Future of Integer Calibration Weighting Methods

Luca Sartore

National Institute of Statistical Sciences (NISS)
USDA National Agricultural Statistics Service

lsartore@niss.org – luca.sartore@nass.usda.gov

SDSS 2018
Data Science at the National Institute of Statistical Sciences

May 17, 2018

Quantum Computing: The Future of Integer Calibration

Luca Sartore

National Institute of Statistical Sciences (NISS)
USDA National Agricultural Statistics Service

lsartore@niss.org – luca.sartore@nass.usda.gov

SDSS 2018
Data Science at the National Institute of Statistical Sciences

May 17, 2018

Acknowledgements

Valbona Bejleri
Katherine Cespedes
Matt Fetter
Thomas Jacob
Johnathan Lisic
Beth Schlein
Clifford Spiegelman
Kelly Toppin
Linda Young

Presentation Outline

PART I: MOTIVATION

1. Census of Agriculture
2. The calibration problem at NASS

PART II: METHODOLOGY

3. Quantum computing
4. Quantum rounding algorithm
5. Quantum integer calibration

PART III: APPLICATION

6. Simulation study
7. Concluding remarks

PART I
MOTIVATION

1. Census of Agriculture
2. The calibration problem at NASS

Census of Agriculture

Every five years, USDA's National Agricultural Statistics Service (NASS) conducts the Census of Agriculture.

- The Census provides a detailed picture of U.S. farms, ranches and the people who operate them.
- It is the only source of uniform, comprehensive agricultural data for every state and county in the United States.
- NASS also obtains information on most commodities from administrative sources or surveys of non-farm populations (e.g. cotton ginning data).

DSE: Dual-System Estimation

NASS uses DSE to adjust its estimates by generating weights assigned to each data-record.

- DSE requires two independent surveys to produce adjusted estimates for under-coverage, non-response and incorrect farm-classification at the national, state and county levels.
- The adjusted weights are used as starting values for the calibration process.
- The weights are calibrated to ensure that the Census estimates are consistent across all levels of aggregation and in agreement with information from other sources.

Calibration problem

A solution $\hat{w}$ such that $T = Aw$, where

- T is a vector partitioned into y known and y^* unknown population totals,
- A is the matrix of collected data from a population, and
- w is a vector of unknown weights.

Calibration finds the solution of the linear system $y = \tilde{A}w$, where

- $\tilde{A}$ is a sub-matrix of the collected data.

NASS publishes its estimates by using
integer weights
to avoid fractional farms.

INCA: Integer Calibration

Currently, NASS uses the following steps to calibrate its weights:

1. All unfeasible weights are truncated to their closest boundary, and to minimize the objective function, non-integer weights are then rounded sequentially according to an importance index based on the gradient.
2. Each weight, according to the magnitude of the gradient, is allowed to move by unit-steps that decrease the objective function.

Limitation

INCA converges to a **local minima**,
not to a global solution.

PART II
METHODOLOGY

3. Quantum computing
4. Quantum rounding algorithm
5. Quantum integer calibration

Quantum Computing and the future of INCA

Quantum computing

Looks promising for applications where complex problems need computationally efficient solutions, such as **finding a discrete global optimum**.

Quantum Integer Calibration (QUINCA)

- Rounds the DSE weights with a quantum search.
- Performs multidimensional adjustments of the rounded weights to match given calibration benchmarks.

The quantum-bit (qubit)

Classical computers use bits to represent either zeros or ones, but quantum computers operate with qubits (or quantum bits), which are allowed to denote values that are *simultaneously* 0 and 1.

