Data Wrangling notes

Table of Contents

[Check out ‘How web works?’ 1](#_Toc501878378)

[Data Wrangling Summary 1](#_Toc501878379)

[Gather 2](#_Toc501878380)

[Assess 2](#_Toc501878381)

[Clean 2](#_Toc501878382)

[Reassess and Iterate 2](#_Toc501878383)

[Store (Optional) 2](#_Toc501878384)

[Ipython (Jupyter Notebook) 3](#_Toc501878385)

[Installing packages from Anaconda.org 3](#_Toc501878386)

[Installing non-conda packages 4](#_Toc501878387)

[**Gather: Summary** 5](#_Toc501878388)

[DATA Assessment 5](#_Toc501878389)

[Data Assessment Types (Visual and Programmatic) 5](#_Toc501878390)

[Data Quality dimensions 6](#_Toc501878391)

[**Data Quality Dimensions** 6](#_Toc501878392)

[Tidy Data 8](#_Toc501878393)

[Assess: Summary 8](#_Toc501878394)

[**Clean: Summary** 8](#_Toc501878395)

[Coding tips: 9](#_Toc501878396)

[Check if a single cell value in pandas dataframe is NaN 9](#_Toc501878397)

# Check out ‘How web works?’

Free [**Web Development course**](https://www.udacity.com/course/web-development--cs253): concepts 2-5 and 24-30 in Lesson 1 ("How the Web Works").

# Data Wrangling Summary

[**Data wrangling template link**](https://d17h27t6h515a5.cloudfront.net/topher/2017/September/59cebf3f_data-wrangling-template/data-wrangling-template.ipynb)

**Stack, unstack, pandas dataframes**

http://nikgrozev.com/2015/07/01/reshaping-in-pandas-pivot-pivot-table-stack-and-unstack-explained-with-pictures/

## Gather

* Depending on the source of your data, and what format it's in, the steps in gathering data vary.
* High-level gathering process: obtaining data (downloading a file from the internet, scraping a web page, querying an API, etc.) and importing that data into your programming environment (e.g. Jupyter Notebook).

## Assess

* Assess data for:
  + Quality: issues with content. Low quality data is also known as dirty data.
  + Tidiness: issues with structure that prevent easy analysis. Untidy data is also known as messy data. Tidy data requirements:
    1. Each variable forms a column.
    2. Each observation forms a row.
    3. Each type of observational unit forms a table.
* Types of assessment:
  + Visual assessment: scrolling through the data in your preferred software application (Google Sheets, Excel, a text editor, etc.).
  + Programmatic assessment: using code to view specific portions and summaries of the data (pandas' head, tail, and info methods, for example).

## Clean

* Types of cleaning:
  + Manual (not recommended unless the issues are one-off occurrences)
  + Programmatic
* The programmatic data cleaning process:
  + Define: convert our assessments into defined cleaning tasks. These definitions also serve as an instruction list so others (or yourself in the future) can look at your work and reproduce it.
  + Code: convert those definitions to code and run that code.
  + Test: test your dataset, visually or with code, to make sure your cleaning operations worked.
* Always make copies of the original pieces of data before cleaning!

## Reassess and Iterate

* After cleaning, always reassess and iterate on any of the data wrangling steps if necessary.

## Store (Optional)

* Store data, in a file or database for example, if you need to use it in the future.

# Ipython (Jupyter Notebook)

1. To use anaconda we need to run 'conda update conda' type of commands. Conda is anaconda's installer.
2. In order that the terminal is able to find this 'conda' cmd, one needs to update the environment variable 'PATH' using 'export PATH=/Users/varsha/anaconda/bin:$PATH' script that needs to added to .bash\_profile.
3. One can open this file using 'subl ~/.bash\_profile' script in terminal. Subl means using sublime text editor open the bash\_profile file.
4. Also to see all the values/locations present in the environment variable 'PATH' run command 'echo $PATH'
5. TO SIMPLY OPEN/ RUN AN IPYTHON NOTEBOOK GO TO THE NECESSARY DIRECTORY WHERE THE IPYNB IS PRESENT AND THEN RUN THIS COMMNAD IN TERMINAL.

"jupyter notebook' this opens the jupyter app in browser, here you can find the directory (you cd into) and thus see all ipynb files. open the desired file here.

1. To move the zip file to desired directory using command line

In the terminal write: mv ~/Downloads/file\_name.zip ~/Documents/blah\_...

Basically: mv source\_path destination\_path

1. To import zipfile into jupyter programmatically, write:
2. Import zipfile (then press shift+enter to execute this command)
3. Above zipfile supports ‘with’ statement as it is also a context manager.

Meaning it ensures that file opened gets closed and the related resources are freed.

