pdfmole

The **pdfmole** package provides a bundle of functions designed to assist in extracting tables from PDF-files. To read-in the data either

- **pdfminer**,
- **pdfboxr** or
- **tesseract** can be used.

In principle, any package which returns the data in a similar format could be used. The packages **pdfminer** and **pdfboxr** can be used if the PDF-file store already the text (in most cases) if the PDF contains only images of the tables **tesseract** can be used.

1. Installation

```r
# install tesseract (optional)
install.packages("tesseract")

# install pdfminer (optional)
remotes::install_github("FlorianSchwendinger/pdfminer")

# install pdfboxr (optional)
remotes::install_gitlab("schwe/pdfboxr")

# install pdfmole
remotes::install_github("ben-schwen/pdfmole")
```

2. Workflow

```r
options(width = 100)
library("pdfboxr")
library("pdfmole")
```

The basic work flow of **pdfmole** can be divided into

1. read data (`pdfboxr::read_chars()`)
2. align rows (`align_rows(x, method, ...)`)
3. aligns columns (`align_columns(x, method, ...)`)
4. group cells (`group_cells(x, collapse)`)
5. transform the data into a rectangular format (`mole(x, header, simplify, keep)`)

3. Example usage

3.1. Read data

Use **pdfboxr** (or **pdfminer**) to read the PDF-file into **R**,

```r
pdf_file <- file.path(system.file("pdfs", package = "pdfmole"), "cars.pdf")
pdf <- pdfboxr::read_chars(pdf_file, pages = 1:2)
pdf
```

```
## A pdf document with 2 pages and
##   metainfo text fonts
## 1        2  313     3
## elements.
```

pdfboxr returns an object of class `"pdf_document"` which is a list containing several `data.frames`, the print method shows the number of rows of each `data.frame`. Each `data.frame` contains specific information about the PDF-file, for example the `data.frame` contains meta information about the PDF-file.

```r
pdf$metainfo
```

```
##   rotation y0 x0  y1  x1 pid
## 1        0  0  0 842 595   1
## 2        0  0  0 842 595   2
```

The columns are defined as follows.

Name	Description
`pid`	page id
`rotation`	the rotation of the page
`x0`	the bounding box start coordinate on the x-axis
`x1`	the bounding box end coordinate on the x-axis
`y0`	the bounding box start coordinate on the y-axis
`y1`	the bounding box end coordinate on the y-axis

The `data.frame` named `text` contains the *page id*, *font name*, *font size*, *name of the color space*, *color* and *bounding box* for each character of the PDF-file.

```r
d0 <- pdf$text
d0[c("rotation", "yscale", "xscale")] <- NULL # so the output fits to the column width
head(d0)
```

```
##   pid text                   font size     x0     y0     x1    y1 width height
## 1   1    c BAAAAA+LiberationSans   10 288.20 769.12 293.20 774.7  5.00   5.58
## 2   1    a BAAAAA+LiberationSans   10 293.20 769.12 298.76 774.7  5.56   5.58
## 3   1    r BAAAAA+LiberationSans   10 298.79 769.12 302.12 774.7  3.33   5.58
## 4   1    s BAAAAA+LiberationSans   10 302.09 769.12 307.09 774.7  5.00   5.58
## 5   1    s           Courier-Bold   12  77.20 747.29  84.40 753.6  7.20   6.31
## 6   1    p           Courier-Bold   12  84.40 747.29  91.60 753.6  7.20   6.31
```

Based on this information we want to assign a row and column id to each row of the `data.frame`. This will allow us to transform character level `data.frame` into the typical format. Additionally we can use information like the location, font size or font type to filter out specific characters / blocks / rows.

