MATLAB in HPC

Large datasets.

Anders Sjöström LUNARC
anders.sjostrom@lunarc.lu.se

Datastore

- A datastore is an object for reading a single file or a collection of files or data.
- A datastore can be created based on the type of data or application.
- Different types of datastores contain properties relevant to the type of data that they support.

```
A complete list of datastores can be found here:
https://se.mathworks.com/help/matlab/import_export/select-
datastore-for-file-format-or-application.html
```

Datastore

- The datastore acts as a repository for data that has the same structure and formatting.
- Each file in a datastore must contain data of the same type.
- Data must appear in the same order.
- Data must be separated by the same delimiter.

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount
2	1	2019-01-01 00:46	2019-01-01 00:53	1	1.50	1	N	151	239	1	7	0.5	0.5	1.65
3	1	2019-01-01 00:59	2019-01-01 01:18	1	2.60	1	N	239	246	1	14	0.5	0.5	1
4	2	2018-12-21 13:48	2018-12-21 13:52	3	.00	1	N	236	236	1	4.5	0.5	0.5	0
5	2	2018-11-28 15:52	2018-11-28 15:55	5	.00	1	N	193	193	2	3.5	0.5	0.5	0
6	2	2018-11-28 15:56	2018-11-28 15:58	5	.00	2	N	193	193	2	52	0	0.5	0
7	2	2018-11-28 16:25	2018-11-28 16:28	5	.00	1	N	193	193	2	3.5	0.5	0.5	0
8	2	2018-11-28 16:29	2018-11-28 16:33	5	.00	2	N	193	193	2	52	0	0.5	0
9	1	2019-01-01 00:21	2019-01-01 00:28	1	1.30	1	N	163	229	1	6.5	0.5	0.5	1.25
10	1	2019-01-01 00:32	2019-01-01 00:45	1	3.70	1	N	229	7	1	13.5	0.5	0.5	3.7
11	1	2019-01-01 00:59	2019-01-01 01:18	1	2.60	1	N	239	246	1	14	0.5	0.5	1
12	2	2018-12-21 13:48	2018-12-21 13:52	3	.00	1	N	236	236	1	4.5	0.5	0.5	0
13	2	2018-11-28 15:52	2018-11-28 15:55	5	.00	1	N	193	193	2	3.5	0.5	0.5	0
14	2	2018-11-28 15:56	2018-11-28 15:58	5	.00	2	N	193	193	2	52	0	0.5	0
15	2	2018-11-28 16:25	2018-11-28 16:28	5	.00	1	N	193	193	2	3.5	0.5	0.5	0
16	2	2018-11-28 16:29	2018-11-28 16:33	5	.00	2	N	193	193	2	52	0	0.5	0
17	1	2019-01-01 00:21	2019-01-01 00:28	1	1.30	1	N	163	229	1	6.5	0.5	0.5	1.25
18	1	2019-01-01 00:32	2019-01-01 00:45	1	3.70	1	N	229	7	1	13.5	0.5	0.5	3.7
19	1	2019-01-01 00:59	2019-01-01 01:18	1	2.60	1	N	239	246	1	14	0.5	0.5	1
20	2	2018-12-21 13:48	2018-12-21 13:52	3	.00	1	N	236	236	1	4.5	0.5	0.5	0
21	2	2018-11-28 15:52	2018-11-28 15:55	5	.00	1	N	193	193	2	3.5	0.5	0.5	0
22	2	2018-11-28 15:56	2018-11-28 15:58	5	.00	2	N	193	193	2	52	0	0.5	0
23	2	2018-11-28 16:25	2018-11-28 16:28	5	.00	1	N	193	193	2	3.5	0.5	0.5	0
24	2	2018-11-28 16:29	2018-11-28 16:33	5	.00	2	N	193	193	2	52	0	0.5	0
25	1	2019-01-01 00:21	2019-01-01 00:28	1	1.30	1	N	163	229	1	6.5	0.5	0.5	1.25
26	1	2019-01-01 00:32	2019-01-01 00:45	1	3.70	1	N	229	7	1	13.5	0.5	0.5	3.7

