Reconstructing the Simulmatics 1960 Simulation: Mathematics, Methods and a Partial Replication

John Hessler
Applied Mathematician & Computer Scientist
Relic {Code} Lab
jhessle1@jhu.edu

22 November 2026

Abstract

In August 1960 the Simulmatics Corporation ran a computer simulation for the Democratic National Committee. It ranked 32 non-Southern states by Kennedy's likely strength if religion dominated the campaign. Pool and Abelson (1961) reported that this ranking correlated .82 with the November result, using no political data later than October 1958. This paper reconstructs the mathematics behind that result from the published record: the 1961 *Public Opinion Quarterly* article and the list of 480 voter types later printed in the appendix of Burdick's *The 480* (1964).

We make five contributions. First, we write the model as a complete system: a 480-type typology, a pooled issue-cluster estimator, a synthetic-state aggregator and nine cross-pressure equation sets. Second, we show that the model's one non-trivial step, estimating how many 1958 Democrats were anti-Catholic, is an odds-ratio constrained 2×2 table with a unique feasible solution for every odds ratio, and we derive its comparative statics. Third, we prove that the authors' "lucky guess" benchmark (add 34 per cent of the Catholic share to the 1956 Democratic vote) is the in-sample ratio of two least-squares coefficients, so its correlation is a two-parameter multiple correlation fitted with hindsight. Fourth, we recompute the published benchmarks, from public returns and from the authors' own data cards, which survive with the program that scored the simulation. The authors' printed ranking has a Spearman correlation of 0.755 with Kennedy's 1960 share across the 32 states ($p < 10^{-6}$). The model's own list of failures, the Mountain states, reappears in our data. The published lucky guess does not reproduce on the authors' own cards: the correlation peaks at $k = 0.52$ with $r = 0.880$, not at 0.34 with .83. Fifth, we transcribe, explain and port the six surviving programs. Among them is the subroutine that solved the fourfold table, which confirms the formulation of our second contribution. Six animations, collected in an online story map, accompany the paper.

On the contrary, we can be sure that they are partly false.

— Ithiel de Sola Pool and Robert Abelson, "The Simulmatics Project" (1961, p. 176)

1. Introduction

The Simulmatics project is usually remembered as a story about politics and computing: a small company, time on the MIT Computation Center's computer, and a candidate who won by a narrow margin. It is less often read as mathematics. Yet the 1961 report by Ithiel de Sola Pool and Robert

Abelson contains an explicit, testable model of voting behaviour. It specifies inputs, equations, parameters and an aggregation rule, and it reports how well the output matched the election.

The model's purpose was narrow. It did not forecast the 1960 result. It answered a contingent question: if the campaign were fought on party and Catholicism alone, which states would Kennedy do best in, and which worst? The answer was a ranking of 32 states, delivered to the campaign on 25 August 1960 together with two other reports (Pool and Abelson 1961, p. 173).

This paper asks three questions of that model.

1. What exactly did it compute? We rebuild the full system from the article's equations and the published list of voter types.
2. Why did it work as well as it did? We analyse each component, the fourfold-table step in particular, and the authors' argument that working voter type by voter type adds information a single correction cannot.
3. How much of the reported performance can be checked today? We recompute every benchmark that public data allow and set out what is needed for the rest.

The contributions are a formal statement of the model (Sections 3–6), three analytical results (Sections 7–9), a partial replication (Section 10), a reusable code base with animated visualisations (Section 11) and an account of the surviving code (Appendices A–F). Section 12 sets out the limits of what can be recovered and a research agenda.

2. Sources and data

Three historical sources define the model. The surviving code and public election returns are used to test it.

The 1961 article. Pool and Abelson describe the data, the 480-by-52 matrix, the synthetic states and the simulation, and print the equations for all nine cross-pressure groups, seven of them in footnote 8 (pp. 177–179). They also print the 32-state ranking (pp. 176–177) and a table of benchmark correlations (p. 182).

The 480 appendix. The full list of voter types was printed in the appendix of Eugene Burdick's novel *The 480* (1964, pp. 311–313), with the company's permission. It gives the defining attributes of every type, the community-size thresholds and the assignment of states to regions.

The underlying archive. Simulmatics pooled 66 surveys held by the Roper Public Opinion Research Center: 50 from the 1952–1958 elections (about 85,000 respondents) and 16 anticipating 1960, well over 100,000 interviews in all. The tabulated matrix was deposited with the Roper Center as a condition of access; the simulation programs and census-based state weights were not (p. 171). Neither the matrix nor the weights is printed in the article. Six programs from the project's code survive in the Ithiel de Sola Pool papers at MIT: the print section of the program that built the matrix, with two assembly-language routines (Appendix A), the program that built the state-by-state results (Appendix B), a program that estimated opinion on every issue cluster in every state (Appendix C), the lucky-guess program with its scoring routine and data cards (Appendix D), the fourfold-table subroutine of the simulation (Appendix E) and a later Guttman scaling program

(Appendix F).

Data used in this paper.

Series	Source	Use
1956 and 1960 presidential returns by state	Official state totals, via Wikipedia's election pages	Benchmarks; outcome variable
Simulmatics ranking of 32 states	Pool and Abelson (1961, pp. 176–177)	Tested against 1960 returns
Voter typology and state regions	Burdick (1964, pp. 311–313)	Model structure; state aggregation
1960 and 1956 vote and Catholic share, as the authors used them	Data cards of the authors' lucky-guess program, box1batch1.bin (Appendix D)	Lucky-guess benchmark (Sections 9 and 10.3)

The returns were checked for internal consistency: every published percentage matches votes divided by total to within 0.005 points, except Alabama 1960, where the unpledged-elector slate breaks the usual arithmetic. Alabama is outside the 32-state sample. The Catholic series, and the vote series used with it in the lucky-guess benchmark, are the authors' own, read from the data cards of their program (Appendix D.3). The cards do not name the source of the Catholic series.

3. The voter typology

The electorate is partitioned into 480 voter types. A type is a cell in a cross-classification of party, region, religion or ethnicity, sex, community size and occupational status. The partition is not a full Cartesian product: cells were merged where samples were thin, which is why the total is 480 rather than a round power of the factor levels.

3.1 Structure

The four main regions share one 36-cell template, repeated for each of three party identifications. The Border states use a smaller 16-cell template without community size.

$$|\mathcal{T}| \;=\; \underbrace{4 \times 3 \times 36}_{\text{East, South, Midwest, West}} \;+\; \underbrace{1 \times 3 \times 16}_{\text{Border}} \;=\; 432 + 48 \;=\; 480$$

Block	Cells	Defining attributes
Protestant, by sex	$2 \times 7 = 14$	Community × occupation (below)
Catholic, by sex	$2 \times 7 = 14$	Community × occupation (below)

Block	Cells	Defining attributes
Jewish	2	Professional and white collar; blue collar
Black	2	Professional and white collar; blue collar
Protestant, no SES rating	1	Non-labour-force families
Catholic, no SES rating	1	Non-labour-force families
Other religion	1	—
No religion	1	—
Total per region × party	**36**	

The seven community × occupation cells are: rural and town, each split into professional-and-white-collar and blue collar; and urban (over 100,000), split three ways into professional, white collar and blue collar. Rural means under 5,000 inhabitants and town 5,000 to 100,000. Professional and white-collar workers were not separated outside cities because rural areas and towns had too few professionals. Types are numbered region by region: 1–108 East, 109–216 South, 217–324 Midwest, 325–432 West and 433–480 Border. Within each region the parties run Democrat, Republican, Independent.

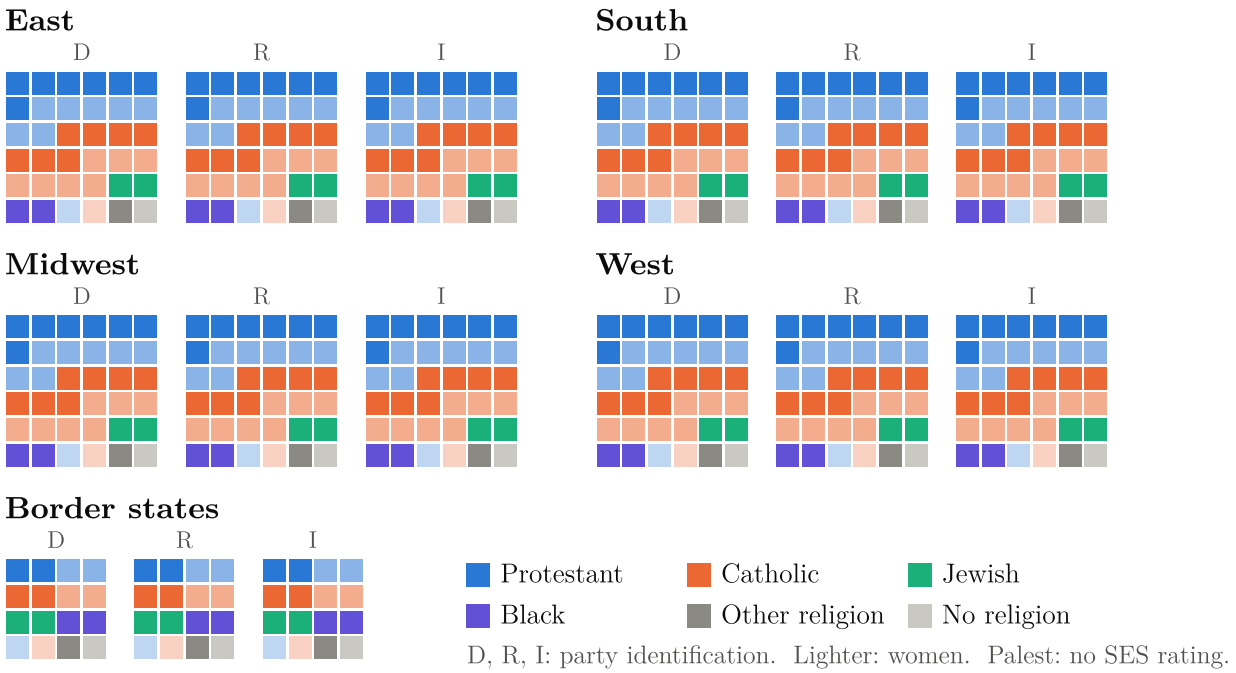


Figure 1: The 480 voter types. One 36-cell template (Protestants and Catholics by sex, community size and occupation; two Jewish and two Black cells; no-SES, other-religion and no-religion cells) is repeated for three party identifications (D, R, I) in each of four regions, giving 432 types. The Border states add 3 × 16 types without a community-size split. A state draws only on its own region's 108 types. Source: Burdick (1964, pp. 311–313). Animated version: Animation S1 in the online story map.

3.2 Regions

The East has 9 states (Maine to Pennsylvania), the South 11 (Virginia to Texas, including Oklahoma), the Midwest 12 (Ohio to Kansas, including Missouri), the West 11 (Montana to Washington) and the Border 5 (Maryland, Delaware, West Virginia, Kentucky, Tennessee). There is no Mountain region. Arizona, Colorado, Idaho, Nevada, New Mexico, Utah and Wyoming are pooled with the Pacific coast.

3.3 Consistency with the 1961 article

Three independent checks connect the appendix to the article. First, the article states that a state contains at most 108 voter types (p. 175); the template gives exactly $3 \times 36 = 108$ per main region. Second, footnote 5 blames the worst misses on "the absence of a 'Mountain Region' in our classification", which the region list confirms. Third, the 32 states the article ranks are exactly the $9 + 12 + 11$ states of the East, Midwest and West: South and Border are excluded, and no state is missing or extra. We verified this set identity in code. A fourth check comes from the code itself: the surviving state-by-state program requires exactly 16 cells in each Border state and 36 in every other, and requires each region's cells to be used by 5, 9, 11, 12 and 11 states, the sizes of the Border, East, South, Midwest and West regions (Appendix B.5).

3.4 Coding rules and anomalies

The appendix adds two rules that matter for estimation. Types marked with one star do not include urban white-collar respondents, while types marked with two stars do. The stars fall on the Jewish, Black and Border-state types: as printed, urban white-collar respondents in those groups are counted with blue collar, not with professional and white collar. We record this as printed.

One entry is irregular. Type 441 (Border, Democrat, Jewish, professional and white collar) is printed with "Female" in the sex column, although no other Jewish type has a sex. We treat it as a typesetting carry-over from type 439.

Finally, the Black types carry no religion, but the simulation's cross-pressure equations need one (Section 6). Most Black voters in 1960 were Protestant, so our default maps these types to the Protestant equations. Because this choice is ours, not the authors', the code exposes it as a parameter.

4. Issue clusters and the data matrix

The survey archive was reduced to a matrix with one row per voter type and one column per "issue cluster": 480 rows by 52 columns. Each cell holds four numbers: the count of respondents asked, and the shares pro, anti and undecided. The surviving print program stores them as a count R and two proportions, P (pro) and Q (anti), leaving the undecided share implicit (Appendix A.6). A separate matrix was built for each biennial election (1952, 1954, 1956, 1958) and one consolidated matrix for all years, so trends could be read by comparing matrices.

4.1 Trichotomisation

Original codes, sometimes up to thirteen per question, were collapsed to three outcomes. For a question with an ordered scale, the authors set cutting points "depending on the distribution" (p. 169). Formally, for question q with ordered codes $1, \dots, m$, a pair of cut points $c_1 \le c_2$ assigns code y to anti if $y \le c_1$, to pro if $y > c_2$ and to undecided otherwise. The choice of cut points is a researcher degree of freedom that the article does not report.

4.2 The pooled cell estimator

Let Q_j be the set of questions assigned to cluster j, and let x^+_{tq} and n_{tq} be the pro replies and cards for type t on question q. The cell value pools replies across every question and survey in the cluster:

$$\hat{p}^+_{tj} \;=\; \frac{\sum_{q \in Q_j} x^+_{tq}}{\sum_{q \in Q_j} n_{tq}} \;=\; \sum_{q \in Q_j} \frac{n_{tq}}{n_{tj}}\, p^+_{tq}, \qquad n_{tj} = \sum_{q \in Q_j} n_{tq}$$

The estimator is a sample-size weighted average of question-level proportions. It is unbiased for a common underlying proportion only if the questions in the cluster measure the same attitude with the same marginal level. Different wordings with different levels add a component of bias that depends on which questions happened to reach which types.

The pooling is also what makes small groups visible. With about 100,000 interviews spread over 480 types, a type averages roughly 210 respondents. A cluster asked in only a tenth of the surveys leaves about 21 per cell, and a sampling standard error near $\sqrt{0.25/21} \approx 0.11$ at $p = 0.5$. This is the reason the authors built synthetic states rather than reading each state's own respondents (Section 5).

4.3 Correlational validation of clusters

Whether two questions belong in one cluster was decided in two steps: an a priori grouping, then an empirical test. For the test the 480 types were consolidated into 15 larger groups, $g = 1, \dots, 15$, so that single-survey percentages were stable. For each pair of candidate questions the authors correlated the pro shares across the 15 groups, and separately the anti shares:

$$\rho^+_{qq'} = \mathrm{corr}_g\,(p^+_{gq},\, p^+_{gq'}), \qquad \rho^-_{qq'} = \mathrm{corr}_g\,(p^-_{gq},\, p^-_{gq'})$$

Questions were retained only when both correlations were high and the items also seemed politically equivalent. No threshold is reported. A Guttman scaling program written by Samuel Popkin in 1962 survives (Appendix F). It tests whether items form a single cumulative scale across respondents, a check at the level of the individual where the 1960 test worked across groups, and it postdates the 1960 matrix.

The second condition was essential, and a simple model shows why. Suppose each question's pro share in group g is a linear function of the group's Democratic share π_g plus noise: $p^+_{gq} =$

$\alpha_q + \beta_q \pi_g + \varepsilon_{gq}$. Then any two strongly party-linked questions correlate highly across groups, whatever their content. The authors give exactly this example: "Which party is better for people like you?" and a question on Adlai Stevenson's image correlate strongly but do not measure the same attitude (p. 170). Correlation across 15 groups is evidence that two questions sort groups the same way. It is not evidence that they measure one attitude.

The validation failed for several foreign-policy clusters, where answers were "highly sensitive to conjunctural influences or shades in wording" (footnote 2). Those were kept as unvalidated a priori clusters and, by the authors' account, barely used. The test also detected a change over time: replies to an identical question on Senator McCarthy stopped tracking the McCarthyism cluster after his censure.

5. Synthetic states

A state's predicted behaviour is the population-weighted average of the behaviour of the voter types living in it. With about 2,000 interviews per state on average, and as few as 300 to 400 in small states, direct state estimates were too noisy. The authors instead estimated how many persons of each voter type lived in each state, from census, poll and voting data, and borrowed the type's behaviour from every respondent of that type in the region.

Let T(s) be the voter types of state s (at most 108, all from s's region), w_{st} the estimated number of persons of type t in s, and V_t any type-level quantity. The synthetic state is

$$\bar{V}_s \;=\; \frac{\sum_{t \in T(s)} w_{st}\, V_t}{\sum_{t \in T(s)} w_{st}}$$

The estimator rests on a homogeneity assumption, which the authors state plainly and call "partly false" (p. 176): a voter type behaves identically in every state of its region. An upper-income rural Protestant Republican man is the same voter in Maine and in New York. All differences between states within a region therefore come from their mix of types, not from place. The surviving state-by-state program (Appendix B) implements this formula directly, within each party and in total, and checks that every state uses every cell of its region. A third surviving program applied the same weights to every issue cluster, producing for each state and party the shares holding the pro and the anti position on each issue (Appendix C): synthetic estimation of state opinion, not only of the vote. It weighted each cell by its population alone. The number of respondents behind a cell played no part, so a thinly sampled cell counted as fully as a well-sampled one.

In modern terms this is post-stratification without the multilevel model: cell estimates pooled within region, reweighted to each state's composition. It is the ancestor of multilevel regression and post-stratification (MRP). The difference is that MRP shrinks each cell towards a fitted model, partly pooling across cells, while Simulmatics pooled completely across states within a region and not at all across types.

