Imputation as Anomaly Detection

Matt Calder
Board of Assessors
City of Framingham

A Remarkable Algorithm

SoftImpute (Mazumder et al 2014)

Minimize the rank of imputed matrix subject to the constraint that the imputation agrees on non-missing values.

$$\min_{\hat{X}} ||X - \hat{X}||_{F_\Omega} + \lambda||\hat{X}||_*$$

Solution is to iterate (1) – (2) until convergence:

(1) $$\begin{aligned} \widehat{X} &= UDV^T \\ \widehat{M} &\leftarrow U\mathcal{S}_\lambda(D)V^T \end{aligned}$$

(2) $$\widehat{X} \leftarrow P_\Omega(X) + P_\Omega^\perp(\widehat{M})$$

A Few Tricks

- Sparse SVD (sparse plus low-rank)
- Alternative: A sequence of regressions (replaces (1))

$$\underset{A,B}{\text{minimize}}\,\frac{1}{2}\|\widehat{X} - AB^T\|_F^2 + \frac{\lambda}{2}(\|A\|_F^2 + \|B\|_B^2)$$

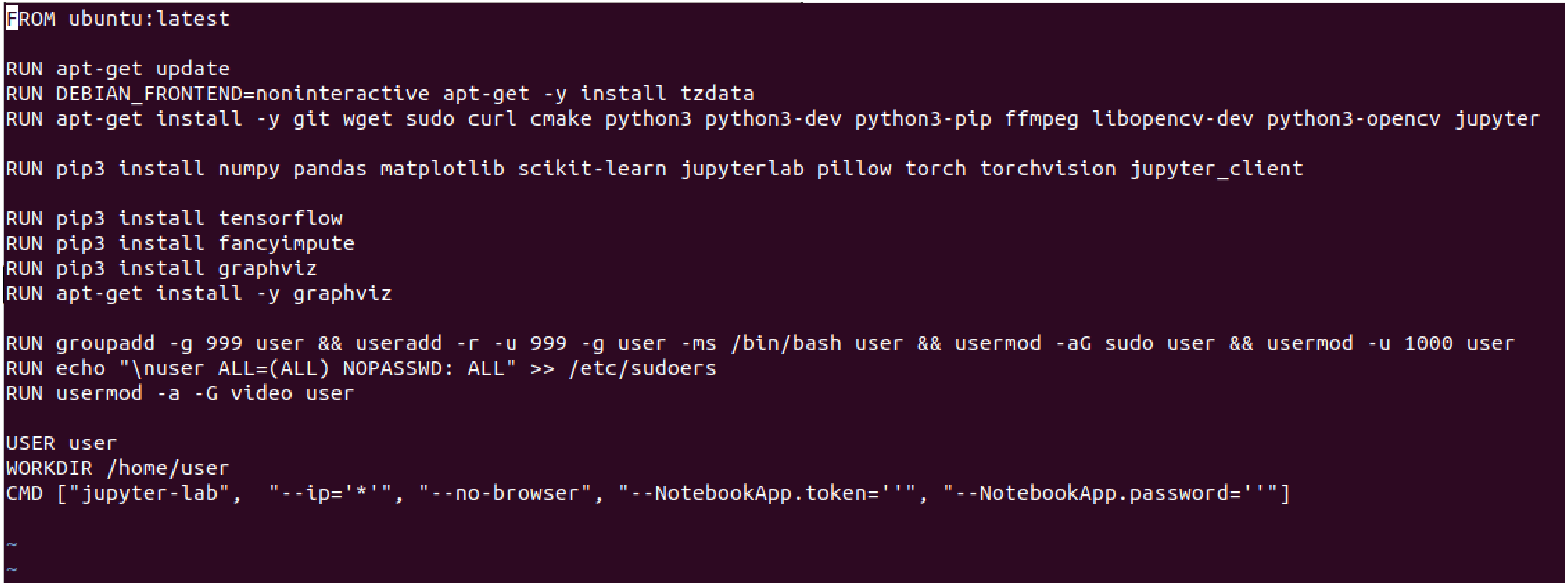
- Spin-off: Row + Column scaling

$$\begin{aligned}\tilde{X}_{ij} &= \frac{X_{ij} - \mu_{ij}}{\sigma_{ij}} \\ &= \frac{X_{ij} - \alpha_i - \beta_j}{\tau_i \gamma_j}\end{aligned}$$

Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares (Mazumder et al 2014)

