SUPPLEMENTARY MATERIAL

RANK ONE UPDATE ALGORITHM

Here we detail the algorithm to update the QR factorization of $\boldsymbol{Z}^t$ for the rank one update

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Z}^t + (\boldsymbol{e}_i - \boldsymbol{H}_i)'\boldsymbol{X}_i \tag{1}$$

We assume that the factorization $\boldsymbol{Z}^t = \boldsymbol{Q}^t\boldsymbol{R}^t$ is known. Let $\boldsymbol{v} = \boldsymbol{e}_i - \boldsymbol{H}_i$. We start by refactoring the update as

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^t\boldsymbol{R}^t + \boldsymbol{v'}\boldsymbol{X}_i = \boldsymbol{Q}^t(\boldsymbol{R}^t + \boldsymbol{w'}\boldsymbol{X}_i) \tag{2}$$

$\boldsymbol{R}^t$ is upper triangular, but $\boldsymbol{w'}\boldsymbol{X}_i$ is not and we would like to convert it to be upper triangular. Givens rotations are a common tool used in QR factorization to convert matrices into upper triangular ones. A Givens rotation can be represented as a rotation matrix $\boldsymbol{G(i,j,\theta)}$, where $G_{[k,k]} = \cos\theta$ for $k = i,j$, $G_{[k,k]} = 1$ for $k \neq i,j$, $G_{[i,j]} = -G_{[j,i]} = -\sin\theta$, and all other entries are zero. The angle of rotation, θ, can be set such that the product of $\boldsymbol{G}$ and a given vector has a zero at index j.

We can compute a set of Givens rotation matrices $\boldsymbol{J}^1, ..., \boldsymbol{J}^{n-1}$ such that $(\boldsymbol{J}^1)'...(\boldsymbol{J}^{n-1})'\boldsymbol{w}' = ||\boldsymbol{w}||\boldsymbol{e_1'}$. This will ensure that $||\boldsymbol{w}||\boldsymbol{e_1'}\boldsymbol{X}_i$ is upper triangular, since only the first row of the product is non-zero. The inverse of a Givens rotation matrix is also its transpose. To maintain equality with the original formula, we must include the transpose of every Givens rotation we introduce. This results in

$$\begin{aligned}\boldsymbol{Z}^{t+1} &= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1(\boldsymbol{J}^1)'\dots(\boldsymbol{J}^{n-1})'(\boldsymbol{R}^t + \boldsymbol{w'}\boldsymbol{X}_i)\\ &= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1(\boldsymbol{A} + ||\boldsymbol{w}||\boldsymbol{e}_1'\boldsymbol{X}_i)\end{aligned}$$

where $\boldsymbol{A} = (\boldsymbol{J}^1)'...(\boldsymbol{J}^{p-1})'\boldsymbol{R}$, which is an upper Hessenberg matrix. Upper Hessenberg matrices are upper triangular matrices with one additional non-zero entry below the diagonal of each column. They can be turned into upper triangular matrices with a linear number of Givens rotations.

$$\begin{aligned}\boldsymbol{Z}^{t+1} &= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1(\boldsymbol{A} + ||\boldsymbol{w}||\boldsymbol{e}_1'\boldsymbol{X}_i)\\ &= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1\tilde{\boldsymbol{A}}\end{aligned}$$

$\tilde{\boldsymbol{A}}$ is also an upper Hessenberg matrix. As such, we can find another set of Givens rotation matrices $\boldsymbol{G}^1, ..., \boldsymbol{G}^{p-1}$ such that $(\boldsymbol{G}^{p-1})'...(\boldsymbol{G}^1)'\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{R}}$, where $\tilde{\boldsymbol{R}}$ is an upper triangular matrix.

$$\begin{aligned}\boldsymbol{Z^{t+1}} &= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1\tilde{\boldsymbol{A}}\\ &= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1\boldsymbol{G}^1, ..., \boldsymbol{G}^{p-1}(\boldsymbol{G}^{p-1})'...(\boldsymbol{G}^1)'\tilde{\boldsymbol{A}}\\ &= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1\boldsymbol{G}^1, ..., \boldsymbol{G}^{p-1}\tilde{\boldsymbol{R}}\end{aligned}$$

This completes the factorization update, with $\boldsymbol{Q}^{t+1} = \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1\boldsymbol{G}^1...\boldsymbol{G}^{p-1}$ and $\boldsymbol{R}^{t+1} = \tilde{\boldsymbol{R}}$.

