pdfautonup Documentation

Release 1.9.0

Louis Paternault

Feb 19, 2024

CONTENTS

Render PDF files to a 'n-up' PDF file of a given page size, guessing the layout.

This program is similar to pdfnup or pdfnup (yes, same name) with the following difference:

- `pdfnup` is focused on layout: "I want my pdf to appear 'n-upped' on a '2x3' layout".
- `pdfautonup` is focused on destination paper size: "I want to fit as many pages on a pdf of a given page size".

CHAPTER

ONE

RATIONALE

As a teacher, I often write A5 (or some weirder format) documents, to be printed on A4 paper, copied and given to my students. I was tired of:

- having to explicitely specify the destination page and the number of source pages to appear in one destination page (since it can be automatically computed using the source and destination page format);
- having to repeat my source file as an argument (when, for example, merging four identical instances of a A6 document to an A4 paper).

Indeed, such a command would look like:

```
pdfnup --no-landscape --nup 2x2 a6.pdf a6.pdf a6.pdf a6.pdf
```

This program `pdfautonup` automatically does this:

- it computes how many source pages fit into one destination page;
- it include source files several times is necessary, not to waste space on the destination file.

Using `pdfautonup`, the following command produces the same result as above:

```
pdfautonup a6.pdf
```

CHAPTER TWO

EXAMPLES

With the default paper size being A4:

- command `pdfautonup trigo.pdf` turns `trigo.pdf` into `trigo-nup.pdf`
- command `pdfautonup --algorithm panel --gap .5cm --margin 1cm pcb.pdf` turns `pcb.pdf` into `pcb-nup.pdf`.
- command `pdfautonup three-pages.pdf` turns `three-pages.pdf` into `three-pages-nup.pdf`.

CHAPTER THREE

DOWNLOAD AND INSTALL

See the main project page for instructions, and changelog.

CHAPTER FOUR

USAGE (COMMAND LINE)

Here are the command line options for *pdfautonup*.

Convert PDF files to 'n-up' file, with multiple input pages per destination pages. The output size is configurable, and the program compute the page layout, to fit as much source pages in per destination pages as possible. If necessary, the source pages are repeated to fill all destination pages.

```
usage: pdfautonup [-h] [--version] [--help-paper] [--list-units]
                  [--list-sizes] [--output [OUTPUT]] [--interactive]
                  [--algorithm {fuzzy,panel}]
                  [--orientation {auto,portrait,landscape}]
                  [--size TARGET_SIZE] [--margin MARGIN] [--gap GAP]
                  [--repeat REPEAT] [--progress PROGRESS] [--verbose]
                  [FILES ...]
```

4.1 Positional Arguments

FILES	PDF files to merge. If their page sizes are different, they are considered to have the same page size, which is the maximum width and height of all pages. To read from standard input, use "-". Default: ['-']

4.2 Named Arguments

--version	Show version
--help-paper	Show an help message about paper sizes, and exit.
--list-units	Display the list of available units, and their meaning, and exit.
--list-sizes	Display the list of available sizes, and their meaning, and exit.
--output, -o	Destination file (or "-" to write to standard output). Default is "-nup" appended to first source file (excepted if first source file is standard input, where default output is standard output).
--interactive, -i	Ask before overwriting destination file if it exists. Default: False

--algorithm, -a
: Possible choices: fuzzy, panel

  Algorithm used to arrange source documents into destination documents. This program tries to put as many copies of the source document into the destination document, given that: - *fuzzy*: documents can overlap, or leave blank spaces between them, but not too much; - *panel*: the gap length between documents is fixed, and a minimum destination margin is respected.

--orientation, -O
: Possible choices: auto, portrait, landscape

  Destination paper orientation. Default is 'auto', which choose the paper orientation to fit the maximum number of source pages on the destination page.

  Default: "auto"

--size, -s
: Size of target paper. Run *pdfautonup –help-paper* for more information.

--margin, -m
: Margin size (e.g. "0.8cm"). Run *pdfautonup –help-paper* for more information.

--gap, -g
: Gap size (e.g. "1.2mm"). Run *pdfautonup –help-paper* for more information about units.

--repeat, -r
: Number of times the input files have to be repeated. Possible values are: - an integer; - 'fit': the input files are repeated enough time to leave no blank

  space in the output file.

  - 'auto': if there is only one input page, equivalent to 'fit'; else, equivalent to 1.

  Default: auto

--progress, -p
: Text to print after processing each page. Strings "{page}", "{pagetotal}", "{percent}" are replaced by their respective values. The following alias are defined: - 'none': no progress; - 'dot': '.'; - 'pages': '{page}/{total}n'; - 'percent': '{percent}%n'.

  Default: ""

--verbose, -v
: Increase verbosity.

  Default: False

The backend Python library used to read and write PDF files can be forced using the environment variable PDFBACKEND. If this variable is not defined (or defined with an invalid value), a library is automatically (and silently) chosen. Available libraries are: 'pymupdf', 'pypdf'.

CHAPTER FIVE

USAGE (LIBRARY)

Pdfautonup can also be used as a library, with the following function.

`pdfautonup.pdfautonup`(*files*, *output=None*, *size=None*, ***, *algorithm=None*, *repeat='auto'*, *more=None*, *interactive=False*, *orientation='auto'*, *progress=<function _none_function>*)

Convert PDF files to 'n-up' PDF files, guessing the output layout.

New in version 1.9.0.

Parameters

- **`files`** (`Sequence[str]`) – Names of the files to process.
- **`output`** (`str`) – Name of the output file. If `None`, writes to standard output.
- **`size`** (`str | None | tuple[Decimal]`) – Size of the pages of the destination file, either as: a tuple of `decimal.Decimal` (width and height, in points); a string to be parsed by `papersize.parse_papersize()`; `None`, in which case the default paper size will be used.
- **`repeat`** (`Union[Literal['auto', 'fit'], int]`) – If a number, repeat the input file this number of times. If `"fit"`, repeat the input files as many times as necessary to fill all pages of the destination file. If `"auto"`, is equivalent to `"fit"` if input file has one page, and equivalent to 1 otherwise.
- **`algorithm`** (`Optional[Literal['fuzzy', 'panel']]`) – Select one algorithm, either `"fuzzy"` (document pages can overlap or leave blank space between them, but not too much) or `"panel"` (the gap length between source pages is fixed, and a minimum destination margin is respected). If `None`, chooses `"panel"` if `"margin"` or `"gap"` are defined in `more`, and `"fuzzy"` otherwise.
- **`orientation`** (`Literal['auto', 'portrait', 'landscape']`) – Force orientation of destination file to portrait or landscape. If `"auto"`, select the one that fits the most input pages.
- **`more`** (`dict`) – Additional arguments for algorithms. The *fuzzy* algorithm does not accept any arguments, while the *panel* algorithm accepts `"margin"` and `"gap"` (as strings to be parsed by `papersize.parse_length()` or `decimal.Decimal` as the length in points).
- **`progress`** (`Callable[[int, int], None]`) – A function that takes to integer arguments (number of pages processed so far, and total number of pages to process), and display a progress. Whether it is displayed on standard input or a GUI or something else is up to you.
- **`interactive`** (`bool`) – If `True`, asks for confirmation before overwriting the destination file, if it already exists.

Warning: If a file is `"-"`, it is read from standard input. I am still undecided about how reading from standard input should be handled, so this might change in the future.

So, right now, reading from standard input is unsupported, and input files cannot be named `"-"`.

CHAPTER SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX