WRANGLING DATA OUT OF PDF FILES

[Robert Gebeloff](mailto:rgebeloff@nytimes.com?subject=Your%20PDF%20Presentation)

The New York Times

Years ago, it was how they got you. Some pesky government agency would comply with your request for data by sending you a PDF file. Then you’d have to negotiate/cajole/badger them incessantly for the underlying data.

Those days, thankfully, are mostly over. Not only are more agencies opening up their public data sets, but the number of tools available for reporters to deal with PDF files is as good as ever.

This tutorial will walk you through a bunch of possible solutions. Why a bunch? Because experience shows that one tool doesn’t handle every situation you’ll encounter, and having a working knowledge of several options will come in handy.

I will divide this discussion into three categories of PDF tools:

**Free software**: Programs that cost $0 but generally have the most limited feature sets.

**Commercial software:** Programs that have powerful additional features that you might need to get a project done.

**Programming techniques:** I will show quick demonstrations of how you can open and manipulate PDF files in Ruby, Python and R, three programming languages that are increasingly common in newsrooms.

But before I get to all of the cool solutions, let’s first stop and talk about a few basics. The solution you choose will depend a lot on what you want to accomplish and what shape your source PDF files are in.

For example, sometimes there’s a table embedded in a document that you want to extract into a spreadsheet. If you try cutting and pasting from your PDF reader, you lose all of the columns and rows and are left with an un-analyzable mess.

Another document might also have the table, but when you try to copy and paste it, you get a discouraging message announcing that the PDF you’re reading isn’t text, but an image of text. This means you’ll have an extra step of applying “Optimal Character Recognition” to the document (otherwise known as OCRing) before you can pull out the data.

Sometimes the table you’re going after is not in a simple rows-and-columns format. There might be tables nested inside tables. Or columns filled with text that wraps over more than one line.

These are three different problems with three different solutions. So as I go through your options, I will make note of what situation each solution is best suited for.

**FREE SOFTWARE**

[**DocumentCloud**](https://www.documentcloud.org/) **(**[**https://www.documentcloud.org/**](https://www.documentcloud.org/)**)**

DocumentCloud is not a tool for working with the kind of tabular data you’d want to load into a spreadsheet, but I list it first because it’s an essential tool for many reporting tasks. The site was built by journalists for journalists, and is now supported by Investigative Reporters & Editors.

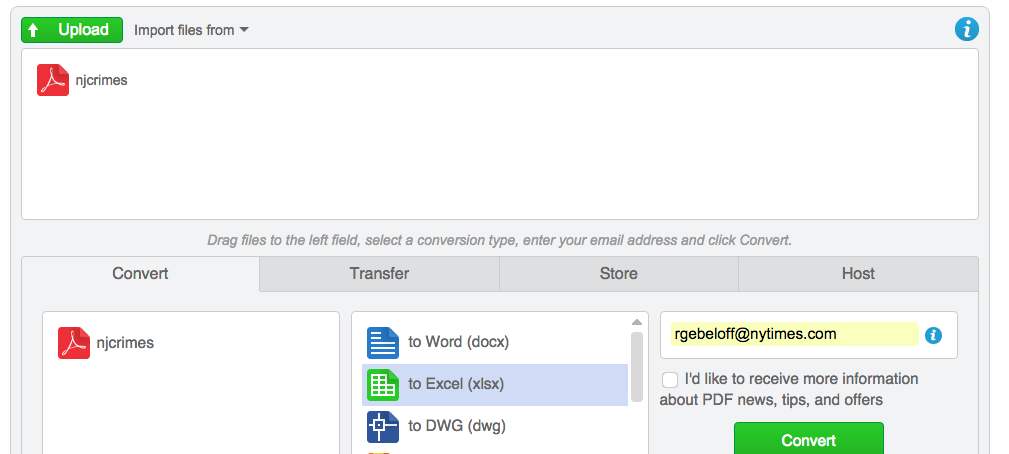
**What it can do:** Most reporters use the service to convert image documents into text documents that can be searched. Built-in algorithms pull out entities from your documents ‘—names, places, institutions – and it’s also easy to add highlights and make your documents public to share with readers.

**Don’t count on:** The service does not preserve table layouts, so it’s not a good choice if you want to extract tables for your spreadsheets.

[**CometDocs**](http://www.cometdocs.com/) **(**[**http://www.cometdocs.com/**](http://www.cometdocs.com/)**)**

CometDocs is the simplest, most effective free tool I know of for getting tables out of PDF files, though there are limits to the free service

**What it can do:** CometDocs is free and easy. Just go to the Web site and upload your file using the easy interface. Then check your e-mail. It works most of the time if your source table has a clear, consistent layout.



**Don’t count on:** The simplicity of CometDocs is also it’s weakness. It doesn’t work every time, and if it fails to interpret the table layout properly, there is not a lot you can do. It also doesn’t handled PDF images for free, but requires a subscription for OCR services.

[**XPDF**](http://www.foolabs.com/xpdf/) **(**[**http://www.foolabs.com/xpdf/**](http://www.foolabs.com/xpdf/)**)**

XPDF is one of the oldest free PDF tools around and was one of the first journalists started to use for extracting tables out of PDF files.

**What it can do:** From the command line, you can convert a PDF table into a text table while (mostly) preserving the layout. On a Mac, for example, this line is all you need:

pdftotext -layout filename.pdf

**Don’t count on:** An easy install. While this has become slightly easier over the years, on a Mac you still need to copy the program binaries into the proper place in your /usr/local/bin directory. And then there are the other shortcomings – it doesn’t do as well with preserving layouts as the commercial software, it doesn’t OCR your files, and you need to pretty much save the table you want as a separate PDF file before you get to work.

[**Tabula**](http://tabula.technology/) **(**[**http://tabula.technology/**](http://tabula.technology/)**)**

Tabula is free software you download and install on your PC or MAC. It allows you to load a PDF file and extract a single table.

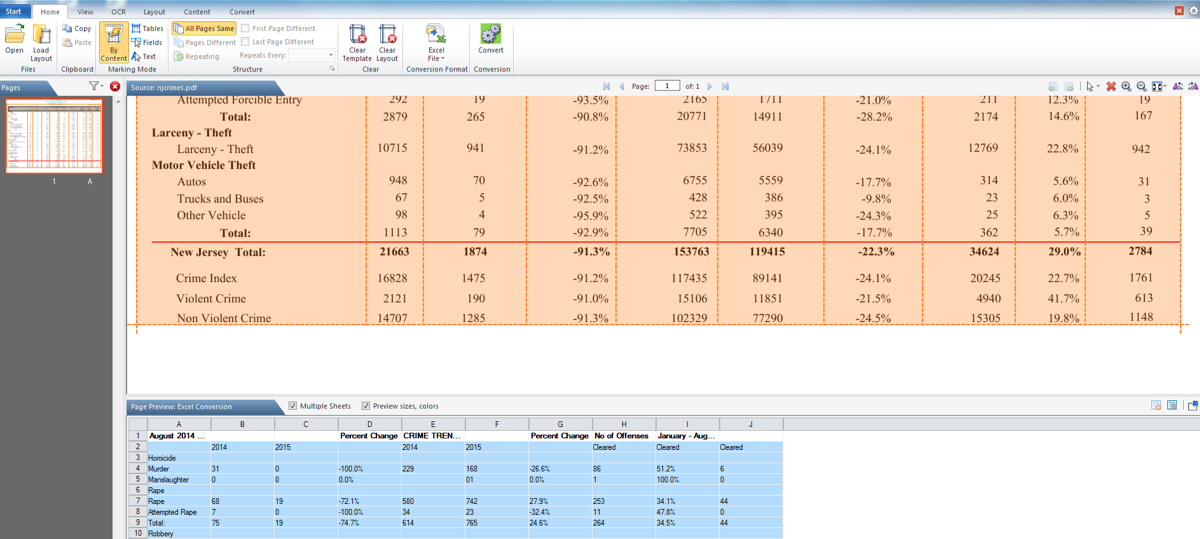
**What it can do:**  Tabula runs like an internal Web service on your computer. You “upload” a PDF file, then draw a box around the table you want to extract, and Tabula attempts to parse the data while maintaining rows and columns. It’s a big improvement over trying to copy and paste data using your free PDF reader and gives you more control over the process than CometDocs.

**Don’t count on:** The program does not OCR documents for you, and is not as fast or accurate as some of the commercial programs mentioned below – for example, if Tabula guesses incorrectly about where to draw the boundaries of columns, you have to clean it up manually.

