DataPrep - The easiest way to prepare data in Python

Jiannan Wang

Simon Fraser University

Apr 21, 2021, Thomson Reuters
From Model-Centric to Data-Centric

Andrew Ng 🎉@AndrewYNg · Apr 17

This Sunday is my birthday! The best gift🎁 to me would be if you can watch this video and let me know what you think. youtube.com/watch?v=06-AZX...

Lets you and I work to shift AI from Model-Centric toward Data-Centric AI development, which will help many teams.

A Chat with Andrew on MLOps: From Model-ccentric AI development to Data-Centric AI development...

“In 80 percent of our work is data preparation, then ensuring data quality is the important work of a machine learning team.”

Andrew Ng

Data Preparation Is Still the Bottleneck!!!

The New York Times

For Big-Data Scientists, ‘Janitor Work’ Is Key Hurdle to Insights

Yet far too much handcrafted work — what data scientists call “data wrangling,” “data munging” and “data janitor work” — is still required. Data scientists, according to interviews and expert estimates, spend from 50 percent to 80 percent of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.


The State of Data Science 2020

Moving from hype toward maturity

We were disappointed, if not surprised, to see that data wrangling still takes the lion’s share of time in a typical data professional’s day. Our respondents reported that almost half of their time is spent on the combined tasks of data loading and cleansing. Data

Why Is Data Preparation Hard?

Collection  Cleaning  Integration  Analysis

How much time is spent on preparation?

1. Too many small problems (e.g., standardize date, dedup address, etc)
2. Humans have different levels of expertise (in data science and programming)
3. Domain specific (finance, social science, healthcare, economics, etc.)
Human-in-the-loop Data Preparation

Three Directions

• Spreadsheet GUI
• Workflow GUI
• Notebook GUI
# Spreadsheet GUI

## Preview

<table>
<thead>
<tr>
<th>IMSI</th>
<th>CONTRACT_END</th>
<th>CONTRACT_START</th>
<th>SUBSCRIBER_AGE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3101790236812721</td>
<td>6/4/16</td>
<td>7/23/09</td>
<td>1</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310169986766700</td>
<td>2/28/16</td>
<td>10/6/13</td>
<td>7</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310179456822541</td>
<td>9/23/16</td>
<td>1/9/07</td>
<td>13</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310085432849230</td>
<td>5/29/15</td>
<td>2/14/01</td>
<td>4</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310926939271995</td>
<td>9/11/15</td>
<td>9/18/10</td>
<td>8</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3100260815466952</td>
<td>8/27/15</td>
<td>3/13/06</td>
<td>16</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>31017948724861</td>
<td>1/16/16</td>
<td>5/11/04</td>
<td>4</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>310179769648971</td>
<td>8/5-July-2011</td>
<td>9/11/06</td>
<td>2 Categories</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>3100268310245556</td>
<td>12/24/15</td>
<td>3/28/01</td>
<td>13</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>31015983429817</td>
<td>3/6/15</td>
<td>7/26/00</td>
<td>14</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310168464252516</td>
<td>9/25/15</td>
<td>4/4/04</td>
<td>10</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310128438750772</td>
<td>4/30/16</td>
<td>9/8/04</td>
<td>10</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3100269195729676</td>
<td>1/16/15</td>
<td>1/3/04</td>
<td>11</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310026261822880</td>
<td>8/13/13</td>
<td>11/23/08</td>
<td>4</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>310085667802048</td>
<td>8/4/16</td>
<td>10/22/14</td>
<td>16</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3101798369208164</td>
<td>1/22/15</td>
<td>10/19/14</td>
<td>8</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310168772677782</td>
<td>11/21/15</td>
<td>12/28/14</td>
<td>14</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310179116249240</td>
<td>27-Sep-2011</td>
<td>2/9/09</td>
<td>INACTIVE</td>
<td></td>
</tr>
<tr>
<td>310026110612337</td>
<td>5/29/15</td>
<td>3/29/05</td>
<td>9</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310126608767670</td>
<td>11/17/16</td>
<td>5/21/07</td>
<td>7</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>31004436658016</td>
<td>9/15/16</td>
<td>7/24/11</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>310128423699542</td>
<td>2/27/15</td>
<td>6/29/11</td>
<td>3</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310023713194729</td>
<td>4/28/16</td>
<td>6/15/04</td>
<td>10</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310092925859214</td>
<td>2/7/15</td>
<td>3/24/12</td>
<td>2</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>310012150088547</td>
<td>13-Jan-2009</td>
<td>12/10/05</td>
<td>3</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>310218378668694</td>
<td>10/1/16</td>
<td>10/25/11</td>
<td>3</td>
<td>ACTIVE</td>
</tr>
</tbody>
</table>

