Contents

1 Getting Started 3
1.1 Requirements ............................................. 3
1.2 Installation ................................................. 3
1.3 Example ..................................................... 4

2 FAQ 5
2.1 tabula-py does not work ................................. 5
2.2 I can’t run from tabula import read_pdf ................ 5
2.3 I got a empty DataFrame. How can I resolve it? ....... 5
2.4 The result is different from tabula-java. Or, stream option seems not to work appropriately .... 6
2.5 Can I use option xxx? ..................................... 6
2.6 How can I ignore useless area? ......................... 6
2.7 I faced ParserError: Error tokenizing data. C error. How can I extract multiple tables? .......... 8
2.8 I want to prevent tabula-py from stealing focus on every call on my mac ............................ 8
2.9 I got ? character with result on Windows. How can I avoid it? ........................................ 8
2.10 I can’t extract file/directory name with space on Windows .................................................. 8
2.11 I want to use a different tabula.jar file ................ 8
2.12 I want to extract multiple tables from a document .......................................................... 9
2.13 Table cell contents sometimes overflow into the next row .................................................. 9
2.14 I got a warning/error message from PDFBox including org.apache.pdfbox.pdmodel.. Is it the cause of empty dataframe? .............................................. 9
2.15 I can’t figure out accurate extraction with tabula-py. Are there any similar Python libraries? .... 9

3 Contributing to tabula-py 11
3.1 Code formatting and testing .............................. 11
3.2 Documentation .............................................. 11

4 tabula 13
4.1 High level interfaces ...................................... 13
4.2 Internal interfaces ........................................ 21

5 tabula.errors 23

6 Indices and tables 25

Python Module Index 27
tabula-py is a simple Python wrapper of tabula-java, which can read table of PDF. You can read tables from PDF and convert into pandas’s DataFrame. tabula-py also enables you to convert a PDF file into CSV/TSV/JSON file.

We highly recommend to look at the example notebook and try it on Google Colab.

For high level API reference, see High level interfaces.
1.1 Requirements

- Java
  - Java 8+
- Python
  - 3.5+

1.2 Installation

Before installing tabula-py, ensure you have Java runtime on your environment.
You can install tabula-py form PyPI with pip command.

```
pip install tabula-py
```

**Note:** conda recipe on conda-forge is not maintained by us. We recommend to install via pip to use latest version of tabula-py.

1.2.1 Get tabula-py working (Windows 10)

This instruction is originally written by @lahoffm. Thanks!

- If you don’t have it already, install Java
- Try to run example code (replace the appropriate PDF file name).
• If there’s a `FileNotFoundError` when it calls `read_pdf()`, and when you type `java` on command line it says 'java' is not recognized as an internal or external command, operable program or batch file, you should set `PATH` environment variable to point to the Java directory.

• Find the main Java folder like `jre...` or `jdk...`. On Windows 10 it was under `C:\Program Files\Java`


• Add the `bin` folder like `C:\Program Files\Java\jre1.8.0_144\bin`, hit OK a bunch of times.

• On command line, `java` should now print a list of options, and `tabula.read_pdf()` should run.

### 1.3 Example

`tabula-py` enables you to extract tables from a PDF into a DataFrame, or a JSON. It can also extract tables from a PDF and save the file as a CSV, a TSV, or a JSON.

```python
import tabula

# Read pdf into a list of DataFrame
dfs = tabula.read_pdf("test.pdf", pages='all')

# Read remote pdf into a list of DataFrame

# convert PDF into CSV
tabula.convert_into("test.pdf", "output.csv", output_format="csv", pages='all')

# convert all PDFs in a directory
tabula.convert_into_by_batch("input_directory", output_format='csv', pages='all')
```

See example notebook for more detail. I also recommend to read the tutorial article written by @aegis4048.

**Note:** If you face some issue, we’d recommend to try `tabula.app` to see the limitation of `tabula-java`. Also, see FAQ as well.
### 2.1 **tabula-py does not work**

There are several possible reasons, but *tabula-py* is just a wrapper of *tabula-java*, make sure you’ve installed Java and you can use *java* command on your terminal. Many issue reporters forget to set PATH for *java* command.