The assumption is testable, which is what made the project scientifically interesting to its authors. If the 1960 result is well predicted, the variation between states within a region is mostly compositional.

Where it fails, place matters. The authors found the largest failures in Arizona, Nevada, New Mexico, Idaho and Colorado, mostly small Mountain states that the classification forced into a West dominated by California (footnote 5). Section 10 shows that the same states stand out in our recomputation.

The weights w_{st} are the binding constraint on any replication. The census can supply each state's population by sex, community size, occupation and race, but religion and party are not census items. They must come from survey distributions within each region, conditional on the census attributes. Simulmatics did this before any 1960 census results were available, presumably from 1950 figures, and its weights were never published. Appendix B shows the form they took: 1,628 records, called T-primes, giving for each state and cell the weight of the cell's Democrats, Republicans and Independents.

6. The cross-pressure equations

The simulation imagines a campaign in which the only issues are party and Catholicism. It draws on the cross-pressure finding of Berelson, Lazarsfeld and McPhee (1954): voters whose group loyalties pull in opposite directions defect more and vote less. The 480 types are sorted into a 3×3 grid of religion (Protestant, Catholic, other) by party (Republican, Democrat, Independent). Each cell gets its own pair of equations, judged by the analysts from what the data showed about that kind of voter.

6.1 Notation

All quantities are proportions for one voter type. The subscripts mix two conventions, which is a trap for readers: 56 and 58 are election years, while 35 and 14 are issue-cluster numbers. The letters themselves are the cluster proportions stored on the Simulmatics tape: P is a cluster's pro share and Q its anti share (Appendix A.6).

Symbol	Meaning
P_{56}, Q_{56}	Share preferring Stevenson / Eisenhower in 1956 polls
P_{58}, Q_{58}	Share intending to vote Democratic / Republican in 1958 polls
P_{35}	Nonvoting index (issue cluster 35)
P_{14}	Share giving anti-Catholic replies (issue cluster 14)
V_k, V_n	Predicted Kennedy / Nixon share of the type in 1960
a	Anti-Catholic 1958 Democrats, from the fourfold table (Section 7)

Because replies were trichotomised, $P + Q < 1$: the undecided remainder is never reallocated. V_k and V_n therefore do not sum to the type's turnout, and the state figure is what the authors call a "Kennedy index", not strictly a percentage. The surviving state-by-state program labels these figures percentages of the eligible electorate and converts them to shares of the two-party vote before comparing them with the result (Appendix B.6).

6.2 The nine cells

Protestant Republicans: no cross-pressure. Their 1960 vote is expected to repeat 1956, discounted by their record of nonvoting:

$$V_k = P_{56}\,(1 - P_{35}), \qquad V_n = Q_{56}\,(1 - P_{35})$$

Protestant Democrats and Protestant Independents: cross-pressured. The base is 1958, because too many of them defected to Eisenhower in 1956 for that year to show normal behaviour. The anti-Catholic Democrats, a, move to Nixon; and the nonvoting index is doubled, because cross-pressured voters stay home more:

$$V_k = (P_{58} - a)\,(1 - 2P_{35}), \qquad V_n = (Q_{58} + a)\,(1 - 2P_{35})$$

Catholic Democrats and Catholic Independents. A third of those who voted Republican in 1958 return to the Democrats; turnout is not penalised:

$$V_k = \Big(P_{58} + \tfrac{1}{3}Q_{58}\Big)(1 - P_{35}), \qquad V_n = \tfrac{2}{3}\,Q_{58}\,(1 - P_{35})$$

Catholic Republicans: cross-pressured the other way. The same one-third switch, with a doubled nonvoting penalty on the Kennedy side and a tripled one on the Nixon side:

$$V_k = \Big(P_{58} + \tfrac{1}{3}Q_{58}\Big)(1 - 2P_{35}), \qquad V_n = \tfrac{2}{3}\,Q_{58}\,(1 - 3P_{35})$$

All others (Jewish, other religion, no religion; all parties). The base averages 1956 and 1958, and the fourfold step is applied to that average:

$$V_k = \Big(\tfrac{P_{58}+P_{56}}{2} - a\Big)(1 - 2P_{35}), \qquad V_n = \Big(\tfrac{Q_{58}+Q_{56}}{2} + a\Big)(1 - 2P_{35})$$

With the margins of the fourfold table set to a + b = $(P_{58} + P_{56})/2$, c + d = $(Q_{58} + Q_{56})/2$ and a + c = $P_{14}\times$ $(P_{58} + Q_{58} + P_{56} + Q_{56})/2$.

6.3 Properties of the system

Three features of the equations are worth stating explicitly.

Conservation within the decided. Every equation moves voters between the two candidates

without creating or destroying them. Before turnout, $V_k + V_n$ equals $P_{58} + Q_{58}$ in the Protestant and Catholic Democratic cells, so the anti-Catholic step and the one-third return are pure transfers. The Catholic Republican cell breaks this symmetry only through its unequal turnout factors.

Domain constraints. The turnout factors are linear in P_{35}, so they become negative for heavy nonvoters: $1 - 2P_{35} < 0$ when $P_{35} > 0.5$, and $1 - 3P_{35} < 0$ when $P_{35} > 1/3$. A type with a nonvoting index above one third would receive a negative Nixon vote in the Catholic Republican cell. Any implementation must check these bounds; ours clips nothing and tests that outputs stay in [0, 1] for realistic inputs. The original programmers guarded against the problem downstream: the state-by-state program sets negative state votes to zero and trims Kennedy-plus-Nixon totals above one (Appendix B.7).

Judgemental parameters. Four constants enter every calculation: the odds ratio 0.6, the return fraction 1/3, and the turnout multipliers 2 and 3. The odds ratio was averaged from the few surveys that asked both questions. The simulation's fourfold-table subroutine held nine odds ratios, as many as the equation sets, though the article reports one (Appendix E.5). The one-third was chosen after reading Kennedy–Nixon trial heats, and the multipliers rest on the cross-pressure literature rather than on an estimate. The authors acknowledge that they did not explore which parameters drove the result (p. 180); Section 12 proposes that exercise.

7. The fourfold-table problem

The model's only non-trivial computation answers one question: of the Democrats who gave anti-Catholic replies, how many had already voted Republican in 1958? Those "hopeless cases" cost Kennedy nothing new. Only anti-Catholics still voting Democratic in 1958 are net losses.

The obstacle was that no respondent answered both questions in most surveys. One respondent of a type was asked his 1958 vote; another, on a different survey, was asked whether he would vote for a Catholic. The joint distribution had to be reconstructed from its margins plus one measure of association.

7.1 Formulation

Within a voter type, cross 1958 vote intention with anti-Catholicism:

1958 vote intention	Anti-Catholic	Not anti-Catholic	Row total
Democratic	a	b	$R = P_{58}$
Republican	c	d	Q_{58}
Column total	$C = P_{14}N$	$N - C$	$N = P_{58} + Q_{58}$

The margins are known for each type. The association was measured on the few surveys carrying both questions: across the larger Protestant Democrat types, the cross-product ratio averaged about 0.6. The unknown a therefore satisfies

$$a+b=R, \qquad a+c=C, \qquad a+b+c+d=N, \qquad \frac{ad}{bc}=\theta=0.6$$

Substituting $b=R-a$, $c=C-a$ and $d=N-R-C+a$ into the odds-ratio condition gives a quadratic in a:

$$(1-\theta)\,a^2 \;+\; \big[\,N-R-C+\theta\,(R+C)\,\big]\,a \;-\; \theta\,R\,C \;=\; 0$$

The article says only that a "was arrived at by the computer". The surviving subroutine EVAL shows that it solved this quadratic. EVAL fixes the margins, imposes the odds ratio and solves for the deviation of a from its value under independence, which is the same equation in a shifted variable (Appendix E.3).

7.2 Existence and uniqueness

Proposition 1. For any margins with $0 < R, C < N$ and any odds ratio $\theta > 0$, the system has exactly one solution with all four cells non-negative.

Proof. All cells are non-negative exactly when a lies in the interval $[L, U]$, with $L = \max(0, R+C-N)$ and $U = \min(R, C)$. Define $g(a) = a\,d(a) - \theta\,b(a)\,c(a)$. On $[L, U]$, a and $d(a)$ are non-negative and increasing in a, while $b(a)$ and $c(a)$ are non-negative and decreasing. So g is strictly increasing. At $a = L$, either $a = 0$ or $d = 0$, so $g(L) = -\theta bc \le 0$. At $a = U$, either $b = 0$ or $c = 0$, so $g(U) = ad \ge 0$. By continuity g has exactly one root in $[L, U]$. □

In practice: when $\theta < 1$ the quadratic's leading coefficient is positive and its constant term negative, so its roots have opposite signs and the feasible root is the positive one. When $\theta = 1$ the equation is linear. EVAL has no branch for that case and would divide by zero; for every other θ it keeps the root of smaller magnitude, which is always the feasible one (Appendix E.3).

7.3 The independence benchmark

At $\theta = 1$ the solution is $a = RC/N = P_{14}P_{58}$. The share of Democrats lost equals the share of the type that is anti-Catholic. This is exactly the naive correction the authors rejected: "it would not suffice merely to subtract the percentage who gave anti-Catholic replies" (p. 178). The odds ratio is therefore the single parameter that measures how far the model departs from naive subtraction. A value below 1 says anti-Catholics were already disproportionately Republican, so fewer Democratic votes are at risk.

7.4 Comparative statics

Implicit differentiation of g(a; θ, C) = 0 gives the two derivatives that drive the model's sensitivity:

$$\frac{\partial a}{\partial \theta} = \frac{b\,c}{a+d+\theta\,(b+c)} > 0, \qquad \frac{\partial a}{\partial C} = \frac{a+\theta\,b}{a+d+\theta\,(b+c)} \in (0,1)$$

Since $V_k = (R-a)(1-2P_{35})$ and $C = P_{14}N$, a Protestant Democrat type's Kennedy share falls

with anti-Catholicism at the rate

$$\frac{\partial V_k}{\partial P_{14}} = -\left(1 - 2P_{35}\right) N \, \frac{a + \theta\, b}{a + d + \theta\,(b + c)}$$

The loss per point of anti-Catholicism is less than one point of the decided vote. It shrinks as θ falls, because more of the anti-Catholics are found among existing Republicans.

7.5 Worked example

Take an illustrative Protestant Democrat type with $P_{58} = 0.55$, $Q_{58} = 0.35$, $P_{14} = 0.30$ and $P_{35} = 0.20$. These are constructed values chosen for clarity, not archive data.

θ	a	b	c	d	V_k	V_n
0.3	0.1105	0.4395	0.1595	0.1905	0.2637	0.2763
0.6 (as used)	0.1417	0.4083	0.1283	0.2217	0.2450	0.2950
1.0 (naive subtraction)	0.1650	0.3850	0.1050	0.2450	0.2310	0.3090
1.5	0.1828	0.3672	0.0872	0.2628	0.2203	0.3197

Every row reproduces its odds ratio to machine precision. Moving from naive subtraction to the authors' $\theta = 0.6$ raises this type's Kennedy index by 1.4 points, about 6 per cent of its value. At $\theta = 0.6$ the derivative $\partial a / \partial \theta$ is 0.076: a 0.1 error in the odds ratio moves the type's Kennedy index by about 0.5 points after the turnout factor.

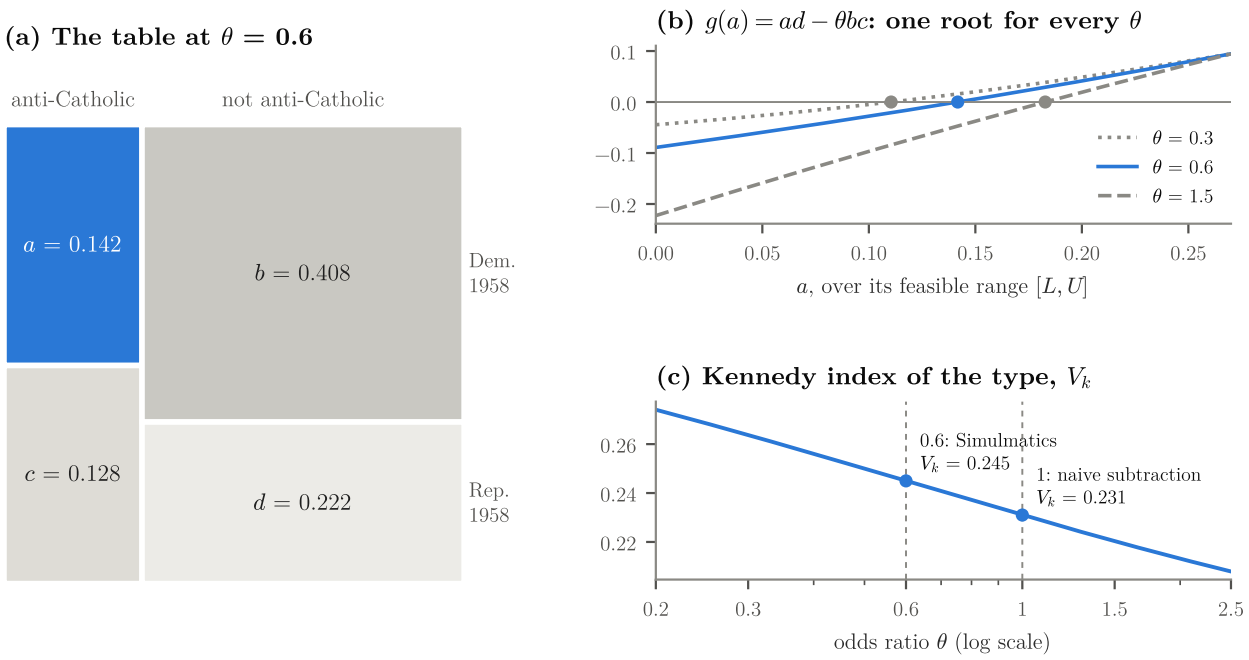


Figure 2: The fourfold-table step for an illustrative Protestant Democrat type ($P_{58} = 0.55$, $Q_{58} = 0.35$, $P_{14} = 0.30$, $P_{35} = 0.20$). (a) With the margins fixed, the authors' odds ratio $\theta = 0.6$ gives $a = 0.142$ anti-Catholic 1958 Democrats. (b) $g(a)$ rises across the feasible range, so every θ has exactly one root (Proposition 1). (c) The type's Kennedy index falls as θ rises; $\theta = 1$ reproduces naive subtraction. Animated version: Animation S2 in the online story map.

8. Why voter-type modelling differs from a global correction

Computing type by type changes the answer only when the model is non-linear and its inputs vary together across types. Pool and Abelson make this argument informally (p. 182): if the same process operated in every type, adding 34 per cent of each Catholic type to the Democrats would equal adding 34 per cent of all Catholics to the total. Interactions, such as turnout varying alongside party loyalty, break the equality. This section makes the argument exact.

8.1 Statement

Let x_t be the input vector of type t (P_{56}, Q_{56}, P_{58}, Q_{58}, P_{35}, P_{14}), f the equation for its cell, and w_t its weight in a state, with weights summing to one. The simulation computes the left-hand side below; a global correction computes something like the right-hand side, the model applied to the state's average voter.

$$\Delta_s \;=\; \sum_t w_t\, f(x_t) \;-\; f\Big(\sum_t w_t\, x_t\Big)$$

Proposition 2. $\Delta_s = 0$ for every set of weights and inputs if and only if f is affine.

This is Jensen's equality case. Every cell equation fails it, because each multiplies a vote share by a turnout factor.

8.2 The bilinear terms are covariances

For the bilinear cells the gap is exactly a weighted covariance. For Protestant Republicans:

$$\sum_t w_t \, P_{56,t} \, (1 - P_{35,t}) \;=\; \bar{P}_{56} \, (1 - \bar{P}_{35}) \;-\; \mathrm{Cov}_w(P_{56},\, P_{35})$$

So working type by type subtracts the covariance between Stevenson support and nonvoting. In the cross-pressured cells the turnout factor is $1 - 2P_{35}$ and the covariance term doubles. The fourfold step adds genuine curvature, since a is a square-root function of the margins. To second order its contribution is $\frac{1}{2}\,\mathrm{tr}(H\Sigma)$, where H is the Hessian of the cell equation and Σ the weighted covariance matrix of the inputs.

8.3 An illustration

Take two illustrative Protestant Democrat types of equal weight. Type A is loyal and participatory: $P_{58} = 0.70$, $Q_{58} = 0.20$, $P_{35} = 0.10$, $P_{14} = 0.10$. Type B is marginal: $P_{58} = 0.40$, $Q_{58} = 0.50$, $P_{35} = 0.35$, $P_{14} = 0.45$. Computed separately and averaged, their Kennedy index is 0.292. Applied once to their averaged inputs, the same equation gives 0.231. Of the 6.1-point gap, 4.9 points are the covariance term: the most Democratic type also votes most. The remaining 1.2 points come from the curvature of the fourfold step.

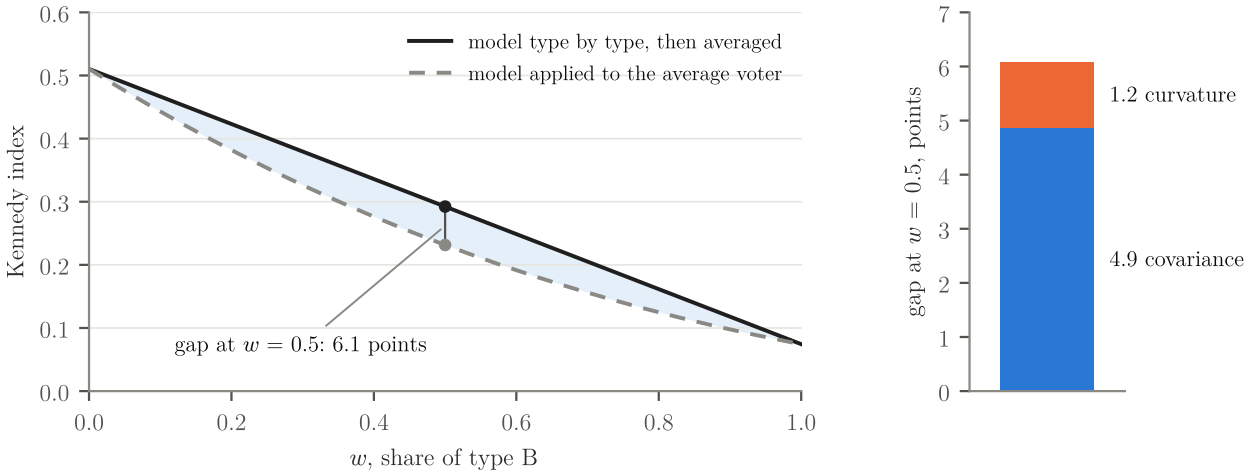


Figure 3: Type-by-type versus average-voter aggregation for the two illustrative Protestant Democrat types of Section 8.3, mixed in shares $1 - w$ and w. The two methods agree only at $w = 0$ and $w = 1$. At $w = 0.5$ the gap is 6.1 points: 4.9 from the covariance of vote and turnout across types and 1.2 from the curvature of the fourfold step. Animated version: Animation S3 in the online story map.

8.4 What this implies for the 1960 result

The type-by-type machinery can only beat a well-chosen global correction where these covariances differ from state to state. Composition effects that are the same everywhere shift every state's index equally and leave the ranking untouched. The authors' own comparison is consistent with that: their simulation (.82) and the best hindsight correction (.83) correlate with the outcome almost

equally. On the authors' own data cards the best correction reaches .880 (Section 10.3), so the gap is wider than they reported. Their claim is therefore not that the simulation was more accurate than the best simple rule. It is that the simulation reached that accuracy before the election, from explicit assumptions, while the simple rule needed a parameter that only hindsight could supply. Section 9 makes the second half of that claim precise.

9. The lucky-guess benchmark

The authors' most quoted comparison is against a simple rule: add k times the state's Catholic percentage to its 1956 Democratic vote. With k = 0.34 the rule correlates .83 with the 1960 result, slightly better than the simulation's .82. No other k does better, and at k = 0.32 or 0.40 the correlation falls below .80 (p. 182). This section shows what that optimum is.

9.1 The correlation as a function of k

Write S for the 1956 Stevenson percentage, C for the Catholic percentage and K for the 1960 Kennedy percentage, all across the 32 states, with variances σ^2 and covariances σ subscripted in the obvious way. Then

$$r(k) \;=\; \operatorname{corr}(S + kC,\; K) \;=\; \frac{\sigma_{SK} + k\,\sigma_{CK}}{\sigma_K \sqrt{\sigma_S^2 + 2k\,\sigma_{SC} + k^2\sigma_C^2}}$$

Setting $r'(k) = 0$ gives a single interior stationary point:

$$k^* \;=\; \frac{\sigma_{CK}\,\sigma_S^2 - \sigma_{SK}\,\sigma_{SC}}{\sigma_{SK}\,\sigma_C^2 - \sigma_{CK}\,\sigma_{SC}}$$

9.2 The optimum is a regression

Proposition 3. Let K = α + $\beta_S S + \beta_C C$ + e be the least-squares regression of the 1960 vote on the 1956 vote and the Catholic share, with $\beta_S > 0$. Then $k^* = \beta_C/\beta_S$, and $r(k^*)$ equals the multiple correlation coefficient R of that regression.

Proof. Correlation is invariant to positive rescaling, so corr(S + kC, K) = corr($\beta_S S + \beta_S kC$, K). Among all linear combinations of S and C, the fitted value $\beta_S S + \beta_C C$ has the highest correlation with K, and that correlation is R by definition. When $\beta_S > 0$, the fitted value is a positive multiple of S + (β_C/β_S) C, so the maximum of r(k) is attained at k = β_C/β_S and equals R. □

We confirmed this numerically on the authors' own series, read from the data cards of their lucky-guess program (Appendix D): Stevenson's 1956 share, Kennedy's 1960 two-party share and their Catholic series, for the 32 states. The closed form gives $k^* = 0.5243$, the regression gives β_C/β_S = 0.33543 / 0.63971 = 0.5243, and both $r(k^*)$ and R equal 0.880.

9.3 Consequences

The lucky guess is not a single fortunate number. It is a two-predictor regression fitted to the very outcome it is scored against. Its .83 is an in-sample multiple correlation, while the simulation's .82 involved no parameter fitted to 1960. A fair comparison must therefore penalise the rule for fitting. Two simple penalties, computed on the authors' own series:

Measure	Value
In-sample r at k^* (= multiple R)	0.880
Adjusted R (32 states, 2 predictors)	0.871
Leave-one-out r (k^* re-estimated without each state)	0.869

The penalties are small here because the rule has only two free parameters and the fit is stable. The point stands nonetheless: the authors' comparison is between a forecast and a fit, and it favours the forecast more than the raw numbers suggest.

Proposition 3 also gives a check on the article. Because r(k) is smooth with a single maximum, the reported values constrain its curvature. A fall from .83 at k = 0.34 to below .80 at k = 0.32 implies a very narrow peak. On the authors' own data the peak is broad: r stays within 0.03 of its maximum for every k from 0.34 to 0.89, and it is .843 at k = 0.32 and .869 at 0.40. The printed table fails this test (Section 10.3).

10. Replication results

The printed ranking holds up: it correlates 0.755 (Spearman) with Kennedy's actual 1960 share across the 32 states. Benchmarks that need only election returns reproduce closely. The lucky guess, recomputed on the authors' own data cards, does not.

Benchmark	Pool and Abelson (1961)	This paper	Status
Simulation vs 1960 result	.82 (Pearson, unpublished index)	0.755 Spearman; 0.573 Kendall	Reproduced from printed ranking
Continuity with 1956	.39	0.427 (returns); 0.436 (authors' cards)	Reproduced closely
Continuity with 1958	.37	—	Needs 1958 House vote by state
Continuity with 1956 and 1958	.44	—	Needs 1958 House vote by state
Contemporaneous trial heats	.53	—	Needs 1958 poll data

Benchmark	Pool and Abelson (1961)	This paper	Status
Best "lucky guess" k and r	0.34, .83	0.52, 0.880	Not reproduced on the authors' own cards

10.1 The printed ranking

The article reports a product-moment correlation of .82 between its Kennedy index and the actual vote. The index itself is not printed, only the rank order it produced. We therefore correlate the ranks. The Spearman coefficient is 0.755 ($p = 5.9 \times 10^{-7}$), Kendall's tau 0.573, and the Pearson correlation of rank with vote share 0.763. Using Kennedy's two-party share instead of his share of all votes changes the Spearman coefficient only to 0.763.

A rank correlation discards the spacing between states, which the Pearson coefficient on the index used. The two coefficients need not agree. Rhode Island and Massachusetts sit far above the other states in both the simulation and the result, and a Pearson coefficient on the index rewards that separation, which ranks ignore. A rank correlation of 0.755 is therefore compatible with the published .82, though it cannot confirm it. The surviving scoring routine, VOTCOR, computes an unweighted product-moment correlation over exactly these 32 states (Appendix D.6), so the .82 was most likely of that kind, like every correlation in this section.

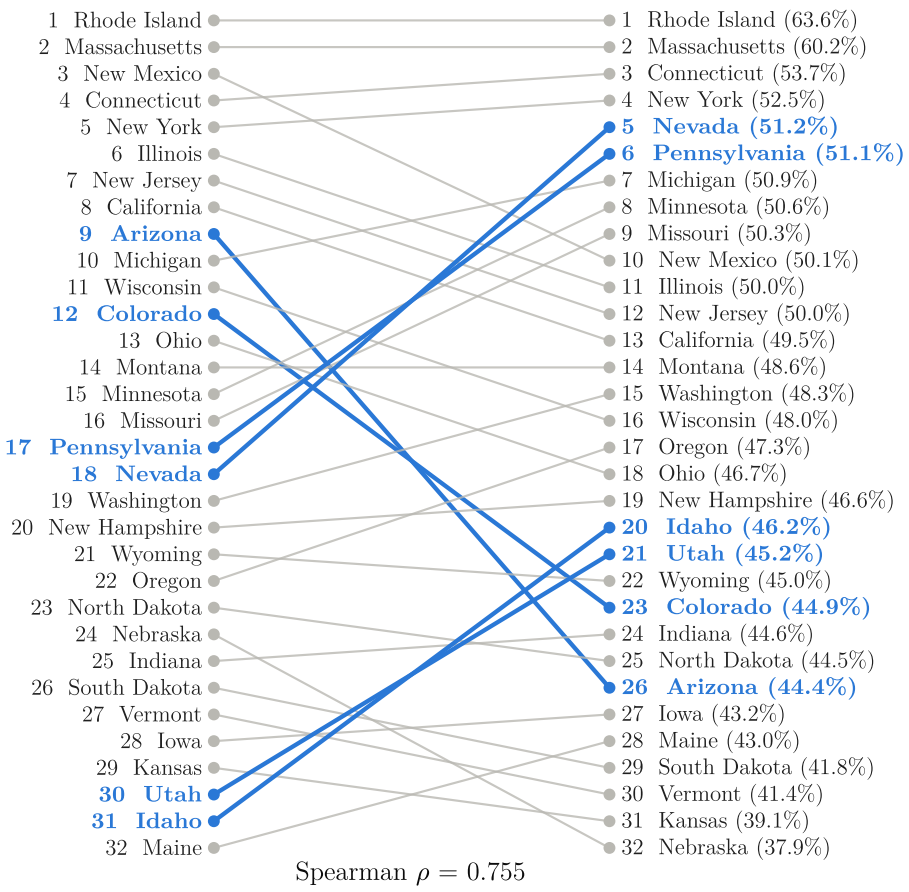


Figure 4: Simulmatics' ranking of the 32 states (August 1960, left) against Kennedy's actual rank by vote share (November 1960, right). Blue: the six states ranked nine or more places from where they finished; five are Western. Spearman $\rho = 0.755$. Sources: Pool and Abelson (1961, pp. 176–177); official 1960 returns. Animated version: Animation S4 in the online story map.

10.2 Where the model failed

The six largest rank errors are:

State	Region	Simulmatics rank	Actual rank	Error
Arizona	West	9	26	17 too high
Nevada	West	18	5	13 too low
Pennsylvania	East	17	6	11 too low
Colorado	West	12	23	11 too high
Idaho	West	31	20	11 too low
Utah	West	30	21	9 too low

Four of the authors' five named failures (Arizona, Nevada, Idaho, Colorado) are among our six

largest errors; New Mexico, the fifth, ranks 3 in the simulation and 10 in fact. Five of the six are in the West, consistent with the authors' diagnosis that pooling Mountain states with the Pacific coast broke the homogeneity assumption. The errors also run in both directions: Arizona and Colorado were overrated while Nevada, Idaho and Utah were underrated. A single regional bias would shift them all one way, so the failure is heterogeneity within the region rather than a regional offset. Pennsylvania is the one large Eastern miss and is not mentioned in the article.

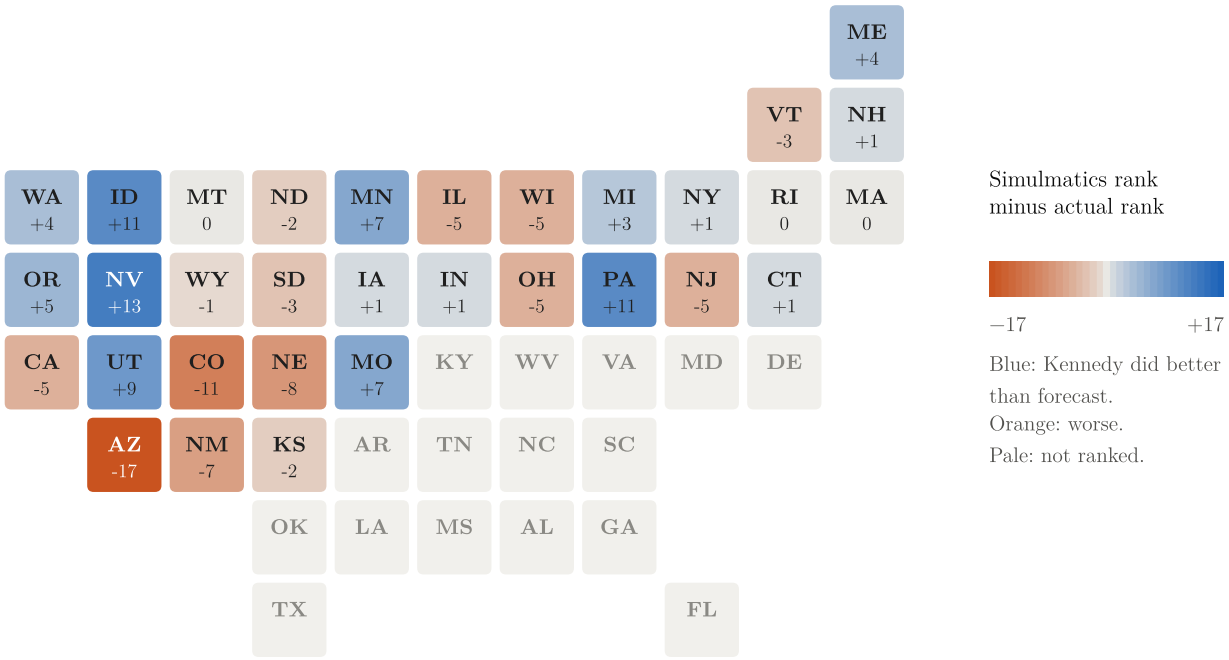


Figure 5: Simulmatics rank minus actual rank on a tile map (tile positions approximate geography). Blue: Kennedy did better than forecast (Nevada +13; Pennsylvania and Idaho +11; Utah +9). Orange: worse (Arizona −17; Colorado −11). The errors cluster in the West and run in both directions. Pale tiles: South and Border states, not ranked. Animated version: Animation S5 in the online story map.

10.3 The lucky guess on the authors' own data

The data cards of the authors' lucky-guess program survive (Appendix D). They give, for all 48 states, Kennedy's 1960 share of the two-party vote, Stevenson's 1956 share and a Catholic series, in the authors' own figures. On the 32 states the rule's correlation is .851 at k = 0.34, not .83, and it is .843 at 0.32 and .869 at 0.40, not below .80. It rises to a maximum of 0.880 at k = 0.52, the least-squares ratio of Proposition 3, and the peak is broad (Figure 6). The article's statement that the simulation's .82 beat every correction except 34 per cent, including 33 and 35, fails too: on these cards every correction from 0.28 to 1 exceeds .82.

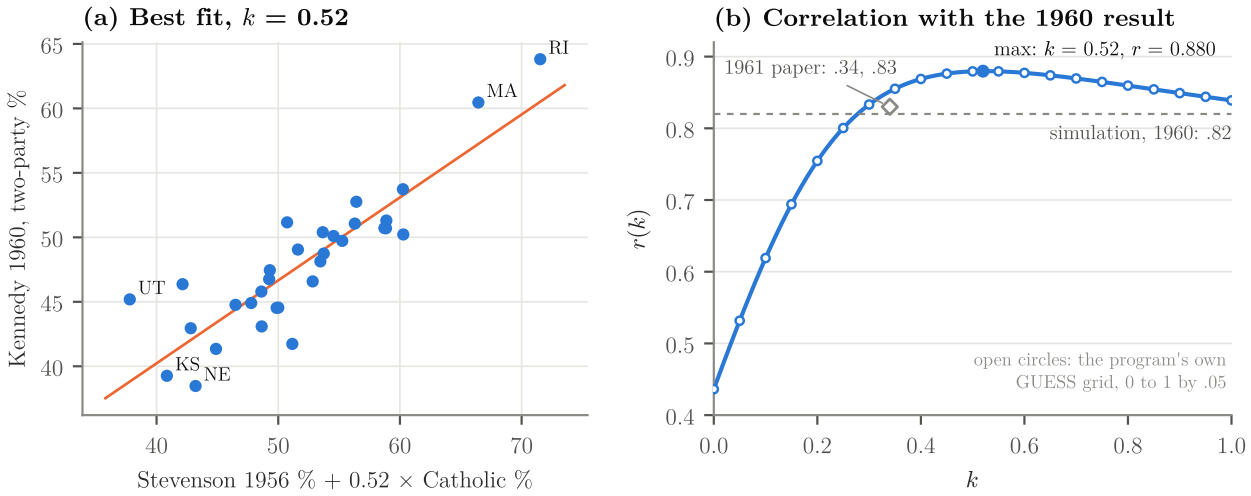


Figure 6: The lucky-guess benchmark on the authors' own data cards (box1batch1.bin, Appendix D), 32 states. (a) Stevenson's 1956 percentage plus 0.52× the Catholic percentage against Kennedy's 1960 two-party percentage. (b) The correlation $r(k)$ peaks at $k = 0.52$, $r = 0.880$; open circles mark the program's own grid of corrections, 0 to 1 by 0.05. The authors reported $k = 0.34$, $r = .83$, with correlations below .80 at 0.32 and 0.40 (Pool and Abelson 1961, p. 182), against the simulation's .82. Animated version: Animation S6 in the online story map.

Weighting states by electoral votes, which the scoring routine was first written to do, moves the maximum to k = 0.65 (r = 0.917). Using all 48 states moves it to 0.39 (r = 0.795). Neither reproduces the published figures. The program's own grid steps by .05 and could not have printed the values at 0.32 to 0.34, so the published numbers come from a run that does not survive, perhaps on different data. One feature of the cards does point to 34 per cent. The correction that makes the average error over the 32 states zero is 0.351, and the one that minimises the root mean square error is 0.353. If the authors chose their optimum by accuracy, the quantity their scoring routine printed first, 34 per cent is close to what they would have found. The correlations they printed with it remain unexplained.

Some results do not depend on these data. The swing from Stevenson in 1956 to Kennedy in 1960 averaged 8.3 points across the 32 states, from returns alone, and Propositions 1–3 are mathematical and hold for any data. On the authors' cards the swing correlates 0.82 with the Catholic share.