Observing the status of a qubit with superposition $\alpha_0|0\rangle + \alpha_1|1\rangle$ will produce

$$x = \begin{cases} 0, & \text{with probability } |\alpha_0|^2, \\ 1, & \text{with probability } |\alpha_1|^2, \end{cases}$$

where

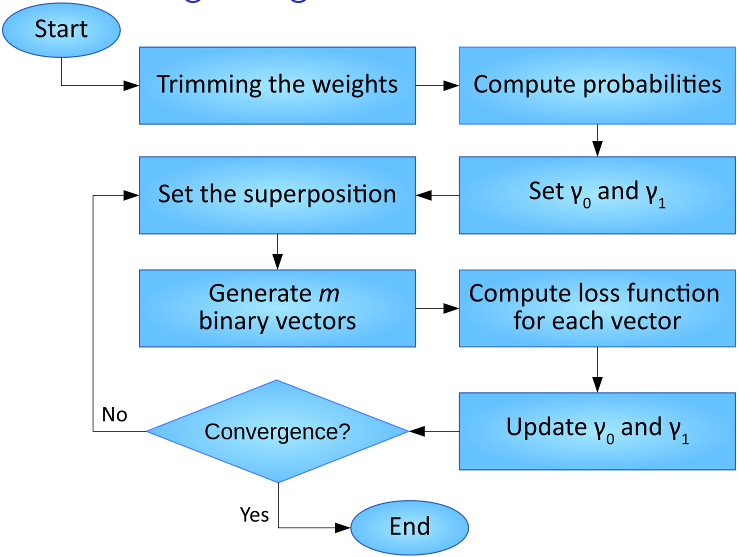
- the amplitudes of α_0 and $\alpha_1 \in \mathbb{C}$ satisfy $|\alpha_0|^2 + |\alpha_1|^2 = 1$,
- $|0\rangle$ denotes the vector $(1, 0)^\top$, and
- $|1\rangle$ represents the vector $(0, 1)^\top$.

Quantum operations

Quantum algorithms manipulate the qubits and produce measurements of the probability distribution of the observed outcomes.

1. The algorithm takes as input *n* classical bits and creates a superposition of 2^n possible states.
2. The superposition is then processed by quantum operations.
3. When the superposition is measured, it randomly collapses to zero or one.
4. Step 3 is iterated to assure convergence over the probability distribution.

Quantum rounding at a glance



Start

Trimming the weights

Compute probabilities

Set γ_0 and γ_1

Set the superposition

Generate *m* binary vectors

Compute loss function for each vector

Update γ_0 and γ_1

Convergence?

No

Yes

End

Example of quantum rounding

- We have a matrix of data

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 6 & 4 & 8 \end{bmatrix}.$$

- We start with the DSE weights $w^* = (2.3, 5.1, 7.9)^\top$.
- Our known totals are $y = (15, 97)^\top$.
- We consider the objective function

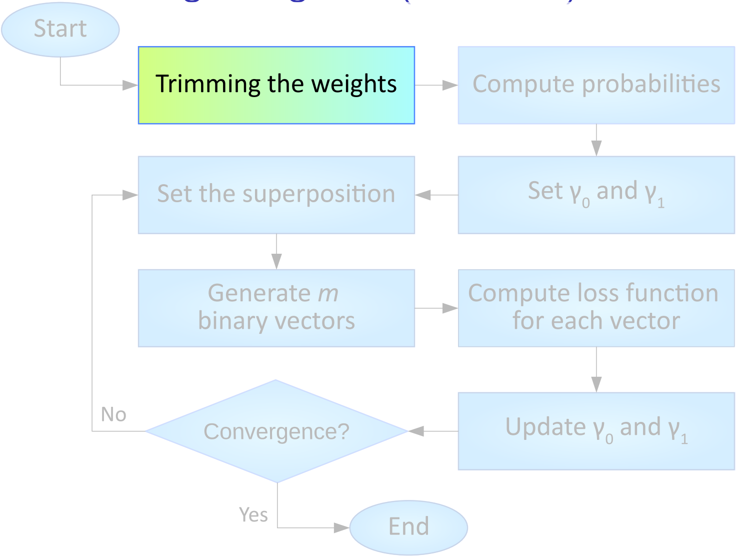
$$L(w) = \sum_i |y_i - w^\top A_i|,$$

  where A_i is the i-th row of the matrix A.

