Importing ARCOS Data with Dask

> *Bárbara Flores*

In [ ]:

```
import pandas as pd
import warnings
import os
from dask.distributed import Client
import dask.dataframe as dd

warnings.simplefilter(action="ignore", category=FutureWarning)
pd.set_option("mode.copy_on_write", True)
```

Last week, we used dask to play with a few datasets to get a feel for how dask works. In order to help us develop code that would run quickly, however, we worked with very small, safe datasets.

Today, we will continue to work with dask, but this time using much larger datasets. This means that (a) doing things incorrectly may lead to your computer crashing (So save all your open files before you start!), and (b) many of the commands you are being asked run will take several minutes each.

For familiarity, and so you can see what advantages dask can bring to your workflow, today we'll be working with the DEA ARCOS drug shipment database published by the Washington Post! However, to strike a balance between size and speed, we'll be working with a slightly thinned version that has only the last two years of data, instead of all six.

Exercise 1

Download the thinned ARCOS data from this link. It should be about 2GB zipped, 25 GB unzipped.

Exercise 2

Our goal today is going to be to find the pharmaceutical company that has shipped the most opioids (`MME_Conversion_Factor * CALC_BASE_WT_IN_GM`) in the US.

When working with large datasets, it is good practice to begin by prototyping your code with a subset of your data. So begin by using `pandas` to read in the first 100,000 lines of the ARCOS data and write pandas code to compute the shipments from each shipper (the group that reported the shipment).

In [ ]:

```python
path = "../../arcos_2011_2012.tsv"
arcos_2011_2012_subset = pd.read_csv(path, sep="\t", nrows=100000)
arcos_2011_2012_subset.head()
```

```
/var/folders/9m/ym86jvl93wq3tx6ssy8ql7kh0000gn/T/ipykernel_22746/323010964.p
y:2: DtypeWarning: Columns (4,6,27) have mixed types. Specify dtype option o
n import or set low_memory=False.
  arcos_2011_2012_subset = pd.read_csv(path, sep="\t", nrows=100000)
```

Out[ ]:

	Unnamed: 0	REPORTER_DEA_NO	REPORTER_BUS_ACT	REPORTER_NAME	REPORTE
0	0	PA0006836	DISTRIBUTOR	ACE SURGICAL SUPPLY CO INC	
1	9	PA0021179	DISTRIBUTOR	APOTHECA INC	
2	10	PA0021179	DISTRIBUTOR	APOTHECA INC	
3	16	PA0021179	DISTRIBUTOR	APOTHECA INC	
4	17	PA0021179	DISTRIBUTOR	APOTHECA INC	

5 rows × 45 columns

> *Then, we group by REPORTER_DEA_NO and REPORTER_NAME. For example, we can see that the reporter 'MCKESSON CORPORATION' has 3 unique IDs of entities reporting shipments to DEA: PM0000771, PF0000012, PM0003094*
>
> *We can observe the number of shipments for our 100,000-row sample for each reporter during the 2011-2012 period from various reporters in the following table.*

In [ ]:

```python
arcos_2011_2012_subset["Total_Shipments"] = (
    arcos_2011_2012_subset["MME_Conversion_Factor"]
    * arcos_2011_2012_subset["CALC_BASE_WT_IN_GM"]
)
```

In [ ]:

```python
arcos_2011_2012_subset.groupby(["REPORTER_DEA_NO", "REPORTER_NAME"])[
    "Total_Shipments"
].sum().sort_values(ascending=False).reset_index(name="Total_Shipments")
```

Out[ ]:

