
tabula-py

Aki Ariga

Nov 20, 2023

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 <code>tabula import read_pdf</code>	5
2.3	I got an empty DataFrame. How can I resolve it?	5
2.4	The result is different from <code>tabula-java</code> . Or, <code>stream</code> option seems not to work appropriately	6
2.5	Can I use option <code>xxx</code> ?	6
2.6	How can I ignore useless area?	6
2.7	I faced <code>ParserError: Error tokenizing data. C error</code> . 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 results on Windows. How can I avoid it?	8
2.10	I can't extract file/directory names with space on Windows	8
2.11	I want to use a different <code>tabula.jar</code> file	9
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 <code>PDFBox</code> including <code>org.apache.pdfbox.pdmodel</code> .. Is it the cause of the empty dataframe?	9
2.15	<code>java_options</code> is ignored once <code>read_pdf</code> or similar function is called.	9
2.16	I can't figure out accurate extraction with tabula-py. Are there any similar Python libraries?	10
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	27
5	tabula.errors	29
6	Indices and tables	31
	Python Module Index	33
	Index	35

`tabula-py` is a simple Python wrapper of [tabula-java](#), which can read table of PDF. You can read tables from PDF and convert them into pandas' DataFrame. `tabula-py` also converts a PDF file into CSV/TSV/JSON file.

We highly recommend looking at [the example notebook](#) and trying it on [Google Colab](#).

For high-level API reference, see [High level interfaces](#).

GETTING STARTED

1.1 Requirements

- Java
 - Java 8+
- Python
 - 3.8+

1.2 Installation

Before installing `tabula-py`, ensure you have Java runtime on your environment.

You can install `tabula-py` from PyPI with `pip` command.

```
pip install tabula-py
```

If you want to leverage faster execution with `jpytype`, install with `jpytype` extra.

```
pip install tabula-py[jpytype]
```

Note: conda recipe on conda-forge is not maintained by us. We recommend installing via `pip` to use the 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 an 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`
- On Windows 10: **Control Panel** -> **System and Security** -> **System** -> **Advanced System Settings** -> **Environment Variables** -> Select **PATH** -> **Edit**

- 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.

```
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
dfs2 = tabula.read_pdf("https://github.com/tabulapdf/tabula-java/raw/master/src/test/
↳resources/technology/tabula/arabic.pdf")

# 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 reading the [tutorial article](#) written by @aegis4048 and another [tutorial](#) written by @tdpetrou.

Note: If you face some issues, we'd recommend trying [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 the 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 conflict with the namespace. You should install `tabula-py` after removing `tabula`.

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

2.3 I got an empty DataFrame. How can I resolve it?

`tabula-py` and `tabula-java` don't support image-based PDFs. It should contain text-based table information.

Before tuning the `tabula-py` option, you have to check you set an appropriate `pages` option. By default, `tabula-py` extracts tables from the 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 accurately.

Tuning points of `tabula-py` are limited:

- Set specific area for accurate table detection
- Try `lattice=True` option for the table having explicit lines. Or try `stream=True` option

To know the limitation of `tabula-java`, I highly recommend using `tabula app`, the GUI version of `tabula-java`.

`tabula app` can:

- specify the area with GUI
- show a preview of the extraction with `lattice` 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 follows. The format is the same as CLI of tabula-java.

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

2.6 How can I ignore useless area?

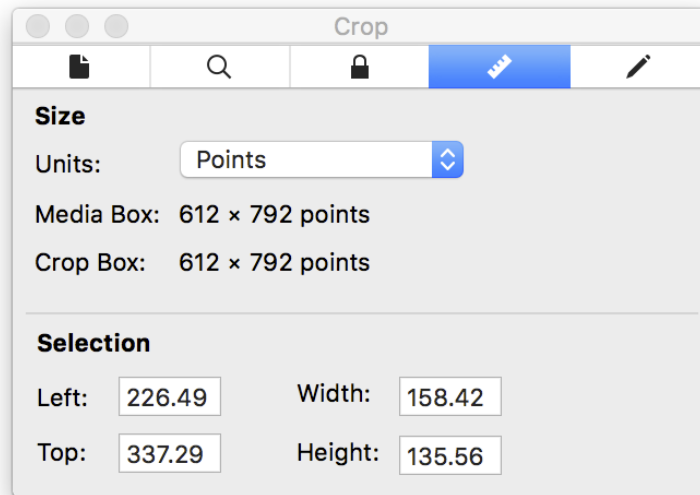
In short, you can extract with area and spreadsheet options.