Installing library packages

# [Installing packages from Anaconda.org](https://conda.io/docs/user-guide/tasks/manage-pkgs.html#id3)

Packages that are not available using conda install can be obtained from Anaconda.org. Formerly Binstar.org, Anaconda.org, is a package management service for both public and private package repositories. Anaconda.org is an Anaconda product, just like Anaconda and Miniconda.

To install a package from Anaconda.org:

1. In a browser, go to [http://anaconda.org](http://anaconda.org/).
2. To find the package named bottleneck, type bottleneck in the top-left box named Search Packages.
3. Find the package that you want and click it to go to the detail page.

The detail page displays the name of the channel. In this example it is the “pandas” channel.

1. Now that you know the channel name, use the conda install command to install the package. In your Terminal window or an Anaconda Prompt, run:
2. conda install **-**c pandas bottleneck

This command tells conda to install the bottleneck package from the pandas channel on Anaconda.org.

1. To check that the package is installed, in your Terminal window or an Anaconda Prompt, run:
2. conda list

A list of packages appears, including bottleneck.

NOTE: For information on installing packages from multiple channels, see [Managing channels](https://conda.io/docs/user-guide/tasks/manage-channels.html).

# [Installing non-conda packages](https://conda.io/docs/user-guide/tasks/manage-pkgs.html#id4)

If a package is not available from conda or Anaconda.org, you may be able to find and install the package with another package manager like pip.

NOTE: Both pip and conda are included in Anaconda and Miniconda, so you do not need to install them separately.

NOTE: Conda environments replace virtualenv, so there is no need to activate a virtualenv before using pip.

To install a non-conda package:

1. Activate the environment where you want to put the program:
   * On Windows, in your Anaconda Prompt, run activate myenv.
   * On macOS and Linux,in your Terminal window, run source activate myenv.
2. To use pip to install a program such as See, in your Terminal window or an Anaconda Prompt, run:
3. pip install see
4. To verify the package was installed, in your Terminal window or an Anaconda Prompt, run:

conda list

* [**Udacity: JSON from an Android Developer's perspective**](https://classroom.udacity.com/nanodegrees/nd002/parts/20a33e0b-0f64-41a3-a9d9-c6f1207265c6/modules/aef902cd-59a3-4cd2-af74-2d39b4b03b80/lessons/96402d84-c99d-4982-9edf-2430ef30d222/concepts/(https://www.youtube.com/watch?v=0IOCgHrTJGU)
* [**Mashery: API Data Exchange: XML vs. JSON**](https://www.mashery.com/blog/api-data-exchange-xml-vs-json)
* [**Reading and Writing JSON to a File in Python**](http://stackabuse.com/reading-and-writing-json-to-a-file-in-python/)

**Gather: Summary**

Gathering is the first step in the data wrangling process:

* **Gather**
* Assess
* Clean

Depending on the source of your data, and what format it's in, the steps in gathering data vary.

The high-level gathering process:

* obtaining data (downloading a file from the internet, scraping a web page, querying an API, etc.)
* importing that data into your programming environment (e.g. Jupyter Notebook)

# DATA Assessment

Dirty data vs. Messy data

Dirty – Content issues (Inaccurate, duplicate, typos)

Messy – structural issues

## Data Assessment Types (Visual and Programmatic)

* + 1. Visual Assessment (Detect and document )

(Why write observations of issues instead of writing observations as well as how to fix them? When your data's issues get complicated, writing how to fix each can get confusing, lengthy, and time-consuming. It can get overwhelming trying to think of how to clean something complicated immediately after documenting it.

If you are separating the assessing and cleaning steps of data wrangling, as we are in this lesson, writing observations only is good practice.

* + 1. Programmatic assessment:

These are the programmatic assessment methods in pandas that you will probably use most often:

* .head (DataFrame and Series)
* .tail (DataFrame and Series)
* .sample (DataFrame and Series)
* .info (DataFrame only)
* .describe (DataFrame and Series)
* .value\_counts (Series only)
* Various methods of indexing and selecting data (.loc and bracket notation with/without boolean indexing, also .iloc)

Try them out below and keep their results in mind. Some will come in handy later in the lesson.

Check out the [pandas API reference](https://pandas.pydata.org/pandas-docs/stable/api.html) for detailed usage information.

# Data Quality dimensions



Data quality can be decided based on above 4 factors viz. Completeness, validity, accuracy, consistency (in decreasing order of importance)

Data should be complete (example: missing values in calculated columns, i.e. if a-b = c then it is easy to derive c values as they can be calculated using a and b)

If data is complete: it should be valid. Example a person’s height is given as -6 ft. Negative height is not possible!

If data is complete, valid: it should be accurate. Example: Listing Seattle in state of Ohio.