Algorithm 1 qr_update

```
Inputs: Q, R, v, u
w' = Q'v'
# Add v' as a new column basis to Q
v' = v'/||v||
Q[·,p+1] = v'
R[p+1,·] = 0
# Use givens rotation to zero out w
for i = p − 1 to 1 do
    G = givens(w_i, w_{i+1})
    Q[·,i:i+1] = Q[·,i:i+1] G
    R[i:i+1,·] = G R[i:i+1,·]
    w[i:i+1] = G w[i:i+1]
end for
R[1,·] = R[1,·] + w_1 u
# Use Givens rotations to make R upper triangular
for i = 1 to p − 1 do
    G = givens(R[i,i], R[i+1,i])
    Q[·,i:i+1] = Q[·,i:i+1] G
    R[i:i+1,·] = G R[i:i+1,·]
end for
# Return first p columns of Q, p rows of R
Q = Q[·,1:p]
R = R[1:p,·]
Return(Q, R)
```

The pseudocode for the rank one QR update is in Algorithm 1. After calculating $\boldsymbol{w}$, we normalize $\boldsymbol{v}$ into the basis of $\boldsymbol{Q}$ and append it as an extra column. We also add a zero row to $\boldsymbol{R}$. Givens rotations are used to zero out $\boldsymbol{w}$, with $\boldsymbol{Q}$ and $\boldsymbol{R}$ updated accordingly. After this we can add $\boldsymbol{wu'} = w_1\boldsymbol{u'}$ to $\boldsymbol{R}$. At this point $\boldsymbol{R}$ is upper Hessenberg, so we make it upper triangular with another series of Givens rotation, updating $\boldsymbol{Q}$ appropriately. We then return the first p columns of $\boldsymbol{Q}$ and the first p rows of $\boldsymbol{R}$.

Givens rotations can be represented as a full matrix product, but in practice it is faster to work with the rows that they operate on. In Algorithm 1, $\boldsymbol{G}$ is a 2×2 Givens rotation matrix which we apply directly to the two columns of $\boldsymbol{Q}$ and rows of $\boldsymbol{R}$ it affects.

SIMULATOR

Our simulator models two different distributions used for data generation: a baseline distribution and a modified or anomalous distribution. The data generated from each distribution is ensured to lie within a spatially contiguous region. These distributions are modeled at specific percentiles of the CDF, which gives us the ability to control at which percentiles they differ.

Our simulation creates a dataset of n points defined by $\boldsymbol{D} = \{\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{L}, \boldsymbol{Q}, \boldsymbol{B_1}, \boldsymbol{B_2}, \boldsymbol{I}\}$.

$\boldsymbol{L}$ is a set of n locations in 2D space, generated uniformly at random.

$\boldsymbol{X}$ is an $n \times p$ covariate matrix, generated uniformly at random between a minimum and maximum value. The first column of $\boldsymbol{X}$ is 1, to denote the intercept term.

$\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ are $k \times p$ distribution matrices representing the default and altered distributions respectively. The ith row of $\boldsymbol{B_j}$ stores the parameters for a regression through the (i/k)th quantile. These parameters are generated such that the quantiles in $\boldsymbol{B_j}$ do not cross within the range of $\boldsymbol{X}$. Each $\boldsymbol{B_j}$ parameterizes a piecewise continuous CDF for the regression data, with k locations in the CDF modeled exactly, and the rest assumed to vary uniformly between them.

$\boldsymbol{I}$ is a $n \times 1$ indicator vector that determines whether each point is generated from $\boldsymbol{B_1}$ or from $\boldsymbol{B_2}$. The set of points generated from $\boldsymbol{B_2}$ make a circle in the space of $\boldsymbol{L}$. This is the target region for the algorithm to identify.

$\boldsymbol{Q}$ is a $n \times 1$ vector indicating what quantile each point is generated from. The values of $\boldsymbol{Q}$ are in the continuous range $[0, k]$ and are generated uniformly at random. We use the function $f(\boldsymbol{B}, Q_i)$ to produce the parameters of a given quantile for distribution matrix $\boldsymbol{B}$.

$$f(\boldsymbol{B}, Q_i) = (Q_i - \lfloor Q_i \rfloor)\boldsymbol{B}_{\lfloor Q_i \rfloor} + (\lceil Q_i \rceil - Q_i)\boldsymbol{B}_{\lceil Q_i \rceil} \tag{3}$$

where $\boldsymbol{B}_i$ indicates the ith row of $\boldsymbol{B}$. If Q_i falls between two rows of $\boldsymbol{B}$, then f returns a weighted average of the two rows, such that the quantiles change continuously with respect to Q_i.

$\boldsymbol{Y}$ is an n vector of response variables. These are generated by

$$Y_i = \boldsymbol{X_i} f(\boldsymbol{B}_{I_i}, Q_i) + \epsilon \tag{4}$$

where $\epsilon \sim Norm(0, \sigma)$ is a random noise term.