```
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]:
  Unnamed: 0  Col2  Col3  Col4  Col5
0          A    B    12    R    G
1         NaN    R    T    23    H
2          B    B    33    R    A
3          C    T    99    E    M
4          D    I    12    34    M
5          E    I    I    W    90
6         NaN    1    2    W    h
7         NaN    4    3    E    H
8          F    E    E4    R    4
```

2.6.1 How to use area option

According to tabula-java wiki, there is an explanation of 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:



This is the header of the table				
Col1	Col2	Col3	Col4	Col5
A	B	12	R	G
	R	T	23	H
B	B	33	R	A
C	T	99	E	M
D	I	12	34	M
E	I	I	W	90
	1	2	W	h
	4	3	E	H
F	E	E4	R	4
G	3	D	R	4

```
java -jar ./target/tabula-1.0.1-jar-with-dependencies.jar -p all -a $y1,$x1,$y2,$x2 -o
↳ $csvfile $filename
```

given

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

```
y1 = top
x1 = left
```

(continues on next page)

(continued from previous page)

```
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 when pandas tries 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 results 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 names with space on Windows

You should escape the file/directory name yourself.

2.11 I want to use a different tabula .jar file

You can specify the jar location via environment 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

```
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 the empty dataframe?

No.

Sometimes, you might see a 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 the silent option that suppresses the warning.

2.15 `java_options` is ignored once `read_pdf` or similar function is called.

Since jpyype doesn't support changing JVM options after the JVM is started, `java_options` is ignored once `read_pdf` or similar function is called. If you want to change JVM options, you need to restart the Python process. See also: <https://jpyype.readthedocs.io/en/latest/api.html#jpyype.shutdownJVM>

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

I know tabula-py has limitations 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>
- <https://camelot-py.readthedocs.io/en/master/>

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 spread 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:

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

For running tests and linter, run nox command.

```
nox .
```

3.2 Documentation

You can build document on your environment as follows:

```
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.

TABULA

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 via jpype.

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

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

```
tabula.io.convert_into(input_path: IO | str | PathLike, output_path: str, output_format: str = 'csv',
                        java_options: List[str] | None = None, pages: str | int | Iterable[int] | None = None,
                        guess: bool = True, area: Iterable[float] | Iterable[Iterable[float]] | None = None,
                        relative_area: bool = False, lattice: bool = False, stream: bool = False, password: str
                        | None = None, silent: bool | None = None, columns: Iterable[float] | None = None,
                        relative_columns: bool = False, format: str | None = None, batch: str | None = None,
                        force_subprocess: bool = False, options: str = "") → None
```

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

Parameters

- **input_path** (*file like obj*) – 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. This option will be ignored once JVM is launched.

Example

"-Xmx256m".

- **pages** (*str*, *int*, *iterable of int*, *optional*) – An optional values specifying pages to extract from. It allows *str*, *int*, *iterable of 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**(*iterable of float*, *iterable of iterable 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** (*iterable*, *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.
- **force_subprocess** (*bool*) – Force to use tabula-java subprocess mode. If you have some issue with jpyype, try this option with same environment. Default `False`.
- **options** (*str*, *optional*) – Raw option string for tabula-java.