**COMMERCIAL SOFTWARE**

[**Cogniview**](http://www.cogniview.com) **(**[**http://www.cogniview.com**](http://www.cogniview.com)**)**

Cogniview is the first program of its kind I can remember using, and it was magical. With Cogniview PDF2XL, you can draw a box around your table, a la Tabula, but then there are myriad editing tools that allow you to adjust your extraction template if the program guesses incorrectly the first time.



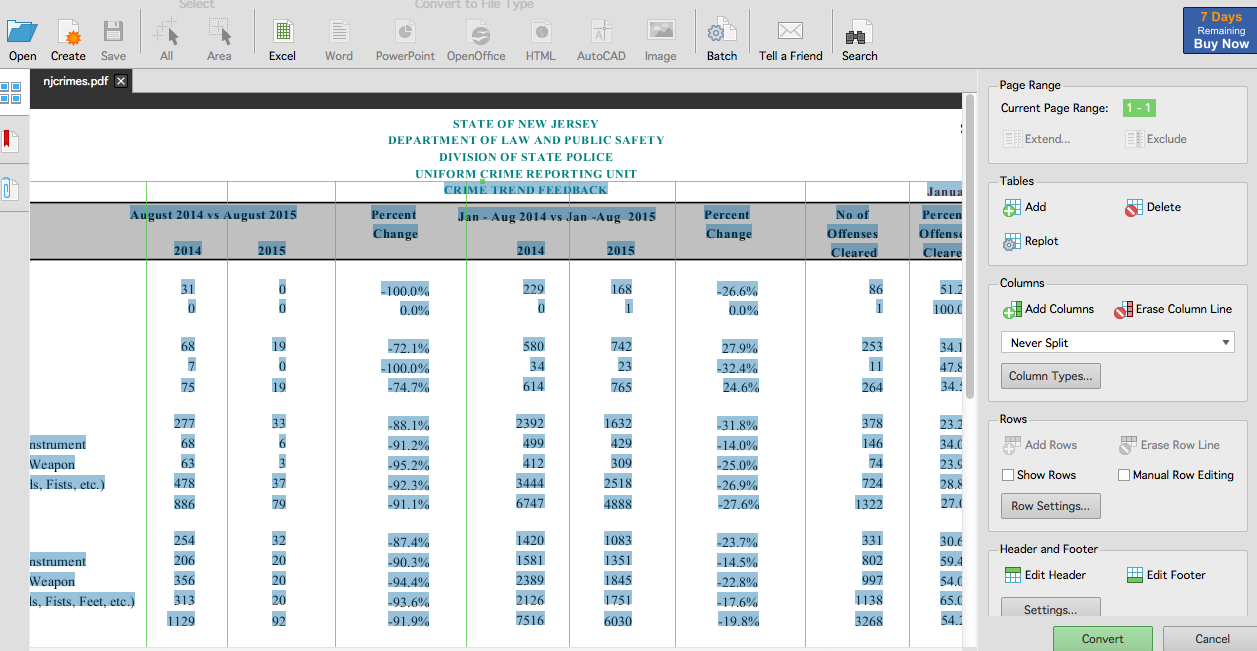
**What it can do:**  Cogniview is brilliant at identifying PDF layouts and then guiding users to fix when it fails. Better yet, there is an OCR version that allows you to operate on files even if they come to you as images.

**Don’t count on:** This is PC only, so if you’re on a Mac, move on. If your table has a lot of word-wrapping, the layouts you built will be more challenging. Also, some colleagues have told me that they find that user interface a bit difficult to master, so it might take you a while to get the hang of it.

**Price:** $199 for the basic, $299 for OCR, but the company often has “sales” – as I write this, there is a 50% off deal going on.

[**ABLE2EXTRACT**](http://www.investintech.com/) **(**[**http://www.investintech.com/**](http://www.investintech.com/)**)**

This is a favorite of the NY Times graphics department. It’s very similar to Cogniview, with perhaps a simpler user interface, plus full Mac support. I’ve found that certain documents work in Cogniview and not Able2Extract, and vice versa.



**What it can do:** Open a file, draw a box around the table, and click a button. That’s Able2Extract at its easiest. If it doesn’t recognize the layout the first time, you can customize the template to get it right.

**Don’t count on:**  While Able2Extract works most of the time, it’s not perfect, and it’s customization scheme is not as sophisticated as Cogniview. For example, if you have a row label that wraps, Cogniview allows you to “merge” two rows into one to preserve the integrity of your spreadsheet, whereas Able2Extract give you two rows, one with part of the label, the other with the remainder of the label and the data.