Create a datastore

```
>> ds = tabularTextDatastore('yellow_tripdata_2019-02.csv')
ds =
  TabularTextDatastore with properties:
                    Files: {
 '/pfs/nobackup/home/h/hukebuck/taxiData/yellow_tripdata_2019-02.csv'
                        }
                                  :
                                  :
                                  :
  Properties that control the table returned by preview, read, readall:
      SelectedVariableNames: {'VendorID', 'tpep_pickup_datetime',
'tpep_dropoff_datetime' ... and 15 more}
            SelectedFormats: {'%f', '%{uuuu-MM-dd HH:mm:ss}D', '%{uuuu-MM-
dd HH:mm:ss}D' ... and 15 more}
                  ReadSize: 20000 rows
```

Show and select variables

```
>> ds.VariableNames

ans =

  1×18 cell array

  Columns 1 through 4

    {'VendorID'}
{'tpep_pickup_dat?'}
{'tpep_dropoff_da?'}
{'passenger_count'}
```

```
>>ds.SelectedVariableNames={'trip_distance','fare_amount'};
>> preview(ds)
ans =
  8×2 table
    trip_distance    fare_amount
    _____________    ___________

        2.1              9
        9.8             32
          0            2.5
        0.8            5.5
        0.8              5
        0.8            4.5
        0.9              5
        2.8             14
```

Sanitize variables

Handle missing data

```
>> ds.TreatAsMissing = 'NA';
```

If the data fits you can read it into memory

```
>> T = readall(ds);
```

If not, read the data in smaller subsets, using **read**. You can define a number of rows to read, default is 20000

```
>> ds.ReadSize = 12000;
```

Read and calculate

- The read function can be called within a while loop.
- Intermediate calculations can be performed on each subset of data, aggregating the intermediate results at the end.

```
>> reset(ds)
X = [ ];
while hasdata(ds)
      T = read(ds);
      X(end+1) = max(T.trip_distance);
end
maxDelay = max(X)

maxDelay =

  701.5000
```

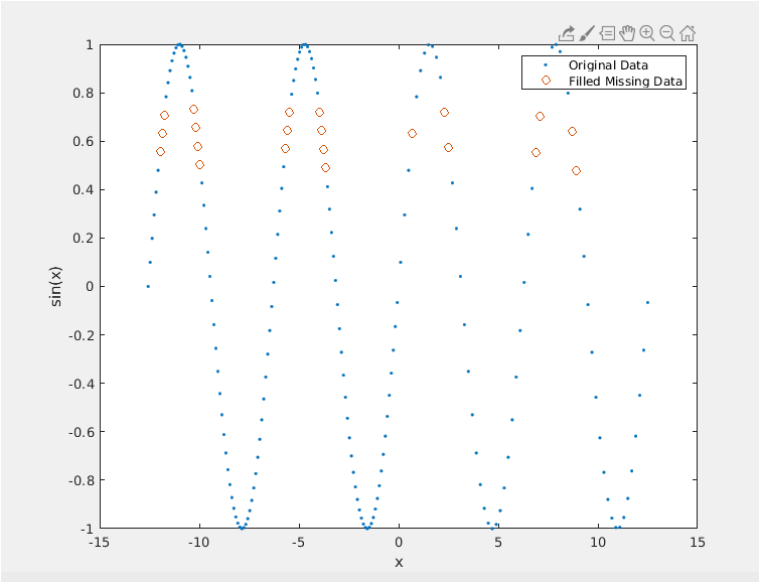
What about missing data?

```
>> A = [1 3 NaN 4 NaN NaN 5];
F = fillmissing(A,'previous')

F =

     1     3     3     4     4     4     5
```

```
>> x = [-4*pi:0.1:0, 0.1:0.2:4*pi];
A = sin(x);
>> A(A < 0.75 & A > 0.5) = NaN;
>> [F,TF] =
fillmissing(A,'linear','SamplePoints',x);
>> plot(x,A,'.', x(TF),F(TF),'o')
xlabel('x');
ylabel('sin(x)')
legend('Original Data','Filled Missing Data')
```



LUND
UNIVERSITY