Data Handling

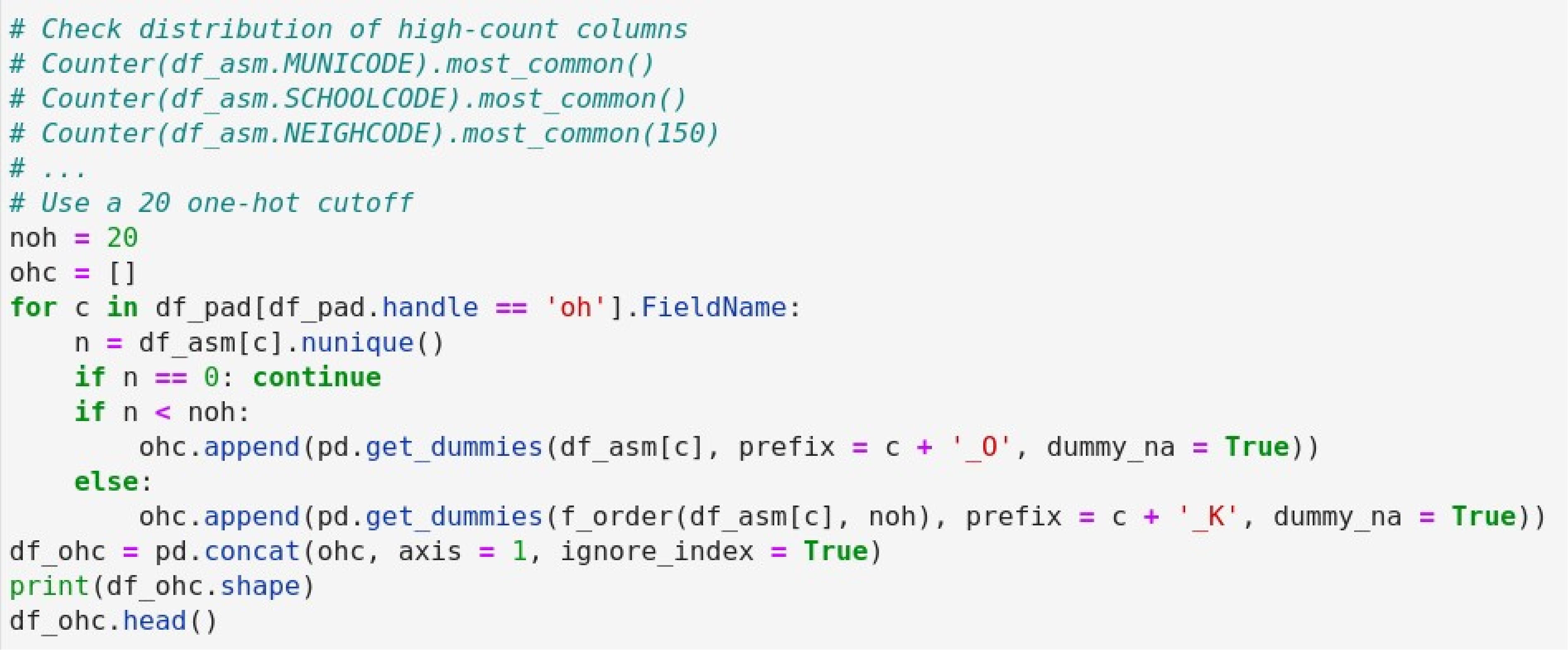
- Containerized Environment



```
FROM ubuntu:latest

RUN apt-get update
RUN DEBIAN_FRONTEND=noninteractive apt-get -y install tzdata
RUN apt-get install -y git wget sudo curl cmake python3 python3-dev python3-pip ffmpeg libopencv-dev python3-opencv jupyter

RUN pip3 install numpy pandas matplotlib scikit-learn jupyterlab pillow torch torchvision jupyter_client

RUN pip3 install tensorflow
RUN pip3 install fancyimpute
RUN pip3 install graphviz
RUN apt-get install -y graphviz

RUN groupadd -g 999 user && useradd -r -u 999 -g user -ms /bin/bash user && usermod -aG sudo user && usermod -u 1000 user
RUN echo "\nuser ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
RUN usermod -a -G video user

USER user
WORKDIR /home/user
CMD ["jupyter-lab",  "--ip='*'", "--no-browser", "--NotebookApp.token=''", "--NotebookApp.password=''"]


~
~
```

Data Handling

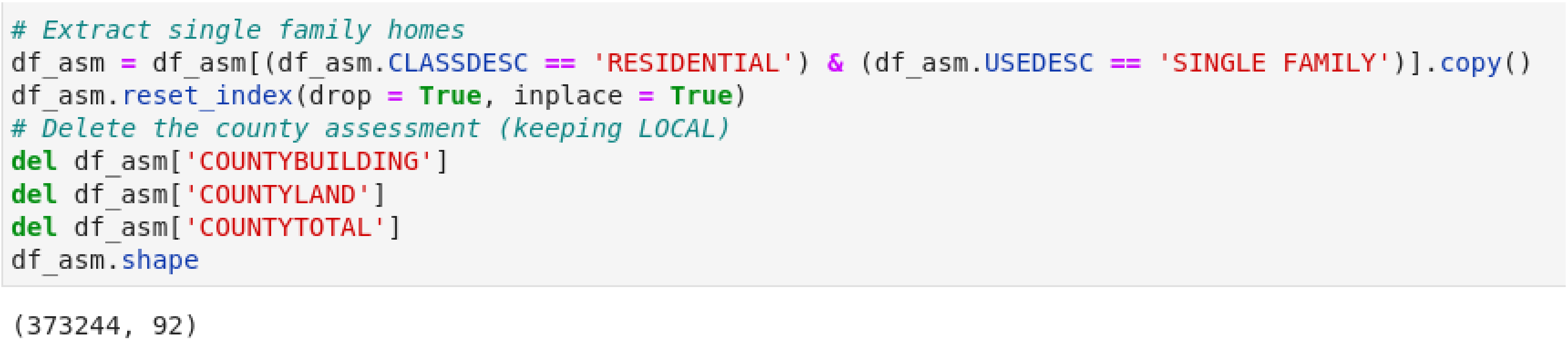
- Inspection & Encoding



```
# Check distribution of high-count columns
# Counter(df_asm.MUNICODE).most_common()
# Counter(df_asm.SCHOOLCODE).most_common()
# Counter(df_asm.NEIGHCODE).most_common(150)
# ...
# Use a 20 one-hot cutoff
noh = 20
ohc = []
for c in df_pad[df_pad.handle == 'oh'].FieldName:
    n = df_asm[c].nunique()
    if n == 0: continue
    if n < noh:
        ohc.append(pd.get_dummies(df_asm[c], prefix = c + '_O', dummy_na = True))
    else:
        ohc.append(pd.get_dummies(f_order(df_asm[c], noh), prefix = c + '_K', dummy_na = True))
df_ohc = pd.concat(ohc, axis = 1, ignore_index = True)
print(df_ohc.shape)
df_ohc.head()
```

Data Handling

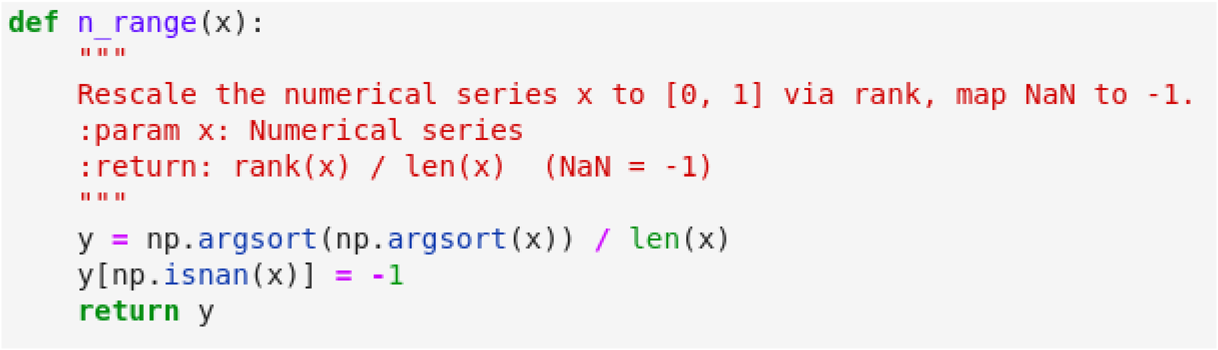
- Filtering



```
# Extract single family homes
df_asm = df_asm[(df_asm.CLASSDESC == 'RESIDENTIAL') & (df_asm.USEDESC == 'SINGLE FAMILY')].copy()
df_asm.reset_index(drop = True, inplace = True)
# Delete the county assessment (keeping LOCAL)
del df_asm['COUNTYBUILDING']
del df_asm['COUNTYLAND']
del df_asm['COUNTYTOTAL']
df_asm.shape
```