You can check whether *tabula-py* can call *java* from Python process with `tabula.environment_info()` function.

### 2.2 **I can’t run from tabula import read_pdf**

If you’ve installed *tabula*, it will be conflict the namespace. You should install *tabula-py* after removing *tabula*.

```
pip uninstall tabula
pip install tabula-py
```

### 2.3 **I got a empty DataFrame. How can I resolve it?**

Before tuning the *tabula-py* option, you have to check you set an appropriate *pages* option. By default, *tabula-py* extracts table from first page of your PDF, with *pages=1* argument. If you want to extract from all pages, you need to set *pages* option like *pages=“all”* or *pages=[1, 2, 3]*. You might want to extract multiple tables from multiple pages, if so you need to set *multiple_tables=True* together.

Depending on the PDF’s complexity, it might be difficult to extract table contents accuracy.

Tuning points of *tabula-py* are limited:

- Set specific *area* for accurate table detection
- Try *lattice=True* option for the table having explicit line. Or try *stream=True* option
To know the limitation of tabula-java, I highly recommend to use tabula app, the GUI version of tabula-java. tabula app can:

- specify the area with GUI
- show preview of the extraction with lattich or stream mode
- export template that is reusable for tabula-py

Even if you can’t extract tabula-py for those table contents which can be extracted tabula app appropriately, file an issue on GitHub.

### 2.4 The result is different from tabula-java. Or, stream option seems not to work appropriately

tabula-py set guess option True by default, for beginners. It is known to make a conflict between stream option. If you feel something strange with your result, please set guess=False.

### 2.5 Can I use option xxx?

Yes. You can use options argument as following. The format is same as cli of tabula-java.

```python
read_pdf(file_path, options="--columns 10.1,20.2,30.3")
```

### 2.6 How can I ignore useless area?

In short, you can extract with area and spreadsheet option.

```python
In [4]: tabula.read_pdf('./table.pdf', spreadsheet=True, area=(337.29, 226.49, 472.85, 384.91))
```

Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8

```
Out[4]:

<table>
<thead>
<tr>
<th></th>
<th>Unnamed</th>
<th>Col1</th>
<th>Col2</th>
<th>Col3</th>
<th>Col4</th>
<th>Col5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>B</td>
<td>12</td>
<td>R</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NaN</td>
<td>R</td>
<td>T</td>
<td>23</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>B</td>
<td>33</td>
<td>R</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>T</td>
<td>99</td>
<td>E</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>I</td>
<td>12</td>
<td>34</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>I</td>
<td>I</td>
<td>W</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NaN</td>
<td>1</td>
<td>2</td>
<td>W</td>
<td>h</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>NaN</td>
<td>4</td>
<td>3</td>
<td>E</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>E</td>
<td>E4</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

### 2.6.1 How to use area option

According to tabula-java wiki, there is a explain how to specify the area: https://github.com/tabulapdf/tabula-java/wiki/Using-the-command-line-tabula-extractor-tool#grab-coordinates-of-the-table-you-want

For example, using macOS’s preview, I got area information of this PDF:
2.6. How can I ignore useless area?

Note the left, top, height, and width parameters and calculate the following:

\[
y_1 = \text{top} \\
x_1 = \text{left}
\]
y2 = top + height
x2 = left + width

I confirmed with tabula-java:
```
java -jar ./tabula/tabula-1.0.1-jar-with-dependencies.jar -a "337.29,226.49,472.85,384.91" table.pdf
```

Without `-r`(same as `--spreadsheet`) option, it does not work properly.

### 2.7 I faced ParserError: Error tokenizing data. C error. How can I extract multiple tables?

This error occurs pandas trys to extract multiple tables with different column size at once. Use `multiple_tables` option, then you can avoid this error.

### 2.8 I want to prevent tabula-py from stealing focus on every call on my mac

Set `java_options=["-Djava.awt.headless=true"]`. [kudos @jakekara](#)

### 2.9 I got ? character with result on Windows. How can I avoid it?

If the encoding of PDF is UTF-8, you should set `chcp 65001` on your terminal before launching a Python process.
```
chcp 65001
```

Then you can extract UTF-8 PDF with `java_options="-Dfile.encoding=UTF8"` option. This option will be added with `encoding='utf-8'` option, which is also set by default.
```
# This is an example for java_options is set explicitly
df = read_pdf(file_path, java_options="-Dfile.encoding=UTF8")
```

Replace 65001 and UTF-8 appropriately, if the file encoding isn’t UTF-8.

### 2.10 I can’t extract file/directory name with space on Windows

You should escape file/directory name yourself.

### 2.11 I want to use a different tabula .jar file

You can specify the jar location via enviroment variable
```
export TABULA_JAR=".../tabula-x.y.z-jar-with-dependencies.jar"
```
2.12 I want to extract multiple tables from a document

You can use the following example code