If data is complete, valid, accurate it should be consistent. Example: format of dates should be same not only within the column but across columns. States names should be either abbreviations or fully spelt, etc.

**Data Quality Dimensions**

Data quality dimensions help guide your thought process while assessing and also cleaning. The four main data quality dimensions are:

* **Completeness**: do we have all of the records that we should? Do we have missing records or not? Are there specific rows, columns, or cells missing?
* **Validity**: we have the records, but they're not valid, i.e., they don't conform to a defined schema. A schema is a defined set of rules for data. These rules can be real-world constraints (e.g. negative height is impossible) and table-specific constraints (e.g. unique key constraints in tables).
* **Accuracy**: inaccurate data is wrong data that is valid. It adheres to the defined schema, but it is still incorrect. Example: a patient's weight that is 5 lbs too heavy because the scale was faulty.
* **Consistency**: inconsistent data is both valid and accurate, but there are multiple *correct* ways of referring to the same thing. Consistency, i.e., a standard format, in columns that represent the same data across tables and/or within tables is desired.

Regarding the other data quality research mentioned in the video, the additional dimensions are super specific cases of these four dimensions listed above. Example: *currency*, defined as follows: *the degree to which data is current with the world that it models. Currency can measure how up-to-date data is.* Currency is a specific case of accuracy data in the sense that out of date data is (usually) valid but wrong. In other words, accuracy is a relaxed definition of currency.

**Data quality issues noticed in given dataset**

#### Quality

##### *patients table*

* Zip code is a float not a string
* Zip code has four digits sometimes
* Tim Neudorf height is 27 in instead of 72 in
* Full state names sometimes, abbreviations other times
* Dsvid Gustafsson
* Missing demographic information (address - contact columns) **(can't clean)**
* Erroneous datatypes (assigned sex, state, zip\_code, and birthdate columns)
* Multiple phone number formats
* Default John Doe data
* Multiple records for Jakobsen, Gersten, Taylor
* kgs instead of lbs for Zaitseva weight

##### *treatments table*

* Missing HbA1c changes
* The letter 'u' in starting and ending doses for Auralin and Novodra
* Lowercase given names and surnames
* Missing records (280 instead of 350)
* Erroneous datatypes (auralin and novodra columns)
* Inaccurate HbA1c changes (leading 4s mistaken as 9s)
* Nulls represented as dashes (-) in auralin and novodra columns

##### *adverse\_reactions table*

* Lowercase given names and surnames

#### Tidiness

* Contact column in patients table should be split into phone number and email
* Three variables in two columns in treatments table (treatment, start dose and end dose)
* Adverse reaction should be part of the treatments table
* Given name and surname columns in patients table duplicated in treatments and adverse\_reactions tables

# Tidy Data

* [**Hadley Wickham: Tidying messy datasets**](https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html)

Issues with current dataset:

1. contact column has email as well as phone numbers
2. drug , dosage data should be 3 columns treatment, start dose and end dose

# Assess: Summary

Assessing is the second step in the data wrangling process:

* Gather
* **Assess**
* Clean

You can assess data for:

* Quality: issues with content. Low quality data is also known as dirty data.
* Tidiness: issues with structure that prevent easy analysis. Untidy data is also known as messy data. Tidy data requirements:
  1. Each variable forms a column.
  2. Each observation forms a row.
  3. Each type of observational unit forms a table.

...using two types of assessment:

* Visual assessment: scrolling through the data in your preferred software application (Google Sheets, Excel, a text editor, etc.).
* Programmatic assessment: using code to view specific portions and summaries of the data (pandas' head, tail, and info methods, for example).

**Clean: Summary**

Cleaning is the third step in the data wrangling process:

* Gather
* Assess
* **Clean**

There are two types of cleaning:

* Manual (not recommended unless the issues are one-off occurrences)
* Programmatic

The programmatic data cleaning process:

1. Define: convert our assessments into defined cleaning tasks. These definitions also serve as an instruction list so others (or yourself in the future) can look at your work and reproduce it.
2. Code: convert those definitions to code and run that code.
3. Test: test your dataset, visually or with code, to make sure your cleaning operations worked.

Always make copies of the original pieces of data before cleaning!

# Coding tips:

## Check if a single cell value in pandas dataframe is NaN

Use pd.isnull(df[‘col\_name])

import pandas as pd

import numpy as np

from pandas import \*

>>> L = [4, nan ,6]

>>> df = Series(L)

>>> df

0 4

1 NaN

2 6

>>> if(pd.isnull(df[1])):

print "Found"

Found

>>> if(np.isnan(df[1])):

print "Found"

Found

Value assignment in dataframe using index, iterrows

df.loc[index,'name'] = np.NaN