- **Hide Example Values**
  - m/dd/yy
  - 9/18/15
  - 6/13/15
  - 5/21/15
  - 12/12/15
  - 1/16/16

- **Month abbreviations**
  - Jan
  - Feb
  - Mar
  - Apr
  - May
  - Jun
  - Jul
  - Aug
  - Sep
  - Oct
  - Nov
  - Dec

- **Date formats**
  - dd-mm-yyyy
  - 14-Nov-2012
  - 11-Jul-2007
  - 20-Jul-2015

- **Data types**
  - 19 columns
  - 20,000 rows
  - 8 data types

- **�**
  - Show only affected
  - Rows

- **Trifacta**
  - 12.65k
  - 5.37k

- **SFU**
Workflow GUI
Notebook GUI

```python
In [1]:
import qgrid
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.rand(3,5))
qg = qgrid.show_grid(df)
qg
```

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 0  | 0.09835 | 0.65988 | 0.24213 | 0.1324 | 0.21811 |
| 1  | 0.11348 | 0.17392 | 0.18452 | 0.25759 | 0.45406 |
| 2  | 0.77328 | 0.2921  | 0.37934 | 0.87462 | 0.93311 |
Which Direction To Go?

“Data Prep Market was valued at USD 3.29 Billion in 2019 and is projected to reach USD 18.11 Billion by 2027, growing at a CAGR of 25.64% from 2020 to 2027.”


Three Directions

- Spreadsheet GUI
  - Targeted at non-programmers
- Workflow GUI
- Notebook GUI
  - Targeted at data scientists
Our Vision

Machine Learning Made Easy

Deep Learning Made Easy

Big Data Made Easy

Visualization Made Easy

Data Preparation Made Easy

SFU

scikit-learn

PyTorch

Koalas

matplotlib

data prep
DataPrep Components

May 2019 - Now  DataPrep.EDA  Simplify Exploratory Data Analysis

Nov 2019 - Now  DataPrep.Connector  Simplify Web Data Collection

Sept 2020 - Now  DataPrep.Clean  Simplify Data Cleaning

Planning  DataPrep.Feature  Simplify Feature Engineering

Planning  DataPrep.Integrate  Simplify Data Integration
Understand your data with a few lines of code in seconds using DataPrep.eda

samdo 4 points · 7 days ago
I'll look into it and get back to you. By the way what you guys are doing is amazing and have the potential to be a game-changer if it cut some time out of data prep.

apivan191 3 points · 5 months ago
This will save me so much time even just exploring my data, not to mention coding all of it up. You've done good in the world

cym13 9 months ago
Time will tell if this is the right solution, but at least I think you're tackling the right problem. Thank's for sharing.
Talk Outline

1. DataPrep Overview

2. Dive into DataPrep
   - DataPrep.EDA
   - DataPrep.Connector

3. Future Direction
Jinglin Peng*, Weiyuan Wu*, et al.
DataPrep.EDA: Task-Centric Exploratory Data Analysis for Statistical Modeling in Python. SIGMOD 2021
Exploratory Data Analysis (EDA)

Understand data and discover insights via data visualization, data summarization, etc.