11. Animation and supplementary materials

The paper is accompanied by six animations (1920 × 1080, 30 frames per second, 14–20 seconds each), collected in an online story map, and a code kit that reproduces every number in Sections 7–10. Figures 1–6 show the final state of Animations S1–S6 in order: the typology (S1), the fourfold table (S2), the aggregation gap (S3), the shift from forecast to result (S4), the tile map of errors (S5) and the lucky-guess sweep (S6).

11.1 The lucky-guess animation (S6)

The animation visualises Section 9. Its left panel is a scatter of the 32 states: Kennedy's 1960 two-party percentage on the vertical axis, and the predicted index S + kC on the horizontal. As k sweeps from 0 to 1, high-Catholic states slide right faster than the rest and the cloud straightens into a line. The right panel traces r(k) as it is computed, with the authors' optimum (k = 0.34, r = .83) and the simulation's .82 drawn as fixed references. After the sweep, k eases back to its optimum and holds.

The animation shows two things a static table cannot. It shows how much the ranking of states changes as k moves, which is what a correlation summarises. And it shows the shape of the peak, which is the testable claim discussed at the end of Section 9.3. Every frame carries a footnote naming its source, the authors' own data cards. The curve's broad peak at 0.52, well to the right of the published 0.34, is the finding of Section 10.3 in moving form.

11.2 The code kit

File	Contents
simulmatics.py	The nine equation sets, the fourfold solver of Section 7 and the synthetic-state aggregator of Section 5
test_simulmatics.py	26 tests: margins and odds ratio recovered, independence at $\theta = 1$, each printed equation reproduced, monotonicity in P_{14} and θ, the full 480-type table run end to end, and the ports of the surviving programs checked against the reconstruction and against their own data cards
build_voter_types.py	Transcription of the 480 types with spot checks against the printed pages, and the state-to-region map
benchmarks.py	All benchmarks in Section 10, with the lucky guess computed on the authors' data cards
render_sweep.py	The animation, taking the same argument
archive/	Transcriptions of the six surviving decks and Python ports of five of the programs (Appendices A–E)

The kit is written so that missing inputs can be dropped in without code changes. The voter-type input table needs six columns per type (P_{56}, Q_{56}, P_{58}, Q_{58}, P_{35}, P_{14}) and the state weights three (state, type, persons).

12. Limitations and a research agenda

The mathematics can be recovered completely; the numbers cannot yet. Three inputs stand between this reconstruction and a full replication.

12.1 What is missing

The 480 × 52 matrix. A printout of the tape was given to the Roper Center as a condition of access (p. 171). None of the surviving decks contains any of its values. Whether it survives is unknown to us. Failing that, the matrix can be rebuilt from the same archive: Gallup and other surveys of 1952–1958 that carry region, community size, sex, race, occupation, religion and party. Without the authors' question-to-cluster assignments and cut points, a rebuilt matrix will be a reconstruction, not a copy.

The state weights. These combine census counts by sex, community size, occupation and race with survey distributions of religion and party within each region. The weights were never released. Appendix B shows their form, but no values survive. A modern version could use the IPUMS 1960 census sample for the census attributes and the rebuilt survey file for the rest.

Benchmark series. The Catholic series the authors used survives on their data cards (Appendix D), but its source is not named; comparison with the 1952 enumeration of church membership by state from the National Council of Churches would identify it or rule it out. The 1958 House vote by state is in the Clerk of the House's official statistics for that election.

12.2 Threats to validity

Even with all inputs, four sources of divergence remain. The cluster definitions involved unreported judgement (Section 4). The Black types need a religion assignment that the authors did not document (Section 3.4). The equations have unstated domain limits (Section 6.3). And the authors admit that clerical errors occurred under the two-week deadline, though none they found changed a conclusion (p. 173).

12.3 Agenda

1. **Sensitivity analysis.** Vary θ over [0.3, 1.5], the return fraction over [0.2, 0.5] and the turnout multipliers over [1, 3], and report the rank correlation over the grid. The authors called this "a full exploration" that "remains to be made" (p. 180). The simulation computed votes for up to 20 values of a "shift parameter" in one pass, and the state-by-state program reported each (Appendices B.7 and E.5), so systematic variation of at least one parameter was planned; which parameter, and with what results, is not recorded.
2. **Homogeneity test.** Re-estimate with a separate Mountain region and measure how much of the Western error disappears.
3. **Shrinkage.** Replace complete pooling within region with partial pooling across states, turning the synthetic states into a multilevel model, and compare out-of-sample.
4. **The peak test.** Done on the authors' own cards, where the published peak does not reappear

(Section 10.3). The run that produced the printed figures, at corrections of 32 to 40 per cent, may yet be found in the Pool papers.

5. **Timing.** Compare the simulation with state-level trial heats month by month in 1960, which the authors could not do, to find when polls first became as predictive as the model.
6. **State opinion.** Rebuild the state issue profiles of Appendix C from the reconstructed matrix and compare them with modern multilevel estimates of state opinion from the same surveys.

13. Conclusion

The Simulmatics model of 1960 was a post-stratified, rule-based simulation with one estimated association and four judgemental constants. Its core is small enough to write on a page: nine pairs of equations, an odds-ratio constrained 2×2 table with a provably unique solution, and a weighted average over voter types.

The model's reported success survives the checks that public data allow. Its printed ranking of 32 states correlates 0.755 in rank with the outcome, and its known failures cluster where its own homogeneity assumption was weakest. The comparison the authors drew against a simple rule is fairer to them than it first appears, because that rule is a regression fitted after the fact. Their own data cards, which survive, do not reproduce the published optimum: the rule does better, and at a larger correction, than they reported.

What cannot be recovered from print or from the surviving code is the matrix and the weights. The value of a reconstruction is that it states exactly what would be needed, so that the remaining steps are archival rather than conceptual.

Acknowledgements

I want to thank Jill Lepore for bringing to light the accomplishments and history of the Simulmatics Corporation and for sending me the archival files that contained much of the surviving Simulmatics code. I would also like to thank Bob Roswell, founder and director of the Systems Source Computer Museum, for many helpful discussions about the IBM hardware of the period. I also want to thank the librarians and archivists at MIT for their guidance in converting the .bin files that make up the Simulmatics code, in the Ithiel de Sola Pool archive. Special thanks also to Samuel Popkin for his comments on the use of Michigan Algorithm Decoder.

Data and code availability

All code, input tables and outputs are in the supplementary kit (simulmatics_kit.zip): the model, tests, voter-type table, state-region map, election returns, the printed ranking, benchmark outputs and the animation script. The authors' own vote and Catholic series, read from the data cards of Appendix D, are in data/simulmatics_original_series.csv. Every result in Section 10 regenerates with two commands. The archive folder holds transcriptions of the six surviving decks and Python ports of five of the programs (Appendices A–E).

References

Berelson, B. R., Lazarsfeld, P. F. and McPhee, W. N. (1954). *Voting: A Study of Opinion Formation in a Presidential Campaign*. Chicago: University of Chicago Press.

Burdick, E. (1964). *The 480*. New York: McGraw-Hill. Appendix, pp. 311–313.

Ford, R. N. (1950). A rapid scoring procedure for scaling attitude questions. *Public Opinion Quarterly* 14(3), 507–532.

Gelman, A. and Little, T. C. (1997). Poststratification into many categories using hierarchical logistic regression. *Survey Methodology* 23(2), 127–135.

Lepore, J. (2020). *If Then: How the Simulmatics Corporation Invented the Future*. New York: Liveright.

McPhee, W. N. (undated). *A Model for Analyzing Macro-dynamics in Voting Systems*. Columbia University, Bureau of Applied Social Research.

Pool, I. de S. and Abelson, R. (1961). The Simulmatics Project. *Public Opinion Quarterly* 25(2), 167–183. JSTOR

Pool, I. de S., Abelson, R. P. and Popkin, S. L. (1964). *Candidates, Issues, and Strategies: A Computer Simulation of the 1960 Presidential Election*. Cambridge, MA: MIT Press.

Data sources

- 1960 United States presidential election, Wikipedia, results by state (accessed 21 November 2026).
- 1956 United States presidential election, Wikipedia, results by state (accessed 21 November 2026).
- Data cards of the lucky-guess program (box1batch1.bin), Ithiel de Sola Pool papers, MIT Libraries: Kennedy 1960 two-party share, Stevenson 1956 share and Catholic share for 48 states.

The surviving code: an overview

Six programs from the Simulmatics project survive as card decks in the Ithiel de Sola Pool papers at MIT; a seventh deck received was a second copy of the state-by-state program, identical to the first byte for byte. Together they show how data moved through the system (Figure 7).

A print program (Appendix A) reads the cluster summary tape, the 480-type issue matrix of Section 4. A cluster-profile program (Appendix C) combines that tape with the T-prime deck, the state weights of Section 5, to estimate opinion on every cluster in every state. One subroutine of the simulation itself survives: EVAL (Appendix E), which solves the fourfold table of Section 7 and whose shared memory shows that the simulation read cell records from the cluster tape and produced votes for every type under up to 20 values of a shift parameter. The state-by-state program (Appendix B) combines those votes with the weights to predict each state's vote, and passes the prediction to VOTCOR, the scoring routine that survives in the lucky-guess program (Appendix D). That program, written in MAD, carries the authors' own data cards, the 1960 and 1956 results and a Catholic series for 48 states, and computes the lucky-guess benchmark of Section 9. The sixth program, GUTM3A (Appendix F), is a Guttman scaling utility from 1962, outside the 1960 pipeline.

The programs that built the matrix and estimated the weights are lost, as is the main program of the simulation. The appendices follow the order in which the decks were received.

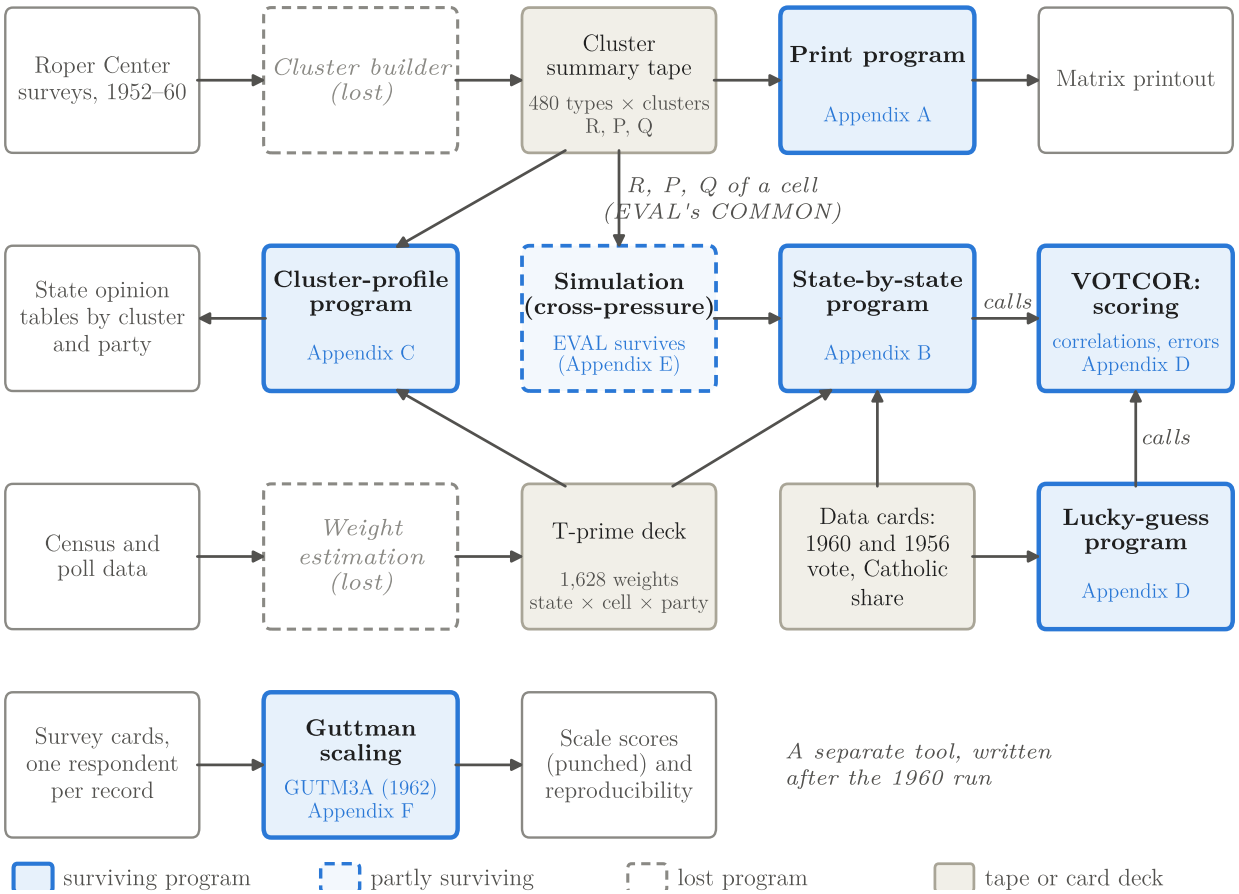


Figure 7: How the surviving programs fit together. Solid blue boxes are surviving programs; the simulation, of which one subroutine survives, is dashed blue; dashed grey boxes are programs known only from their inputs or outputs; shaded boxes are tapes and card decks. The simulation passes the votes V_k and V_n of every voter type to the state-by-state program.

Appendix A. The surviving Simulmatics code

One card deck from the project's code, preserved in the Ithiel de Sola Pool papers at MIT as box5batch2.bin, is transcribed and explained here. It is the print section of the program that built the issue-cluster matrix of Section 4, with two assembly-language subroutines and the control and data cards for one run. It contains no survey data, none of the cluster-building logic and no part of the simulation of Section 6 (one subroutine of which survives separately, Appendix E). Its value is that it shows how the matrix was stored and read, and it settles what the letters P and Q in Pool and Abelson's equations denote. The program that built the state-by-state results is transcribed in Appendix B, the program that estimated opinion on every cluster in every state in Appendix C, the lucky-guess program with its scoring routine VOTCOR and its data cards in Appendix D, the fourfold-table subroutine of the simulation in Appendix E, and a later Guttman scaling program in Appendix F.

A.1 Provenance

The file holds 134 cards, one of them a binary card whose contents are not transcribed here; its fingerprint is c82a1b-421a64. The text was converted from the archive's card images with the guidance of the MIT librarians and archivists (see Acknowledgements). The comment cards identify the main program as "Program No. 7090-0", recoded by Sam Popkin, and the subroutines as coded by George E. McCord III. The program is written in FORTRAN II for the IBM 7090 under the Fortran Monitor System (FMS); the subroutines are in FAP, the 7090's assembly language.

A.2 Transcription

The listing below reproduces the converted deck line for line. In FORTRAN II a character in column 6 marks a continuation line; statement labels occupy columns 1–5.

```
*     END TAPE
*M888-1505,FMS,RESULT,10,15,30000,200
*     XEQ
*     DEAR OPERATOR(2PLY PAPER PLEASE)
*     LIST8
C     SIMULMATICS PROJECT, PROGRAM NO. 7090-0,RECODED BY SAM POPKIN
C     GENERAL PROGRAM TO CREATE CLUSTERS FROM SURVEY DATA.
C
      DIMENSION ITITLE(21), TITLE(14), IYEAR(2), IARRAY(1380), INDEX1(22
     1,60), INDEX2(60), I3(11), I4(11), A(15,20), B(15,20), C(20), AA(30
     20), BB(300), R(60), P(60), Q(60), IR(5), IP(5), IQ(5)
     3 ,RM(60),PM(60),QM(60)
      COMMON ITITLE, IO,JO,KO,LO, R,P,Q
      EQUIVALENCE (ITITLE, TITLE), (ITITLE(15), NSURV), (ITITLE(16), NCL
     1US), (ITITLE(19), IYEAR), (IARRAY, INDEX1), (IARRAY(1321), INDEX2)
     2, (A, AA), (B, BB), (IBOOL1, BOOL1), (IBOOL2, BOOL2), (IBOOL3, BOO
     3L3),(IBOOL4,BOOL4),(ITITLE(17),NYEARS),(ITITLE(18),NMASTR)
      REWIND 9
      DO 13 I=1,481
  13   READ TAPE 9,SAM
    1 READ 100, ITITLE
       DO 42 LL=9,10
      DO 42 ICLUS=1,NCLUS,5
C     PRINT AND PUNCH CLUSTERS SECTION.
C
       LL=9
      IF (ICLUS-1)71,71,69
 69   CONTINUE
      DO 70 K=1,481
```

```
 70   BACKSPACE LL
 71   CONTINUE
      READ TAPE LL,FILEM,NCLUST
      MCLUS = ICLUS + 4
      ILINE = 54
      DO 42 I=1,480
      READ TAPE LL,IO,JO,KO,LO,(R(I),P(I),Q(I),I=1,NCLUS)
      DO 39 J = 1, 5
      IJCLUS = ICLUS + J - 1
      IF (NCLUS - IJCLUS) 38, 37, 37
   37 IR(J) = XBCD1F(R(IJCLUS))
      IP(J) = XBCD2F(P(IJCLUS))
      IQ(J) = XBCD2F(Q(IJCLUS))
      GO TO 39
   38 MCLUS = NCLUS
      IR(J) = IBOOL3
      IP(J) = IBOOL3
      IQ(J) = IBOOL3
   39 CONTINUE
      IF (ILINE - 54) 41, 40, 40
   40 ILINE = 0
      IPAGE = IPAGE + 1
      WRITE OUTPUT TAPE 2,110,IYEAR(LL-8),TITLE,IPAGE,(II,II=ICLUS,MCLUS
     1)
   41 ILINE = ILINE + 1
   42  WRITE OUTPUT TAPE 2, 111,IO,JO,KO,LO,(R(II),P(II),Q(II),
     1 II= ICLUS, MCLUS)
      GO TO 1
C
C
  100  FORMAT (14A6/4I6,2(4XA2)A6)
  101 FORMAT (A3,I3,11(I3,1XI2))
  102 FORMAT (31HONUMBER OF QUESTIONS IN CLUSTERI3,12H EXCEEDS 22./)
  103  FORMAT (I1,1X4I1,F4.0,I2,8X30F2.0)
  104 FORMAT (61HOYEAR IDENTIFICATION IS NOT CONSISTENT THROUGHOUT INPUT
     1 DECK./)
  105 FORMAT (61HOCELL IDENTIFICATION IS NOT CONSISTENT THROUGHOUT INPUT
     1 DECK./)
  106 FORMAT (50HOSURVEYS ARE NOT IN ASCENDING ORDER WITHIN A CELL./)
  107 FORMAT (1H114A6,6H  PAGEI3//)
  108 FORMAT (8H IN YEARI2,7H, CELL I4,8H, SURVEYI3,10H, QUESTIONI3,7H..
```

```
     1.  A=F4.0,3H B=F4.0,3H C=F4.0)
  109 FORMAT (5(A4,1XA4,1XA4)1XA2,I2,1XA4)
  110 FORMAT (6H1YEAR A4,14A6,6H  PAGEI4//5H0CELL2X5(5X1HR7X1HP7X1HQ)/7X
     15(13XI2,8X)/)
  111  FORMAT (1X4I1,4X5(F6.0,F8.4,F8.4))
C
      END
*      FAP
       COUNT   53
*      SIMULMATICS PROJECT, SUBROUTINES XBCD1, XBCD2, CNVRT
*      CODED BY GEORGE E. MCCORD III
*
       ENTRY   XBCD1
       ENTRY   XBCD2
       ENTRY   CNVRT
*
XBCD2  XCA
       FMP     =1000.0
       SLN     4
XBCD1  UFA     =O233000000000
       ANA     =O777777777
       STZ     BCD
       SXA     XZ2,4
       AXT     4,4
       LRS     35
XZ1    DVP     TEN+4,4
       XCA
       ORA     BCD
       ALS     6
       SLW     BCD
       PXD     0,0
       TIX     XZ1,4,1
       SLT     4
       CLS     =O2000000000
       ORA     BCD
       ALS     6
       ORA     =H0000
XZ2    AXT     **,4
       TRA     1,4
*
*
```

```
CNVRT  CAL*    1,4
       LGR     30
       ALS     12
       SLW*    1,4
       PXD     0,0
       LGL     12
       ALS     18
       ORS*    1,4
       PXD     0,0
       LGL     6
       ALS     30
       ORS*    1,4
       TRA     2,4
*
TEN    DEC     1000,100,10,1
*
BCD    BSS     1
*
       END
*     DATA
 MARCH 1 YEAR THREE
    15    50    01    01    03    M3
```

A.3 Control cards

Cards beginning with an asterisk are instructions to the monitor system or the machine-room operator, not part of the program.