**Price:** $100 standard, $130 with OCR.

[**ABBY FineReader**](http://www.abbyy.com/finereader/) **(**[**http://www.abbyy.com/finereader/**](http://www.abbyy.com/finereader/)**)**

This software is primarily known for being a very good OCR engine, but some journalists rave about it’s abilities to parse PDF tables – it is better at handling table layouts that are unusual. For more on the pros and cons, I will turn it over to [a tutorial written](https://github.com/sarahcnyt/stabile/blob/master/pdf/extracting_image_pdfs.pdf) by Rob Barry of The Wall Street Journal.

**Price:** $120 for Mac, $170 for PC.

[**Adobe Acrobat Pro**](http://acrobat.adobe.com) **(**[**http://acrobate.adobe.com**](http://acrobate.adobe.com)**)**

Acrobat is a full-fledged PDF processing program. You might have the free Adobe reader on your machine, but with Pro, there is an enormous suite of tools for extracting pages, OCRing text, redacting sensitive information, etc. as well as a tool for extracting tables.

**What it can do:** If you see a table, highlight it, then right click to copy as a table. Then fire up Excel and paste. It doesn’t get much easier and it preserves the layout almost all of the time. It’s also possible that somebody in your news organization already has the Pro version of the software – if so, it might be an easy solution for a one-time conversion job. As mentioned, it can also do a TON of other PDF functions beyond extracting tables that might come in handy – I use it often for OCRing and for combining or splitting PDF files.

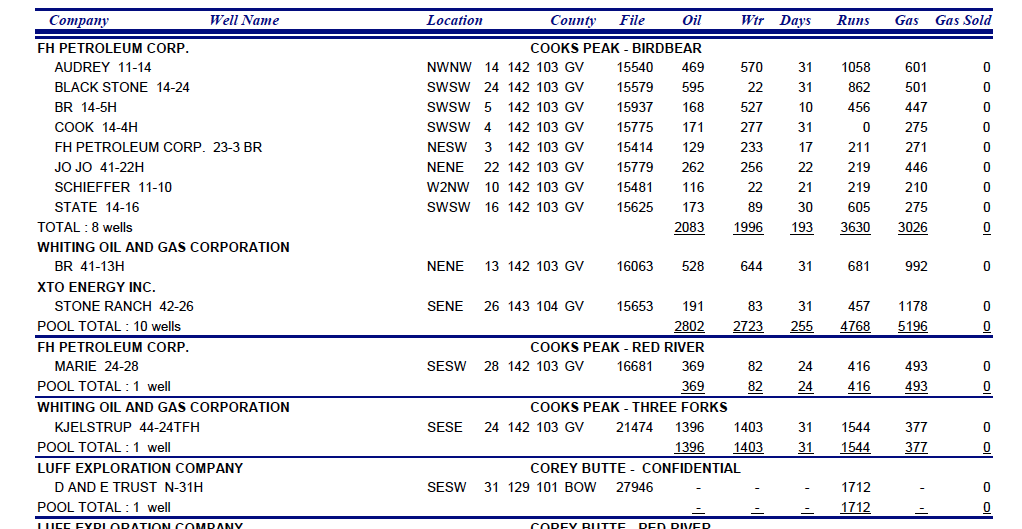
**Don’t count on:** This is not software designed just for table extraction – it’s a high-end document management system. Therefore, it’s not cheap. It’s also a pain to deal with tables that span more than one page., or anything with an unusual layout.

**Price:** As of this writing, Adobe is attempting to develop a cloud-pricing model, where customers “rent” the software month by month. The price I’m seeing is $25 a month, cheaper for annual commitments. The desktop version goes for about $450.