Understand “Age” column

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0.42</td>
</tr>
<tr>
<td>5-th Percentile</td>
<td>4</td>
</tr>
<tr>
<td>Q1</td>
<td>20.125</td>
</tr>
<tr>
<td>Median</td>
<td>28</td>
</tr>
<tr>
<td>Q3</td>
<td>38</td>
</tr>
<tr>
<td>95-th Percentile</td>
<td>56</td>
</tr>
<tr>
<td>Maximum</td>
<td>80</td>
</tr>
<tr>
<td>Range</td>
<td>79.58</td>
</tr>
<tr>
<td>IQR</td>
<td>17.875</td>
</tr>
</tbody>
</table>
Solution 1: Pandas + Matplotlib

😊 Hard to Use

- Beginner: Need to know how to write plotting code
- Expert: Need to write lengthy and repetitive code

Understand “Age” column
Current EDA Solutions in Python

Solution 2: Pandas-profiling

😊 Slow
😊 Hard to Customize

```python
profile = ProfileReport(df, title="Pandas Profiling Report")
```
# DataPrep.EDA Design Goals

<table>
<thead>
<tr>
<th>EDA Solutions</th>
<th>Easy to Use</th>
<th>Interactive Speed</th>
<th>Easy to Customize</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Pandas + Matplotlib</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>2. Pandas-profiling</td>
<td>😊</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>3. DataPrep.EDA</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
</tr>
</tbody>
</table>
Key Idea

Task-Centric API Design

- Declarative
- Support both coarse-grained and fine-grained EDA tasks

Example

- `plot(df)`: “I want to see an overview of the dataset”
- `plot_missing(df)`: “I want to understand the missing values of the dataset”
- `plot(df, x)`: “I want to understand the column x”
- `plot(df, x, y)`: “I want to understand the relationship between x and y”
- ...
from dataprep.eda import plot, plot_missing, plot_correlation, create_report

import pandas as pd

df = pd.read_csv("titanic.csv")

I want an overview of the dataset

plot(df)

Understand Missing Value

plot_missing(df)

Understand Correlation

plot_correlation(df)

Understand Numerical Column

plot(df, "Age")

Understand Text Column

plot(df, "Name")

Understand Column Relationship
Under the Hood

plot(df, "price", bins = 50) → ?? → Mapping Rules

Data Processing Pipeline
# Mapping Rules

N = Numerical, C = Categorical

<table>
<thead>
<tr>
<th>Task-Centric API Design</th>
<th>Corresponding Stats/Plots</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot(df)</td>
<td>Dataset statistics, histogram or bar chart for each column</td>
</tr>
</tbody>
</table>
| plot(df, col₁)          | (1) col₁ = N → Column statistics, histogram, kde plot, qq-normal plot, box plot  
                        | (2) col₁ = C → Column statistics, bar chart, pie chart, word cloud, word frequencies |
| plot(df, col₁, col₂)    |                          |
| plot_correlation(df)    |                          |
| plot_correlation(df, col₁) |                       |
| plot_correlation(df, col₁, col₂) |              |
| plot_missing(df)        |                          |
| plot_missing(df, col₁)  |                          |
| plot_missing(df, col₁, col₂) |                        |

[1] [https://www.data-to-viz.com/](https://www.data-to-viz.com/)
[2] Exploratory data analysis with R
Data Processing Pipeline

1. Config Manager
   - Config
     - hist.bins: 50,
     - kde.bins: 50,
     - hist.width: 90,
     ...

2. Compute Module
   - Intermediates
     - missing_count: 0,
     - hist.counts: [2, 10,...],
     - hist.endpoints: [334, 421,...],
     ...

3. Render Module

```
plot(df, "price", bins = 50)
```
Interactive Speed

Ubuntu 16.04 Linux server with 64 GB memory and 8 Intel E7-4830 cores

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>#Rows</th>
<th>#Cols (N/C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>11KB</td>
<td>303</td>
<td>14 (14/0)</td>
</tr>
<tr>
<td>diabetes</td>
<td>23KB</td>
<td>768</td>
<td>9 (9/0)</td>
</tr>
<tr>
<td>automobile</td>
<td>26KB</td>
<td>205</td>
<td>26 (10/16)</td>
</tr>
<tr>
<td>titanic</td>
<td>64KB</td>
<td>891</td>
<td>12 (7/5)</td>
</tr>
<tr>
<td>women</td>
<td>500KB</td>
<td>8553</td>
<td>10 (5/5)</td>
</tr>
<tr>
<td>credit</td>
<td>2.7MB</td>
<td>30K</td>
<td>25 (25/0)</td>
</tr>
<tr>
<td>solar</td>
<td>2.8MB</td>
<td>33K</td>
<td>11 (7/4)</td>
</tr>
<tr>
<td>suicide</td>
<td>2.8MB</td>
<td>28K</td>
<td>12 (6/6)</td>
</tr>
<tr>
<td>diamonds</td>
<td>3MB</td>
<td>54K</td>
<td>11 (8/3)</td>
</tr>
<tr>
<td>chess</td>
<td>7.3MB</td>
<td>20K</td>
<td>16 (6/10)</td>
</tr>
<tr>
<td>adult</td>
<td>5.7MB</td>
<td>49K</td>
<td>15 (6/9)</td>
</tr>
<tr>
<td>basketball</td>
<td>9.2MB</td>
<td>53K</td>
<td>31 (21/10)</td>
</tr>
<tr>
<td>conflicts</td>
<td>13MB</td>
<td>34K</td>
<td>25 (10/15)</td>
</tr>
<tr>
<td>rain</td>
<td>13.5MB</td>
<td>142K</td>
<td>24 (17/7)</td>
</tr>
<tr>
<td>hotel</td>
<td>16MB</td>
<td>119K</td>
<td>32 (20/12)</td>
</tr>
</tbody>
</table>
## Efficiency Comparison

**DataPrep.EDA vs Pandas-Profilin**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>#Rows</th>
<th>#Cols (N/C)</th>
<th>PP</th>
<th>EDA&lt;sup&gt;x&lt;/sup&gt;</th>
<th>Faster</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>11KB</td>
<td>303</td>
<td>14 (14/0)</td>
<td>17.7s</td>
<td>2.0s</td>
<td>8.6×</td>
</tr>
<tr>
<td>diabetes</td>
<td>23KB</td>
<td>768</td>
<td>9 (9/0)</td>
<td>28.3s</td>
<td>1.6s</td>
<td>17.7×</td>
</tr>
<tr>
<td>automobile</td>
<td>26KB</td>
<td>205</td>
<td>26 (10/16)</td>
<td>38.2s</td>
<td>3.9s</td>
<td>9.8×</td>
</tr>
<tr>
<td>titanic</td>
<td>64KB</td>
<td>891</td>
<td>12 (7/5)</td>
<td>17.8s</td>
<td>2.1s</td>
<td>8.5×</td>
</tr>
<tr>
<td>women</td>
<td>500KB</td>
<td>8553</td>
<td>10 (5/5)</td>
<td>19.8s</td>
<td>2.3s</td>
<td>8.6×</td>
</tr>
<tr>
<td>credit</td>
<td>2.7MB</td>
<td>30K</td>
<td>25 (25/0)</td>
<td>127.0s</td>
<td>6.1s</td>
<td>20.8×</td>
</tr>
<tr>
<td>solar</td>
<td>2.8MB</td>
<td>33K</td>
<td>11 (7/4)</td>
<td>25.1s</td>
<td>2.7s</td>
<td>9.3×</td>
</tr>
<tr>
<td>suicide</td>
<td>2.8MB</td>
<td>28K</td>
<td>12 (6/6)</td>
<td>20.6s</td>
<td>2.8s</td>
<td>7.4×</td>
</tr>
<tr>
<td>diamonds</td>
<td>3MB</td>
<td>54K</td>
<td>11 (8/3)</td>
<td>28.2s</td>
<td>3.1s</td>
<td>9×</td>
</tr>
<tr>
<td>chess</td>
<td>7.3MB</td>
<td>20K</td>
<td>16 (6/10)</td>
<td>23.6s</td>
<td>4.3s</td>
<td>5.5×</td>
</tr>
<tr>
<td>adult</td>
<td>5.7MB</td>
<td>49K</td>
<td>15 (6/9)</td>
<td>23.2s</td>
<td>4.0s</td>
<td>5.8×</td>
</tr>
<tr>
<td>basketball</td>
<td>9.2MB</td>
<td>53K</td>
<td>31 (21/10)</td>
<td>126.2s</td>
<td>9.9s</td>
<td>12.7×</td>
</tr>
<tr>
<td>conflicts</td>
<td>13MB</td>
<td>34K</td>
<td>25 (10/15)</td>
<td>34.9s</td>
<td>8.6s</td>
<td>4×</td>
</tr>
<tr>
<td>rain</td>
<td>13.5MB</td>
<td>142K</td>
<td>24 (17/7)</td>
<td>100.1s</td>
<td>11.6s</td>
<td>8.6×</td>
</tr>
<tr>
<td>hotel</td>
<td>16MB</td>
<td>119K</td>
<td>32 (20/12)</td>
<td>83.2s</td>
<td>13s</td>
<td>6.4×</td>
</tr>
</tbody>
</table>
DataPrep.EDA Takeaways