- The gradient of L

$$g(w) = -A^\top \mathrm{sign}(y - Aw).$$

Quantum rounding at a glance (continued)



Start
Trimming the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
End

Step 1: Trimming the weights

- The weights that do not satisfy given constraints are trimmed and all the others are truncated.
- Update the weights

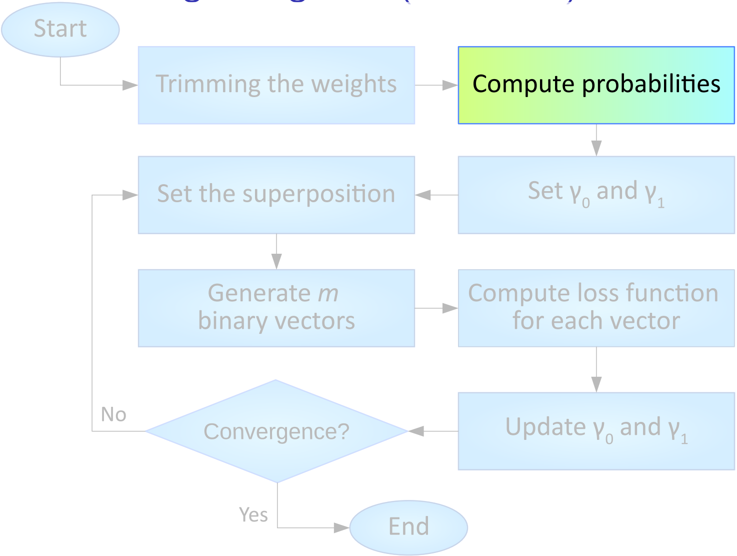
$$w_i = \begin{cases} 1, & \text{if } w_i^* < 1, \\ \lfloor w_i^* \rfloor, & \text{if } 1 \leq w_i^* < 6, \\ 6, & \text{if } w_i^* \geq 6, \end{cases}$$

where w_i^* represents the i-th DSE weight.

In our example

$$w^* = (2.3,\ 5.1,\ 7.9)^\top$$
$$w = (2,\ \ 5,\ \ 6)^\top$$

Quantum rounding at a glance (continued)


Start
Trimming the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
End

Step 2: Compute probabilities

A qubit $|\Psi_i\rangle$ is initialized as $|\Psi_i\rangle = \sqrt{1 - p_i}|0\rangle + \sqrt{p_i}|1\rangle$, where the initial probabilities are computed as

$$p_i = \begin{cases} g_{(1)}^{-1} g_i (w^* - w), & \text{if } g_i < 0 \text{ and } 0 < w^* - w < 1, \\ 0, & \text{otherwise,} \end{cases}$$

where $g_{(1)}^{-1}$ denotes the inverse of the smallest component of the gradient, g_i represents the i-th component of the gradient, for any $i = 1, \ldots, n$.

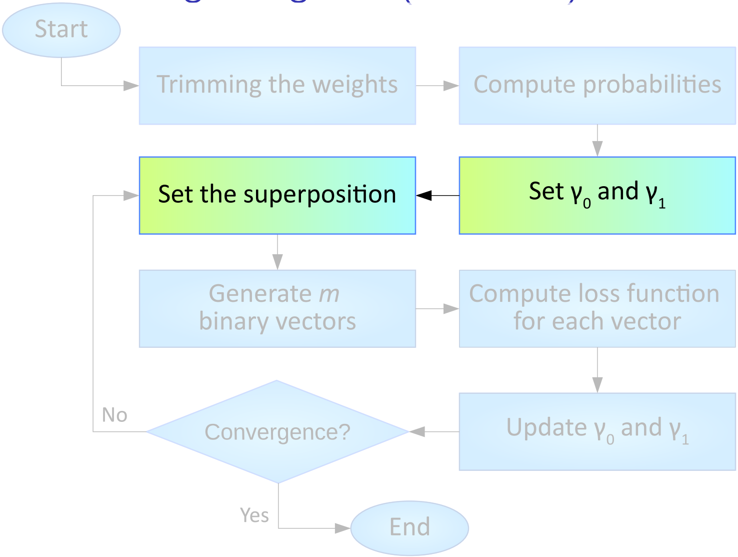
In our example

$$g = (-7, \ -5, \ -9)^\top$$

$$w^* - w = (0.3, \ 0.1, \ 0)^\top$$

$$p = (0.23, \ 0.06, \ 0)^\top$$

Quantum rounding at a glance (continued)