```python
df = read_pdf(file_path, multiple_tables=True)
```

The result will be a list of DataFrames. If you want separate tables across all pages in a document, use the `pages` argument.

2.13 Table cell contents sometimes overflow into the next row.

You can try using `lattice=True`, which will often work if there are lines separating cells in the table.

2.14 I got a warning/error message from PDFBox including `org.apache.pdfbox.pdmodel.. Is it the cause of empty dataframe?`

No.

Sometimes, you might see message like ‘‘Jul 17, 2019 10:21:25 AM org.apache.pdfbox.pdmodel.font.PDType1Font WARNING: Using fallback font NimbusSanL-Regu for Univers. Nothing was parsed from this one.’’ This error message came from Apache PDFBox which is used under tabula-java, and this is caused by the PDF itself. Neither tabula-py nor tabula-java can’t handle the warning itself, except for silent option that suppress the warning.

2.15 I can’t figure out accurate extraction with tabula-py. Are there any similar Python libraries?

I know tabula-py has limitation depending on tabula-java. Sometimes your PDF is too complex to tabula-py. If you want to find plan B, there are similar packages as the following:

- https://github.com/jsvine/pdfplumber
CHAPTER 3

Contributing to tabula-py

Interested in helping out? I’d love to have your help!

You can help by:

• Reporting a bug.
• Adding or editing documentation.
• Contributing code via a Pull Request.
• Write a blog post or spreading the word about tabula-py to people who might be able to benefit from using it.

3.1 Code formatting and testing

If you want to become a contributor, you can install dependency after cloning the repo as follows:

```bash
pip install -e .[dev, test]
pip install nox
```

For running text and liter, run nox command.

```bash
nox .
```

3.2 Documentation

You can build document on your environment as follows:

```bash
pip install -e .[doc]
cd docs && make html
```

The documentation source is under docs/ directory and the document is published on Read the Docs automatically.
CHAPTER 4

---

4.1 High level interfaces

4.1.1 tabula.io

This module is a wrapper of tabula, which enables table extraction from a PDF.

This module extracts tables from a PDF into a pandas DataFrame. Currently, the implementation of this module uses subprocess.

Instead of importing this module, you can import public interfaces such as `read_pdf()`, `read_pdf_with_template()`, `convert_into()`, `convert_into_by_batch()` from `tabula` module directory.

**Note:** If you want to use your own tabula-java JAR file, set `TABULA_JAR` to environment variable for JAR path.

**Example**