Innovation
The first task-centric EDA system in Python

Achieve three design goals
Easy to use
Interactive speed
Easy to customize
DataPrep.Connector

A Unified API Wrapper to Simplify Web Data Collection
Data Collection Through Restful APIs

Social Data
- Twitter
- Yelp

Business Data

Event Data
- Eventbrite

Publication Data
- dblp

Index
- Animals
- Anime
- Anti-Malware
- Art & Design
- Books
- Business
- Calendar
- Cloud Storage & File Sharing
- Continuous Integration
- Cryptocurrency
- Currency Exchange
- Data Validation
- Development
- Dictionaries
- Documents & Productivity
- Environment
- Events
- Finance
## Restful API Example

### Request

```plaintext
GET https://api.yelp.com/v3/businesses/search
```

### Parameters

These parameters should be in the query string.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>term</td>
<td>string</td>
<td>Optional. Search term, for example &quot;food&quot; or &quot;restaurant&quot; business names.</td>
</tr>
<tr>
<td>location</td>
<td>string</td>
<td>Required if either latitude or longitude is not provided. Latitude of city.</td>
</tr>
<tr>
<td>latitude</td>
<td>decimal</td>
<td>Required if location is not provided. Latitude of city.</td>
</tr>
<tr>
<td>longitude</td>
<td>decimal</td>
<td>Required if location is not provided. Longitude of city.</td>
</tr>
<tr>
<td>radius</td>
<td>int</td>
<td>Optional. A suggested search radius in meters. The actual search radius may be less than 40000 meters (about 25 miles).</td>
</tr>
</tbody>
</table>

### Response Body

```json
{
  "total": 8228,
  "businesses": [
    {
      "rating": 4,
      "price": "$$",
      "phone": "+14152520800",
      "id": "EBR3k3dcmqtyoPMj0_0lg",
      "alias": "four-barrel-coffee-san-francisco",
      "is_closed": false,
      "categories": [
        {
          "alias": "coffee",
          "title": "Coffee & Tea"
        }
      ],
      "review_count": 1738,
      "name": "Four Barrel Coffee",
      "url": "https://www.yelp.com/biz/four-barrel-coffee-san-francisco",
      "coordinates": {
        "latitude": 37.670169511878,
        "longitude": -122.42184275
      },
      "image_url": "http://s3-media2.fl.yelpcdn.com/bphoto/MnUtASp3l_t4tPCL1iAsCg/
      "location": {
        "city": "San Francisco",
        "country": "US",
        "address1": "",
        "address2": "",
        "state": "CA",
        "address3": "375 Valencia St",
        "zip_code": "94103"
      },
      "distance": 1604.23,
      "transactions": ["pickup", "delivery"]
    },
    // ...
  ],
  "region": {
    "center": {
      "latitude": 37.767413217936834,
      "longitude": -122.42820739746094
    }
  }
}
```
Restful API Wrapper

Wrap API calls into Easy-to-Use Python Functions

- **bear / python-twitter**
  
  A Python wrapper around the Twitter API.
  