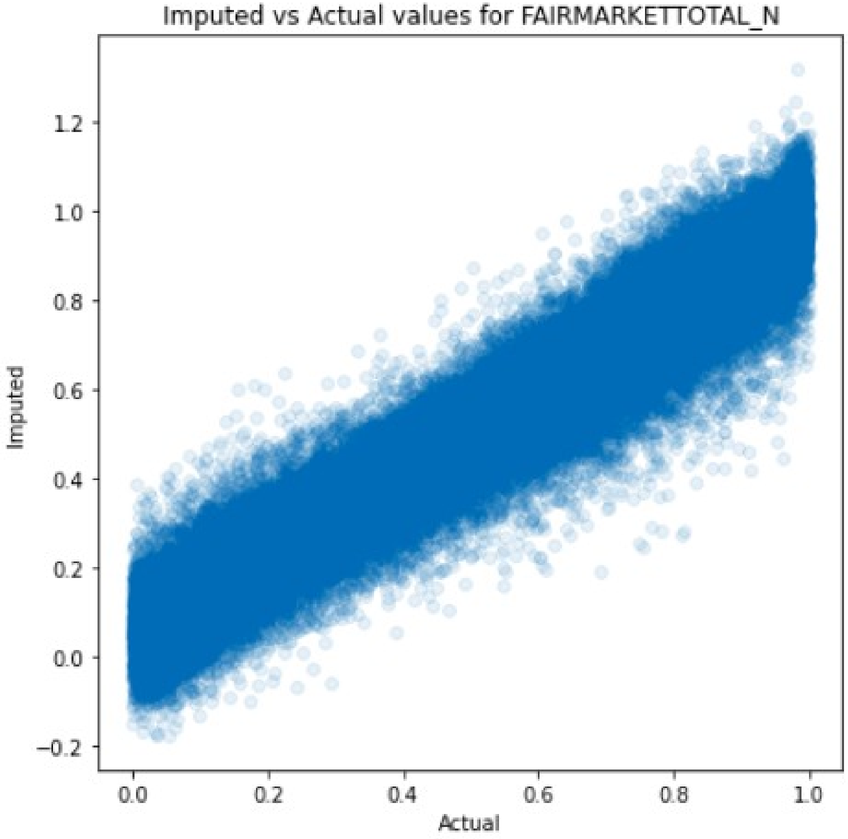
```
(373244, 92)
```

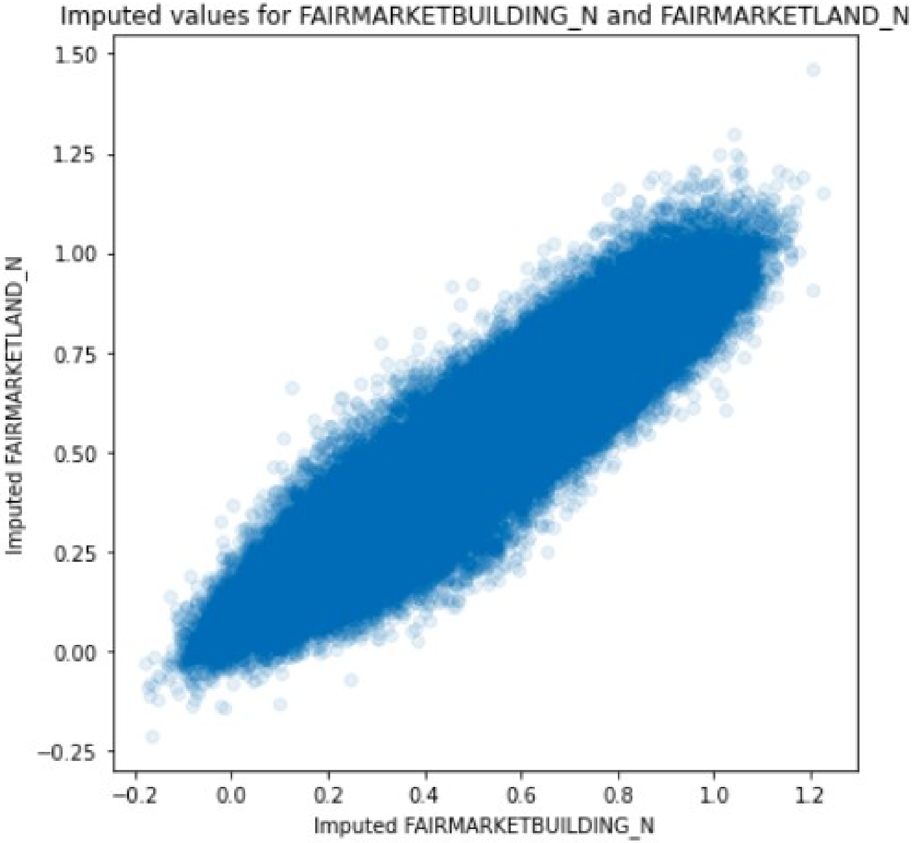
- Scaling



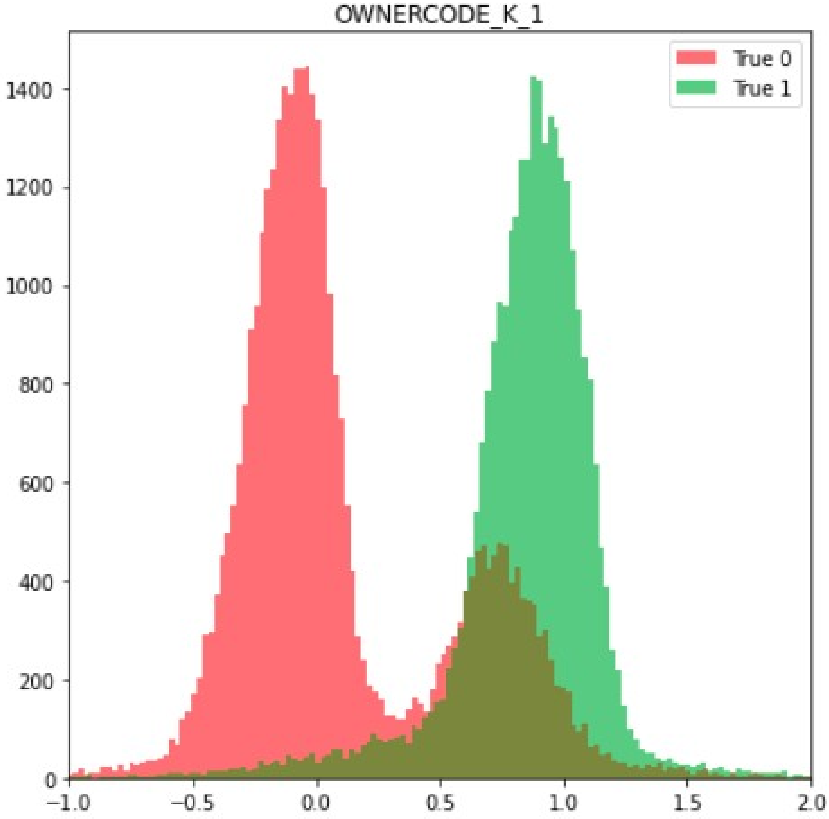
```
def n_range(x):
    """
    Rescale the numerical series x to [0, 1] via rank, map NaN to -1.
    :param x: Numerical series
    :return: rank(x) / len(x)  (NaN = -1)
    """
    y = np.argsort(np.argsort(x)) / len(x)
    y[np.isnan(x)] = -1
    return y
```