```python
>>> import tabula
>>> df = tabula.read_pdf("/path/to/sample.pdf", pages="all")
```

`tabula.io.build_options`  
`pages=None, guess=True, area=None, relative_area=False, lattice=False, stream=False, password=None, silent=None, columns=None, format=None, batch=None, output_path=None, options=""")

Build options for tabula-java

**Parameters**

- **pages** *(str, int, list of int, optional) – An optional values specifying pages to extract from. It allows str,’int’, list :int. Default: 1)*
Examples

'1-2,3', 'all', [1,2]

• **guess (bool, optional)** – Guess the portion of the page to analyze per page. Default True If you use “area” option, this option becomes False.

**Note:** As of tabula-java 1.0.3, guess option becomes independent from lattice and stream option, you can use guess and lattice/stream option at the same time.

• **area (list of float, list of list of float, optional)** – Portion of the page to analyze(top,left,bottom,right). Default is entire page.

**Note:** If you want to use multiple area options and extract in one table, it should be better to set multiple_tables=False for read_pdf()

Examples

[269.875, 12.75, 790.5, 561]. [[12.1, 20.5, 30.1, 50.2], [1.0, 3.2, 10.5, 40.2]]

• **relative_area (bool, optional)** – If all area values are between 0-100 (inclusive) and preceded by '%', input will be taken as % of actual height or width of the page. Default False.

• **lattice (bool, optional)** – Force PDF to be extracted using lattice-mode extraction (if there are ruling lines separating each cell, as in a PDF of an Excel spreadsheet)

• **stream (bool, optional)** – Force PDF to be extracted using stream-mode extraction (if there are no ruling lines separating each cell, as in a PDF of an Excel spreadsheet)

• **password (str, optional)** – Password to decrypt document. Default: empty

• **silent (bool, optional)** – Suppress all stderr output.

• **columns (list, optional)** – X coordinates of column boundaries.

**Example**

[10.1, 20.2, 30.3]

• **format (str, optional)** – Format for output file or extracted object. ("CSV", "TSV", "JSON")

• **batch (str, optional)** – Convert all PDF files in the provided directory. This argument should be directory path.

• **output_path (str, optional)** – Output file path. File format of it is depends on format. Same as --outfile option of tabula-java.

• **options (str, optional)** – Raw option string for tabula-java.

**Returns** Built list of options

**Return type** list
tabula.io.convert_into(input_path, output_path, output_format='csv', java_options=None, **kwargs)

Convert tables from PDF into a file. Output file will be saved into output_path.

Parameters

- **input_path** (file like object) – File like object of target PDF file.
- **output_path** (str) – File path of output file.
- **output_format** (str, optional) – Output format of this function (csv, json or tsv). Default: csv
- **java_options** (list, optional) – Set java options

Example

"-Xmx256m".

- **kwargs** – Dictionary of option for tabula-java. Details are shown in build_options()

Raises

- FileNotFoundError – If downloaded remote file doesn’t exist.
- ValueError – If output_format is unknown format, or if downloaded remote file size is 0.
- tabula.errors.JavaNotFoundError – If java is not installed or found.

tabula.io.convert_into_by_batch(input_dir, output_format='csv', java_options=None, **kwargs)

Convert tables from PDFs in a directory.

Parameters

- **input_dir** (str) – Directory path.
- **output_format** (str, optional) – Output format for returned object (dataframe or json)
- **java_options** (list, optional) – Set java options like -Xmx256m.
- **kwargs** – Dictionary of option for tabula-java. Details are shown in build_options()

Returns

Nothing. Outputs are saved into the same directory with input_dir.

Raises

- ValueError – If input_dir doesn’t exist.
- tabula.errors.JavaNotFoundError – If java is not installed or found.

tabula.io.read_pdf(input_path, output_format=None, encoding='utf-8', java_options=None, pandas_options=None, multiple_tables=True, user_agent=None, **kwargs)

Read tables in PDF.

Parameters

- **input_path** (str, path object or file-like object) – File like object of target PDF file. It can be URL, which is downloaded by tabula-py automatically.
- **output_format** (str, optional) – Output format for returned object (dataframe or json)
• encoding (str, optional) – Encoding type for pandas. Default: utf-8
• java_options (list, optional) – Set java options.

Example

["-Xmx256m"]

• pandas_options (dict, optional) – Set pandas options.

Example

{'header': None}

Note: With multiple_tables=True (default), pandas_options is passed to pandas.DataFrame, otherwise it is passed to pandas.read_csv. Those two functions are different for accept options like dtype.

• multiple_tables (bool) – It enables to handle multiple tables within a page. Default: True

Note: If multiple_tables option is enabled, tabula-py uses not pd.read_csv(), but pd.DataFrame(). Make sure to pass appropriate pandas_options.

• user_agent (str, optional) – Set a custom user-agent when download a pdf from a url. Otherwise it uses the default urllib.request user-agent.

• kwargs – Dictionary of option for tabula-java. Details are shown in build_options().

Returns list of DataFrames or dict.

Raises

• FileNotFoundError – If downloaded remote file doesn’t exist.
• ValueError – If output_format is unknown format, or if downloaded remote file size is 0.
• tabula.errors.CSVParseError – If pandas CSV parsing failed.
• tabula.errors.JavaNotFoundError – If java is not installed or found.
• subprocess.CalledProcessError – If tabula-java execution failed.

Examples

Here is a simple example. Note that read_pdf() only extract page 1 by default.

Notes: As of tabula-py 2.0.0, read_pdf() sets multiple_tables=True by default. If you want to get consistent output with previous version, set multiple_tables=False.