  - View license
  - 3k stars  922 forks

- **Yelp / yelp-fusion**
  
  Yelp Fusion API
  
  - yelp.com/developers
  - MIT License
  - 331 stars  305 forks

- **plamere / spotipy**
  
  A lightweight Python library for the Spotify Web API
  
  - MIT License
  - 2.6k stars  568 forks

- **srcscde / python-youtube-api**
  
  A basic Python YouTube v3 API to fetch data from YouTube Key without OAuth
  
  - GPL-3.0 License
  - 73 stars  35 forks

- **scholarly / dblp-python**
  
  A simple Python wrapper around search and author and publication
  
  - 61 stars  34 forks

- 

- 

-
Build a New API Wrapper is Tedious!

- **HTTP Connection**: Connect to the website server
- **Authorization**: Handle authorization schemes
- **Pagination**: Request data from multiple pages
- **Concurrency**: Retrieve data in parallel with less time
- **Result Parsing**: Convert Json string to Pandas Dataframe

...
If we don’t unify API wrappers, then ...
If we don’t unify API wrappers, then...

- **Bad for developers** (repetitive building efforts)
- **Bad for users** (burden to learn many API wrappers)
DataPrep.Connector
A Unified API Wrapper

Reusable Components

HTTP Connection
Authorization
Pagination
Concurrency
Result Parsing

......

Configuration Files

- Yelp Config File
- Spotify Config File
- Youtube Config File
- Twitter Config File
- DBLP Config File
- Facebook Config File
- Reddit Config File

......

Good for developers (No repetitive building efforts)
The Unified API

1. Connect

```python
conn = connect(website_name, _auth, _concurrency)
```

2. Understand (Optional)

```python
conn.table_names
conn.show_schema(table_name)
```

3. Query

```python
conn.query(table_name, query_parameter_list, _count)
```

Good for users (No burden to learn many API wrappers)
A Unified API Design

DBLP

```python
In [ ]: conn dblp = Connector("dblp")
In [ ]: conn dblp.table_names
In [ ]: conn dblp.show_schema("publication")
In [ ]: df = await conn dblp.query("publication", q = "machine learning")
In [ ]: df.head(5)
```  

Youtube

```python
In [ ]: conn youtube = Connector('youtube', _auth={"access_token":youtube_auth_token})
In [ ]: conn youtube.table_names
In [ ]: conn youtube.show_schema("videos")
In [ ]: df = await conn youtube.query("videos", q = "data science", part = "snippet", type = "videos")
```
DataPrep.Connector Takeaways

Innovation
The first unified API Wrapper in Python

Good For Developers
Speed up wrapper development process

Good For Users
Speed up data collection from Web APIs
Talk Outline

1. DataPrep Overview

2. Dive into DataPrep
   - DataPrep. EDA
   - DataPrep. Connector

3. Future Direction
Future Direction

DataPrep.EDA

• Make plots look attractive
• Understand multiple dataframes (plot_diff, plot_db, ...)

DataPrep.Connector  [http://cx.dataprep.ai](http://cx.dataprep.ai)

• Speed up `read_sql()` with arrow and parallel connection
Future Direction

DataPrep.Clean

• **Goal:** Implement 100+ `clean_{type}(df, x)` functions
• **Example:** `clean_email`, `clean_date`, `clean_phone`, `clean_country`, etc.
• **Application:** Data Validation, Data Standardization, Semantic Type Detection

```python
import pandas as pd
from dataprep.clean import clean_country

df = pd.DataFrame({'country': ['USA', 'country: Canada', 'France', '233', 'tr']})
clean_country(df, 'country')
```

<table>
<thead>
<tr>
<th>country</th>
<th>country_clean</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>United States</td>
</tr>
<tr>
<td>country: Canada</td>
<td>Canada</td>
</tr>
<tr>
<td>France</td>
<td>France</td>
</tr>
<tr>
<td>233</td>
<td>Estonia</td>
</tr>
<tr>
<td>tr</td>
<td>Turkey</td>
</tr>
</tbody>
</table>
The easiest way to prepare data in Python

pip install -U dataprep

Thank you!

http://dataprep.ai