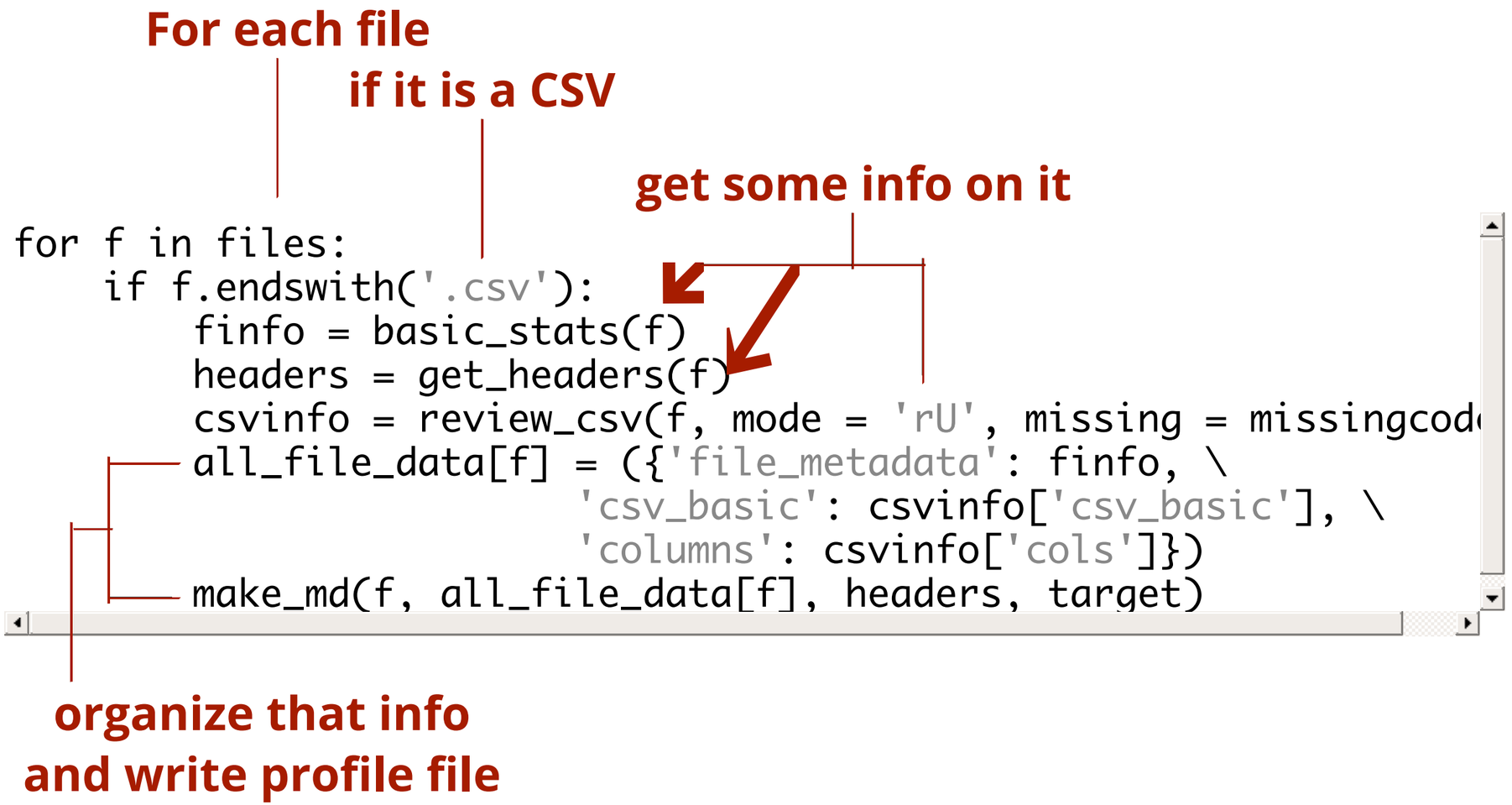
Column name

Questions to fill out

Descriptive statistics

```
{
    "MiddlesexVagrants1777-1786v1.1.csv": {
        "columns": {
            "Gender of Lead Vagrant": {
                "percent_digit": "0%",
                "percent_missing": "0%",
                "min_digit": "no digits",
                "missing": 0,
                "unique_value_content":
                        "The values are:\n\t* [unknown]\n\t* M\n\t* F\n",
                "unique_values": "3 (this includes missing values)",
                "max_digit": "no digits"
            },
            [more columns...]
        },
        "csv_basic": {
            "num_rows": 14789,
            "missing": "[missing]",
            "num_columns": 29
        },
        "file_metadata": {
            "last_access": "2016-08-11 21:02:30",
            "size": 4641076,
            "last_modified": "2016-04-24 17:31:01",
            "filename": "MiddlesexVagrants1777-1786v1.1.csv"
        }
    },
    [more files...]
}
```

So what's going on in the code?


For each file
if it is a CSV
get some info on it
for f in files:
if f.endswith('.csv'):
finfo = basic_stats(f)
headers = get_headers(f)
csvinfo = review_csv(f, mode = 'rU', missing = missingcod
all_file_data[f] = ({'file_metadata': finfo, \
'csv_basic': csvinfo['csv_basic'], \
'columns': csvinfo['cols']})
make_md(f, all_file_data[f], headers, target)
organize that info
and write profile file

BYOPF

"bring your own profiling functions"

```
for f in files:
    if f.endswith('¯\_(ツ)_/¯'):
        finfo = ٩(◔)۶(f)
        headers = ಠДಠ(f)
        csvinfo = ಥДಥ(f, mode = 'rU', missing = missingcode)
        all_file_data[f] = ({'file_metadata': finfo, \
                             'csv_basic': csvinfo['csv_basic'], \
                             'columns': csvinfo['cols']})
        make_md(f, all_file_data[f], headers, target)
```

(valid python3 code ^_^)

pip not required

- Uses all standard packages in 2.7

```
import os
from os.path import isfile, join
import csv
import datetime
import glob
import sys
import json
```

Future directions

Just a proof of concept

- Easier to start with something than nothing goes for code as well
- CSVs are common and easy, so the best of low hanging fruit
- Needs more work for data type, more granular control, etc.

Features I'd like to add:

- Turn this into a web tool that can be locally launched
- Better auto detection for data types
- More statistics
- Prettier outputs

Questions?

https://github.com/elliewix/data-profile-tool

@elliewix