```python
>>> import tabula
>>> tabula.read_pdf(pdf_path, stream=True)
```

(continues on next page)
<table>
<thead>
<tr>
<th>carb</th>
<th>Unnamed: 0</th>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mazda RX4</td>
<td>21.0</td>
<td>6</td>
<td>160.0</td>
<td>110</td>
<td>3.90</td>
<td>2.620</td>
<td>16.46</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Mazda RX4 Wag</td>
<td>21.0</td>
<td>6</td>
<td>160.0</td>
<td>110</td>
<td>3.90</td>
<td>2.875</td>
<td>17.02</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Datsun 710</td>
<td>22.8</td>
<td>4</td>
<td>108.0</td>
<td>93</td>
<td>3.85</td>
<td>2.320</td>
<td>18.61</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Hornet 4 Drive</td>
<td>21.4</td>
<td>6</td>
<td>258.0</td>
<td>110</td>
<td>3.08</td>
<td>3.215</td>
<td>19.44</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>Hornet Sportabout</td>
<td>18.7</td>
<td>8</td>
<td>360.0</td>
<td>175</td>
<td>3.15</td>
<td>3.440</td>
<td>17.02</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Valiant</td>
<td>18.1</td>
<td>6</td>
<td>225.0</td>
<td>105</td>
<td>2.76</td>
<td>3.460</td>
<td>20.22</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>Duster 360</td>
<td>14.3</td>
<td>8</td>
<td>360.0</td>
<td>245</td>
<td>3.21</td>
<td>3.570</td>
<td>15.84</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Merc 240D</td>
<td>24.4</td>
<td>4</td>
<td>146.7</td>
<td>62</td>
<td>3.69</td>
<td>3.190</td>
<td>20.00</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Merc 230</td>
<td>22.8</td>
<td>4</td>
<td>140.8</td>
<td>95</td>
<td>3.92</td>
<td>3.150</td>
<td>22.90</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Merc 280</td>
<td>19.2</td>
<td>6</td>
<td>167.6</td>
<td>123</td>
<td>3.92</td>
<td>3.440</td>
<td>18.30</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Merc 280C</td>
<td>17.8</td>
<td>6</td>
<td>167.6</td>
<td>123</td>
<td>3.92</td>
<td>3.440</td>
<td>18.90</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Merc 450SE</td>
<td>16.4</td>
<td>8</td>
<td>275.8</td>
<td>180</td>
<td>3.07</td>
<td>4.070</td>
<td>17.40</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Merc 450SL</td>
<td>17.3</td>
<td>8</td>
<td>275.8</td>
<td>180</td>
<td>3.07</td>
<td>3.730</td>
<td>17.60</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Merc 450SLC</td>
<td>15.2</td>
<td>8</td>
<td>275.8</td>
<td>180</td>
<td>3.07</td>
<td>3.780</td>
<td>18.00</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Cadillac Fleetwood</td>
<td>10.4</td>
<td>8</td>
<td>472.0</td>
<td>205</td>
<td>2.93</td>
<td>5.250</td>
<td>17.98</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Lincoln Continental</td>
<td>10.4</td>
<td>8</td>
<td>460.0</td>
<td>215</td>
<td>3.00</td>
<td>5.424</td>
<td>17.82</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Chrysler Imperial</td>
<td>14.7</td>
<td>8</td>
<td>440.0</td>
<td>230</td>
<td>3.23</td>
<td>5.345</td>
<td>17.42</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Fiat 128</td>
<td>32.4</td>
<td>4</td>
<td>78.7</td>
<td>66</td>
<td>4.08</td>
<td>2.200</td>
<td>19.47</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Honda Civic</td>
<td>30.4</td>
<td>4</td>
<td>75.7</td>
<td>52</td>
<td>4.93</td>
<td>1.615</td>
<td>18.52</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Toyota Corolla</td>