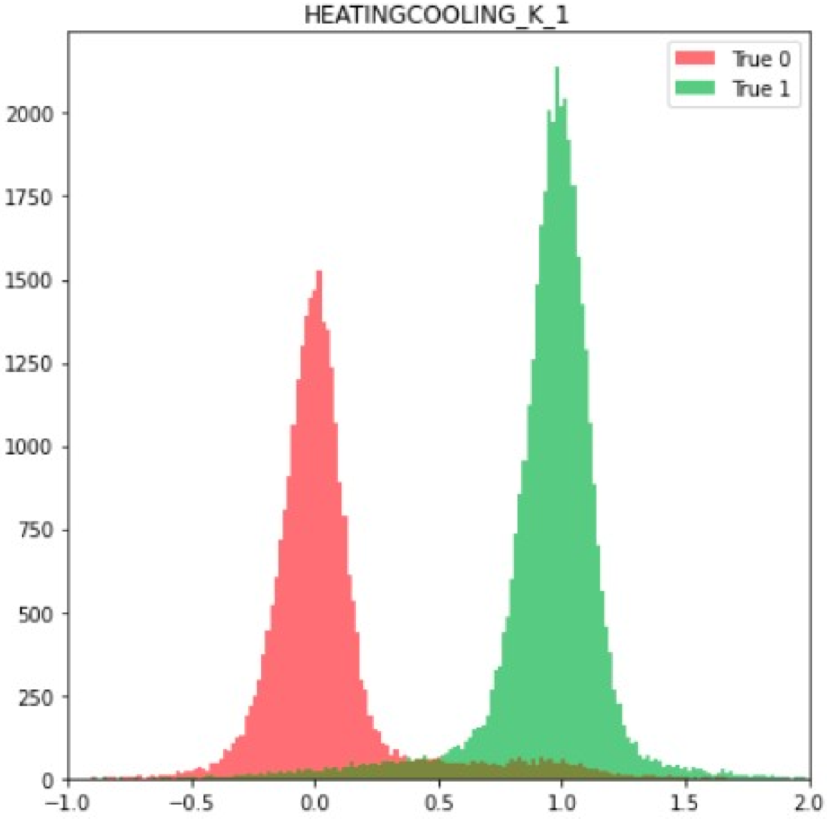
Imputation


Imputed vs Actual values for FAIRMARKETTOTAL_N

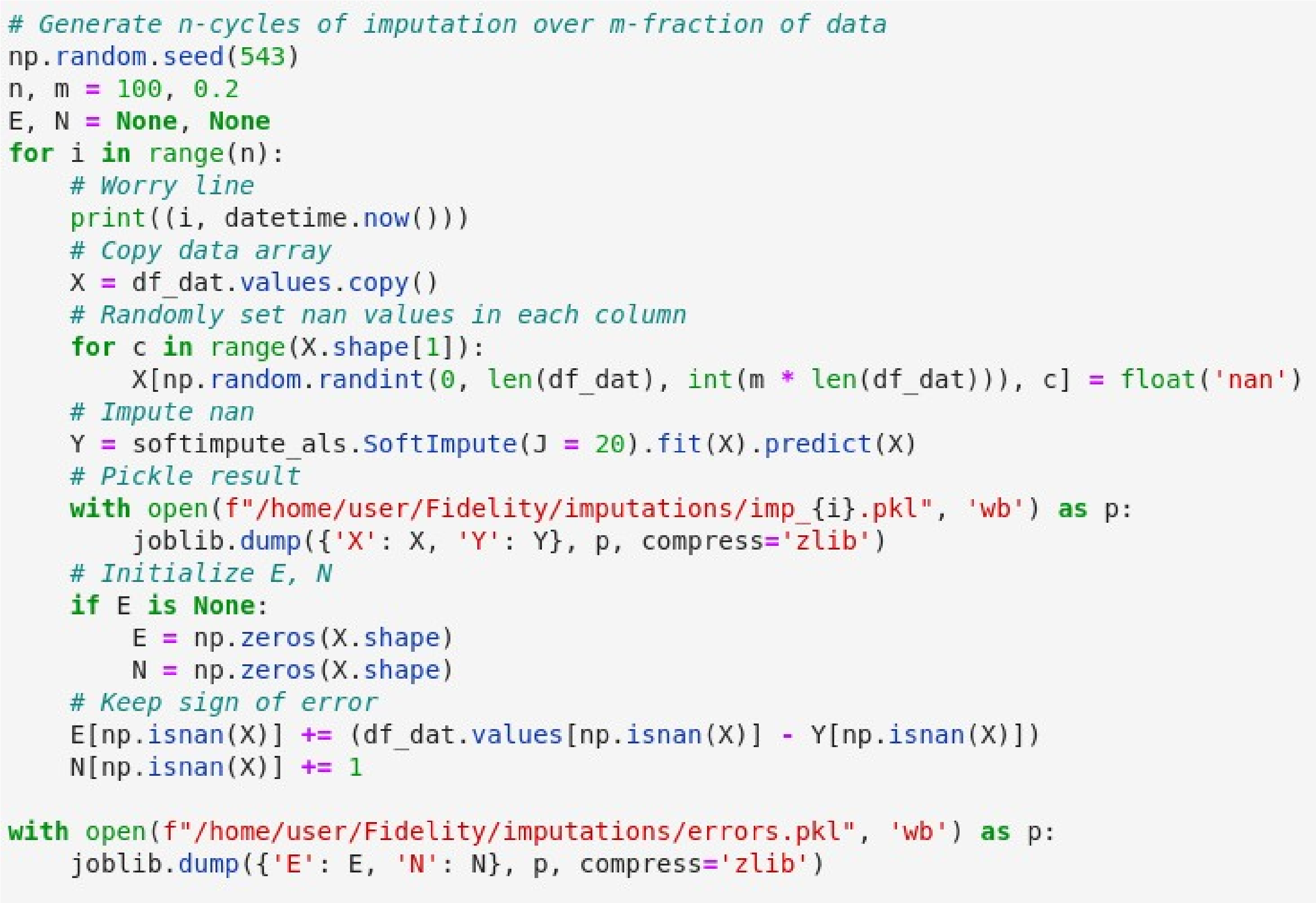

Imputed values for FAIRMARKETBUILDING_N and FAIRMARKETLAND_N

Imputation


OWNERCODE_K_1
🟥 True 0
🟩 True 1
0
200
400
600
800
1000
1200
1400
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