Start
Trimming the weights
Compute probabilities
Set the superposition
Set γ_0 and γ_1
Generate *m* binary vectors
Compute loss function for each vector
No
Convergence?
Update γ_0 and γ_1
Yes
End

Step 3 & 4: Stable setting of γ_0 and γ_1

- The quantities $\gamma_{0i} = 1 - p_i$ and $\gamma_{1i} = p_i$ are set so that

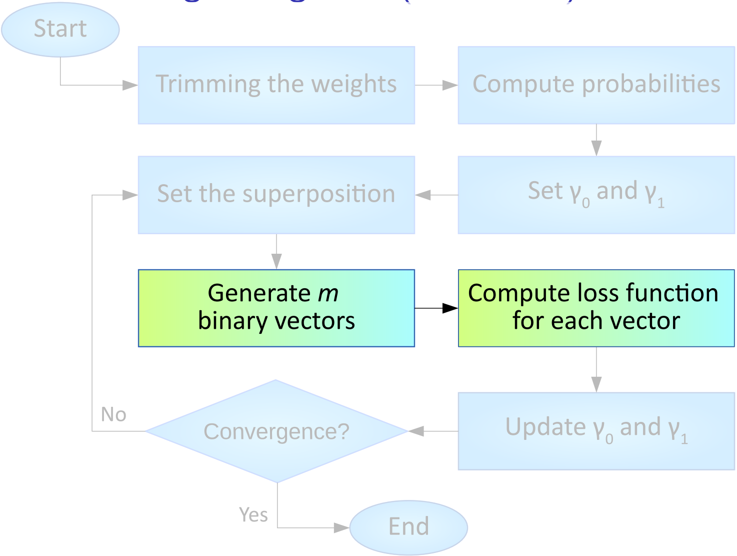
$$p_i = \frac{\gamma_{1i}}{\gamma_{0i} + \gamma_{1i}}.$$

- These two values, γ_{0i} and γ_{1i}, will be updated at each iteration.

In our example

$$\gamma_0 = (0.77,\ 0.94,\ 1)^\top$$
$$\gamma_1 = (0.23,\ 0.06,\ 0)^\top$$

Quantum rounding at a glance (continued)



Start
Trimming the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
End

Step 5 & 6: Observing the quantum state

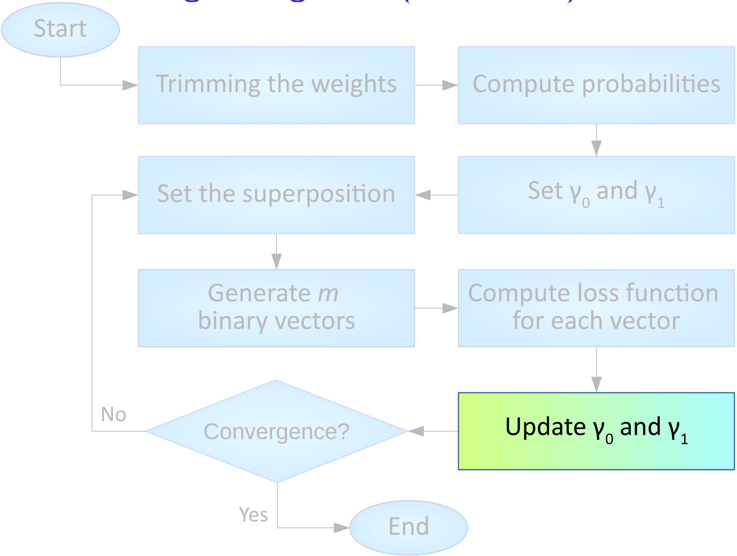
- m binary vectors x_j are generated by observing the quantum states of the qubits $j = 1, \ldots, m$.
- The performance of these vectors is evaluated by a loss function L_j associated to x_j.