Card	Reading
`* END TAPE`	Monitor control card, probably closing a preceding tape section of the job stream
`*M888-1505,FMS,RESULT,10,15,30000,200`	Job card: account or job number M888-1505, run under FMS, job name RESULT; the remaining fields are probably limits on running time, lines printed (30,000) and cards punched (200)
`* XEQ`	Compile the program, then execute it
`* DEAR OPERATOR(2PLY PAPER PLEASE)`	A request to the operator to load two-ply printer paper, which yields a carbon copy
`* LIST8`	A listing option for the compiled program

Card	Reading
`* FAP, COUNT 53`	Start of the assembly-language section, with an estimate of its length in cards
`* DATA`	Start of the data cards

A.4 What the program does

1. **Skip a matrix.** Tape 9 is rewound and 481 records are read into a dummy variable: one header record and 480 voter-type records, that is, one complete matrix. The tape therefore holds at least two matrices, and the run prints the second. The cluster-profile program of Appendix C makes the same choice with a parameter, ITAPE.
2. **Read the title cards** (FORMAT 100): an 84-character page title, then the number of surveys, the number of clusters, the number of years, a master-file number and two year labels.
3. **Print the clusters in bands of five.** For each band the tape is backspaced 481 records to the start of the matrix (except for the first band), the header is read, and all 480 cell records are read in turn. Each record holds a four-digit cell code (IO, JO, KO, LO) followed by R, P and Q for every cluster. A page header (FORMAT 110) is printed every 54 lines, and each cell gives one line (FORMAT 111): the cell code and the five (R, P, Q) triples of the band.
4. **Return to step 2** for the next title card, so that one job could print several reports.

With the 50 clusters of this run the output is 10 bands of 9 pages, or 90 pages, each line 118 printer columns wide.

A.5 Record layouts

Record	Source	Fields
Title card 1	Cards, FORMAT 100	Page title, 84 characters
Title card 2	Cards, FORMAT 100	NSURV, NCLUS, NYEARS, NMASTR (six digits each); two year labels; one further six-character field
Tape header	Tape 9	FILEM, NCLUST
Cell record	Tape 9, 480 per matrix	Cell code IO, JO, KO, LO; then R, P, Q for each cluster
Printed line	FORMAT 111	Cell code; then R (whole number), P and Q (four decimals) for five clusters

Record	Source	Fields
Input card (not used by this section)	FORMAT 103	One digit; four-digit cell code; a four-column count; two-digit survey number; thirty two-digit fields

The data cards of this run give the title MARCH 1 YEAR THREE, 15 surveys, 50 clusters, one year, master file 1, and the year labels 03 and M3. "Year three" plausibly means the third biennial matrix, 1956, but the deck does not say so.

A.6 What R, P and Q are

The printed widths identify the three numbers. R is a whole number, the respondents in the cell who were asked the cluster's questions. P and Q are proportions, the shares giving the pro and the anti reply. The undecided share, $1 - P - Q$, is not stored. These are the "four numbers" of Section 4, with one left implicit.

The same letters appear in the equations of Section 6. Their reading follows directly: P and Q with a year subscript (P_{56}, Q_{58}) are the pro and anti shares of a voting cluster in that year's matrix, and P with a cluster subscript (P_{14}, P_{35}) is the pro share of that cluster. The deck thus confirms the notational point of Section 6.1 from the program side.

This run has 50 clusters where the article reports 52. The deck cannot say whether it predates the final matrix or covers one year in which fewer clusters were available.

A.7 Traces of the cluster builder

Much of what the program declares is never used, and it records what the full program did before this copy was reduced to its print section.

- **Capacity.** The index arrays allow up to 22 questions in each of up to 60 clusters, and FORMAT 102 prints "NUMBER OF QUESTIONS IN CLUSTER … EXCEEDS 22".
- **Input checks.** FORMATs 104–106 report inconsistent year or cell codes and surveys "not in ascending order within a cell": the input was sorted by cell, then survey.
- **Per-question counts.** FORMAT 108 prints a year, cell, survey and question with three counts, A, B and C, most likely the pro, anti and undecided counts that were pooled into each cluster cell (Section 4.2).
- **Card output.** The section is titled "PRINT AND PUNCH", and the values are converted to card characters (IR, IP, IQ), but no punch statement remains.

A.8 The assembly routines

XBCD1 and XBCD2 convert a floating-point number to printable decimal characters in BCD, the 7090's six-bit character code. XBCD1 handles whole numbers such as R. XBCD2 first multiplies by

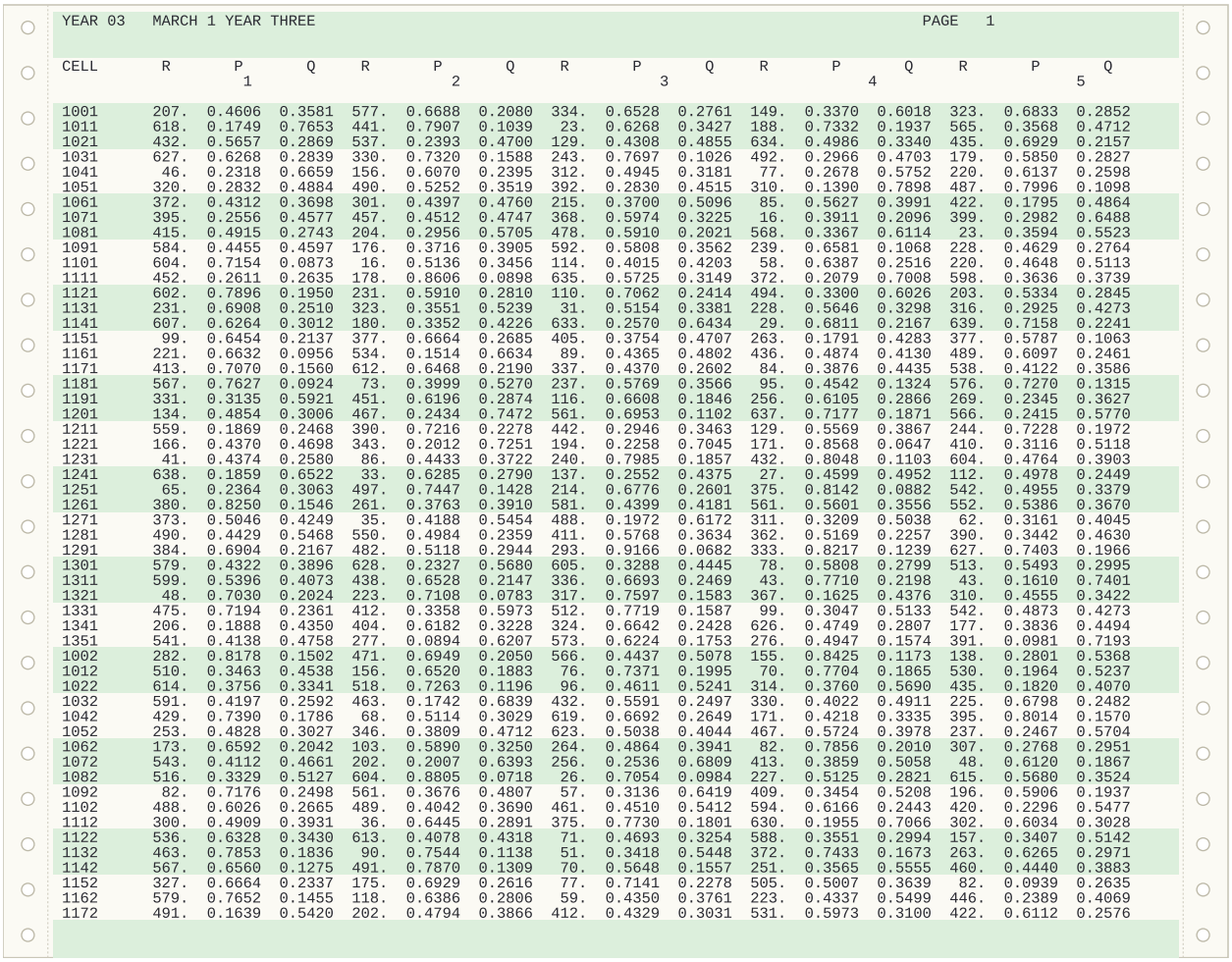
1000, so a proportion keeps three decimal places, and sets sense light 4 so that the shared code can tell the two entries apart. Both use idioms of the period: an unnormalised floating add of a constant (`UFA =O233000000000`) to turn the number into an integer, a loop dividing by 1000, 100, 10 and 1 to extract four digits, and a return address saved by rewriting the program's own instruction (`SXA XZ2,4`, then `AXT **,4`). CNVRT rearranges the six-bit fields within a word; nothing in this deck calls it.

A.9 Open questions

- **A hand patch.** Inside the loop `DO 42 LL=9,10` the statement `LL=9` resets the tape number, so tape 10 is never read. With the 481-record skip in step 1, this looks like a patch to print one particular matrix.
- **A reused counter.** The loop `DO 42 I=1,480` reuses I as the index of the implied loop in `READ TAPE ...,(R(I),P(I),Q(I),I=1,NCLUS)`. In modern Fortran this corrupts the outer loop; whether it did on the 7090 depends on how FORTRAN II kept loop counters.
- **An uninitialised page counter.** IPAGE is incremented but never set to zero.
- **Header alignment.** Running FORMAT 110 through the port shows that the column letters R, P and Q sit two columns left of the right edge of their fields, and that the cluster numbers advance 23 columns per band against 22 for the data, drifting one column per band (Figure 8).
- **The cell code.** The four digits IO, JO, KO, LO are the region (0–4, Border first), two digits for the cell within the region's template, and the party (1 Democrat, 2 Republican, 3 Independent). The cluster-profile program (Appendix C), which reads this same tape, requires its records in exactly this order, by cell and then party, and uses the first digit to choose the region's states; the state-by-state program (Appendix B) uses the same scheme for votes. What the two middle digits encode is not recorded.

A.10 The printout reconstructed

A Python port of the print section reproduces FORMATs 110 and 111 column for column, with the program's paging and banding. Figure 8 shows the first page it produces.



YEAR 03 MARCH 1 YEAR THREE PAGE 1

CELL	R	P	Q	R	P	Q	R	P	Q	R	P	Q	R	P	Q
		1			2			3			4			5	
1001	207.	0.4606	0.3581	577.	0.6688	0.2080	334.	0.6528	0.2761	149.	0.3370	0.6018	323.	0.6833	0.2852
1011	618.	0.1749	0.7653	441.	0.7907	0.1039	23.	0.6268	0.3427	188.	0.7332	0.1937	565.	0.3568	0.4712
1021	432.	0.5657	0.2869	537.	0.2393	0.4700	129.	0.4308	0.4855	634.	0.4986	0.3340	435.	0.6929	0.2157
1031	627.	0.6268	0.2839	330.	0.7320	0.1588	243.	0.7697	0.1026	492.	0.2966	0.4703	179.	0.5850	0.2827
1041	46.	0.2318	0.6659	156.	0.6070	0.2395	312.	0.4945	0.3181	77.	0.2678	0.5752	220.	0.6137	0.2598
1051	320.	0.2832	0.4884	490.	0.5252	0.3519	392.	0.2830	0.4515	310.	0.1390	0.7898	487.	0.7996	0.1098
1061	372.	0.4312	0.3698	301.	0.4397	0.4760	215.	0.3700	0.5096	85.	0.5627	0.3991	422.	0.1795	0.4864
1071	395.	0.2556	0.4577	457.	0.4512	0.4747	368.	0.5974	0.3225	16.	0.3911	0.2096	399.	0.2982	0.6488
1081	415.	0.4915	0.2743	204.	0.2956	0.5705	478.	0.5910	0.2021	568.	0.3367	0.6114	23.	0.3594	0.5523
1091	584.	0.4455	0.4597	176.	0.3716	0.3905	592.	0.5808	0.3562	239.	0.6581	0.1068	228.	0.4629	0.2764
1101	604.	0.7154	0.0873	16.	0.5136	0.3456	114.	0.4015	0.4203	58.	0.6387	0.2516	220.	0.4648	0.5113
1111	452.	0.2611	0.2635	178.	0.8606	0.0898	635.	0.5725	0.3149	372.	0.2079	0.7008	598.	0.3636	0.3739
1121	602.	0.7896	0.1950	231.	0.5910	0.2810	110.	0.7062	0.2414	494.	0.3300	0.6026	203.	0.5334	0.2845
1131	231.	0.6908	0.2510	323.	0.3551	0.5239	31.	0.5154	0.3381	228.	0.5646	0.3298	316.	0.2925	0.4273
1141	607.	0.6264	0.3012	180.	0.3352	0.4226	633.	0.2570	0.6434	29.	0.6811	0.2167	639.	0.7158	0.2241
1151	99.	0.6454	0.2137	377.	0.6664	0.2685	405.	0.3754	0.4707	263.	0.1791	0.4283	377.	0.5787	0.1063
1161	221.	0.6632	0.0956	534.	0.1514	0.6634	89.	0.4365	0.4802	436.	0.4874	0.4130	489.	0.6097	0.2461
1171	413.	0.7070	0.1560	612.	0.6468	0.2190	337.	0.4370	0.2602	84.	0.3876	0.4435	538.	0.4122	0.3586
1181	567.	0.7627	0.0924	73.	0.3999	0.5270	237.	0.5769	0.3566	95.	0.4542	0.1324	576.	0.7270	0.1315
1191	331.	0.3135	0.5921	451.	0.6196	0.2874	116.	0.6608	0.1846	256.	0.6105	0.2866	269.	0.2345	0.3627
1201	134.	0.4854	0.3006	467.	0.2434	0.7472	561.	0.6953	0.1102	637.	0.7177	0.1871	566.	0.2415	0.5770
1211	559.	0.1869	0.2468	390.	0.7216	0.2278	442.	0.2946	0.3463	129.	0.5569	0.3867	244.	0.7228	0.1972
1221	166.	0.4370	0.4698	343.	0.2012	0.7251	194.	0.2258	0.7045	171.	0.8568	0.0647	410.	0.3116	0.5118
1231	41.	0.4374	0.2580	86.	0.4433	0.3722	240.	0.7985	0.1857	432.	0.8048	0.1103	604.	0.4764	0.3903
1241	638.	0.1859	0.6522	33.	0.6285	0.2790	137.	0.2552	0.4375	27.	0.4599	0.4952	112.	0.4978	0.2449
1251	65.	0.2364	0.3063	497.	0.7447	0.1428	214.	0.6776	0.2601	375.	0.8142	0.0882	542.	0.4955	0.3379
1261	380.	0.8250	0.1546	261.	0.3763	0.3910	581.	0.4399	0.4181	561.	0.5601	0.3556	552.	0.5386	0.3670
1271	373.	0.5046	0.4249	35.	0.4188	0.5454	488.	0.1972	0.6172	311.	0.3209	0.5038	62.	0.3161	0.4045
1281	490.	0.4429	0.5468	550.	0.4984	0.2359	411.	0.5768	0.3634	362.	0.5169	0.2257	390.	0.3442	0.4630
1291	384.	0.6904	0.2167	482.	0.5118	0.2944	293.	0.9166	0.0682	333.	0.8217	0.1239	627.	0.7403	0.1966
1301	579.	0.4322	0.3896	628.	0.2327	0.5680	605.	0.3288	0.4445	78.	0.5808	0.2799	513.	0.5493	0.2995
1311	599.	0.5396	0.4073	438.	0.6528	0.2147	336.	0.6693	0.2469	43.	0.7710	0.2198	43.	0.1610	0.7401
1321	48.	0.7030	0.2024	223.	0.7108	0.0783	317.	0.7597	0.1583	367.	0.1625	0.4376	310.	0.4555	0.3422
1331	475.	0.7194	0.2361	412.	0.3358	0.5973	512.	0.7719	0.1587	99.	0.3047	0.5133	542.	0.4873	0.4273
1341	206.	0.1888	0.4350	404.	0.6182	0.3228	324.	0.6642	0.2428	626.	0.4749	0.2807	177.	0.3836	0.4494
1351	541.	0.4138	0.4758	277.	0.0894	0.6207	573.	0.6224	0.1753	276.	0.4947	0.1574	391.	0.0981	0.7193
1002	282.	0.8178	0.1502	471.	0.6949	0.2050	566.	0.4437	0.5078	155.	0.8425	0.1173	138.	0.2801	0.5368
1012	510.	0.3463	0.4538	156.	0.6520	0.1883	76.	0.7371	0.1995	70.	0.7704	0.1865	530.	0.1964	0.5237
1022	614.	0.3756	0.3341	518.	0.7263	0.1196	96.	0.4611	0.5241	314.	0.3760	0.5690	435.	0.1820	0.4070
1032	591.	0.4197	0.2592	463.	0.1742	0.6839	432.	0.5591	0.2497	330.	0.4022	0.4911	225.	0.6798	0.2482
1042	429.	0.7390	0.1786	68.	0.5114	0.3029	619.	0.6692	0.2649	171.	0.4218	0.3335	395.	0.8014	0.1570
1052	253.	0.4828	0.3027	346.	0.3809	0.4712	623.	0.5038	0.4044	467.	0.5724	0.3978	237.	0.2467	0.5704
1062	173.	0.6592	0.2042	103.	0.5890	0.3250	264.	0.4864	0.3941	82.	0.7856	0.2010	307.	0.2768	0.2951
1072	543.	0.4112	0.4661	202.	0.2007	0.6393	256.	0.2536	0.6809	413.	0.3859	0.5058	48.	0.6120	0.1867
1082	516.	0.3329	0.5127	604.	0.8805	0.0718	26.	0.7054	0.0984	227.	0.5125	0.2821	615.	0.5680	0.3524
1092	82.	0.7176	0.2498	561.	0.3676	0.4807	57.	0.3136	0.6419	409.	0.3454	0.5208	196.	0.5906	0.1937
1102	488.	0.6026	0.2665	489.	0.4042	0.3690	461.	0.4510	0.5412	594.	0.6166	0.2443	420.	0.2296	0.5477
1112	300.	0.4909	0.3931	36.	0.6445	0.2891	375.	0.7730	0.1801	630.	0.1955	0.7066	302.	0.6034	0.3028
1122	536.	0.6328	0.3430	613.	0.4078	0.4318	71.	0.4693	0.3254	588.	0.3551	0.2994	157.	0.3407	0.5142
1132	463.	0.7853	0.1836	90.	0.7544	0.1138	51.	0.3418	0.5448	372.	0.7433	0.1673	263.	0.6265	0.2971
1142	567.	0.6560	0.1275	491.	0.7870	0.1309	70.	0.5648	0.1557	251.	0.3565	0.5555	460.	0.4440	0.3883
1152	327.	0.6664	0.2337	175.	0.6929	0.2616	77.	0.7141	0.2278	505.	0.5007	0.3639	82.	0.0939	0.2635
1162	579.	0.7652	0.1455	118.	0.6386	0.2806	59.	0.4350	0.3761	223.	0.4337	0.5499	446.	0.2389	0.4069
1172	491.	0.1639	0.5420	202.	0.4794	0.3866	412.	0.4329	0.3031	531.	0.5973	0.3100	422.	0.6112	0.2576