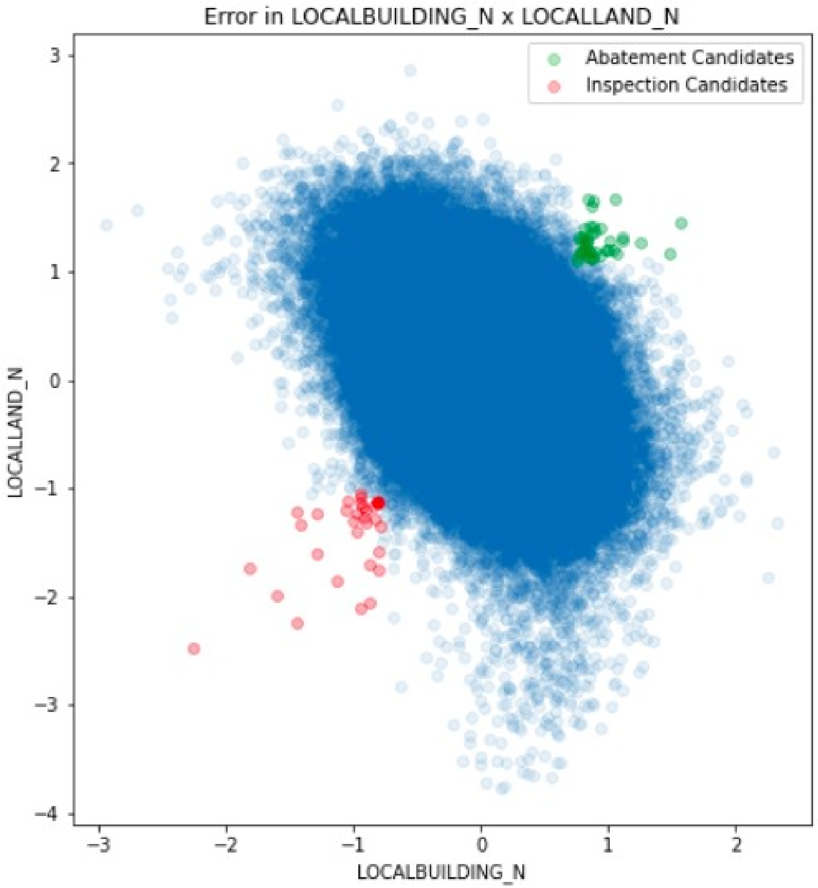

HEATINGCOOLING_K_1
🟥 True 0
🟩 True 1
0
250
500
750
1000
1250
1500
1750
2000
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Anomaly Detection

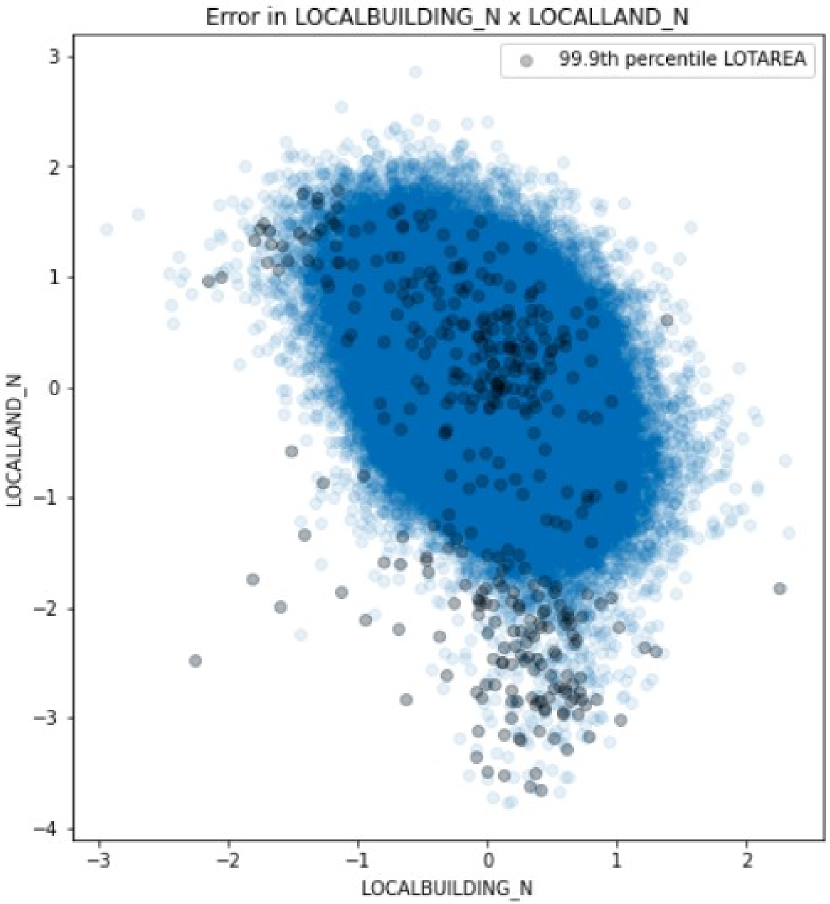


```python
# Generate n-cycles of imputation over m-fraction of data
np.random.seed(543)
n, m = 100, 0.2
E, N = None, None
for i in range(n):
    # Worry line
    print((i, datetime.now()))
    # Copy data array
    X = df_dat.values.copy()
    # Randomly set nan values in each column
    for c in range(X.shape[1]):
        X[np.random.randint(0, len(df_dat), int(m * len(df_dat))), c] = float('nan')
    # Impute nan
    Y = softimpute_als.SoftImpute(J = 20).fit(X).predict(X)
    # Pickle result
    with open(f"/home/user/Fidelity/imputations/imp_{i}.pkl", 'wb') as p:
        joblib.dump({'X': X, 'Y': Y}, p, compress='zlib')
    # Initialize E, N
    if E is None:
        E = np.zeros(X.shape)
        N = np.zeros(X.shape)
    # Keep sign of error
    E[np.isnan(X)] += (df_dat.values[np.isnan(X)] - Y[np.isnan(X)])
    N[np.isnan(X)] += 1

with open(f"/home/user/Fidelity/imputations/errors.pkl", 'wb') as p:
    joblib.dump({'E': E, 'N': N}, p, compress='zlib')
```