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.
- **subprocess.CallProcessError** – If tabula-java execution failed.

```
tabula.io.convert_into_by_batch(input_dir: str, output_format: str = 'csv', java_options: List[str] | None = None, pages: str | int | Iterable[int] | None = None, guess: bool = True, area: Iterable[float] | Iterable[Iterable[float]] | None = None, relative_area: bool = False, lattice: bool = False, stream: bool = False, password: str | None = None, silent: bool | None = None, columns: Iterable[float] | None = None, relative_columns: bool = False, format: str | None = None, output_path: str | None = None, force_subprocess: bool = False, options: str = "") → None
```

Convert tables from PDFs in a directory.

Parameters

- **input_dir** (*str*) – Directory path.
- **output_format** (*str*, *optional*) – Output format of this function (csv, json or tsv)
- **java_options** (*list*, *optional*) – Set java options like `-Xmx256m`. This option will be ignored once JVM is launched.
- **pages** (*str*, *int*, *iterable of int*, *optional*) – An optional values specifying pages to extract from. It allows `str`int``, *iterable of :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** (*iterable of float*, *iterable of iterable 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** (*iterable*, *optional*) – X coordinates of column boundaries.

Example

```
[10.1, 20.2, 30.3]
```

- **relative_columns** (*bool*, *optional*) – If all values are between 0-100 (inclusive) and preceded by '%', input will be taken as % of actual width of the page. Default `False`.
- **format** (*str*, *optional*) – Format for output file or extracted object. ("CSV", "TSV", "JSON")
- **force_subprocess** (*bool*) – Force to use tabula-java subprocess mode. If you have some issue with jpyype, try this option with same environment. Default `False`.
- **options** (*str*, *optional*) – Raw option string for tabula-java.

Returns

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

Raises

ValueError – If *input_dir* doesn't exist.

```
tabula.io.read_pdf(input_path: IO | str | PathLike, output_format: str | None = None, encoding: str = 'utf-8',
    java_options: List[str] | None = None, pandas_options: Dict[str, Any] | None = None,
    multiple_tables: bool = True, user_agent: str | None = None, use_raw_url: bool = False,
    pages: str | int | Iterable[int] | None = None, guess: bool = True, area: Iterable[float] |
    Iterable[Iterable[float]] | None = None, relative_area: bool = False, lattice: bool = False,
    stream: bool = False, password: str | None = None, silent: bool | None = None, columns:
    Iterable[float] | None = None, relative_columns: bool = False, format: str | None = None,
    batch: str | None = None, output_path: str | None = None, force_subprocess: bool = False,
    options: str = "") → List[DataFrame] | Dict[str, Any]
```

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) Giving this option enforces to ignore *multiple_tables* option.
- **encoding** (*str, optional*) – Encoding type for pandas. Default: utf-8
- **java_options** (*list, optional*) – Set java options. This option will be ignored once JVM is launched.

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.
- **use_raw_url** (*bool*) – It enforces to use *input_path* string for url without quoting/dequoting. Default: False
- **pages** (*str, int, iterable of int, optional*) – An optional values specifying pages to extract from. It allows *str*, *int*, *iterable of 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**(*iterable of float, iterable of iterable 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**(*iterable, optional*) – X coordinates of column boundaries.

Example

```
[10.1, 20.2, 30.3]
```

- **relative_columns**(*bool, optional*) – If all values are between 0-100 (inclusive) and preceded by '%', input will be taken as % of actual width of the page. Default False.
- **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.
- **force_subprocess**(*bool*) – Force to use tabula-java subprocess mode. If you have some issue with jpye, try this option with same environment. Default False.
- **options**(*str, optional*) – Raw option string for tabula-java.