	REPORTER_DEA_NO	REPORTER_NAME	Total_Shipments
0	PM0000771	MCKESSON CORPORATION	91928.451192
1	PF0000012	MCKESSON CORPORATION	64118.325379
2	PC0003044	CARDINAL HEALTH 110, LLC	54352.323711
3	PD0029567	MCKESSON CORPORATION	47680.418755
4	PL0032627	AMERISOURCEBERGEN DRUG CORP	34561.394892
5	PM0003094	MCKESSON CORPORATION	34332.441204
6	PM0001951	MCKESSON CORPORATION	30383.924484
7	PK0070297	KINRAY INC	28620.315246
8	PM0018425	MCKESSON CORPORATION	23845.207505
9	PL0184933	LOUISIANA WHOLESALE DRUG CO	14787.765559
10	PF0031120	FRANK W KERR INC	8730.016283
11	PM0001014	MCKESSON CORPORATION	6977.562707
12	PH0035964	H D SMITH WHOLESALE DRUG CO	6399.324050
13	PK0132706	KAISER FOUNDATION HOSPITALS	3891.329580
14	PB0020139	BURLINGTON DRUG COMPANY	3889.490325
15	PG0149650	AMERICAN SALES COMPANY	3432.058005
16	PD0058063	DIK DRUG CO	3278.514405
17	PB0020052	KPH HEALTHCARE SERVICES, INC.	1988.890350
18	PB0167127	BLOODWORTH WHOLESALE DRUGS	1782.827325
19	PD0203377	DISCOUNT DRUG MART	1272.596205
20	PK0132821	KAISER FNDTN HEALTH PLAN NW	1159.892040
21	PH0234928	H J HARKINS COMPANY INC	352.393956
22	PC0049507	CAPITAL WHOLESALE DRUG & CO	188.318175
23	PA0021179	APOTHECA INC	23.913300
24	PH0187117	HALS MED DENT SUPPLY CO., INC.	18.162000
25	PC0044696	CESAR CASTILLO INC	3.027000
26	PM0005721	MERRITT VETERINARY SUPPLIES INC	1.816200
27	PA0006836	ACE SURGICAL SUPPLY CO INC	0.605400

Exercise 3

Now let's turn to dask. Re-write your code for dask, and calculate the total shipments by reporting company. Remember:

- Activate a conda environment with a clean dask installation.
- Start by spinning up a distributed cluster.
- Dask won't read compressed files, so you have to unzip your ARCOS data.
- Start your cluster in a cell all by itself since you don't want to keep re-running the "start a cluster" code.

If you need to review dask basic code, check here.

As you run your code, make sure to click on the Dashboard link below where you created your cluster:

dask_dashboard

Among other things, the bar across the bottom should give you a sense of how long your task will take:

dask_progress

(For context, my computer (which has 10 cores) only took a couple seconds. My computer is fast, but most computers should be done within a couple minutes, tops).

In [ ]:

```
# client = Client()
# client
```

In [ ]:

```
arcos_2011_2012_dask = dd.read_csv(
    path,
    sep="\t",
    dtype={
        "Unnamed: 0": "str",
        "REPORTER_DEA_NO": "str",
        "REPORTER_BUS_ACT": "str",
        "REPORTER_NAME": "str",
        "REPORTER_ADDL_CO_INFO": "str",
        "REPORTER_ADDRESS1": "str",
        "REPORTER_ADDRESS2": "str",
        "REPORTER_CITY": "str",
        "REPORTER_STATE": "str",
        "REPORTER_ZIP": "str",
        "REPORTER_COUNTY": "str",
        "BUYER_DEA_NO": "str",
        "BUYER_BUS_ACT": "str",
        "BUYER_NAME": "str",
        "BUYER_ADDL_CO_INFO": "str",
        "BUYER_ADDRESS1": "str",
        "BUYER_ADDRESS2": "str",
        "BUYER_CITY": "str",
        "BUYER_STATE": "str",
        "BUYER_ZIP": "str",
        "BUYER_COUNTY": "str",
        "TRANSACTION_CODE": "str",
        "DRUG_CODE": "str",
        "NDC_NO": "str",
        "DRUG_NAME": "str",
        "QUANTITY": "str",
        "UNIT": "str",
        "ACTION_INDICATOR": "str",
        "ORDER_FORM_NO": "str",
        "CORRECTION_NO": "str",
        "STRENGTH": "str",
        "TRANSACTION_DATE": "str",
        "CALC_BASE_WT_IN_GM": "float",
        "DOSAGE_UNIT": "str",
        "TRANSACTION_ID": "str",
        "Product_Name": "str",
        "Ingredient_Name": "str",
        "Measure": "str",
        "MME_Conversion_Factor": "float64",
        "Combined_Labeler_Name": "str",
        "Revised_Company_Name": "str",
        "Reporter_family": "str",
        "dos_str": "str",
        "date": "str",
        "year": "str",
    },
)
```