Figure 8: Page 1 of the program's output, reconstructed by the Python port of FORMATs 110 and 111 (supplementary kit, archive/printout.py) and set on 10-character-per-inch, 6-line-per-inch line-printer paper. Layout, paging and header are the program's. The values are synthetic, because the deck contains no data. The cell codes follow the scheme of A.9 (region, two cell digits, party), confirmed by Appendices B and C; the assignment of cells to the two middle digits is illustrative.

Appendix B. The state-by-state program

The second surviving deck is the program that turned simulated voter-type results into state predictions and compared them with the actual vote. It implements the synthetic states of Section 5 almost exactly as described there, and its built-in checks confirm the structure of the 480-type typology independently of Burdick's appendix.

B.1 Identification

The deck is preserved as Box1batch2.bin in the Ithiel de Sola Pool papers at MIT. It opens with a monitor LABEL card and the tag C*SXS*, and its first comment names it the "program to produce state-by-state simulation results". It is FORTRAN II for the IBM 7090, like the deck in Appendix A. It calls one subroutine, VOTCOR, which is not in the deck, and it contains no data cards. Its own comments (lines 6–24) document its input in unusual detail.

B.2 Transcription

The listing reproduces the converted deck line for line. Five lines (93, 169, 170, 182 and 183, counting from the top) contain a tab character where a comma belongs; the tabs appear here as runs of spaces (see B.8).

```
*     LABEL
C*SXS*
C     PROGRAM TO PRODUCE STATE-BY-STATE SIMULATION RESULTS
C
C
C     FIRST TWO DATA CARDS CONTAIN TITLE, 12A6/12A6.
C     THIRD DATA CARD CONTAINS KAPPA1,KAPPA2,INPUT1,INPUT2,NJOB, 5I10.
C          KAPPA1 AND KAPPA2 CONTROL SUBROUTINE VOTCOR.
C          IF INPUT1 IS LESS THAN, EQUAL, OR GREATER THAN 1, THE TPRIMES
C             ARE READ IN ON CARDS, INPUT TAPE 9, OR TAPE 9,
C             RESPECTIVELY, ACCORDING TO VARIABLE FORMAT FORMT1.
C          IF INPUT2 IS LESS THAN, EQUAL, OR GREATER THAN 1, THE KENNEDY
C             AND NIXON VOTES FOR EACH CELL ARE READ IN ON CARDS, INPUT
C             TAPE 10, OR TAPE 10, RESPECTIVELY, ACCORDING TO FORMT2.
C          THE PROGRAM READS IN DATA, COMPUTES AND WRITES OUTPUT FOR
C             NJOB SETS OF KENNEDY, NIXON VOTES. IF NJOB IS LESS THAN 1,
C             IT IS SET EQUAL TO 1.
C     FOURTH DATA CARD CONTAINS THE VARIABLE FORMAT FORMT1, 12A6.
C     FIFTH DATA CARD CONTAINS THE VARIABLE FORMAT FORMT2, 12A6.
C     NEXT FOUR DATA CARDS CONTAIN VKACT FOR THE 48 STATES, 12F6.5.
C     NEXT FOUR DATA CARDS CONTAIN WEIGHTS FOR 48 STATES, 12F6.5.
C     IF INPUT1 IS LESS THAN 1, DECK OF TPRIMES FOLLOWS, FORMT1.
C     IF INPUT2 IS LESS THAN 1, NJOB DECKS OF KENNEDY, NIXON VOTES FOR
```

```
C          EACH CELL FOLLOW, FORMT2.
C
      DIMENSION TITLE(24),VKACT(48),WEIGHT(48),LCOUNT(5,10,10)
     1,VK(48,4),VN(48,4),SUM(48,4),II(1628),JJ(1628),KK(1628),MM(1628),
     2TPRIME(1628,3),III(480),JJJ(480),KKK(480),LLL(480),VOTEK(480),
     3VOTEN(480),XK1(5,10,10),XN1(5,10,10),XK2(5,10,10),XN2(5,10,10),
     4XK3(5,10,10),XN3(5,10,10),VKNORM(48,4),VNNORM(48,4),VOTE(48)
     5,MCOUNT(48),FORMT1(12),FORMT2(12)
      DIMENSION NAME(48)
C
C     READ PARAMETER CARDS AND PRINT TITLE
C
      READ 200, (TITLE(I),I=1,24)
      PRINT 300, (TITLE(I),I=1,24)
      READ 201, KAPPA1,KAPPA2,INPUT1,INPUT2,NJOB,(FORMT1(IQ),IQ=1,12),
     1(FORMT2(IQ),IQ=1,12),(VKACT(IQ),IQ=1,48),(WEIGHT(IQ),IQ=1,48)
      READ 200,NAME
      IS = 1
      IF (NJOB) 1,5,5
    1 NJOB = 1
C
C     INITIALIZE SUMS AND COUNTERS
C
      PRINT 311
    5 DO 6 I=1,5
      DO 6 J=1,10
      DO 6 K=1,10
    6 LCOUNT(I,J,K) = -111
      DO 7 M=1,48
      MCOUNT(M) = 0
      DO 7 L=1,4
      VK(M,L) = 0.0
      VN(M,L) = 0.0
    7 SUM(M,L) = 0.0
C
C     READ IN T-PRIMES FOR ALL CELLS
C
      IF (IS-1) 9,9,20
    9 IF (INPUT1-1) 10,12,14
   10 READ FORMT1,(II(I),JJ(I),KK(I),MM(I),(TPRIME(I,L),L=1,3),I=1,1628)
      GO TO 20
```

```
   12 READ INPUT TAPE 9, FORMT1, (II(I),JJ(I),KK(I),MM(I),(TPRIME(I,L),
     1L=1,3),I=1,1628)
      GO TO 20
   14 READ TAPE 9, (II(I),JJ(I),KK(I),MM(I),(TPRIME(I,L),L=1,3)I=1,1628)

C
C     READ IN AND INDEX KENNEDY AND NIXON VOTES
C
   20 IF (INPUT2-1) 21,22,23
   21 READ FORMT2,CSHIFT,(III(I),JJJ(I),KKK(I),LLL(I),VOTEK(I),VOTEN(I),
     1I=1,480)
      GO TO 24
   22 READ INPUT TAPE 10, FORMT2, CSHIFT,(III(I),JJJ(I),KKK(I),LLL(I),
     1VOTEK(I),VOTEN(I),I=1,480)
      GO TO 24
   23 READ TAPE 10, CSHIFT,(III(I),JJJ(I),KKK(I),LLL(I),VOTEK(I),
     1VOTEN(I),I=1,480)
   24 DO 25 IV=1,480
      I = III(IV) + 1
      J = JJJ(IV) + 1
      K = KKK(IV) + 1
      L = LLL(IV)
      GO TO (241,242,243),L
  241 XK1(I,J,K) = VOTEK(IV)
      XN1(I,J,K) = VOTEN(IV)
      LCOUNT(I,J,K) = LCOUNT(I,J,K) + 100
      GO TO 25
  242 XK2(I,J,K) = VOTEK(IV)
      XN2(I J,K) = VOTEN(IV)
      LCOUNT(I,J,K) = LCOUNT(I,J,K) + 10
      GO TO 25
  243 XK3(I,J,K) = VOTEK(IV)
      XN3(I,J,K) = VOTEN(IV)
      LCOUNT(I,J,K) = LCOUNT(I,J,K) + 1
   25 CONTINUE
C
C     CHECK FOR ERRORS IN READIND IN VOTEK, VOTEN
C
      DO 28 I=1,5
      DO 28 J=1,10
      DO 28 K=1,10
```

```
      IF (LCOUNT(I,J,K)) 27,28,29
   27 IF (LCOUNT(I,J,K)+111) 29,28,29
   28 CONTINUE
      GO TO 30
   29 IA = I - 1
      JA = J - 1
      KA = K - 1
      PRINT 301,IA,JA,KA,LCOUNT(I,J,K)
      CALL EXIT
C
C     COMPUTE STATE-BY-STATE RESULTS
C
   30 DO 33 IALL=1,1628
      I = II(IALL) + 1
      J = JJ(IALL) + 1
      K = KK(IALL) + 1
      M = MM(IALL)
      IF (LCOUNT(I,J,K)) 60,31,31
   60 PRINT 302, II(IALL),JJ(IALL),KK(IALL),MM(IALL),LCOUNT(I,J,K)
      GO TO 33
   31 VK(M,1) = XK1(I,J,K)*TPRIME(IALL,1) + VK(M,1)
      VN(M,1) = XN1(I,J,K)*TPRIME(IALL,1) + VN(M,1)
      VK(M,2) = XK2(I,J,K)*TPRIME(IALL,2) + VK(M,2)
      VN(M,2) = XN2(I,J,K)*TPRIME(IALL,2) + VN(M,2)
      VK(M,3) = XK3(I,J,K)*TPRIME(IALL,3) + VK(M,3)
      VN(M,3) = XN3(I,J,K)*TPRIME(IALL,3) + VN(M,3)
      VK(M,4) = XK1(I,J,K)*TPRIME(IALL,1) + XK2(I,J,K)*TPRIME(IALL,2) +
     1 XK3(I,J,K)*TPRIME(IALL,3) + VK(M,4)
      VN(M,4) = XN1(I,J,K)*TPRIME(IALL,1) + XN2(I,J,K)*TPRIME(IALL,2) +
     1 XN3(I,J,K)*TPRIME(IALL,3) + VN(M,4)
      SUM(M,4) = TPRIME(IALL,1) + TPRIME(IALL,2) + TPRIME(IALL,3)
     1 + SUM(M,4)
      DO 32 L=1,3
   32 SUM(M,L) = TPRIME(IALL,L) + SUM(M,L)
      LCOUNT(I,J,K) = LCOUNT(I,J,K) + 1
   33 MCOUNT(M) = MCOUNT(M) + 1
      DO 36 M=1,48
      DO 35 L=1,4
      VK(M,L) = VK(M,L)/SUM(M,L)
      VN(M,L) = VN(M,L)/SUM(M,L)
      VKNORM(M,L) = VK(M,L)/(VK(M,L) + VN(M,L))
```

```
   35 VNNORM(M,L) = 1.0 - VKNORM(M,L)
   36 VOTE(M) = VKNORM(M,4)
C
C     CHECK FOR ERRORS IN COMPUTATION
C
   39 DO 46 M=1,48
      IF (M-5) 40,40,41
   40 IF (MCOUNT(M)-16) 61,42,61
   61 PRINT 303, M,MCOUNT(M)
      GO TO 42
   41 IF (MCOUNT(M)-36) 62,42,62
   62 PRINT 304, M,MCOUNT(M)
   42 DO 45 L=1,4
      IF (VK(M,L)) 63,43,43
   63 PRINT 305, M,L,VK(M,L)
      VK(M,L) = 0.0
   43 IF (VN(M,L)) 64,44,44
   64 PRINT 306, M,L,VN(M,L)
      VN(M,L) = 0.0
   44 IF (VK(M,L)+VN(M,L)-1.0) 45,45,65
   65 PRINT 307, M,L,VK(M,L),M,L,VN(M,L)
      ADJ = (VK(M,L) + VN(M,L) - 1.0)/2.0
      VK(M,L) = VK(M    L) - ADJ
      VN(M  L) = VN(M   L) - ADJ
   45 CONTINUE
      IF (SUM(M,4)-SUM(M,1)-SUM(M,2)-SUM(M,3)) 66,46,66
   66  CONTINUE
   46 CONTINUE
      DO 56 I=1,5
      DO 56 J=1,10
      DO 56 K=1,10
      IF (LCOUNT(I,J,K)) 56,50,50
   50 GO TO (51,52,53,54,55),I
   51 IF (LCOUNT(I,J,K)-5) 71,56,71
   52 IF (LCOUNT(I,J,K)-9) 72,56,72
   53 IF (LCOUNT(I  J,K)-11) 73,56,73
   54 IF (LCOUNT(I,J    K)-12) 74,56,74
   55 IF (LCOUNT(I,J,K)-11) 75,56,75
   56 CONTINUE
      GO TO 80
   71 ITEM = 5
```

```
      GO TO 76
   72 ITEM = 9
      GO TO 76
   73 ITEM = 11
      GO TO 76
   74 ITEM = 12
      GO TO 76
   75 ITEM = 11
   76 IA = I - 1
      JA = J - 1
      KA = K - 1
      PRINT 309, IA,JA,KA,LCOUNT(I,J,K),ITEM
      GO TO 56
C
C     COMPUTE CORRELATION AND WRITE OUTPUT
C
   80 PRINT 310, IS,CSHIFT
      CALL VOTCOR(KAPPA1,KAPPA2,WEIGHT,VOTE,VKACT)
      PUNCH 9999,VOTE
9999  FORMAT(12F6.5)
      PRINT 315,((NAME(M),VKACT(M),VOTE(M)),M=1,48)
      PRINT 311
      WRITE OUTPUT TAPE 2,312,(TITLE(N),N=1,24),IS,CSHIFT,(NAME(M),
     1(VK(M,L),
     2VN(M,L),L=1,4),M=1,48)
      WRITE OUTPUT TAPE 2,313,(TITLE(N),N=1,24),IS,CSHIFT,
     1 (NAME(M),(VKNORM(M,
     2L),VNNORM(M,L) ,L=1,4),M=1,48)
      IS = IS + 1
      IF (NJOB-IS) 81,5,5
C
C     END OF RUN PROCEDURE
C
   81 IF (INPUT1-1) 83,82,82
   82 REWIND 9
   83 IF (INPUT2-1) 85,84,84
   84 REWIND 10
   85 PRINT 314
   86 CALL EXIT
C
C     FORMAT SPECIFICATIONS
```

```
C
  200 FORMAT (12A6/12A6)
  201 FORMAT (5I10/12A6/12A6/(12F6.5))
  300 FORMAT (93H1SIMULMATICS PROJECT, PROGRAM TO PRODUCE STATE-BY-STATE
     1 SIMULATION RESULTS, NOW IN OPERATION.//12A6/12A6/1H1)
  301 FORMAT (39H ERROR IN READING VOTEK, VOTEN. LCOUNT(I1,1H,I1,1H,I1,
     13H) =I4,17H. RUN TERMINATED.)
  302 FORMAT (10H FOR CELL 3I1,10H IN STATE I2,10H, LCOUNT =I5,41H. VK,
     1VN, SUM COMPUTED WITHOUT THIS CELL./1H0)
  303 FORMAT (8H MCOUNT(I2,3H) =I3,18H. IT SHOULD BE 16./1H0)
  304 FORMAT (8H MCOUNT(I2,3H) =I3,18H. IT SHOULD BE 36./1H0)
  305 FORMAT (4H VK(I2,1H,I1,4H) = F8.5,29H. IT WILL BE SET EQUAL TO 0.0
     1/1H0)
  306 FORMAT (4H VN(I2,1H,I1,4H) = F8.5,29H. IT WILL BE SET EQUAL TO 0.0
     1/1H0)
  307 FORMAT (4H VK(I2,1H,I1,4H) = F8.5,8H AND VN(I2,1H,I1,4H) = F8.5,
     137H. THEY WILL BE ADJUSTED TO SUM TO 1.0/1H0)
  308 FORMAT (5H SUM(I2,21H,L) FOR L = 1,2,3 IS 3F7.4,29H, WHICH DO NOT
     1ADD UP TO SUM(I2,5H,4) = F7.0/1H0)
  309 FORMAT (8H LCOUNT(I1,1H,I1,1H,I1,3H) =I3,17H. IT SHOULD EQUALI3//)
  310 FORMAT (30H1CORRELATIONS FOLLOW FOR CYCLEI3,35H, WITH THE SHIFT PA
     1RAMETER EQUAL TOF8.5/)
  311 FORMAT (1H1)
  312 FORMAT (1H1,12A6/12A6/1H 20X, 12HOUTPUT CYCLEI3,20H.  SHIFT PARAME
     1TER =F8.5/1H028X,59HKENNEDY, NIXON VOTES AS PERCENTAGES OF ELIGIBL
     2E ELECTORATE./1H013X,9HDEMOCRATS14X,11HREPUBLICANS13X,12HINDEPENDE
     3NTS27X,5HTOTAL/6H STATE5X/
     4               5H  NO.9X,7HKENNEDY6X,5HNIXON6X,7HKENNEDY6X,5HNIXON
     56X,7HKENNEDY6X,5HNIXON18X,7HKENNEDY6X,5HNIXON/1H0/(1XA6 ,6F12.5,12
     6X,2F12.5))
  313 FORMAT (1H1,12A6/12A6/1H 20X,  12HOUTPUT CYCLEI3,20H.  SHIFT PARAM
     1ETER =F8.5/1H028X,54HKENNEDY, NIXON VOTES AS PERCENTAGES OF TWO-PA
     2RTY VOTE./1H013X,9HDEMOCRATS14X,    11HREPUBLICANS13X,12HINDEPENDE
     3NTS27X,5HTOTAL/6H STATE5X/
     4               5H  NO.9X,7HKENNEDY6X,5HNIXON6X,7HKENNEDY6X,5HNIXON
     56X,7HKENNEDY6X,5HNIXON/1H0/(1XA6 ,6F12.5,12
     6X,2F12.5))
  314 FORMAT (1H140X,11HEND OF RUN.)
 315  FORMAT(50H ACTUAL AND PREDICTED KENNEDY VOTE FOR EACH STATE   /
     128H STATE   ACTUAL   PREDICTED   /  (1XA6,F8.5,F9.5   ) )
C
```

```
      END
```