<td>33.9</td>
<td>4</td>
<td>71.1</td>
<td>65</td>
<td>4.22</td>
<td>1.835</td>
<td>19.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Toyota Corona</td>
<td>21.5</td>
<td>4</td>
<td>120.1</td>
<td>97</td>
<td>3.70</td>
<td>2.465</td>
<td>20.01</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>Dodge Challenger</td>
<td>15.5</td>
<td>8</td>
<td>318.0</td>
<td>150</td>
<td>2.76</td>
<td>3.520</td>
<td>16.87</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>AMC Javelin</td>
<td>15.2</td>
<td>8</td>
<td>304.0</td>
<td>150</td>
<td>3.15</td>
<td>3.435</td>
<td>17.30</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Camaro 228</td>
<td>13.3</td>
<td>8</td>
<td>350.0</td>
<td>245</td>
<td>3.73</td>
<td>3.840</td>
<td>15.41</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Pontiac Firebird</td>
<td>19.2</td>
<td>8</td>
<td>400.0</td>
<td>175</td>
<td>3.08</td>
<td>3.845</td>
<td>17.05</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Fiat X1-9</td>
<td>27.3</td>
<td>4</td>
<td>79.0</td>
<td>66</td>
<td>4.08</td>
<td>1.935</td>
<td>18.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Porsche 914-2</td>
<td>26.0</td>
<td>4</td>
<td>120.3</td>
<td>91</td>
<td>4.43</td>
<td>2.140</td>
<td>16.70</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Lotus Europa</td>
<td>30.4</td>
<td>4</td>
<td>95.1</td>
<td>113</td>
<td>3.77</td>
<td>1.513</td>
<td>16.90</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

(continues on next page)
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sepal.Length</td>
<td>Sepal.Width</td>
<td>Petal.Length</td>
<td>Petal.Width</td>
<td>Species</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2 setosa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2 setosa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2 setosa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2 setosa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2 setosa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4 setosa,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Read tables in PDF with a Tabula App template.

**Parameters**

- `input_path(str, path object or file-like object)` - File like object of target PDF file. It can be URL, which is downloaded by tabula-py automatically.

- `template_path(str, path object or file-like object)` - File like object for Tabula app template. It can be URL, which is downloaded by tabula-py automatically.

- `pandas_options(dict, optional)` - Set pandas options like `{'header': None}`.

- `encoding(str, optional)` - Encoding type for pandas. Default is `utf-8`.

- `java_options(list, optional)` - Set java options like `['-Xmx256m']`.

- `user_agent(str, optional)` - Set a custom user-agent when download a pdf from a url. Otherwise it uses the default `urllib.request` user-agent.

- `**kwargs` - Dictionary of option for `tabula-java`. Details are shown in `build_options()`.

**Returns** list of DataFrame.

**Raises**

- `FileNotFoundError` - If downloaded remote file doesn’t exist.

- `ValueError` - If output_format is unknown format, or if downloaded remote file size is 0.

- `tabula.errors.CSVParseError` - If pandas CSV parsing failed.

- `tabula.errors.JavaNotFoundError` - If java is not installed or found.

Examples

You can use template file extracted by tabula app.