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`.

```
>>> import tabula
>>> pdf_path = "https://github.com/chezou/tabula-py/raw/master/tests/resources/data.
↳pdf"
>>> tabula.read_pdf(pdf_path, stream=True)
[
  Unnamed: 0  mpg  cyl  disp  hp  drat    wt   qsec  vs  am  gear
↳carb
0      Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46  0  1    4
↳ 4
1      Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02  0  1    4
↳ 4
2      Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61  1  1    4
↳ 1
3      Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44  1  0    3
↳ 1
4      Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02  0  0    3
↳ 2
5      Valiant  18.1    6  225.0  105  2.76  3.460  20.22  1  0    3
↳ 1
6      Duster 360  14.3    8  360.0  245  3.21  3.570  15.84  0  0    3
↳ 4
7      Merc 240D  24.4    4  146.7   62  3.69  3.190  20.00  1  0    4
↳ 2
8      Merc 230  22.8    4  140.8   95  3.92  3.150  22.90  1  0    4
↳ 2
9      Merc 280  19.2    6  167.6  123  3.92  3.440  18.30  1  0    4
↳ 4
10     Merc 280C  17.8    6  167.6  123  3.92  3.440  18.90  1  0    4
↳ 4
11     Merc 450SE  16.4    8  275.8  180  3.07  4.070  17.40  0  0    3
↳ 3
12     Merc 450SL  17.3    8  275.8  180  3.07  3.730  17.60  0  0    3
↳ 3
13     Merc 450SLC  15.2    8  275.8  180  3.07  3.780  18.00  0  0    3
↳ 3
14  Cadillac Fleetwood  10.4    8  472.0  205  2.93  5.250  17.98  0  0    3
↳ 4
15  Lincoln Continental  10.4    8  460.0  215  3.00  5.424  17.82  0  0    3
↳ 4
16  Chrysler Imperial  14.7    8  440.0  230  3.23  5.345  17.42  0  0    3
↳ 4
17      Fiat 128  32.4    4   78.7   66  4.08  2.200  19.47  1  1    4
↳ 1
18     Honda Civic  30.4    4   75.7   52  4.93  1.615  18.52  1  1    4
```

(continues on next page)

(continued from previous page)

→ 2												
19	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	┐
→ 1												
20	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	┐
→ 1												
21	Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	┐
→ 2												
22	AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	┐
→ 2												
23	Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	┐
→ 4												
24	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	┐
→ 2												
25	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	┐
→ 1												
26	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	┐
→ 2												
27	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	┐
→ 2												
28	Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	┐
→ 4												
29	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	┐
→ 6												
30	Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	┐
→ 8												
31	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	┐
→ 2]												

If you want to extract all pages, set `pages="all"`.

```
>>> dfs = tabula.read_pdf(pdf_path, pages="all")
>>> len(dfs)
4
>>> dfs
[      0      1      2      3      4      5      6      7      8      9
0  mpg  cyl  disp  hp  drat    wt   qsec  vs  am  gear
1  21.0    6  160.0  110  3.90  2.620  16.46   0   1    4
2  21.0    6  160.0  110  3.90  2.875  17.02   0   1    4
3  22.8    4  108.0   93  3.85  2.320  18.61   1   1    4
4  21.4    6  258.0  110  3.08  3.215  19.44   1   0    3
5  18.7    8  360.0  175  3.15  3.440  17.02   0   0    3
6  18.1    6  225.0  105  2.76  3.460  20.22   1   0    3
7  14.3    8  360.0  245  3.21  3.570  15.84   0   0    3
8  24.4    4  146.7   62  3.69  3.190  20.00   1   0    4
9  22.8    4  140.8   95  3.92  3.150  22.90   1   0    4
10 19.2    6  167.6  123  3.92  3.440  18.30   1   0    4
11 17.8    6  167.6  123  3.92  3.440  18.90   1   0    4
12 16.4    8  275.8  180  3.07  4.070  17.40   0   0    3
13 17.3    8  275.8  180  3.07  3.730  17.60   0   0    3
14 15.2    8  275.8  180  3.07  3.780  18.00   0   0    3
15 10.4    8  472.0  205  2.93  5.250  17.98   0   0    3
16 10.4    8  460.0  215  3.00  5.424  17.82   0   0    3
```

(continues on next page)

(continued from previous page)