3.2 Align blocks (optional)

The block information is assigned by **pdfminer** and signals which characters belong to the same text block. The function `group_blocks` can be use to paste together all the characters belonging to the same block. **pdfmoler** provides no block information therefore we skip this part.

```r
d <- group_blocks(d0)
head(d, 3)
```

3.3 Align rows

The function `align_rows(x, method, ...)` currently implements three methods `"exact_match"`, `"hclust"` and `"fixed_width"`.

```
d <- align_rows(d0)
head(d, 30)
```

```
##    pid text                  font size     x0     y0     x1    y1 width height row
## 1    1    c BAAAAA+LiberationSans   10 288.20 769.12 293.20 774.7  5.00   5.58   1
## 2    1    a BAAAAA+LiberationSans   10 293.20 769.12 298.76 774.7  5.56   5.58   1
## 3    1    r BAAAAA+LiberationSans   10 298.79 769.12 302.12 774.7  3.33   5.58   1
## 4    1    s BAAAAA+LiberationSans   10 302.09 769.12 307.09 774.7  5.00   5.58   1
## 5    1    s           Courier-Bold   12  77.20 747.29  84.40 753.6  7.20   6.31   2
## 6    1    p           Courier-Bold   12  84.40 747.29  91.60 753.6  7.20   6.31   2
## 7    1    e           Courier-Bold   12  91.60 747.29  98.80 753.6  7.20   6.31   2
## 8    1    e           Courier-Bold   12  98.80 747.29 106.00 753.6  7.20   6.31   2
## 9    1    d           Courier-Bold   12 106.00 747.29 113.20 753.6  7.20   6.31   2
## 10   1    d           Courier-Bold   12 124.40 747.29 131.60 753.6  7.20   6.31   2
## 11   1    i           Courier-Bold   12 131.60 747.29 138.80 753.6  7.20   6.31   2
## 12   1    s           Courier-Bold   12 138.80 747.29 146.00 753.6  7.20   6.31   2
## 13   1    t           Courier-Bold   12 146.00 747.29 153.20 753.6  7.20   6.31   2
## 14   1    1                Courier   12  67.90 732.27  75.10 738.6  7.20   6.33   4
## 15   1    4                Courier   12 115.20 732.27 122.40 738.6  7.20   6.33   4
## 16   1    2                Courier   12 154.80 732.27 162.00 738.6  7.20   6.33   4
## 17   1    2                Courier   12  67.90 717.27  75.10 723.6  7.20   6.33   5
## 18   1    4                Courier   12 115.20 717.27 122.40 723.6  7.20   6.33   5
## 19   1    1                Courier   12 147.60 717.27 154.80 723.6  7.20   6.33   5
## 20   1    0                Courier   12 154.80 717.27 162.00 723.6  7.20   6.33   5
## 21   1    3                Courier   12  67.90 702.27  75.10 708.6  7.20   6.33   6
## 22   1    7                Courier   12 115.20 702.27 122.40 708.6  7.20   6.33   6
## 23   1    4                Courier   12 154.80 702.27 162.00 708.6  7.20   6.33   6
## 24   1    4                Courier   12  67.90 687.27  75.10 693.6  7.20   6.33   7
## 25   1    7                Courier   12 115.20 687.27 122.40 693.6  7.20   6.33   7
## 26   1    2                Courier   12 147.60 687.27 154.80 693.6  7.20   6.33   7
## 27   1    2                Courier   12 154.80 687.27 162.00 693.6  7.20   6.33   7
## 28   1    5                Courier   12  67.90 672.27  75.10 678.6  7.20   6.33   8
## 29   1    8                Courier   12 115.20 672.27 122.40 678.6  7.20   6.33   8
## 30   1    1                Courier   12 147.60 672.27 154.80 678.6  7.20   6.33   8
```