```python
>>> import tabula
>>> tabula.read_pdf_with_template(pdf_path, "path/to/data.tabula-template.json")
```

<table>
<thead>
<tr>
<th>Unnamed</th>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>...</th>
<th>gsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>21.0</td>
<td>6</td>
<td>160.0</td>
<td>110</td>
<td>...</td>
<td>16.46</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>21.0</td>
<td>6</td>
<td>160.0</td>
<td>110</td>
<td>...</td>
<td>17.02</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>22.8</td>
<td>4</td>
<td>108.0</td>
<td>93</td>
<td>...</td>
<td>18.61</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>21.4</td>
<td>6</td>
<td>258.0</td>
<td>110</td>
<td>...</td>
<td>19.44</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>18.7</td>
<td>8</td>
<td>360.0</td>
<td>175</td>
<td>...</td>
<td>17.02</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>18.1</td>
<td>6</td>
<td>225.0</td>
<td>105</td>
<td>...</td>
<td>20.22</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>14.3</td>
<td>8</td>
<td>360.0</td>
<td>245</td>
<td>...</td>
<td>15.84</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>24.4</td>
<td>4</td>
<td>146.7</td>
<td>62</td>
<td>...</td>
<td>20.00</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>22.8</td>
<td>4</td>
<td>140.8</td>
<td>95</td>
<td>...</td>
<td>22.90</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>19.2</td>
<td>6</td>
<td>167.6</td>
<td>123</td>
<td>...</td>
<td>18.30</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>17.8</td>
<td>6</td>
<td>167.6</td>
<td>123</td>
<td>...</td>
<td>18.90</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>16.4</td>
<td>8</td>
<td>275.8</td>
<td>180</td>
<td>...</td>
<td>17.40</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>17.3</td>
<td>8</td>
<td>275.8</td>
<td>180</td>
<td>...</td>
<td>17.60</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>15.2</td>
<td>8</td>
<td>275.8</td>
<td>180</td>
<td>...</td>
<td>18.00</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>10.4</td>
<td>8</td>
<td>472.0</td>
<td>205</td>
<td>...</td>
<td>17.98</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>10.4</td>
<td>8</td>
<td>460.0</td>
<td>215</td>
<td>...</td>
<td>17.82</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>14.7</td>
<td>8</td>
<td>440.0</td>
<td>230</td>
<td>...</td>
<td>17.42</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>12.8</td>
<td>4</td>
<td>78.7</td>
<td>66</td>
<td>...</td>
<td>19.47</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>30.4</td>
<td>4</td>
<td>75.7</td>
<td>52</td>
<td>...</td>
<td>18.52</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>33.9</td>
<td>4</td>
<td>71.1</td>
<td>65</td>
<td>...</td>
<td>19.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>21.5</td>
<td>4</td>
<td>120.1</td>
<td>97</td>
<td>...</td>
<td>20.01</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>15.5</td>
<td>8</td>
<td>318.0</td>
<td>150</td>
<td>...</td>
<td>16.87</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>15.2</td>
<td>8</td>
<td>304.0</td>
<td>150</td>
<td>...</td>
<td>17.30</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>13.3</td>
<td>8</td>
<td>350.0</td>
<td>245</td>
<td>...</td>
<td>15.41</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>24</td>
<td>19.2</td>
<td>8</td>
<td>400.0</td>
<td>175</td>
<td>...</td>
<td>17.05</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>27.3</td>
<td>4</td>
<td>79.0</td>
<td>66</td>
<td>...</td>
<td>18.90</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>26.0</td>
<td>4</td>
<td>120.3</td>
<td>91</td>
<td>...</td>
<td>16.70</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>30.4</td>
<td>4</td>
<td>95.1</td>
<td>113</td>
<td>...</td>
<td>16.90</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>28</td>
<td>15.8</td>
<td>8</td>
<td>351.0</td>
<td>264</td>
<td>...</td>
<td>14.50</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>29</td>
<td>19.7</td>
<td>6</td>
<td>145.0</td>
<td>175</td>
<td>...</td>
<td>15.50</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>15.0</td>
<td>8</td>
<td>301.0</td>
<td>335</td>
<td>...</td>
<td>14.60</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>31</td>
<td>21.4</td>
<td>4</td>
<td>121.0</td>
<td>109</td>
<td>...</td>
<td>18.60</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
```