17	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	
18	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	
19	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	
20	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	
21	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	
22	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	
23	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	
24	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	
25	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	
26	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	
27	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	
28	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	
29	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	
30	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	
31	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5,	0
→	1		2		3		4				
0	Sepal.Length		Sepal.Width		Petal.Length		Petal.Width			Species	
1		5.1		3.5		1.4		0.2		setosa	
2		4.9		3.0		1.4		0.2		setosa	
3		4.7		3.2		1.3		0.2		setosa	
4		4.6		3.1		1.5		0.2		setosa	
5		5.0		3.6		1.4		0.2		setosa	
6		5.4		3.9		1.7		0.4		setosa,	0
→	1		2		3		4		5		
0	NaN	Sepal.Length		Sepal.Width		Petal.Length		Petal.Width		Species	
1	145		6.7		3.3		5.7		2.5	virginica	
2	146		6.7		3.0		5.2		2.3	virginica	
3	147		6.3		2.5		5.0		1.9	virginica	
4	148		6.5		3.0		5.2		2.0	virginica	
5	149		6.2		3.4		5.4		2.3	virginica	
6	150		5.9		3.0		5.1		1.8	virginica,	0
0	supp										
1	VC										
2	VC										
3	VC										
4	VC										
5	VC										
6	VC										
7	VC										
8	VC										
9	VC										
10	VC										
11	VC										
12	VC										
13	VC										
14	VC]										

```
tabula.io.read_pdf_with_template(input_path: IO | str | PathLike, template_path: IO | str | PathLike,
                                pandas_options: Dict[str, Any] | None = None, encoding: str = 'utf-8',
                                java_options: List[str] | None = None, user_agent: str | None = None,
                                use_raw_url: bool = False, pages: str | int | Iterable[int] | None = None,
                                guess: bool = False, area: Iterable[float] | Iterable[Iterable[float]] |
                                None = None, relative_area: bool = False, lattice: bool = False, stream:
                                bool = False, password: str | None = None, silent: bool | None = None,
                                columns: List[float] | None = None, relative_columns: bool = False,
                                format: str | None = None, batch: str | None = None, output_path: str |
                                None = None, force_subprocess: bool = False, options: str | None =
                                None) → List[DataFrame]
```

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"]. This option will be ignored once JVM is launched.
- **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.
- **use_raw_url** (*bool*) – It enforces to use *input_path* string for url without quoting/dequoting. Default: False
- **pages** (*str, int, iterable of int, optional*) – An optional values specifying pages to extract from. It allows *str*, *int*, *iterable of 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** (*iterable of float, iterable of iterable 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** (*iterable*, *optional*) – X coordinates of column boundaries.

Example

```
[10.1, 20.2, 30.3]
```

- **relative_columns** (*bool*, *optional*) – If all values are between 0-100 (inclusive) and preceded by '%', input will be taken as % of actual width of the page. Default `False`.
- **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`.
- **force_subprocess** (*bool*) – Force to use `tabula-java` subprocess mode. If you have some issue with `jpy`, try this option with same environment. Default `False`.
- **options** (*str*, *optional*) – Raw option string for `tabula-java`.

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.
- **`subprocess.CallProcessError`** – If `tabula-java` execution failed.

Examples

You can use template file extracted by tabula app.