B.3 Inputs

Input	Source	Contents
Title	Two cards (12A6/12A6)	Run title, printed on every table
Parameters	One card (5I10)	KAPPA1, KAPPA2 (passed to VOTCOR); INPUT1, INPUT2 (where the T-primes and votes come from); NJOB (number of vote sets)
Formats	Two cards (12A6)	FORMT1 and FORMT2, read at run time and used as the formats for the T-primes and votes
Actual vote	Four cards (12F6.5)	VKACT, the actual Kennedy vote in each of 48 states
State weights	Four cards (12F6.5)	WEIGHT, one per state, passed to VOTCOR
State names	Four cards	NAME, six characters per state
T-primes	Cards, BCD tape 9 or binary tape 9	1,628 records: cell (II, JJ, KK), state (MM), and three weights, one per party
Votes	Cards, BCD tape 10 or binary tape 10; NJOB sets	A shift parameter CSHIFT, then 480 records: cell (III, JJJ, KKK), party (LLL), Kennedy vote VOTEK, Nixon vote VOTEN

The T-primes are the weights w_{st} of Section 5, given separately for each party: the weight of a cell's Democrats, Republicans and Independents in a state. The votes are the V_k and V_n of Section 6 for each of the 480 voter types.

B.4 What the program does

1. **Read the parameters, the actual vote, the state weights and the names.** A negative NJOB is set to 1.
2. **Read the T-primes once.** They do not change between vote sets.

3. **Read a set of votes** and file each under its cell and party. Each cell starts at −111 and gains 100, 10 or 1 when its Democratic, Republican or Independent vote arrives, so a complete cell reaches 0. Any cell that is neither complete nor empty stops the run.
4. **Build the states.** For each T-prime record, if its cell has votes, the program adds the weighted Kennedy and Nixon votes of each party, $T'_\ell \times V_{k,\ell}$ and $T'_\ell \times V_{n,\ell}$, and the weights T'_ℓ, to the state's running totals, by party ($\ell = 1, 2, 3$) and in total. It then divides each total by its weight. For party ℓ in state s the result is $\sum_c T'_{cs\ell} V_{k,c\ell} / \sum_c T'_{cs\ell}$, the formula of Section 5 applied within each party; the total pools all three parties. A cell with no votes is skipped, with a message.
5. **Convert to two-party shares**, $V_k/(V_k + V_n)$, by party and in total. The total two-party share is the state's prediction, VOTE.
6. **Check the results** (B.5) and guard against impossible values (B.7).
7. **Report.** It calls VOTCOR with KAPPA1, KAPPA2, the state weights, the prediction and the actual vote; punches the 48 predictions on cards in the same format as the actual-vote cards; prints actual against predicted for each state (FORMAT 315); and writes two tables to the output tape, one as "percentages of eligible electorate" (FORMAT 312) and one as "percentages of two-party vote" (FORMAT 313), each with Kennedy and Nixon columns for Democrats, Republicans, Independents and the total.
8. **Repeat** steps 3–7 for each of the NJOB vote sets, each labelled with its cycle number and shift parameter.

B.5 What its checks establish

The program's checks hard-code the structure of the typology.

- **Cells per state.** States 1–5 must have exactly 16 cells and every other state 36 (messages 303 and 304). With five Border states first, $5 \times 16 + 43 \times 36 = 1{,}628$, the number of T-prime records. At three parties per cell this is 48 and 108 voter types per state, as in Burdick's appendix and the article's "at most 108 voter types" (p. 175).
- **States per region.** After the states are built, each cell's counter has risen by one for every state that used it, and the program requires 5, 9, 11, 12 and 11 for regions 0 to 4. These are the sizes of the Border, East, South, Midwest and West regions in Burdick's list, in that order. Every state therefore uses every cell of its region: the homogeneity assumption of Section 5 is built into the code.
- **The voter-type code.** A voter type is indexed by region (0–4, Border first), two digits for its cell within the region's template, and party (1 Democrat, 2 Republican, 3 Independent, following the order of the output columns). Appendix C confirms the same scheme on the cluster summary tape, and settles the reading of the cell code in Appendix A.9.

B.6 Outputs

The eligible-electorate table gives, for each state and each party, the Kennedy and Nixon votes as shares of everyone of voting age, nonvoters included; this is the "Kennedy index" of the 1961 article,

which is "not strictly a percentage". The two-party table renormalises the same figures, and the two-party total is the prediction that is correlated with the actual vote and punched. The program does not print a ranking; the ranking in the 1961 article would have been read off these predictions.

B.7 What the deck adds to the paper

- **The basis of the index.** The simulated votes are shares of the eligible electorate, as the turnout factors of Section 6 imply, and the comparison with the result uses two-party shares.
- **An unweighted correlation.** VOTCOR receives a weight for every state and two control switches. It is not in this deck, but a MAD version with the same five arguments survives (Appendix D). That version ignores the weight and both switches, scores eight groups of states including the 32 that the 1961 article ranks, and computes every statistic unweighted, having been switched from electoral-vote weights. If the published .82 came from it, it is an unweighted correlation over the 32 states, which is what Section 10 computes.
- **Parameter variation.** The program runs repeated cycles, each with its own vote set and a "shift parameter" CSHIFT that is printed in every heading. The 1961 article says a full exploration of the parameters remained to be made (p. 180); this program was built for systematic variation of at least one. The simulation's own shared memory holds votes for all 480 types under up to 20 values of CSHIFT (Appendix E.5), so the variation was produced in a single simulation pass. Which parameter CSHIFT is, and when the runs were made, neither deck says.
- **Guards against impossible values.** A negative state vote is set to zero, and a Kennedy-plus-Nixon total above one is reduced by half the excess on each side, each with a message. Section 6.3 notes that the turnout factors can go negative for heavy nonvoters; the programmers evidently anticipated trouble of this kind. The guards act after the two-party shares and the prediction have been computed, so they change only the eligible-electorate table.

B.8 Transcription issues and quirks

- **Tabs for commas.** In five lines a tab stands where a comma belongs, as in `XN2(I J,K)` and `VK(M L)`. FORTRAN II could not have compiled these statements, so the tabs are almost certainly faults of the card-image conversion, not of the cards.
- **A missing comma.** Line 68 lacks the comma before `I=1,1628` in its input list. This could be a conversion fault or a slip on the original card.
- **A disabled check.** The test that the three party weights of a state add up to its total weight leads to a CONTINUE, and its message (FORMAT 308) is never printed.
- **A misplaced page eject.** The page eject at line 47 runs only when NJOB is negative, because the branch for a non-negative NJOB jumps past it.

B.9 A Python port

A line-by-line Python port of the program is in the supplementary kit (archive/sxs.py). Given the 480 voter types of Section 3, the region lists of Burdick's appendix and the region codes of B.5, it

reproduces the program's input structure exactly: 1,628 T-prime records, and no messages from any of its structural checks. Given identical synthetic votes and weights, its predictions agree with the synthetic-state function of the reconstruction (Section 5) to within 2.2×10^{-16}, the limit of machine precision. Both results are part of the kit's test suite. VOTCOR is not in this deck; the port passes its results to the port of the surviving MAD version (Appendix D) through the same five-argument call.

Appendix C. The cluster-profile program

The third surviving deck applies the synthetic states of Section 5 to the whole issue matrix. For every state, every cluster and each party it computes the share of the eligible population holding the pro and the anti position, and totals across parties. In modern terms it is synthetic estimation of state opinion on every issue in the archive, the task that multilevel regression and post-stratification now serves. It is also the deck that ties the other two together.

C.1 Identification

The deck is preserved as Box1batch10.bin in the Ithiel de Sola Pool papers at MIT. Its job card, `M888-1505,FMS,DEBUG,3,3,200,000`, carries the same account number as the deck in Appendix A, but the job is named DEBUG and its limits are far smaller. The first comment names the program the “program to produce state-by-state printout of all clusters”. It is FORTRAN II for the IBM 7090, calls no subroutines and contains no data cards.

C.2 Transcription

The listing reproduces the converted deck line for line.

```
*     M888-1505,FMS,DEBUG,3,3,200,000
*     LIST
C     PROGRAM TO PRODUCE STATE-BY-STATE PRINTOUT OF ALL CLUSTERS
C     FIRST DATA CARD CONTAINS TITLE, 12A6.
C     SECOND DATA CARD CONTAINS ITAPE, I1.
C     THIRD DATA CARD CONTAINS THE VARIABLE TPRIME FORMAT, 12A6.
C     NEXT FOUR DATA CARDS CONTAIN STATE NAMES, 4(12A6).
C     DECK OF 1628 TPRIMES FOLLOWS, FORMAT
C
      DIMENSION R(60),P(60),Q(60),IMISS(20),PP(60,3),QQ(60,3),
     1PSUM(48,60,3),QSUM(48,60,3),TSUM(48,60,3),ISTATE(5),T(3),N(3),
     2STATE(48),TITLE(12),FORMAT(12)
      COMMON NCLUS,I,J,K,L,R,P,Q,NMISS,IMISS,PP,QQ,PSUM,QSUM,TSUM
C
C     READ PARAMETERS AND PRINT TITLE
C
      READ 200, (TITLE(I),I=1,12)
      PRINT 300, (TITLE(I),I=1,12)
      READ 201, ITAPE,(FORMAT(K),K=1,12),(STATE(M),M=1,48)
C
C     INITIALIZE SUMS AND COUNTERS, AND POSITION INPUT TAPE
C
      DO 1 M=1,48
      DO 1 N=1,60
```

```
      DO 1 L=1,3
      PSUM(M,N,L) = 0.0
      QSUM(M,N,L) = 0.0
    1 TSUM(M,N,L) = 0.0
      ISTATE(1) = 5
      ISTATE(2) = 9
      ISTATE(3) = 11
      ISTATE(4) = 12
      ISTATE(5) = 11
      IF (ITAPE)  2,2,3
    2 ITAPE = 1
    3   INDEX = 1
  495 READ TAPE 9, FILE, NCLUS
      IF (INDEX - ITAPE) 50, 20, 20
   50 DO 51 IXYZ=1,480
   51 READ TAPE 9, I,J,K,L,(R(IX),P(IX),Q(IX),IX=1,NCLUS)
      INDEX = INDEX + 1
      GO TO 495
C
C     BEGIN CYCLE AND READ IN DATA
C
   20 DO 36 ICYCLE=1,160
      DO 25 LQ=1,3
      READ TAPE 9, I,J,K,L,(R(IX),P(IX),Q(IX),IX=1,NCLUS)
      IF (L-LQ) 70,21,70
   21 DO 25 NQ=1,NCLUS
      PP(NQ,LQ) = P(NQ)
   25 QQ(NQ,LQ) = Q(NQ)
C
C     READ IN T-PRIMES
C
   30 IIALL = ISTATE(I+1)
      DO 39 IALL=1,IIALL
      READ FORMAT,IA,JA,KA,M,(T(LQ),LQ=1,3)
      IF (IA-I) 74,31,74
   31 IF (JA-J) 74,32,74
   32 IF (KA-K) 74,33,74
C
C     COMPUTE SUMS
C
   33 DO 35 LQ=1,3
```

```
      DO 35 NQ=1,NCLUS
      PSUM(M,NQ,LQ) = PP(NQ,LQ)*T(LQ) + PSUM(M,NQ,LQ)
      QSUM(M,NQ,LQ) = QQ(NQ,LQ)*T(LQ) + QSUM(M,NQ,LQ)
   35 TSUM(M,NQ,LQ) = T(LQ) + TSUM(M,NQ,LQ)
   39 CONTINUE
   36 CONTINUE
C
C     COMPUTE FINAL OUTPUT
C
      DO 47 M=1,48
      DO 45 N=1,NCLUS
      A = 0.0
      B = 0.0
      C = 0.0
      DO 43 L=1,3
      A = A + PSUM(M,N,L)
      B = B + QSUM(M,N,L)
      C = C + TSUM(M,N,L)
      IF (TSUM(M,N,L)) 75,75,41
   41 PSUM(M,N,L) = PSUM(M,N,L)/TSUM(M,N,L)
      QSUM(M,N,L) = QSUM(M,N,L)/TSUM(M,N,L)
   43 CONTINUE
      IF (C) 76,76,44
   44 TSUM(M,N,1) = A/C
      TSUM(M,N,2) = B/C
   45 CONTINUE
   47 CONTINUE
C
C     WRITE OUTPUT TWO WAYS
C
      DO 55 M=1,48
      WRITE OUTPUT TAPE 2, 310, (TITLE(I),I=1,12),M,STATE(M),
     1(N,(PSUM(M,N,L),QSUM(M,N,L),L=1,3),(TSUM(M,N,L),L=1,2),N=1,NCLUS)
   55 CONTINUE
      DO 59 N=1,NCLUS
      WRITE OUTPUT TAPE 2, 311, (TITLE(I),I=1,12),N,(M,STATE(M),
     1(PSUM(M,N,L),QSUM(M,N,L),L=1,3),(TSUM(M,N,L),L=1,2),M=1,48)
   59 CONTINUE
C
      REWIND 9
      PRINT 312
```

```
      CALL EXIT
C
C     ERROR SECTION
C
   70 PRINT 302, I,J,K,L,LQ
      CALL EXIT
   74 PRINT 304, I,J,K,IA,JA,KA,M
      CALL EXIT
   75 PRINT 305, M,N,L,TSUM(M,N,L)
      PSUM(M,N,L) = 0.0
      QSUM(M,N,L) = 0.0
      GO TO 43
   76 LA = 4
      PRINT 305, M,N,LA,C
      TSUM(M,N,1) = 0.0
      TSUM(M,N,2) = 0.0
      GO TO 45
C
C     FORMAT SPECIFICATIONS
C
  200 FORMAT (12A6)
  201 FORMAT (I1/12A6/12A6/12A6/12A6/12A6)
  300 FORMAT (99H1SIMULMATICS PROJECT, PROGRAM TO PRODUCE STATE-BY-STATE
     1 PRINTOUT OF ALL CLUSTERS, NOW IN OPERATION.//12A6/1H1)
  302 FORMAT (10HOFOR CELL 4I1,30H ON CLUSTER SUMMARY TAPE, LQ =I2,17H.
     1RUN TERMINATED.)
  304 FORMAT (6HOCELL 3I1,39H ON THE CLUSTER SUMMARY TAPE, AND CELL 3I1,
     146HIN THE TPRIME DECK ENCOUNTERED SIMULTANEOUSLY./
     216H RUN TERMINATED.)
  305 FORMAT (6HOTSUM(I2,1H,I2,1H,I1,3H) =F8.5,64H. THE CORRESPONDING CL
     1USTER PERCENTAGES WILL BE SET EQUAL TO 0.0)
  310 FORMAT (1H125X,12A6/1HO14X,92HPROS AND ANTIS FOR ALL CLUSTERS, AS
     1PERCENTAGES OF ELIGIBLE VOTING POPULATION IN THIS STATE./1HO56X,
     25HSTATEI3,3H.  A6//8HOCLUSTER6X,9HDEMOCRATS14X,11HREPUBLICANS13X,
     312HINDEPENDENTS27H,5HTOTAL/8H   NO.  3X,3HPRO8X,4HANTI9X,3HPRO8X,
     44HANTI9X,3HPRO8X,4HANTI21X,3HPRO8X,4HANTI//(I4,6F12.5,12X,2F12.5))
  311 FORMAT (1H125X,12A6/1HO14X,93HPROS AND ANTIS FOR THIS CLUSTER, AS
     1PERCENTAGES OF ELIGIBLE VOTING POPULATIONS IN ALL STATES./1HO55X,
     27HCLUSTERI4//7H0 STATE13X,      9HDEMOCRATS14X,11HREPUBLICANS13X,
     312HINDEPENDENTS27H,5HTOTAL/7H0 STATE13X,3HPRO8X,4HANTI9X,3HPRO8X,
     44HANTI9X,3HPRO8X,4HANTI21X,3HPRO8X,4HANTI//(I3,1X,A6,6F12.5,12X,
```

```
   52F12.5))
312 FORMAT (12H1END OF RUN.)
    END
```