In our example

The measurements (with $m = 5$) and their losses are

$$\begin{array}{lcl} x_1 = (1, 0, 0)^\top & \rightarrow & L_1 = 12 \\ x_2 = (0, 0, 0)^\top & \rightarrow & L_2 = 19 \\ x_3 = (1, 0, 0)^\top & \rightarrow & L_3 = 12 \\ x_4 = (1, 1, 0)^\top & \rightarrow & L_4 = 7 \\ x_5 = (0, 0, 0)^\top & \rightarrow & L_5 = 19 \end{array}$$

Quantum rounding at a glance (continued)


Start
Trimming the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
End

Step 7: Updating the probabilities via γ_0 and γ_1

$$\gamma_{0i}^{[\tau+1]} \leftarrow \gamma_{0i}^{[\tau]}(1-\lambda) + \lambda \sum_{j=1}^{m} (1 - x_{ji}) \left(\frac{L_{(m)} - L_j}{L_{(m)} - L_{(1)}} \right)^2 ,$$

$$\gamma_{1i}^{[\tau+1]} \leftarrow \gamma_{1i}^{[\tau]}(1-\lambda) + \lambda \sum_{j=1}^{m} x_{ji} \left(\frac{L_{(m)} - L_j}{L_{(m)} - L_{(1)}} \right)^2 ,$$

where

- $L_{(1)}$ denotes the minimum loss associated with the best fit,
- $L_{(m)}$ represents the maximum loss associated with the worst fit,
- the scalar $\lambda \in [0, 1]$ is used to speed-up convergence.

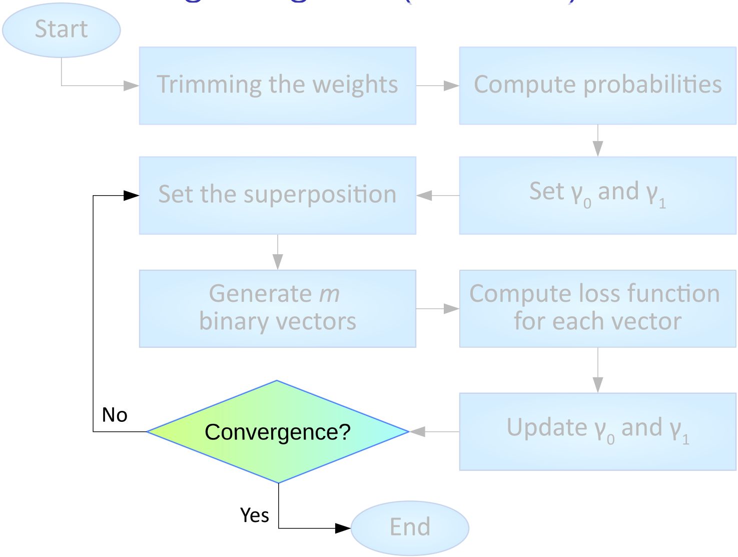
In our example

For $\lambda = 0.5$,

$$\gamma_0 = (0.38,\ 0.81,\ 1.34)^\top, \quad \text{and} \quad \gamma_1 = (0.96,\ 0.53,\ 0)^\top$$

$$p = (0.71,\ 0.39,\ 0)^\top$$

Quantum rounding at a glance (continued)


Start
Trimming the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
End

Step 8: Convergence

These operations are iterated until convergence is achieved.

$$\gamma_{1i}/(\gamma_{0i} + \gamma_{1i}) \to 0 \;\Rightarrow\; w_i = \lfloor w_i^* \rfloor$$