In [ ]:

```
arcos_2011_2012_dask.head()
```

Out[ ]:

	Unnamed: 0	REPORTER_DEA_NO	REPORTER_BUS_ACT	REPORTER_NAME	REPORTE
0	0	PA0006836	DISTRIBUTOR	ACE SURGICAL SUPPLY CO INC	
1	9	PA0021179	DISTRIBUTOR	APOTHECA INC	
2	10	PA0021179	DISTRIBUTOR	APOTHECA INC	
3	16	PA0021179	DISTRIBUTOR	APOTHECA INC	
4	17	PA0021179	DISTRIBUTOR	APOTHECA INC	

5 rows × 45 columns

In [ ]:

```python
arcos_2011_2012_dask = arcos_2011_2012_dask[
    [
        "REPORTER_DEA_NO",
        "REPORTER_NAME",
        "BUYER_STATE",
        "BUYER_COUNTY",
        "MME_Conversion_Factor",
        "CALC_BASE_WT_IN_GM",
        "year",
    ]
]
```

> *Now, we perform the same operation as before, but on our Dask DataFrame to see how many shipments we have per reporter.*

In [ ]:

```python
arcos_2011_2012_dask["Total_Shipments"] = (
    arcos_2011_2012_dask["MME_Conversion_Factor"]
    * arcos_2011_2012_dask["CALC_BASE_WT_IN_GM"]
)

arcos_2011_2012_dask.groupby(["REPORTER_DEA_NO", "REPORTER_NAME"])[
    "Total_Shipments"
].sum().compute().reset_index(name="Total_Shipments").sort_values(
    by="Total_Shipments", ascending=False
)
```

Out[ ]:

	REPORTER_DEA_NO	REPORTER_NAME	Total_Shipments
236	RW0294493	WALGREEN CO	1.991478e+06
120	RW0277752	WALGREEN CO	1.247551e+06
105	RW0204026	WALGREEN CO	1.211052e+06
22	PM0000771	MCKESSON CORPORATION	8.440511e+05
187	RO0153609	CARDINAL HEALTH	7.499573e+05
...	...	...	...
401	RV0315956	VETESSA PHARMACEUTICAL, INC	3.027000e-01
391	RI0236681	IVESCO, LLC	3.027000e-01
405	RK0202123	KING PHARMACEUTICALS	3.027000e-01
385	RV0357675	VET PHARM, INC.	3.027000e-01
398	RR0350126	REMEDYREPACK	1.816200e-01

409 rows × 3 columns

Exercise 4

Now let's calculate, *for each state*, what company shipped the most pills?

Note you will quickly find that you can't sort in dask -- sorting in parallel is *really* tricky! So you'll have to work around that. Do what you need to do on the big dataset first, then compute it all so you get it as a regular pandas dataframe, then finish.

In [ ]:

```
grouped_table = (
    arcos_2011_2012_dask.groupby(["REPORTER_DEA_NO", "REPORTER_NAME", "BUYER
        "Total_Shipments"
    ]
    .sum()
    .compute()
    .reset_index(name="Total_Shipments")
    .sort_values(by="Total_Shipments", ascending=False)
)

grouped_table.reset_index()
```

