Overview

Note that important datasets for this paper are not publicly available.

The code in this replication package constructs the figures and tables in the paper using Stata and Python. The replicator should have access to a large-scale cluster to run the code.

Data Availability and Provenance Statements

Statement about Rights

- I certify that the author(s) of the manuscript have legitimate access to and permission to use the data used in this manuscript.

Summary of Availability

- Key components of the data are **not** publicly available. For any publicly available data, I include the data files in the replication package for convenience.

Details on each Data Source

Dataset name	Files
1. Rig status updates **(proprietary/not publicly available)**	`data_py/raw/contracts/dayrates_rigzone_new.xls`
2. Rig order book pre-2000 **(proprietary/not publicly available)**	`data_py/raw/Order Book Prior to 2000.xls`
3. Rig order book post-2000 **(proprietary/not publicly available)**	`data_py/raw/Order Book 2000-2015 v2.xls`
4. Rig contracts (US) **(proprietary/not publicly available)**	`data_py/raw/contracts/US GoM fixtures 1996 to May 2016 with Turnkey fixtures highlighted.xlsx`
5. Gas prices (monthly)	`data_py/raw/gas_prices/gas_price_ym.dta`
6. Oil prices	`data_py/raw/gas_prices/PET_PRI_SPT_S1_W.xls`
7. Gas prices	`data_py/raw/gas_prices/RNGWHHDw.xls`
8. Applications for permits to drill (non-eWell)	`data_py/raw/wells/APDRawData.zip`
9. Bottomhole Pressure Survey data	`data_py/raw/wells/BHPSRawData.zip`
10. Borehole data	`data_py/raw/wells/BoreholeRawData.zip`
11. Applications for permits to drill (eWell)	`data_py/raw/wells/eWellAPDRawData.zip`
12. Applications for permits to modify (eWell)	`data_py/raw/wells/eWellAPMRawData.zip`
13. End of operations report (eWell)	`data_py/raw/wells/eWellEORRawData.zip`
14. Well activity reports (eWell)	`data_py/raw/wells/eWellWARRawData.zip`
15. Well production data	`data_py/raw/wells/ogora[year]delimit.zip` for year in [2000, ..., 2015]
16. Lease data	`data_py/raw/wells/lsetapefixed.zip`
17. Price deflator	`data_py/raw/deflator_daily.csv`
18. Rig name/ID mapping	`data_py/external/rig_map_VALIDATE.xls`

Datasets in 1.-3.:

These are from Rigzone, a commercial data provider. I have permission to use these data for research purposes. The data are not publicly available, but researchers can contact Rigzone to purchase the data.

Dataset 4.:

This is from IHS, a commercial data provider. I have permission to use these data for research purposes. The data are not publicly available, but researchers can contact IHS to purchase the data.

Datasets in 5.-7.:

This is from the U.S. Energy Information Administration (EIA) (where I changed dataset 5. to STATA format). The data are publicly available at https://www.eia.gov/. I include these data files in the replication package for convenience.

Datasets in 8.-15.:

This is from the Bureau of Safety and Environmental Enforcement (BSEE). The data are publicly available at https://www.bsee.gov/. I include these data files in the replication package for convenience.

Dataset 16.:

This is from the Bureau of Ocean Energy Management (BOEM). The data are publicly available at https://www.boem.gov/. I include these data files in the replication package for convenience.

Dataset 17.:

This is from FRED (Federal Reserve Economic Data). The data are publicly available at https://fred.stlouisfed.org/. I include these data files in the replication package for convenience.

Dataset 18.:

The is a mapping between IHS and Rigzone rig names/IDs. Where rig names are identical in both datasets, I use these to match the rigs. Where they are not identical, I manually matched the rigs using ex-names from sources documented in the file. This file is included in the replication package for convenience.

Computational requirements

Software Requirements

- MacTeX 2025
- Stata (code was last run with version SE 19.0)
    - `egenmore`
    - Load this via ssc install
- Python (for intel Mac, with builds:)
    - python=3.9.7=h38b4d05_3_cpython
    - numpy=1.20
    - scipy=1.10.1
    - scikit-learn=0.24.2=py39hd4eea88_1
    - pandas=1.3.2=py39h4d6be9b_0
    - statsmodels=0.12.2=py39h329c335_0
    - dask=2021.9.0
    - openpyxl=3.1.5
    - libblas=3.9.0=11_osx64_openblas
    - liblapack=3.9.0=11_osx64_openblas
    - libopenblas=0.3.17=openmp_h3351f45_1
    - joblib=1.1.0
    - threadpoolctl=3.0.0
    - numba=0.54.1
    - snakemake-minimal=7.32.4
    - geopy=2.2.0
    - xlrd=2.0.1
    - pandasql=0.7.3
    - sqlalchemy=1.4.25
    - matplotlib=3.4.2=py39h6e9494a_0
    - pip:
        - humanize==3.10.0
        - mpire==2.2.1
        - pandas-flavor==0.2.0
        - pandas-log==0.1.7
        - xarray==0.19.0
        - stata_setup==0.1.3
    - the file "`env_bbm.yml`" lists these dependencies, see https://pip.pypa.io/en/stable/user_guide/#ensuring-repeatability for further instructions on creating and using the "`env_bbm.yaml`" file.
        - E.g. use terminal command `mamba env create -f env_bbm.yml` (which will make the env with name `bbm_39`).
    - I used mamba with `minforge3-25.3.0` to install the environment
- Portions of the code use bash scripting, which may require Linux.