[32 rows x 12 columns],

<table>
<thead>
<tr>
<th>NaN</th>
<th>Sepal.Width</th>
<th>Petal.Length</th>
<th>Petal.Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NaN</th>
<th>Sepal.Length</th>
<th>Sepal.Width</th>
<th>Petal.Length</th>
<th>Petal.Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>145</td>
<td>6.7</td>
<td>3.3</td>
<td>5.7</td>
<td>2.5</td>
</tr>
<tr>
<td>2</td>
<td>146</td>
<td>6.7</td>
<td>3.0</td>
<td>5.2</td>
<td>2.3</td>
</tr>
<tr>
<td>3</td>
<td>147</td>
<td>6.3</td>
<td>2.5</td>
<td>5.0</td>
<td>1.9</td>
</tr>
<tr>
<td>4</td>
<td>148</td>
<td>6.5</td>
<td>3.0</td>
<td>5.2</td>
<td>2.0</td>
</tr>
<tr>
<td>5</td>
<td>149</td>
<td>6.2</td>
<td>3.4</td>
<td>5.4</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Unnamed: 0 supp dose

0 | 4.2 | VC | 0.5 |
1 | 11.5| VC | 0.5 |
2 | 7.3 | VC | 0.5 |
4.1.2 tabula.util

Utility module providing some convenient functions.

**tabula.util.environment_info()**
Show environment information for reporting.

Returns Detailed information like Python version, Java version, or OS environment, etc.
Return type str

**tabula.util.java_version()**
Show Java version

Returns Result of java -version
Return type str

4.2 Internal interfaces

4.2.1 tabula.template

**tabula.template.load_template(path_or_buffer)**
Build tabula-py option from template file

Parameters path_or_buffer (str, path object or file-like object) – File
like object of Tabula app template.

Returns tabula-py options
Return type dict

4.2.2 tabula.file_util

**tabula.file_util.is_file_like(obj)**
Check file like object

Parameters obj – file like object.

Returns file like object or not
Return type bool
tabula.file_util.localize_file (path_or_buffer, user_agent=None, suffix='.pdf')

Ensure localize target file.

If the target file is remote, this function fetches into local storage.

Parameters

- **path_or_buffer** (str) – File path or file like object or URL of target file.
- **user_agent** (str, optional) – Set a custom user-agent when download a pdf from a url. Otherwise it uses the default urllib.request user-agent.
- **suffix** (str, optional) – File extension to check.

Returns tuple of str and bool, which represents file name in local storage and temporary file flag.

Return type (str, bool)
exception tabula.errors.CSVParseError(message, cause)
    Bases: pandas.errors.ParserError

    Error represents CSV parse error, which mainly caused by pandas.

exception tabula.errors.JavaNotFoundError
    Bases: Exception

    Error represents Java doesn’t exist.
CHAPTER 6

Indices and tables

- genindex
- modindex
- search
Python Module Index

t
  tabula.errors, 23
  tabula.file_util, 21
  tabula.io, 13
  tabula.template, 21
  tabula.util, 21
Index

B
build_options() (in module tabula.io), 13

C
convert_into() (in module tabula.io), 14
convert_into_by_batch() (in module tabula.io), 15
CSVParseError, 23

E
environment_info() (in module tabula.util), 21

I
is_file_like() (in module tabula.file_util), 21

J
java_version() (in module tabula.util), 21
JavaNotFoundError, 23

L
load_template() (in module tabula.template), 21
localize_file() (in module tabula.file_util), 21

R
read_pdf() (in module tabula.io), 15
read_pdf_with_template() (in module tabula.io), 19

T
tabula.errors (module), 23
tabula.file_util (module), 21
tabula.io (module), 13
tabula.template (module), 21
tabula.util (module), 21