Out[ ]:

	index	REPORTER_DEA_NO	REPORTER_NAME	BUYER_STATE	Total_Shipment
0	967	RW0277752	WALGREEN CO	FL	885224.01956
1	121	PM0000771	MCKESSON CORPORATION	FL	843970.58874
2	2271	RC0182080	CARDINAL HEALTH	FL	531054.22774
3	59	PF0000012	MCKESSON CORPORATION	CA	432201.14897
4	1517	RO0153609	CARDINAL HEALTH	OH	411709.92607
...	...	...	...	...	.
3966	3579	RH0302567	HAWTHORN PHARMACEUTICALS INC	OR	0.06054
3967	209	RC0231148	ALTURA PHARMACEUTICALS, INC	TN	0.06054
3968	3472	RH0302567	HAWTHORN PHARMACEUTICALS INC	HI	0.06054
3969	3292	RH0302567	HAWTHORN PHARMACEUTICALS INC	NV	0.04540
3970	3475	RH0319663	HSB VETERINARY SUPPLIES INC	LA	0.01513

3971 rows × 5 columns

> *In order to carry out the operation, we first had to perform the computation in Dask and then apply the missing "group by" operations.*
>
> *Reviewing the results, we can see that there are several companies involved, but the main one appears to be MCKESSON CORPORATION.*

In [ ]:

```
max_indices = grouped_table.groupby("BUYER_STATE")["Total_Shipments"].idxmax
grouped_table.loc[max_indices][
    ["BUYER_STATE", "REPORTER_NAME", "REPORTER_DEA_NO", "Total_Shipments"]
]
```

Out[ ]:

	BUYER_STATE	REPORTER_NAME	REPORTER_DEA_NO	Total_Shipments
852	AK	CARDINAL HEALTH	RW0191813	19447.768327
1430	AL	MCKESSON CORPORATION	RM0336950	313080.933988
2697	AR	AMERISOURCEBERGEN DRUG CORPORATION	RA0316958	83172.511420
166	AZ	MCKESSON CORPORATION	PM0021131	343507.894122
59	CA	MCKESSON CORPORATION	PF0000012	432201.148970
153	CO	MCKESSON CORPORATION	PM0018425	181587.285809
265	CT	CARDINAL HEALTH	RD0108200	162078.687446
960	DC	CARDINAL HEALTH	RW0269654	22902.037314
1950	DE	WALGREEN CO	RW0294493	82099.678748
967	FL	WALGREEN CO	RW0277752	885224.019566
886	GA	MCKESSON CORPORATION	PR0040357	208514.646666
1315	GU	AMERISOURCEBERGEN DRUG CORP	RA0289048	1370.742129
124	HI	MCKESSON CORPORATION	PM0001014	62476.302371
2711	IA	AMERISOURCEBERGEN DRUG	RA0326276	38767.267612
187	ID	MCKESSON CORPORATION	PM0023046	38637.707998
1040	IL	WALGREEN CO	PW0211158	112614.722070
934	IN	CARDINAL HEALTH	RW0231908	247749.675133
2713	KS	AMERISOURCEBERGEN DRUG	RA0326276	44533.007987
3059	KY	AMERISOURCEBERGEN DRUG CORP	RA0314562	110778.613941
117	LA	LOUISIANA WHOLESALE DRUG CO	PL0184933	117206.957966
266	MA	CARDINAL HEALTH	RD0108200	288906.803174
48	MD	MCKESSON CORPORATION	PD0029567	194954.586093
267	ME	CARDINAL HEALTH	RD0108200	90982.957180
194	MI	MCKESSON	PM0030849	290038.591542