Memory, Runtime, Storage Requirements

Summary

Some code (but not all) is feasible to run on a desktop machine, as described below.

Details

Steps 1, 3, 5 of the code in 'Instruction to Replications Section' were last run on an **Apple 2019 Intel Mac Pro with 32GB of RAM, with Operating System Sequoia 15.5**. As mentioned above, I have provided the exact builds of the python packages, which need to be run on this kind of machine. This is potentially important for precise numerical reproducability.

Steps 2 and 4 of the code in 'Instruction to Replications Section' were last run on a **120-node university SLURM cluster, for several days per batch job, with 4GB of attached storage on each node**. The cluster is a Linux machine using an AMD x86-64 chip.

Description of programs/code

- The files `config.yaml` contain the configuration for code. You will need to edit these files to match your local path names.
- The folder `src/data_py` contains the code to construct the data. You can run this using the snakemake command in step 1 below.
- The folder `src/descriptives` contains additional code to produce some descriptive statistics and figures in the paper. These files are run in Step 5. below.
- The folder `src/interpolation` is a bug fixed version of the `interpolation` package, which is used in the code to interpolate the data.
- The folder `src/models_new` contains code/functions used in the model. These functions are called in the model-based steps below (e.g. Step 2,3,4,5).
- The folder `src/rules` contains the Snakemake rules to run the code. These are used by the Snakemake command in the steps below.
- The folder `src/run_scripts` contains scripts to run various parts of the code. These are used in various Snakemake rules and in the batch scripts used on the cluster.
- The folder `src/tex` contains LaTex skeletons that are used to produce the tables in the paper. These are used automatically by various programs in the code.
- The file `src/settings.json` contains settings for the table and figures.
- The various scripts named `run_*.sh` in the root directory contain example SLURM batch scripts to run the code on a cluster. You will need to edit these scripts to match your local cluster configuration.
- `image.sif` : this is the Apptainer image used to run the code on the cluster. It contains all the necessary software and dependencies to run the code. You will need to have Apptainer installed on your cluster to use this image.

Instructions to Replicators

Note: Steps 1, 3, 5 of the code should be run on an Intel Mac machine. Steps 2 and 4 of the code should not be run on a desktop machine, but rather on a cluster with an x86-64 processor (for the Apptainer container) and with sufficient resources.

0. Initial Setup

- Edit: `config.yaml` with your local path names for `overleaf_path` (use the local outputs directory); `stata_path` , `root_path` .
- Edit the batch submission scripts `run_*.sh` in the root directory with your specific configuration and path names (I used the ASU SLURM Cluster).
- Install conda environment using `env_bbm.yml` (e.g. use terminal: `mamba env create -f env_bbm.yml`)

1. Get descriptives and data construction

- 1.1 In terminal activate environment (e.g. `mamba activate bbm_39`)
- 1.2 In `config.yaml` set:
  - `run_data: True`
  - `run_first_stage: True`
  - `run_bootstrap_draws: True`
  - and the rest of the flags to `False`
- 1.3 Write: `snakemake -c8 -F --latency-wait=100` in the terminal in the root directory of the folder (this will run the data construction/main descriptives)
  - You might need to experiment with the 100 number; this is the max. number of seconds the code will wait for each Stata file to complete. This number works on my machine but if yours is slower the Stata code may exit before it finishes.
  - Note: can change flag `use_previous_extensions` to not use previous numerical values of extension models (set to `True` as default since there are tiny numerical differences with scikit-learn Logit across ARM, Intel, AMD machines)

2. Run the estimation and bootstrap on a cluster

- 2.1 Transfer the entire folder to the computing cluster.
- 2.2 Run batch scripts:
  - Type `sbatch run1_local.sh` in the terminal in the root directory of the folder on the cluster
  - Type `sbatch run_bootstrap.sh` in the terminal in the root directory of the folder on the cluster