Abatement or Inspection

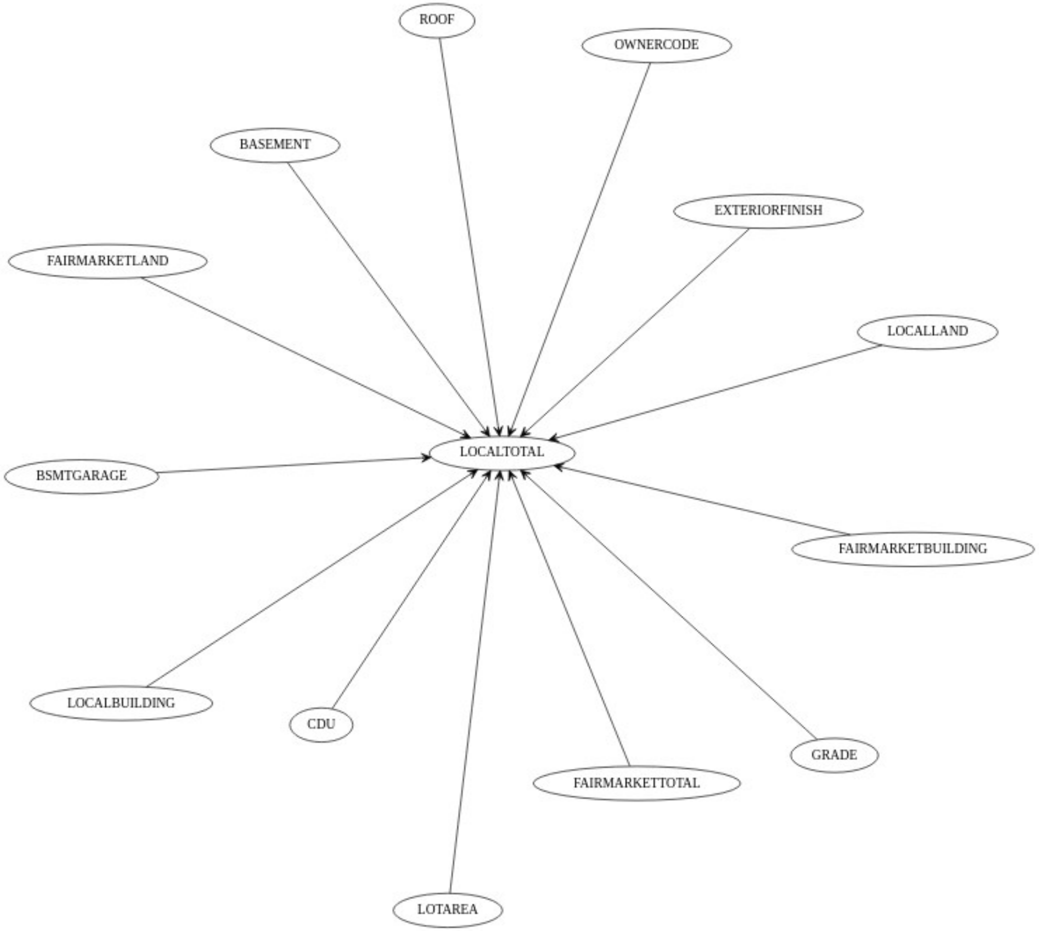


What’s That Blob?



Extremely large lot sizes lead to an over estimate of land value in the imputation

Causal Connections



Measure MSE of LOCALTOTAL imputation when each other variable is NaN / not NaN. Most MSE reducing variables shown here.