# Extracting price data from PDF

February 2, 2025

## 1 Extracting MX spot prices from PDF

As part of their monthly reporting procedure,Master Data Management team extracts prices of raw materials by copy pasting them from the downloaded PDFs into their existing Excel files that store price data.

Prices for the current period are extracted,as well as the ones for 3 future periods.

To minimize the need of this manual data entry procedure,this script reads the data,indexes the required row and applies data transformation techniques to prepare the data for export to Excel.

## 2 Introduction

This notebook is divided into several chapters: Data Loading,Data Preparation and Writing Data to an Excel workbook.The goal of the first one is to read the data from the PDF and select only the rows required.

In the Data Preparation section,using date functions,the headers will be changed to reflect the current and future periods. As a last step,the extracted and transformed data is written to the respective Excel workbook.

### 2.1 Data loading

#### 2.1.1 Import required libraries

[31]:
```
from tabula.io import read_pdf
import pandas as pd
import numpy as np
import openpyxl
import datetime
from dateutil import relativedelta
import os
import calendar
import datetime
```

### 2.1.2 Read the data

[32]:
```
df = read_pdf("C:\\Users\\user\\Downloads\\Argus Toluene and Xylenes Outlook
↪(2024-06-26).pdf", pages="all")
```

[33]:
```
df[0]
```

[33]:

| | US | Unnamed: 0 | Unnamed: 1 | Unnamed: 2 | Unnamed: 3 |
|---|---|---|---|---|---|
| 0 | NaN | Jun 17 | Jul 17 | Aug 17 | Sep 17 |
| 1 | NaN | NaN | NaN | NaN | NaN |
| 2 | Tol spot ¢/USG | 202 | 193 | 191 | 199 |
| 3 | Tol spot $/t | 614 | 586 | 580 | 605 |
| 4 | MX spot ¢/USG | 209 | 203 | 201 | 214 |
| 5 | MX spot $/t | 636 | 619 | 611 | 653 |
| 6 | OX contract ¢/lb | 39 | 38 | 37 | 37 |
| 7 | PX contract ¢/lb | 41 | 39 | 39 | 40 |
| 8 | PTA contract ¢/lb | 42 | 41 | 41 | 42 |
| 9 | MEG contract ¢/lb | 42 | 41 | 36 | 35 |
| 10 | PET resin contract ¢/lb | 72 | 69 | 67 | 65 |
| 11 | NaN | NaN | NaN | NaN | NaN |
| 12 | Europe | NaN | NaN | NaN | NaN |
| 13 | NaN | Jun 17 | Jul 17 | Aug 17 | Sep 17 |
| 14 | NaN | NaN | NaN | NaN | NaN |
| 15 | Tol contract $/t | 638 | 575 | 543 | 553 |
| 16 | Tol spot $/t | 604 | 546 | 540 | 565 |
| 17 | MX spot $/t | 576 | 569 | 561 | 583 |
| 18 | OX contract fob €/t | 805 | 805 | 805 | 805 |
| 19 | PX contract fob €/t | 760 | 746 | 743 | 743 |
| 20 | PX spot $/t | 718 | 702 | 700 | 721 |
| 21 | PTA €/t | 650 | 641 | 639 | 639 |
| 22 | MEG €/t | 859 | 822 | 728 | 719 |
| 23 | PET resin €/t | 1,136 | 1,115 | 1,076 | 1,072 |
| 24 | NaN | NaN | NaN | NaN | NaN |
| 25 | Asia-Pacific | NaN | NaN | NaN | NaN |
| 26 | NaN | Jun 17 | Jul 17 | Aug 17 | Sep 17 |
| 27 | NaN | NaN | NaN | NaN | NaN |
| 28 | Tol spot $/t | 607 | 606 | 608 | 614 |
| 29 | MX spot $/t | 625 | 606 | 603 | 619 |
| 30 | OX spot $/t | 765 | 746 | 743 | 749 |
| 31 | PX spot $/t | 785 | 771 | 773 | 774 |
| 32 | PTA spot $/t | 608 | 599 | 595 | 596 |
| 33 | MEG contract $/t | 850 | 850 | 730 | 720 |
| 34 | PET resin $/t | 900 | 911 | 840 | 819 |
| 35 | NaN | NaN | NaN | NaN | NaN |
| 36 | Market assumptions | NaN | NaN | NaN | NaN |
| 37 | NaN | Jun 17 | Jul 17 | Aug 17 | Sep 17 |
| 38 | NaN | NaN | NaN | NaN | NaN |
| 39 | Crude WTI $/bl | 47 | 48 | 49 | 49 |