	BUYER_STATE	REPORTER_NAME	REPORTER_DEA_NO	Total_Shipments
		CORPORATION		
2845	MN	MCKESSON DRUG COMPANY	PM0036334	100612.980509
2855	MO	MCKESSON CORPORATION	PM0037374	180586.620517
125	MP	MCKESSON CORPORATION	PM0001014	319.393875
1186	MS	AMERISOURCEBERGEN DRUG CORP	RA0223432	62268.751039
188	MT	MCKESSON CORPORATION	PM0023046	39406.049010
947	NC	CARDINAL HEALTH	RW0243903	364795.440762
2847	ND	MCKESSON DRUG COMPANY	PM0036334	14582.671790
2861	NE	MCKESSON CORPORATION	PM0038693	35480.452780
162	NH	MCKESSON CORPORATION	PM0020850	70450.564911
963	NJ	CARDINAL HEALTH	RW0269654	251436.342877
882	NM	WALGREEN CO	RW0204026	73520.483175
953	NV	CARDINAL HEALTH	RW0263056	172070.019210
2914	NY	CARDINAL HEALTH 110, LLC	RK0416900	291145.357435
1517	OH	CARDINAL HEALTH	RO0153609	411709.926079
2956	OK	MCKESSON CORPORATION	RM0138328	201301.607036
184	OR	MCKESSON CORPORATION	PM0022929	212413.796470
2472	PA	MCKESSON CORPORATION	RM0173055	308694.402288
430	PR	DROGUERIA BETANCES, LLC	RD0369947	15052.314020
269	RI	CARDINAL HEALTH	RD0108200	62522.927694
888	SC	MCKESSON CORPORATION	PR0040357	210851.068656
2862	SD	MCKESSON CORPORATION	PM0038693	17698.590515
215	TN	CARDINAL HEALTH	RC0238104	249234.692995
3022	TX	WALGREEN CO	RW0281953	219753.631410

	BUYER_STATE	REPORTER_NAME	REPORTER_DEA_NO	Total_Shipments
1324	UT	AMERISOURCEBERGEN DRUG CORP	RA0289098	108870.239304
949	VA	CARDINAL HEALTH	RW0243903	146810.590605
2211	VI	CARDINAL HEALTH P.R. 120, INC.	RB0374683	610.669425
165	VT	MCKESSON CORPORATION	PM0020850	31049.274965
2872	WA	MCKESSON CORPORATION	PM0150538	157014.588206
1980	WI	WALGREEN CO	RW0294493	168164.427102
1520	WV	CARDINAL HEALTH	RO0153609	105784.410141
159	WY	MCKESSON CORPORATION	PM0018425	14779.469197

Exercise 5

Now go ahead and try and re-do the chunking you did by hand for your project (with this 2 years of data) -- calculate, for each year, the total morphine equivalents sent to each county in the US.

In [ ]:

```
arcos_2011_2012_dask.groupby(["year", "BUYER_STATE", "BUYER_COUNTY"])[
    "Total_Shipments"
].sum().compute().reset_index(name="Total_Shipments").sort_values(
    by="Total_Shipments", ascending=False
)
```

Out[ ]:

	year	BUYER_STATE	BUYER_COUNTY	Total_Shipments
1367	2012	AZ	MARICOPA	337023.243956
71	2011	AZ	MARICOPA	314707.706697
96	2011	CA	LOS ANGELES	283866.848898
1392	2012	CA	LOS ANGELES	278747.192600
224	2011	FL	HILLSBOROUGH	233352.040536
...	...	...	...	...
6140	2011	SD	DEWEY	0.302700
6148	2012	MT	CARTER	0.302700
6145	2011	TX	THROCKMORTON	0.181620
6146	2012	TX	SCHLEICHER	0.121080
6099	2011	OR	WHEELER	0.072648

6149 rows × 4 columns

Exercise 6

Now, re-write your opioid project's initial opioid import using dask. Each person on your team should create a NEW branch to try this. The person who wrote the initial chunking code can help everyone else understand what they did originally and the data, but everyone should write their own code.

WARNING: You will probably run into a lot of type errors (depending on how the ARCOS data has changed since last year). With real world messy data one of the biggest problems with dask is that it struggles if halfway through dataset it discovers that the column it *thought* was floats contains text. That's why, in the dask reading, I specified the column type for so many columns as `objects` explicitly. Then, because occasionally there data cleanliness issues, I had to do some converting data types by hand.