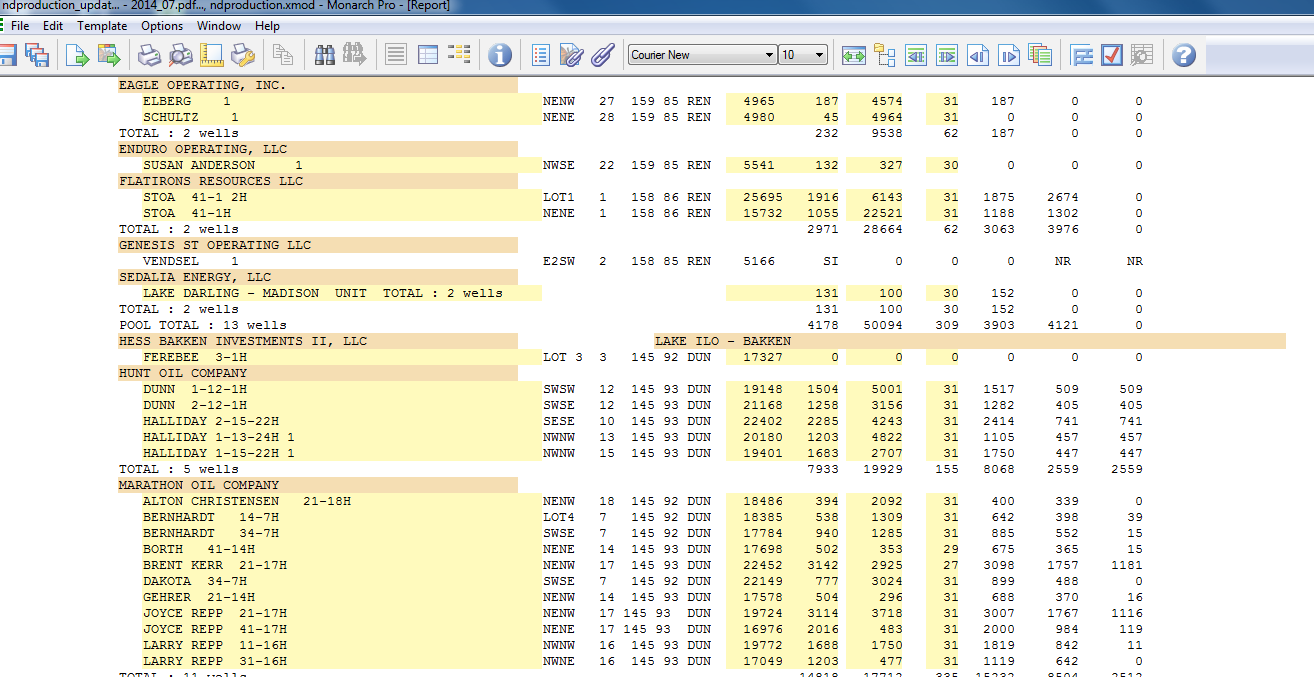
[**DATAWATCH MONARCH**](http://www.datawatch.com/products/datawatch-monarch/) **(**[**http://www.datawatch.com/products/datawatch-monarch/**](http://www.datawatch.com/products/datawatch-monarch/)**)**

This is the superstar software of the bunch, but it isn’t cheap. In fact, Datawatch doesn’t even list the price on its Web site (as they say, if you have to ask…). However, if you’re working on a major long-term project and you HAVE to be able to extract data that is locked in a difficult-to-parse format, Monarch is highly recommended.

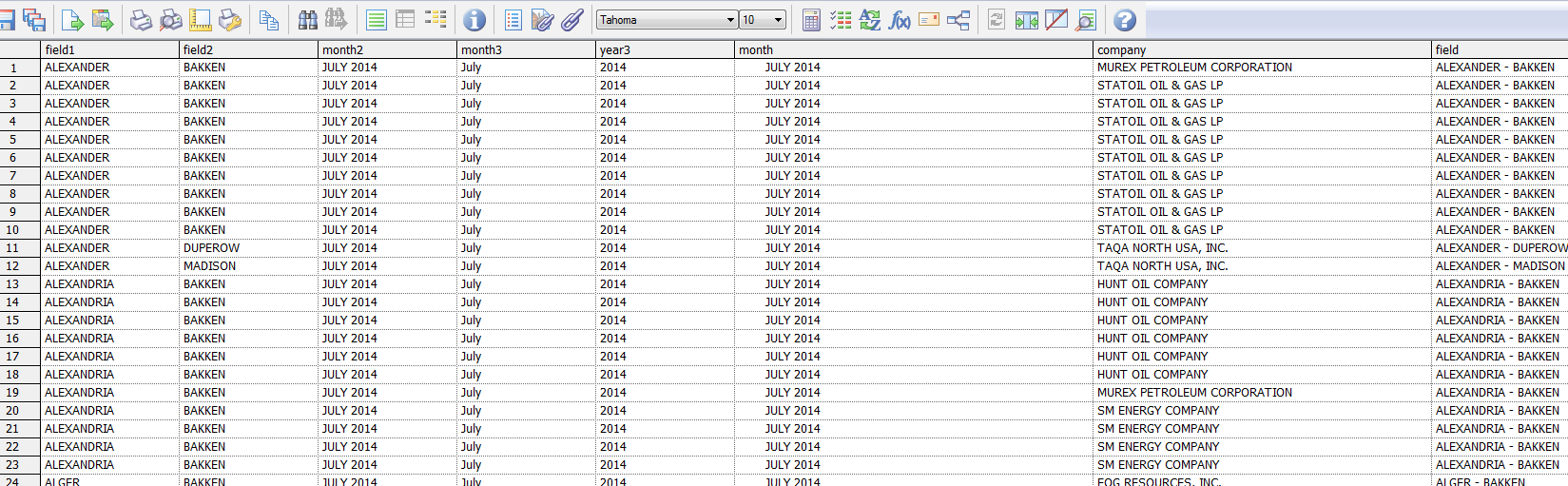
**What it can do:**  Where Monarch really shines is when you have a source file in a “report” layout. For example, in 2014 I had to parse North Dakota oil production data that was reported by oil field, by company and by well – and each detail level was nested in the layout of the page.