| | | | | | |
|---|---|---|---|---|---|
| 40 | North Sea Dated $/bl | 49 | 50 | 51 | 51 |
| 41 | Dubai $/bl | 48 | 49 | 49 | 50 |
| 42 | Gasoline 87 USGC ¢/USG | 148 | 143 | 142 | 141 |
| 43 | Naphtha ARA $/t | 407 | 410 | 414 | 418 |
| 44 | Naphtha cfr Japan $/t | 412 | 416 | 423 | 429 |

| | Unnamed: 4 | Unnamed: 5 | Unnamed: 6 | Unnamed: 7 | Unnamed: 8 | Unnamed: 9 |
|---|---|---|---|---|---|---|
| 0 | Oct 17 | Nov 17 | Dec 17 | Jan 18 | Feb 18 | Mar 18 |
| 1 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 202 | 210 | 213 | 206 | 210 | 201 |
| 3 | 614 | 638 | 648 | 624 | 638 | 612 |
| 4 | 215 | 221 | 217 | 210 | 213 | 202 |
| 5 | 656 | 674 | 661 | 641 | 650 | 616 |
| 6 | 39 | 39 | 38 | 37 | 36 | 36 |
| 7 | 40 | 42 | 41 | 40 | 40 | 39 |
| 8 | 42 | 42 | 42 | 41 | 41 | 41 |
| 9 | 35 | 35 | 34 | 34 | 34 | 33 |
| 10 | 64 | 64 | 63 | 64 | 63 | 63 |
| 11 | NaN | NaN | NaN | NaN | NaN | NaN |
| 12 | NaN | NaN | NaN | NaN | NaN | NaN |
| 13 | Oct 17 | Nov 17 | Dec 17 | Jan 18 | Feb 18 | Mar 18 |
| 14 | NaN | NaN | NaN | NaN | NaN | NaN |
| 15 | 559 | 566 | 583 | 586 | 591 | 585 |
| 16 | 554 | 578 | 588 | 584 | 598 | 572 |
| 17 | 586 | 584 | 571 | 581 | 590 | 576 |
| 18 | 806 | 776 | 756 | 736 | 743 | 754 |
| 19 | 724 | 711 | 686 | 655 | 653 | 648 |
| 20 | 723 | 731 | 725 | 712 | 711 | 705 |
| 21 | 626 | 618 | 601 | 580 | 579 | 576 |
| 22 | 719 | 710 | 706 | 697 | 694 | 685 |
| 23 | 1,064 | 1,054 | 1,042 | 1,024 | 1,022 | 1,016 |
| 24 | NaN | NaN | NaN | NaN | NaN | NaN |
| 25 | NaN | NaN | NaN | NaN | NaN | NaN |
| 26 | Oct 17 | Nov 17 | Dec 17 | Jan 18 | Feb 18 | Mar 18 |
| 27 | NaN | NaN | NaN | NaN | NaN | NaN |
| 28 | 619 | 618 | 620 | 624 | 612 | 614 |
| 29 | 634 | 623 | 610 | 609 | 587 | 579 |
| 30 | 754 | 733 | 730 | 719 | 687 | 669 |
| 31 | 774 | 773 | 775 | 769 | 737 | 719 |
| 32 | 596 | 590 | 592 | 583 | 561 | 545 |
| 33 | 720 | 710 | 700 | 690 | 690 | 680 |
| 34 | 824 | 795 | 822 | 808 | 799 | 784 |
| 35 | NaN | NaN | NaN | NaN | NaN | NaN |
| 36 | NaN | NaN | NaN | NaN | NaN | NaN |
| 37 | Oct 17 | Nov 17 | Dec 17 | Jan 18 | Feb 18 | Mar 18 |
| 38 | NaN | NaN | NaN | NaN | NaN | NaN |
| 39 | 50 | 50 | 50 | 50 | 49 | 48 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 40 | 51 | 51 | 52 | 51 | 50 | 49 |
| 41 | 49 | 49 | 50 | 50 | 49 | 48 |
| 42 | 141 | 140 | 141 | 138 | 139 | 143 |
| 43 | 427 | 437 | 448 | 451 | 440 | 433 |
| 44 | 439 | 448 | 460 | 464 | 452 | 444 |