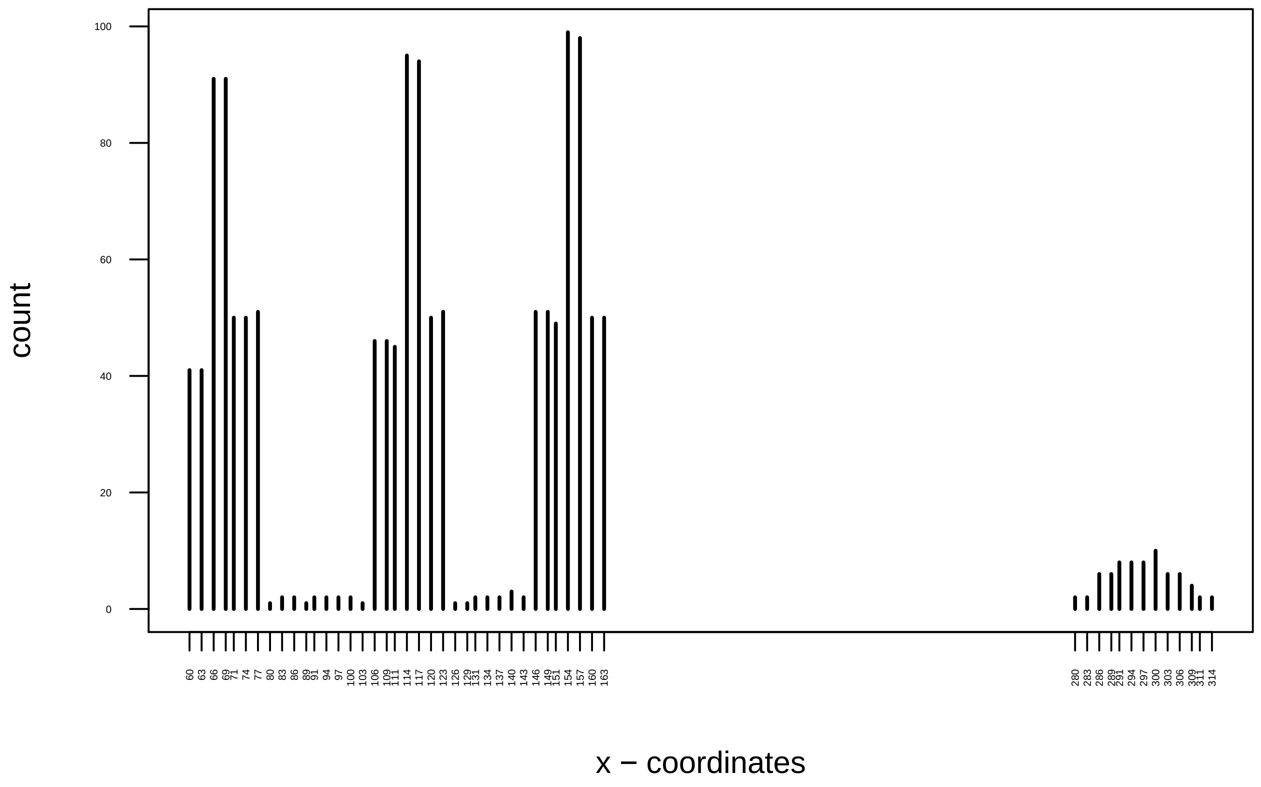
After applying the function the column row is added to the `data.frame`.