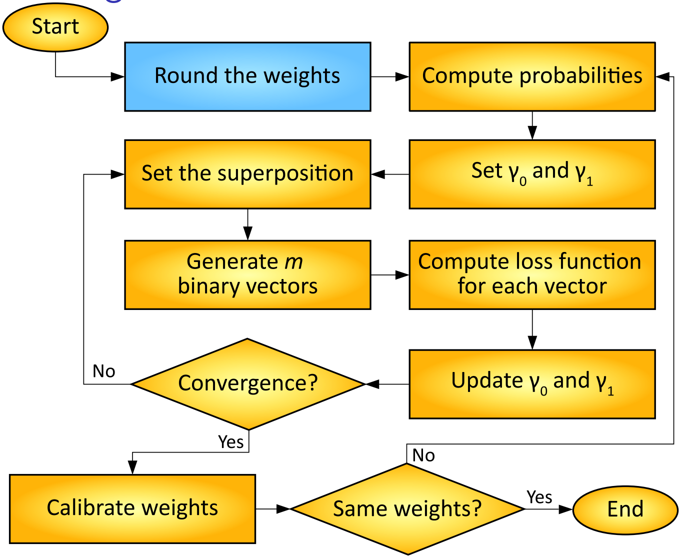
$$\gamma_{1i}/(\gamma_{0i} + \gamma_{1i}) \to 1 \;\Rightarrow\; w_i = \lceil w_i^* \rceil$$

In our example

The probabilities $p \approx (1, 1, 0)^\top$, so the vector of rounded weights is

$$w + p = \begin{pmatrix} 2 \\ 5 \\ 6 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 6 \\ 6 \end{pmatrix}$$

QUINCA at a glance



Start
Round the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
Calibrate weights
Same weights?
No
Yes
End

The calibration setting

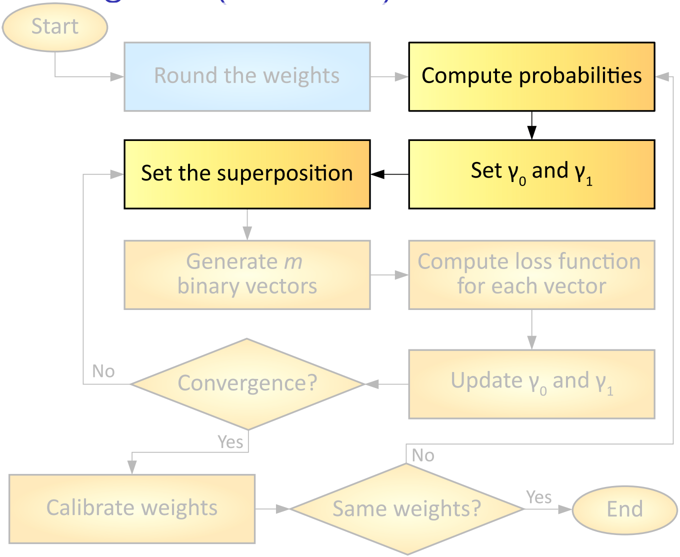
- Final integer weights are computed iteratively by unit adjustments according to the sign of the gradient.
- The feasible steps are computed by $s \odot x$.
- The components of the vector s satisfy the equality $s_i = -\mathrm{sign}(g_i)$ and those of x are generated by measuring the status of the qubits $|\Psi_i\rangle$, for any $i = 1, \ldots, n$.

In our example

The new gradient g and the vector s are

$$g = (-6,\ -4,\ -8)^\top$$
$$s = (1,\ 1,\ 1)^\top$$

QUINCA at a glance (continued)


Start
Round the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
Calibrate weights
Same weights?
No
Yes
End

Step 1: Qubits initialization

Qubits $|\Psi_i\rangle$ are initialized to take into account only for feasible adjustments in the opposite direction of the gradient; i.e.