| | Unnamed: 10 | Unnamed: 11 | Unnamed: 12 |
|---|---|---|---|
| 0 | Apr 18 | May 18 | Jun 18 |
| 1 | NaN | NaN | NaN |
| 2 | 207 | 206 | 213 |
| 3 | 630 | 627 | 647 |
| 4 | 211 | 220 | 231 |
| 5 | 644 | 669 | 703 |
| 6 | 34 | 35 | 36 |
| 7 | 39 | 40 | 41 |
| 8 | 41 | 41 | 42 |
| 9 | 33 | 34 | 33 |
| 10 | 64 | 64 | 65 |
| 11 | NaN | NaN | NaN |
| 12 | NaN | NaN | NaN |
| 13 | Apr 18 | May 18 | Jun 18 |
| 14 | NaN | NaN | NaN |
| 15 | 581 | 588 | 597 |
| 16 | 590 | 587 | 607 |
| 17 | 594 | 599 | 633 |
| 18 | 752 | 773 | 793 |
| 19 | 654 | 668 | 678 |
| 20 | 712 | 728 | 739 |
| 21 | 580 | 589 | 596 |
| 22 | 680 | 689 | 680 |
| 23 | 1,017 | 1,027 | 1,027 |
| 24 | NaN | NaN | NaN |
| 25 | NaN | NaN | NaN |
| 26 | Apr 18 | May 18 | Jun 18 |
| 27 | NaN | NaN | NaN |
| 28 | 605 | 621 | 629 |
| 29 | 570 | 581 | 574 |
| 30 | 660 | 671 | 674 |
| 31 | 705 | 711 | 699 |
| 32 | 540 | 549 | 541 |
| 33 | 680 | 690 | 680 |
| 34 | 803 | 802 | 784 |
| 35 | NaN | NaN | NaN |
| 36 | NaN | NaN | NaN |
| 37 | Apr 18 | May 18 | Jun 18 |
| 38 | NaN | NaN | NaN |
| 39 | 49 | 50 | 52 |

```
40          49          51          52
41          48          49          51
42         148         152         158
43         424         429         437
44         435         441         449
```

### 2.1.3 Select the required row of data

[34]:
```
df_new=df[0][26:30].iloc[3:]
```

### 2.1.4 Display data

[35]:
```
df_new
```

[35]:
```
              US Unnamed: 0 Unnamed: 1 Unnamed: 2 Unnamed: 3 Unnamed: 4  \
29  MX spot $/t         625         606         603         619         634

   Unnamed: 5 Unnamed: 6 Unnamed: 7 Unnamed: 8 Unnamed: 9 Unnamed: 10  \
29         623         610         609         587         579         570

   Unnamed: 11 Unnamed: 12
29         581         574
```

## 2.2 Data preparation

The goal of the Data reparation module is to prepare the Actual and Forecast prices for export the Excel. That is achieved by extracting the current and forecast month and year and concatenating them.Afterwards,the header names are changed respectively.