3.3 Align columns

Visualization

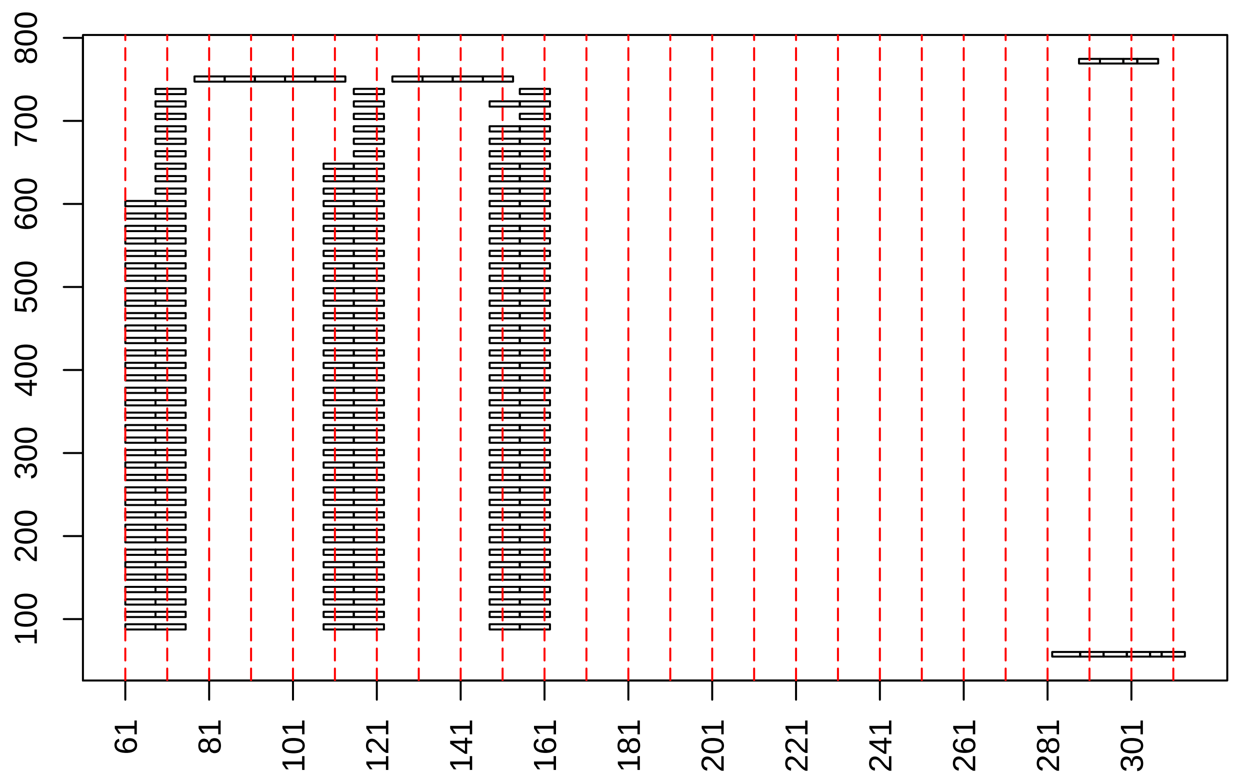
Pixelplot

```
pixelplot(d, scale = 0.35, cex.axis = 0.35)
```



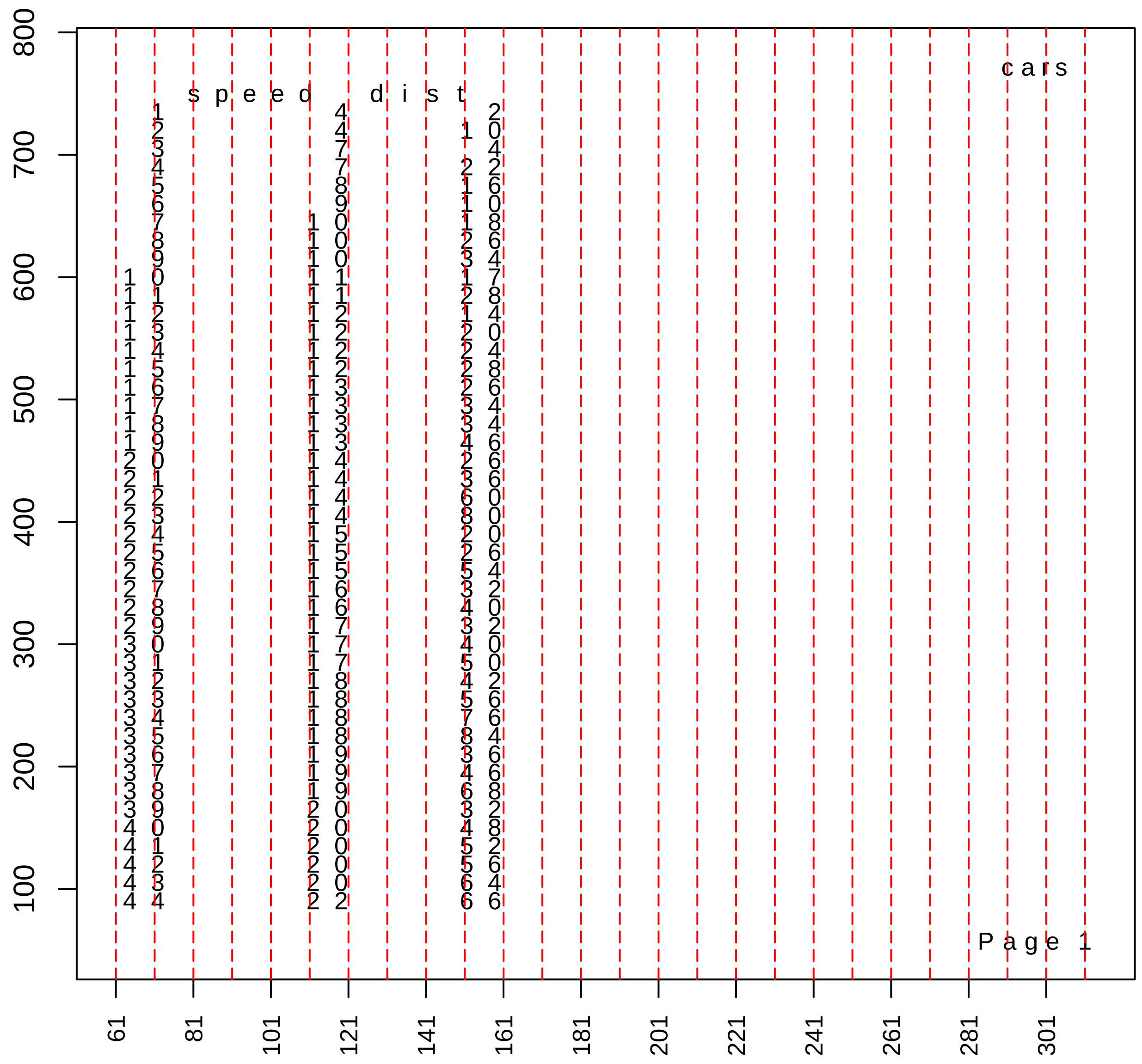
Bounding box plot

```r
bboxplot(d, pid = 1L)
```