Using Monarch, I was able to design an extraction template for every layer based on the row/column layout for that level of data.



And in the end, Monarch “flattens” each layer into a single table, and provides loads of data-cleaning features such as date parsing, eliminating leading/trail spaces, exporting, etc.



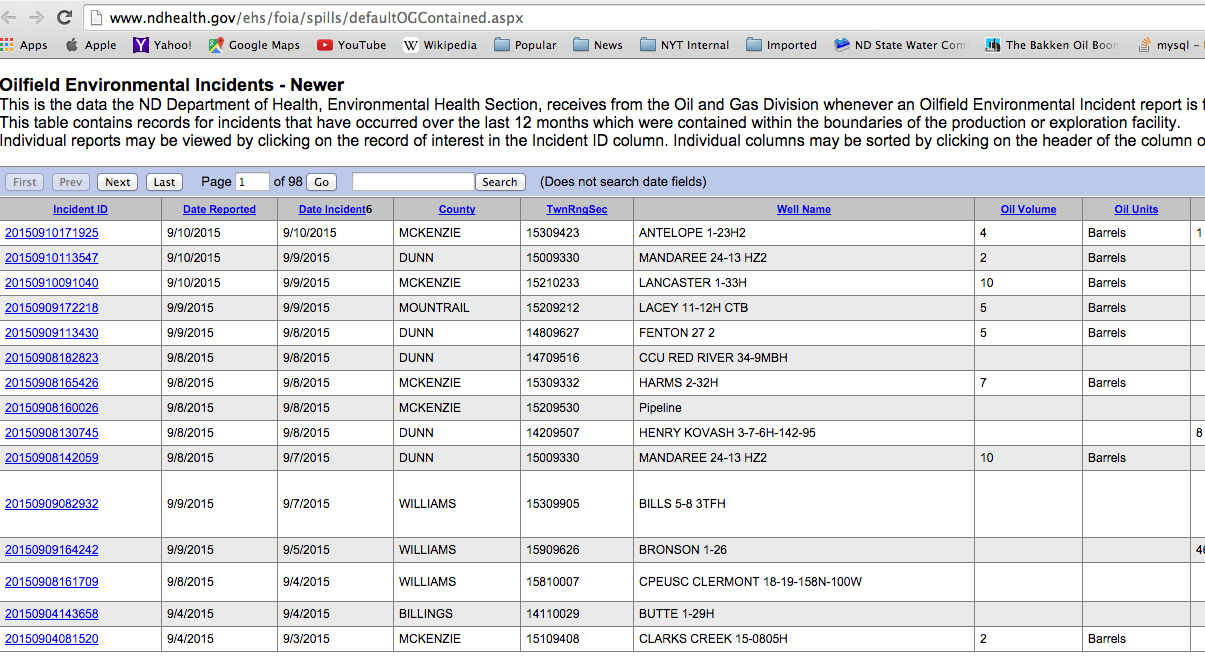
**Don’t count on:** The primary drawback to Monarch is the price. As of September, 2015, the company is asking for $1,600 for a one-year license, which might be ok if you’re a large media company with a big budget.