3. Run some robustness checks

- 3.1 Transfer the entire folder `models/smm` from cluster to local computer.
- 3.2 In terminal activate environment (e.g. `mamba activate bbm_39`)
- 3.3 In `config.yaml` set:
  - `run_robustness: True`
  - and the rest of the flags to `False`
- 3.4 Write: `snakemake -c1 --latency-wait=100` in the terminal in the root directory of the folder on your local computer (this will run the robustness checks)

4. Run some more computationally intensive robustness tests on the cluster

- 4.1 Transfer the entire folder `models/robustness` from local computer to cluster (keeping the other directories in place on the cluster).
- 4.2 Run batch script:
  - Type `sbatch run_robustness.sh` in the terminal in the root directory of the folder on the cluster

5. Run counterfactuals and produce paper inputs

- 5.1 Transfer entire directory `models/smm` from cluster to local computer.
- 5.2 In terminal activate environment (e.g. `mamba activate bbm_39`)
- 5.3 In `config.yaml` set:
  - `run_produce_paper_inputs: True`
  - `run_counterfactuals: True`
  - and the rest of the flags to `False`
- 5.4 Write: `snakemake -c1 --latency-wait=100` in the terminal in the root directory of the folder (this will run the counterfactuals and produce paper inputs)

List of tables and programs

The provided code reproduces all tables and figures in the paper (note: you need to run all the code, these just make the final figures from the results).

Figure/Table #	Program	Output file
Table 1	src/run_scripts/run_descriptive_figures.py	output/tables/table_summary.tex
Figure 2	src/run_scripts/run_descriptive_figures.py	output/figures/figure_positive_assortive_matching.pdf
Figure 3	src/run_scripts/run_descriptive_figures.py	output/figures/figure_dayrate.pdf
Figure 4	src/run_scripts/run_descriptive_figures.py	output/figures/figure_boom_bust_2.pdf
Table 2	src/run_scripts/run_descriptive_figures.py	output/tables/table_dispersion.tex
Table 3	src/run_scripts/run_descriptive_figures.py	output/tables/table_mismatch.tex
Figure 6	src/run_scripts/run_descriptive_figures.py	output/figures/figure_value_search.pdf
Table 5	src/run_scripts/run_descriptive_figures.py	output/tables/table_smm.tex
Figure 7(a), (b)	src/run_scripts/run_descriptive_figures.py	output/figures/figure_counterfactual_benchmark.pdf
Figure 7(c)	src/run_scripts/run_descriptive_figures.py	output/tables/table_benchmark.tex
Figure 8(a), (b)	src/run_scripts/run_descriptive_figures.py	output/figures/figure_counterfactual_intermediary.pdf
Figure 8(c)	src/run_scripts/run_descriptive_figures.py	output/tables/table_intermediary.pdf
Figure 9(a), (b)	src/run_scripts/run_descriptive_figures.py	output/figures/figure_counterfactual_demand_smoothing.pdf
Figure 9(c)	src/run_scripts/run_descriptive_figures.py	output/tables/table_demand_smoothing.tex
Table A-1	src/descriptives/descriptives_reviewer_2_hedonic.py	output/tables/table_price_hedonic.tex
Table A-2	src/descriptives/descriptives_reviewer_2_synergies.py	output/tables/table_synergies.tex
Table A-3	src/descriptives/descriptives_reviewer_2_drilling_speed.py	output/tables/table_duration.tex
Figure A-3	src/run_scripts/run_descriptive_figures.py	output/figures/figure_boom_bust_composition.pdf
Table A-4	src/run_scripts/run_descriptive_figures.py	output/tables/table_sorting.tex
Table A-5	src/run_scripts/run_descriptive_figures.py	output/tables/table_utilization.tex
Table A-6	src/run_scripts/run_descriptive_figures.py	output/tables/table_moments_detail.tex
Figure A-4	src/run_scripts/run_descriptive_figures.py	output/figures/figure_acceptance.pdf
Figure A-5	src/run_scripts/run_descriptive_figures.py	output/figures/figure_out_of_sample.pdf
Figure A-6	src/descriptives/descriptives_reviewer_2_lump_oil_gas.py	output/figures/oil_gas_het.pdf
Table A-7	src/descriptives/descriptives_reviewer_2_lump_oil_gas.py	output/tables/table_proportion_gas.pdf
Table A-8	src/run_scripts/run_robustness_myopic_do_sim.py	output/tables/table_robust_p_exit.tex
Table A-9	src/run_scripts/run_descriptive_figures.py	output/tables/table_rig_target_counterfactual.tex
Table A-10	src/run_scripts/run_descriptive_figures.py	output/tables/table_two_week_counterfactual.tex