Textplot

```
textplot(d, pid = 1L)
```



```
d <- align_columns(d, split_points = c(78, 126, 220))
head(d, 3)
```

```
##   pid text                   font size     x0     y0     x1    y1 width height row col
## 1   1    c BAAAAA+LiberationSans   10 288.20 769.12 293.20 774.7  5.00   5.58   1   4
## 2   1    a BAAAAA+LiberationSans   10 293.20 769.12 298.76 774.7  5.56   5.58   1   4
## 3   1    r BAAAAA+LiberationSans   10 298.79 769.12 302.12 774.7  3.33   5.58   1   4
```

3.5 Filter (optional)

```
d <- d[grep("Courier", d$font),]
```

3.6 Group cells

```r
d <- group_cells(d)
head(d, 3)
```

```
##   pid row col  text
## 1   1   2   2 speed
## 2   1   2   3  dist
## 3   1   4   1     1
```

3.7 Transform into a rectangular (dense) format

After the grouping the data is available similar to a simple triplet sparse matrix format (sometimes called coordinate format). To obtain the usual (dense) `data.frame` / `matrix` format the function `mole()` can be used.

```r
x <- mole(d, header = TRUE, simplify = TRUE)
x
```

```
## X1   speed  dist
## 1    4      2
## 2    4      10
## 3    7      4
## 4    7      22
## 5    8      16
## 6    9      10
## ...  ...    ...
## 8    10     26
## 9    10     34
## 11   11     28
## 12   12     14
## 13   12     20
## 14   12     24
## 16   13     26
## 18   13     34
## 21   14     36
## 24   15     20
## 26   15     54
## 28   16     40
## 32   18     42
## 34   18     76
## 35   18     84
## 39   20     32
## ...  ...    ...
## 45   23     54
## 46   24     70
## 47   24     92
## 48   24     93
## 49   24     120
## 50   25     85
```

The function `mole()` returns an object of class `mole`,

```r
str(x)
```

```
## List of 3
##  $ X1   : int [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##  $ speed: int [1:50] 4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : int [1:50] 2 10 4 22 16 10 18 26 34 17 ...
##  - attr(*, "class")= chr "mole"
```

which can be coerced to either `data.frame`

```
head(as.data.frame(x))
```

```
##   X1 speed dist
## 1  1     4    2
## 2  2     4   10
## 3  3     7    4
## 4  4     7   22
## 5  5     8   16
## 6  6     9   10
```

or `matrix`.

```
head(as.matrix(x))
```

```
##      X1 speed dist
## [1,]  1     4    2
## [2,]  2     4   10
## [3,]  3     7    4
## [4,]  4     7   22
## [5,]  5     8   16
## [6,]  6     9   10
```