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

	Unnamed: 0	mpg	cyl	disp	hp	...	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	...	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	...	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	...	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	...	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	...	17.02	0	0	3	2
5	Valiant	18.1	6	225.0	105	...	20.22	1	0	3	1
6	Duster 360	14.3	8	360.0	245	...	15.84	0	0	3	4
7	Merc 240D	24.4	4	146.7	62	...	20.00	1	0	4	2
8	Merc 230	22.8	4	140.8	95	...	22.90	1	0	4	2
9	Merc 280	19.2	6	167.6	123	...	18.30	1	0	4	4
10	Merc 280C	17.8	6	167.6	123	...	18.90	1	0	4	4
11	Merc 450SE	16.4	8	275.8	180	...	17.40	0	0	3	3
12	Merc 450SL	17.3	8	275.8	180	...	17.60	0	0	3	3
13	Merc 450SLC	15.2	8	275.8	180	...	18.00	0	0	3	3
14	Cadillac Fleetwood	10.4	8	472.0	205	...	17.98	0	0	3	4
15	Lincoln Continental	10.4	8	460.0	215	...	17.82	0	0	3	4
16	Chrysler Imperial	14.7	8	440.0	230	...	17.42	0	0	3	4
17	Fiat 128	32.4	4	78.7	66	...	19.47	1	1	4	1
18	Honda Civic	30.4	4	75.7	52	...	18.52	1	1	4	2
19	Toyota Corolla	33.9	4	71.1	65	...	19.90	1	1	4	1
20	Toyota Corona	21.5	4	120.1	97	...	20.01	1	0	3	1
21	Dodge Challenger	15.5	8	318.0	150	...	16.87	0	0	3	2
22	AMC Javelin	15.2	8	304.0	150	...	17.30	0	0	3	2
23	Camaro Z28	13.3	8	350.0	245	...	15.41	0	0	3	4
24	Pontiac Firebird	19.2	8	400.0	175	...	17.05	0	0	3	2
25	Fiat X1-9	27.3	4	79.0	66	...	18.90	1	1	4	1
26	Porsche 914-2	26.0	4	120.3	91	...	16.70	0	1	5	2
27	Lotus Europa	30.4	4	95.1	113	...	16.90	1	1	5	2
28	Ford Pantera L	15.8	8	351.0	264	...	14.50	0	1	5	4
29	Ferrari Dino	19.7	6	145.0	175	...	15.50	0	1	5	6
30	Maserati Bora	15.0	8	301.0	335	...	14.60	0	1	5	8
31	Volvo 142E	21.4	4	121.0	109	...	18.60	1	1	4	2

[32 rows x 12 columns],

	0	1	2	3	4
0	NaN	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa,

	0	1	2	3	4	5
0	NaN	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	145	6.7	3.3	5.7	2.5	virginica
2	146	6.7	3.0	5.2	2.3	virginica
3	147	6.3	2.5	5.0	1.9	virginica
4	148	6.5	3.0	5.2	2.0	virginica

(continues on next page)

(continued from previous page)

```

5 149          6.2          3.4          5.4          2.3 virginica,
   Unnamed: 0 supp  dose
0          4.2  VC   0.5
1         11.5  VC   0.5
2          7.3  VC   0.5
3          5.8  VC   0.5
4          6.4  VC   0.5
5         10.0  VC   0.5
6         11.2  VC   0.5
7         11.2  VC   0.5
8          5.2  VC   0.5
9          7.0  VC   0.5
10        16.5  VC   1.0
11        16.5  VC   1.0
12        15.2  VC   1.0
13        17.3  VC   1.0]

```

4.1.2 tabula.util

Utility module providing some convenient functions.

```

class tabula.util.TabulaOption(pages: str | int | Iterable[int] | None = None, guess: bool = True, area:
    Iterable[float] | Iterable[Iterable[float]] | None = None, relative_area: bool
    = False, lattice: bool = False, stream: bool = False, password: str | None =
    None, silent: bool | None = None, columns: Iterable[float] | None = None,
    relative_columns: bool = False, format: str | None = None, batch: str | None
    = None, output_path: str | None = None, options: str | None = "",
    multiple_tables: bool = True)

```

Bases: object

Build options for tabula-java

Parameters

- **pages** (str, int, *iterable* of int, optional) – An optional values specifying pages to extract from. It allows str, `int`, *iterable* of :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**(*iterable* of float, *iterable* of *iterable* 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** (*iterable*, *optional*) – X coordinates of column boundaries.