### 2.2.1 Get month and year (actual and forecast

[36]:
```
month =datetime.date.today()-relativedelta.relativedelta(months=1)
year = str(datetime.date.today().year)
strMonth=str(month.month)
```

[37]:
```
currentdate= "-".join([strMonth,year])
```

[38]:
```
currentdate
```

[38]:
```
'5-2024'
```

[39]:
```
nextmonth1 = datetime.date.today()
```

[40]:
```
nextmonth=str(nextmonth1.month)
```

[41]:
```
nextmonthyear=str(nextmonth1.year)
```

[42]:
```
concatenatedcurrent="-".join([nextmonth, nextmonthyear])
```

[43]:
```
secondmonth1=datetime.date.today()+ relativedelta.relativedelta(months=1)
```

[44]:
```
secondmonth=str(secondmonth1.month)
```

[45]:
```
secondmonthyear=str(secondmonth1.year)
```

[46]:
```
concatenatednext="-".join([secondmonth, secondmonthyear])
```

[47]:
```
thirdmonth1 =datetime.date.today()+ relativedelta.relativedelta(months=2)
```

[48]:
```
thirdmonth=str(thirdmonth1.month)
```

[49]:
```
thirdmonthyear=str(thirdmonth1.year)
```

[50]:
```
concatenatedlast="-".join([thirdmonth, thirdmonthyear])
```

[51]:
```
concatenatedcurrent
```

[51]:
```
'6-2024'
```

### 2.3 Change header names

[52]:
```
dict = {'US':'Product',
    'Unnamed: 0':currentdate,
    'Unnamed: 1':concatenatedcurrent,
        'Unnamed: 2':concatenatednext,
        'Unnamed: 3':concatenatedlast}

# call rename () method
df_new.rename(columns=dict,
          inplace=True)

# print Data frame after rename columns
display(df_new)
```

```
        Product 5-2024 6-2024 7-2024 8-2024 Unnamed: 4 Unnamed: 5 Unnamed: 6  \
29  MX spot $/t    625    606    603    619        634        623        610

   Unnamed: 7 Unnamed: 8 Unnamed: 9 Unnamed: 10 Unnamed: 11 Unnamed: 12
29        609        587        579         570         581         574
```

[53]:
```
df_final=df_new.melt(id_vars=['Product'], var_name='Date', value_name='Price').
↪iloc[0:4]
```

[54]:
```
df_current=df_final.iloc[:1]
```

[55]:
```
df_forecast=df_final.iloc[1:]
```

[56]:
```
df_forecast
```

[56]:

| | Product | Date | Price |
|---|---|---|---|
| 1 | MX spot $/t | 6-2024 | 606 |
| 2 | MX spot $/t | 7-2024 | 603 |
| 3 | MX spot $/t | 8-2024 | 619 |

[57]:
```
df_current
```

[57]:

| | Product | Date | Price |
|---|---|---|---|
| 0 | MX spot $/t | 5-2024 | 625 |

[58]:
```
df_current2=df_current.values.tolist()
```

### 2.3.1 Data writing to Excel workbook

[59]:
```
book = openpyxl.load_workbook("C:\\Users\\user\\Downloads\\toluene1.xlsx")
writer = pd.ExcelWriter("C:\\Users\\user\\Downloads\\toluene1.xlsx",␣
 ↪engine='openpyxl')
writer.book = book
writer.sheets = {ws.title: ws for ws in book.worksheets}

ws_current=book["Actual Toluene"]
for sheetname in writer.sheets:
    df_forecast.to_excel(writer,sheet_name="Forecast Toluene", index = False)
    print("Forecast prices written successfully to Excel file")

for i in range(len(df_current2)):
    ws_current.append(df_current2[i])
    print("Actual prices written successfully to Excel file")
```

```
Forecast prices written successfully to Excel file
Forecast prices written successfully to Excel file
Actual prices written successfully to Excel file
```

[60]:
```
writer.close()
```