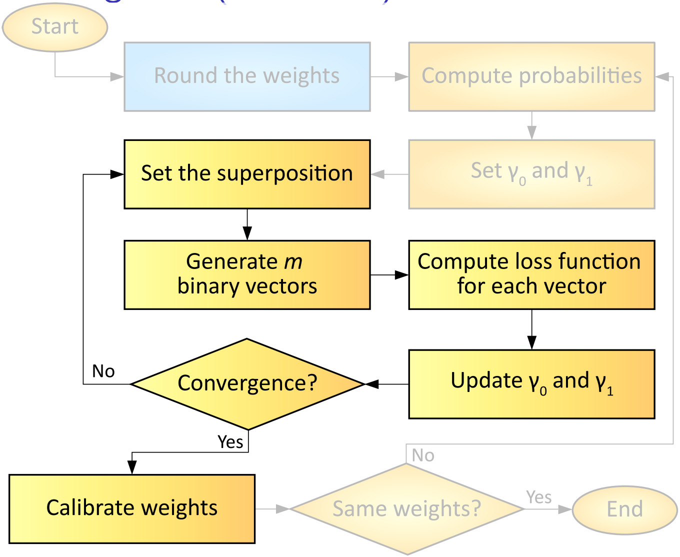
$$p_i = \begin{cases} 0, & \text{if } g_i > 0 \text{ and } w_i < 2, \\ 0, & \text{if } g_i < 0 \text{ and } w_i > \lfloor u_i \rfloor - 1, \\ |g_i| / \max\left(|g_{(1)}|, |g_{(n)}|\right), & \text{otherwise.} \end{cases}$$

These are used to initialize γ_0 and γ_1.

In our example

$$\begin{aligned} p &= (0.75,\ 0,\ 0)^\top \\ \gamma_0 &= (0.25,\ 1,\ 1)^\top \\ \gamma_1 &= (0.75,\ 0,\ 0)^\top \end{aligned}$$

QUINCA at a glance (continued)


Start
Round the weights
Compute probabilities
Set γ_0 and γ_1
Set the superposition
Generate *m* binary vectors
Compute loss function for each vector
Update γ_0 and γ_1
Convergence?
No
Yes
Calibrate weights
Same weights?
No
Yes
End

Step 2: Quantum adjustments

- The performance of the m binary vectors is evaluated by the loss function L_j, for any $j = 1, \ldots, m$.
- γ_0 and γ_1 are updated as in the rounding algorithm.
- When the ratio $\gamma_{1i}/(\gamma_{0i} + \gamma_{1i})$ converges, $w_i \leftarrow w_i + s_i \hat{x}_i$, where

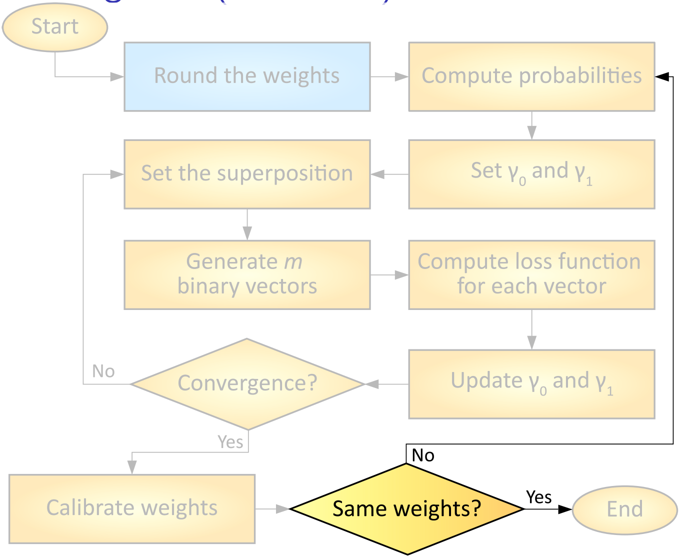
$$\hat{x}_i = \begin{cases} 0, & \text{if } \gamma_{0i} > \gamma_{1i}, \\ 1, & \text{otherwise.} \end{cases}$$

In our example

The probabilities $p \approx (1,\ 0,\ 0)^\top$, therefore

$$w + s \odot p = \begin{pmatrix} 3 \\ 6 \\ 6 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \odot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 6 \end{pmatrix}$$

QUINCA at a glance (continued)



Step 3: Convergence

- At each iteration the gradient is updated and a new set of probabilities are computed to initialize the qubits.
- The quantum calibration algorithm terminates when the ratio $\gamma_{1i}/(\gamma_{0i}+\gamma_{1i}) \rightarrow 0$ for any $i = 1, \ldots, n$.