**PROGRAMMING**

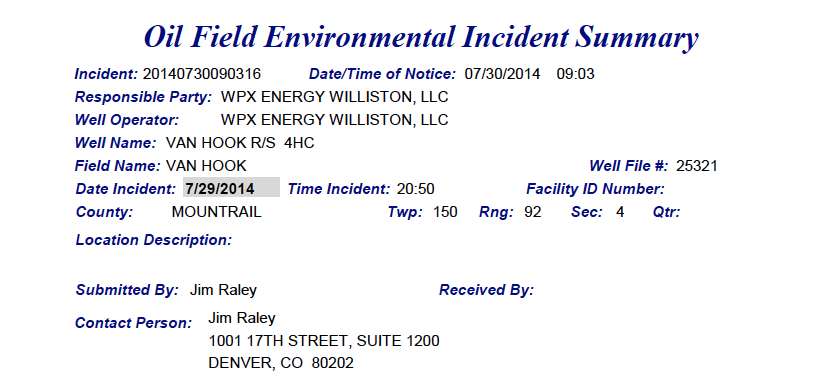
Finally, as journalists become more adept at incorporating programming techniques into their reporting, I wanted to provide a couple of examples of how to work with PDF in a scripting environment.

Not only is this a good excuse for you to burnish your programming skills, it’s free and allows you to do customized parsing that might not be possible with software.

Example number one is my own. For a story last year, I had to build a database of oilfield environmental incidents in North Dakota.



What I needed to do was a write a program that harvested all of the incident reports from the Web site, grabbing the data displayed on the screen, but ALSO clicking on the incident ID hyperlink to get at the incident details, which were stored in a PDF file that looked like this:



While it was possible that I could have tried doing this parsing in Monarch, I instead came up with a plan within the Ruby programming language that I was using to do the scraping.

A program to get and parse a batch of PDF files [might look something like this](https://github.com/gebelo/nd_oil_incidents/blob/master/sample.rb):

require 'pdf/reader'

require 'net/http'

require 'open-uri'

**# get a list of incidents from the database**

toget=Incidents.all(:limit=>10) **#limit to 10 to test our your script first**

toget.each do |t|

the\_url="http://www.ndhealth.gov/EHS/FOIA/Spills/Summary\_Reports/"<<t.incident\_id.to\_s<<"\_summary\_report.pdf"

web\_contents = open(the\_url)

**#load contents into a variable called reader**

reader = PDF::Reader.new(web\_contents)

the\_text=""

reader.pages.each do |page|

the\_text << page.text

end

**#dump text into a table in the database**

n=FullText.new

n.incident\_id=t.incident\_id

n.the\_doc=the\_text

**#regex to pull out responsible party**

n.responsible\_party=the\_text.match(/**\b**Responsible Party:**\s**?(.+)/)[1].lstrip.rstrip

n.save!

puts t.incident\_id

end

You can do much the same in Python, another popular language used in newsrooms. Martin Burch of The Wall Street Journal [shared this code](https://github.com/wsjdata/clinton-email-cruncher/blob/master/pdfTextToDatabase.py) he recently authored to pull Hilary Clinton e-mails out of PDF and into a database.

#!/usr/bin/env python

# encoding: utf-8

"""

pdfTextToDatabase.py

1. Get a list of PDF names from the database

2. Open them

3. Extract text contents

4. Write contents back to the database

"""

from hrcemail\_common import \*

import string

import slate

import sys

def extract(filename):

try:

file = open('pdfs/'+filename+'.pdf', 'rb')

except:

return None;

try:

#this removes non-ASCII characters. Might not be desirable in some circumstances.

return filter(lambda x: x in string.printable,"\n".join(slate.PDF(file)))

except:

return None;

docIDs = Document.select(Document.docID)

for docID in docIDs:

filename = docID.docID

print "Working on",filename

output\_string = extract(filename)

insert\_query = Document.update(docText = output\_string).where(Document.docID == filename)

insert\_query.execute()

**Conclusion**: My objective here was not to show you THE correct way of wrangling data out of PDF files, but to demonstrate how there is no single solution. Knowing what’s possible is an excellent starting point, and becoming familiar with one or more of these tools will help you deal with whatever source data gets thrown your way.

Also, please note that the information I’m providing is current, to the best of my knowledge, as of September, 2015. Things change rapidly in this area – if you know of another tool, or think I should change or update one of my entries, please [let me know](mailto:rgebeloff@nytimes.com?subject=Your%20PDF%20Presentation).