Example

```
[10.1, 20.2, 30.3]
```

- **relative_columns** (*bool*, *optional*) – If all values are between 0-100 (inclusive) and preceded by '%', input will be taken as % of actual width of the page. Default `False`.
- **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`.
- **multiple_tables** (*bool*, *optional*) – Extract multiple tables into a dataframe. Default: `True`

area: `Iterable[float] | Iterable[Iterable[float]] | None = None`

batch: `str | None = None`

build_option_list() `→ List[str]`

Convert to `tabula-java` option list

columns: `Iterable[float] | None = None`

format: `str | None = None`

guess: `bool = True`

lattice: bool = False

merge(*other*: *TabulaOption*) → *TabulaOption*

Merge two *TabulaOption*. self will overwrite other fields' values.

multiple_tables: bool = True

options: str | None = ''

output_path: str | None = None

pages: str | int | Iterable[int] | None = None

password: str | None = None

relative_area: bool = False

relative_columns: bool = False

silent: bool | None = None

stream: bool = False

tabula.util.environment_info() → None

Show environment information for reporting.

Returns

Detailed information like Python version, Java version, or OS environment, etc.

Return type

str

tabula.util.java_version() → str

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*: IO | str | PathLike) → List[*TabulaOption*]

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: IO | str | PathLike) → bool`

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: IO | str | PathLike, user_agent: str | None = None, suffix: str = '.pdf', use_raw_url=False) → Tuple[str, bool]`

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.
- **use_raw_url** (*bool*) – Use *path_or_buffer* without quoting/dequoting.

Returns

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

Return type

(str, bool)

TABULA.ERRORS

exception `tabula.errors.CSVParseError` (*message: Any, cause: Any*)

Bases: `ParserError`

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

exception `tabula.errors.JavaNotFoundError`

Bases: `Exception`

Error represents Java doesn't exist.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

t

- `tabula.errors`, [29](#)
- `tabula.file_util`, [28](#)
- `tabula.io`, [13](#)
- `tabula.template`, [27](#)
- `tabula.util`, [25](#)

A

area (*tabula.util.TabulaOption attribute*), 26

B

batch (*tabula.util.TabulaOption attribute*), 26

build_option_list() (*tabula.util.TabulaOption method*), 26

C

columns (*tabula.util.TabulaOption attribute*), 26

convert_into() (*in module tabula.io*), 13

convert_into_by_batch() (*in module tabula.io*), 15

CSVParseError, 29

E

environment_info() (*in module tabula.util*), 27

F

format (*tabula.util.TabulaOption attribute*), 26

G

guess (*tabula.util.TabulaOption attribute*), 26

I

is_file_like() (*in module tabula.file_util*), 28

J

java_version() (*in module tabula.util*), 27

JavaNotFoundError, 29

L

lattice (*tabula.util.TabulaOption attribute*), 26

load_template() (*in module tabula.template*), 27

localize_file() (*in module tabula.file_util*), 28

M

merge() (*tabula.util.TabulaOption method*), 27

module

tabula.errors, 29

tabula.file_util, 28

tabula.io, 13

tabula.template, 27

tabula.util, 25

multiple_tables (*tabula.util.TabulaOption attribute*), 27

O

options (*tabula.util.TabulaOption attribute*), 27

output_path (*tabula.util.TabulaOption attribute*), 27

P

pages (*tabula.util.TabulaOption attribute*), 27

password (*tabula.util.TabulaOption attribute*), 27

R

read_pdf() (*in module tabula.io*), 16

read_pdf_with_template() (*in module tabula.io*), 21

relative_area (*tabula.util.TabulaOption attribute*), 27

relative_columns (*tabula.util.TabulaOption attribute*), 27

S

silent (*tabula.util.TabulaOption attribute*), 27

stream (*tabula.util.TabulaOption attribute*), 27

T

tabula.errors

module, 29

tabula.file_util

module, 28

tabula.io

module, 13

tabula.template

module, 27

tabula.util

module, 25

TabulaOption (*class in tabula.util*), 25