In our example

The optimal solution was found in $\hat{w} = c(4, 6, 6)^\top$ with a final loss of 2. No further adjustments are needed.

PART III
APPLICATION

6. Simulation study
7. Concluding remarks

A simulation study

1) The data

- 150 weights $\omega_i \sim \text{Gamma}(3.333, 1)$.
- A 201 × 150 matrix A is simulated such that

$$a_{ki} = \begin{cases} 1, & \text{if } k = 1, \\ b_{ki} c_{ki}, & \text{otherwise,} \end{cases}$$

  where $b_{ki} \sim \text{Bernoulli}(0.3)$ and $c_{ki} \sim \text{Poisson}(4)$.
- Calibration benchmarks are computed as $y = A\omega$.
- DSE weights are simulated from a $\text{U}(0, 7.5)$.
- Final weights are restricted such that $\hat{w}_i \in [1, 6]$.

A simulation study (continued)

2) The loss function and its gradient

The loss function is

$$L_j = \sum_{k=1}^{201} \left| y_k - \sum_{i=1}^{150} a_{ki} w_i \right| ,$$

and its gradient is

$$g_i = - \sum_{k=1}^{201} \mathrm{sign}(\varepsilon_k) a_{ki} ,$$

where $\varepsilon_j = y_k - \sum_{i=1}^{150} a_{ki} w_i$.

A simulation study (continued)

3) The setting for the experiments

- Investigating the performance of the algorithm with respect to
  - number of measurements $m \in \{64, 101, 161, 256\}$,
  - learning rate $\lambda \in \{0.50, 0.60, 0.69, 0.75\}$.
- The simulated vector of the DSE weights is the same for all the combinations of m and λ.

A simulation study (continued)

4) The results

Table: Final loss after QUINCA

Number of	Learning Rate			
Measurements	0.50	0.60	0.69	0.75
64	1588	1532	1529	1437
101	1498	1362	1403	1330
161	1585	1402	1468	1431
256	1610	1590	1343	1327

Table: Elapsed time in seconds

Number of	Learning Rate			
Measurements	0.50	0.60	0.69	0.75
64	0.75s	0.59s	0.82s	0.67s
101	1.74s	2.37s	1.61s	1.30s
161	2.99s	3.27s	2.28s	3.04s
256	3.19s	2.35s	6.03s	5.02s

Concluding remarks

- QUINCA is a *preliminary* improvement of INCA.
- The proposed methodology performs a quantum search that, by design, overcomes the limitations of INCA and finds *better* vectors of integer calibrated weights.
- QUINCA adjusts the weights by performing multidimensional steps and has the potential of converging *heuristically* to a *global solution*.
- Future research can exploit quantum entanglement to move towards a global solution in one step.

Selected References

Barnett, S. (2009). *Quantum information*, volume 16. Oxford University Press.

Dasgupta, S., Papadimitriou, C. H., and Vazirani, U. (2006). *Algorithms*. McGraw-Hill, Inc.

Grover, L. and Rudolph, T. (2002). Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*.

Koenker, R. (2005). *Quantile regression*. Cambridge University Press, New York.

Sartore, L. and Toppin, K. (2016). *inca: Integer Calibration*. R package version 0.0.2.

Sartore, L., Toppin, K., Young, L., and Spiegelman, C. (2018). Developing integer calibration weights for Census of Agriculture. *Journal of Agricultural, Biological and Environmental Statistics*, Accepted.

Young, L. J., Lamas, A. C., and Abreu, D. A. (2017). The 2012 Census of Agriculture: a capture–recapture analysis. *Journal of Agricultural, Biological and Environmental Statistics*, 22(4):523–539.

Thank you!

Questions?

Luca Sartore, PhD

`lsartore@niss.org`
`luca.sartore@nass.usda.gov`