> *Let's first try using pandas.*

In [ ]:

```
file_path = "../../arcos_all_washpost.tsv"
arcos_all_washpost = pd.read_csv(file_path, sep="\t", nrows=100)
arcos_all_washpost.head()
```

Out[ ]:

	REPORTER_DEA_NO	REPORTER_BUS_ACT	REPORTER_NAME	REPORTER_ADDL_CC
0	RM0220688	DISTRIBUTOR	MCKESSON CORPORATION	
1	RM0220688	DISTRIBUTOR	MCKESSON CORPORATION	
2	RM0220688	DISTRIBUTOR	MCKESSON CORPORATION	
3	RM0220688	DISTRIBUTOR	MCKESSON CORPORATION	
4	RM0220688	DISTRIBUTOR	MCKESSON CORPORATION	

5 rows × 33 columns

> *Now let's try with Dask*

In [ ]:

```
arcos_all_washpost_dask = dd.read_csv(
    file_path,
    sep="\t",
    dtype={
        "REPORTER_DEA_NO": "str",
        "REPORTER_BUS_ACT": "str",
        "REPORTER_NAME": "str",
        "REPORTER_ADDL_CO_INFO": "str",
        "REPORTER_ADDRESS1": "str",
        "REPORTER_ADDRESS2": "str",
        "REPORTER_CITY": "str",
        "REPORTER_STATE": "str",
        "REPORTER_ZIP": "str",
        "REPORTER_COUNTY": "str",
        "BUYER_DEA_NO": "str",
        "BUYER_BUS_ACT": "str",
        "BUYER_NAME": "str",
        "BUYER_ADDL_CO_INFO": "str",
        "BUYER_ADDRESS1": "str",
        "BUYER_ADDRESS2": "str",
        "BUYER_CITY": "str",
        "BUYER_STATE": "str",
        "BUYER_ZIP": "str",
        "BUYER_COUNTY": "str",
        "TRANSACTION_CODE": "str",
        "DRUG_CODE": "str",
        "NDC_NO": "str",
        "DRUG_NAME": "str",
        "Measure": "str",
        "MME_Conversion_Factor": "float64",
        "Dosage_Strength": "str",
        "TRANSACTION_DATE": "str",
        "Combined_Labeler_Name": "str",
        "Reporter_family": "str",
        "CALC_BASE_WT_IN_GM": "float64",
        "DOSAGE_UNIT": "str",
        "MME": "float64",
    },
)
```

In [ ]:

```
arcos_all_washpost_dask = arcos_all_washpost_dask[
    [
        "REPORTER_DEA_NO",
        "REPORTER_NAME",
        "BUYER_STATE",
        "BUYER_COUNTY",
        "MME_Conversion_Factor",
        "CALC_BASE_WT_IN_GM",
        "TRANSACTION_DATE",
        "MME",
    ]
]
```

In [ ]:

```
arcos_all_washpost_dask["Total_Shipments"] = (
    arcos_all_washpost_dask["MME_Conversion_Factor"]
    * arcos_all_washpost_dask["CALC_BASE_WT_IN_GM"]
)

arcos_all_washpost_dask["year_month"] = arcos_all_washpost_dask["TRANSACTION
    0:7
]
```

In [ ]:

```
groupped_arcos_all_washpost_dask = (
    arcos_all_washpost_dask.groupby(["year_month", "BUYER_STATE", "BUYER_COU
        "Total_Shipments", "MME"
    ]
    .sum()
    .compute()
)
```

In [ ]:

```
groupped_arcos_all_washpost_dask = groupped_arcos_all_washpost_dask.reset_ir
groupped_arcos_all_washpost_dask.head()
```

Out[ ]:

	year_month	BUYER_STATE	BUYER_COUNTY	Total_Shipments	MME
0	2006-01	IN	ADAMS	190.651252	1.906513e+05
1	2006-01	IN	ALLEN	2515.866036	2.515866e+06
2	2006-01	IN	BARTHOLOMEW	1441.961333	1.441961e+06
3	2006-01	IN	BOONE	487.815865	4.878159e+05
4	2006-01	IN	CASS	247.482270	2.474823e+05

In [ ]:

```
groupped_arcos_all_washpost_dask.to_parquet(
    "../20_intermediate_files/arcos_all_washpost_collapsed.parquet"
)
```

> *Finally, the branch with this code is located in:*
>
> *https://github.com/MIDS-at-Duke/IDS720_PracticalDataScience_JBR/blob/PDS_assigment_barbara/10_code/20_lc*