C.3 Inputs

Input	Source	Contents
Title	One card (12A6)	Run title
Matrix choice	One card (I1)	ITAPE, which matrix on the tape to use (1 if zero or negative)
T-prime format	One card (12A6)	Run-time format for the T-prime records
State names	Four cards	Names of the 48 states
Cluster summary tape	Tape 9	One or more matrices, each a header (FILE, NCLUS) and 480 records: cell (I, J, K), party (L), then R, P, Q for each cluster, ordered by cell and then party
T-prime deck	Cards	1,628 records (IA, JA, KA, M, three weights), sorted by cell and, within a cell, by the states of the cell's region

C.4 What the program does

1. **Read the title, ITAPE, the T-prime format and the state names.**
2. **Move to the chosen matrix**, reading and discarding a header and 480 records for each earlier matrix on the tape.
3. **Read the matrix one cell at a time**, 160 cells in all. Each cell must arrive as three consecutive records for parties 1, 2 and 3, or the run stops. The program keeps each party's P and Q for every cluster; it reads R but never uses it.
4. **Read the cell's T-prime records**, one for each state in the cell's region: 5, 9, 11, 12 or 11 records for regions 0 to 4. Each must name the same cell as the tape, or the run stops with the message that the two cells were "encountered simultaneously".
5. **Accumulate**, for each of those states, the weighted pro and anti shares of each party, $T'_\ell P_\ell$ and $T'_\ell Q_\ell$, and the weights themselves.
6. **Divide** to obtain, for state s, cluster n and party ℓ, the pro share $\sum_c T'_{cs\ell} P_{c\ell n} / \sum_c T'_{cs\ell}$, the same for the anti share, and the corresponding totals across parties. A zero weight sets the shares to zero, with a message.

7. **Print two ways**: for each state, all clusters (FORMAT 310); and for each cluster, all 48 states (FORMAT 311). Both are headed "pros and antis … as percentages of eligible voting population". With 50 clusters a run produces 98 reports.

C.5 What it establishes

- **State opinion on every issue.** The synthetic states were not used only for the vote. The system produced a profile of every state, by party, on every cluster in the matrix. The 1961 article describes the synthetic states only in connection with the simulation.
- **A shared pipeline.** The tape this program reads, which it calls the "cluster summary tape", has the record layout that the program in Appendix A prints, down to the header record and the 480 cell records. The weights it reads are the T-prime records of Appendix B, in the same layout. The three programs are parts of one system (Figure 7).
- **The cell code, confirmed.** Records on the tape run by cell and then party, so the fourth digit of the cell code is the party. The first digit is the region, since it selects how many states' weights follow: 5, 9, 11, 12 and 11 for regions 0 to 4, the counts that Appendix B also hard-codes. The program assumes exactly 160 cells, $4 \times 36 + 16$, and together with the region sizes this gives the 1,628 T-prime records of Appendix B. The T-prime deck was sorted by cell and then by state.
- **No weighting by sample size.** The respondent counts R are read but never used. Each cell's proportions enter a state's estimate with their full population weight, however few respondents they rest on. This is the complete pooling within region described in Section 5, with no allowance for the precision of each cell.

C.6 Quirks

- **A garbled header.** In both report formats the column heading for the totals reads `27H,5HTOTAL`, where the state-by-state program has `27X,5HTOTAL`. As written, the H makes the next 27 characters print as literal text, which would garble the heading. This fits the job name DEBUG, though a conversion fault cannot be ruled out.
- **An array used as a counter.** N is declared as an array, `N(3)`, but used throughout as a loop counter and in the output lists.
- **A variable named FORMAT.** The run-time format is held in an array called FORMAT, a name FORTRAN II did not reserve.
- **Memory.** The three main arrays, each $48 \times 60 \times 3 = 8{,}640$ words, take 25,920 of the 7090's 32,768 words of memory. This plausibly explains the limit of 60 clusters, which the program in Appendix A shares.
- **The job card.** DEBUG's limits (3, 3, 200, 000) against RESULT's (10, 15, 30000, 200) in Appendix A support the reading that the last two fields cap the lines printed and the cards punched; a debugging run punches none.

C.7 A Python port

A line-by-line Python port is in the supplementary kit (archive/stcl.py). Built from the typology of Section 3 and the region lists of Burdick's appendix, a synthetic cluster tape and T-prime deck satisfy every ordering check the program makes. On those inputs the port's state profiles agree with the synthetic-state function of the reconstruction, applied cluster by cluster and party by party, to within 2.8×10^{-16}. A deck whose T-prime records are out of step with the tape stops the port as it stopped the original. All three checks are in the kit's test suite.

Appendix D. The lucky-guess program and VOTCOR

The fourth deck is the only one that carries data. It holds a short main program that computes the "lucky guess" of Section 9 over a grid of corrections, the subroutine VOTCOR that scores any set of state predictions against the result, and the data cards for both: the 1960 result, the 1956 result and a Catholic series for all 48 states. It is the program behind the comparison on p. 182 of the article, or a close relative of it, and VOTCOR is the scoring routine that the state-by-state program of Appendix B calls.

D.1 Identification

The deck is preserved as box1batch1.bin in the Ithiel de Sola Pool papers at MIT (fingerprint a0e328-f11be5). Its job card, `M888-1505,FMS,RESULT,5,5,12000,2000`, carries the account number of Appendices A and C and the job name RESULT. Unlike the other decks it is written in MAD, the Michigan Algorithm Decoder, an ALGOL-like language that ran on the IBM 7090 alongside FORTRAN. MAD statements read almost as English: `THROUGH LOOP, FOR GUESS=0.,.05,GUESS.G.1.0` is a loop, `WHENEVER ... OR WHENEVER ... OTHERWISE ... END OF CONDITIONAL` a conditional, `.G.` and `.LE.` comparisons, `.P.` a power, and a name ending in a period, such as `SQRT.` or `VOTCOR.`, a function. `VECTOR VALUES` presets data, `R` in column 11 marks a remark, and formats are written between dollar signs.

D.2 Transcription

The listing reproduces the converted deck line for line, including the data cards; columns 73–80 of the data cards carry identification labels (VKACT 1 to 4, V56ACT to V56ACT4). A trailing line naming the file and its fingerprint was added by the conversion and is omitted.

```
*M888-1505,FMS,RESULT,5,5,12000,2000
*     XEQ
*     MAD
 MAIN     R PROGRAM TO SET UP ALL VOTE STATISTICS
           DIMENSION  NAME(48),VKACT(48),PREDIC(48)
           DIMENSION TITLE(12)
           READ FORMAT LIST1,NAME(1)...NAME(48)
            VECTOR VALUES LIST1=$(12A6)$
           READ FORMAT LIST2,VKACT(1)...VKACT(48)
           DIMENSION  CATH(48),ADLAI(48)
           READ FORMAT LIST2,ADLAI(1)...ADLAI(48), CATH(1)...CATH(48)
           VECTOR VALUES LIST2=$(12F6.5)$
           A=0.
           B=0.
           C=0.
           THROUGH LOOP,FOR GUESS=0.,.05,GUESS.G.1.0
           THROUGH MIX,FOR I=1,1,I.G.48
```

```
MIX          PREDIC(I)=ADLAI(I)+GUESS*CATH(I)
            PRINT FORMAT LIST10,GUESS
            VECTOR VALUES LIST10=$(1H1/13H LUCKY GUESS'   F8.5)$
            EXECUTE VOTCOR.(A,B,C,PREDIC(1),VKACT(1))
            THROUGH SET1,FOR I=1,12,I.G.37
            INTEGER  I
            PRINT FORMAT LIST4,NAME(I)...NAME(I+11),VKACT(I)...VKACT(I+11
           1),PREDIC(I)...PREDIC(I+11)
 SET1       CONTINUE
            VECTOR VALUES LIST4=$(6HOSTATE  12(A6,3X)/6H ACT.   12(F6.5,3X
           1X)/6HOPRED.  12(F6.5,3X))$
LOOP         CONTINUE
            END OF PROGRAM
*     MAD
             EXTERNAL FUNCTION(K1,K2,F,V1,V2)
             DIMENSION                      AVE1(8),AVE2(8),RIGHT(8),
           1  R(8),WEIGHT(384,A),XX(48)
            DIMENSION DSUM(8)
             VECTOR VALUES A=2,1,48
            VECTOR VALUES ELECT(1)=9,3,8,10,11,5,4,3,  16,4,8,45,16,32,
           112,14,8,12,10,11,8,8,10,8,24,25,20,13,27,12,11,10,13,4,4,6,
           28,4,4,6,4,3,4,3,4,32,6,9
            VECTOR VALUES NUMBER(1)=41.,133.,125.,153.,79.,166.,365.,531.
            VECTOR VALUES NUMBER(1)=5.,9.,11.,12.,11.,16.,32.,48.
             VECTOR VALUES WEIGHT(1)=1.,1.,1.,1.,1.
             VECTOR VALUES WEIGHT (54)=1.,1.,1.,1.,1.,1.,1.,1.,1.
             VECTOR VALUES WEIGHT(111)=1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.
             VECTOR VALUES WEIGHT(170)=1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
           1 1.
             VECTOR VALUES WEIGHT(230)=1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.
             ENTRY TO VOTCOR.
             THROUGH SET1,FOR I=1,1,I.G.48
            ELECT(I)=1.
             WEIGHT (8,I)=1.
             WHENEVER I.LE.5
                    WEIGHT(6,I)=1.
             OR WHENEVER I.GE.15.AND.I.LE.25
                    WEIGHT(6,I)=1.
             OTHERWISE
                    WEIGHT(7,I)=1.
 SET1        END OF CONDITIONAL
```

```
            THROUGH SET2,FOR J=1,1,J.G.8
            AVE1(J)=0.
            AVE2(J)=0.
           DSUM(J)=0.
SET2        RIGHT(J)=0.
            INTEGER I,J
            FLOATING POINT  NUMBER
            THROUGH SET5,FOR J=1,1,J.G.8
            THROUGH SET3,FOR I=0,1,I.G.47
SET3        XX(I)=WEIGHT(J,I+1)*ELECT(I+1)
            EN=0.
            SV1=0.
            SV2=0.
            SV1SQ=0.
            SV2SQ=0.
            SV1V2=0.
            THROUGH SET4,FOR I=0,1,I.G.47
            X=(V1(I)-V2(I))*XX(I)
            AVE1(J)=AVE1(J)+X/NUMBER(J)
           AVE2(J)=AVE2(J)+(X*X)/(NUMBER(J)*ELECT(I+1))
           WHENEVER V1(I).G..5,DSUM(J)=DSUM(J)+XX(I)
            WHENEVER (V1(I)-.5)*(V2(I)-.5).GE.0.,RIGHT(J)=RIGHT(J)+XX(I)
            SV1=SV1+XX(I)*V1(I)
            SV2=SV2+XX(I)*V2(I)
           SV1SQ=SV1SQ+XX(I)*V1(I)*V1(I)
           SV2SQ=SV2SQ+XX(I)*V2(I)*V2(I)
            SV1V2=SV1V2+XX(I)*V1(I)*V2(I)
SET4        EN=EN +XX(I)
            COV=(SV1V2-(SV1*SV2/EN)).P.2
            SIG1=SV1SQ -((SV1.P.2)/EN )
            SIG2=SV2SQ -((SV2.P.2)/EN)
           RSQD=COV/(SIG1*SIG2)
           AVE2(J)=SQRT.(AVE2(J))
SET5        R(J)=SQRT.(RSQD)
            PRINT FORMAT LIST1,AVE1(1)...AVE1(8),AVE2(1)...AVE2(8),
          1 RIGHT(1)...RIGHT(8),NUMBER(1)...NUMBER(8),R(1)...R(8)
           PRINT FORMAT LIST2,DSUM(1)...DSUM(8)
           VECTOR VALUES LIST1=$(12X6HREGION12X80HBORDER    EAST      SO
          1UTH     MIDWEST   WEST      (16)      (32)      ALL STATES /
          21H011X7HAVERAGE /14X5HERROR 11X 8(F7.3,3X)/12X10HMEAN SQRT.
          38X 8(F8.4,2X)//12X11HPREDICTIONS 7X 8(F4.0,6X)/1H+32X
```

```
          48(1H/F4.0,5X)//12X11HCORRELATION  7X 8(F7.4,3X))$
           VECTOR VALUES LIST2=$(  10X19HJFK ELECTORAL VOTES
          1  8(F8.3,2X))$
            FUNCTION RETURN
            END OF FUNCTION
*     DATA
MD.   DEL.  W.VA. KY.   TENN. MAINE N.H.  VT.   MASS. R.I.  CONN. N.Y.
N.J.  PENN. VA.   N.C.  S.C.  GA.   FLA.  ALA.  MISS. ARK.  LA.   OKLA.
TEXAS OHIO  MICH. IND.  ILL.  WIS.  MINN. IOWA  MO.   N.D.  S.D.  NEB.
KANSASMONT. ARIZ. COL.  IDAHO WYO.  UTAH  NEV.  N.MEX.CALIF.ORE.  WASH.
.53758.50818.52621.46405.46364.42952.46582.41350.60458.63814.53731.52759 VKACT 1
.50397.51304.47282.52106.51240.62580.48485.57657.59565.53861.63814.40980 VKACT 2
.50989.46755.51066.44765.50103.48135.50716.43091.50704.44547.41735.38473 VKACT 3
.39262.48744.44532.45794.46367.44907.45189.51162.50223.49725.47450.49052 VKACT 4
.399  .446  .459  .452  .486  .291  .338  .278  .404  .417  .363  .387  V56ACT
.342  .433  .384  .507  .454  .664  .427  .565  .582  .525  .395  .449  V56ACT2
.440  .389  .441  .397  .403  .378  .461  .407  .501  .381  .416  .345  V56ACT3
.342  .429  .389  .393  .388  .399  .354  .420  .418  .443  .447  .454  V56ACT4
.13185.18377.05750.09208.02224.26361.36597.32852.50087.57367.46050.34071
.37420.29969.03937.01055.01553.01601.05833.03372.02935.02691.31531.04261
.18182.19908.23446.13056.27394.30120.24280.15265.16833.22891.18377.16740
.12783.20842.21091.17911.06417.15146.04611.16787.35532.21074.08860.11955
```

D.3 The data cards

Cards	Format	Contents
4	12A6	State names, in the Simulmatics order: Border (1–5), East (6–14), South (15–25), Midwest (26–37), West (38–48)
4	12F6.5, labelled VKACT	Kennedy's 1960 share of the two-party vote, five decimals
4	12F6.5, labelled V56ACT	Stevenson's 1956 share of the total vote, three decimals (read as ADLAI)
4	12F6.5, unlabelled	The Catholic share of each state's population, five decimals (read as CATH)

The state order is the order of Burdick's appendix and of the state-by-state program, which requires the five Border states first (Appendix B.5). The vote cards use the 12F6.5 layout that the state-by-state program reads for its actual vote (Appendix B.3), so one deck of VKACT cards could serve both programs.

We checked the two vote series against the official returns. ADLAI agrees with Stevenson's share of the total vote to within 0.005 in every state (Colorado; the median difference is 0.0003). VKACT agrees with Kennedy's share of the two-party vote to within 0.0068 (Colorado again), and exactly to five decimals in 15 states. The small differences elsewhere are what one would expect from returns compiled before the official canvass. Rhode Island and Louisiana carry the identical value .63814, which is Louisiana's two-party share to five places; Rhode Island's is .63628. A duplicated field is the likeliest explanation. The Catholic series is not identified on the cards. Its values (Rhode Island .574, Massachusetts .501, Utah .046, Tennessee .022) are of the order of Catholic church membership as a share of population in the 1950s.

D.4 The main program

The main program reads the cards and steps a correction GUESS from 0 to 1 in steps of .05. At each step it forms, for every state,

$$\text{PREDIC}_s \;=\; \text{ADLAI}_s \;+\; \text{GUESS} \times \text{CATH}_s,$$

the rule of Section 9 with $k = \text{GUESS}$, prints a page headed LUCKY GUESS, calls VOTCOR to score PREDIC against VKACT, and lists the 48 states with their actual and predicted values. The run produces 21 pages. Figure 9 shows the page for GUESS = .35, regenerated from the deck's own cards by the Python port.

```
LUCKY GUESS' 0.35000
          REGION            BORDER    EAST      SOUTH     MIDWEST   WEST      (16)      (32)      ALL STATES

          AVERAGE
            ERROR           -0.017    -0.017    -0.020     0.018    -0.006    -0.019    -0.000    -0.007
          MEAN SQRT.         0.0492    0.0254    0.0543    0.0289    0.0346    0.0528    0.0301    0.0392

          PREDICTIONS         3/  5.    8/  9.   10/ 11.   11/ 12.    8/ 11.   13/ 16.   27/ 32.   40/ 48.

          CORRELATION        0.5226    0.9668    0.7404    0.8881    0.7106    0.6827    0.8551    0.7926
        JFK ELECTORAL VOTES   1.000     5.000     7.000     3.000     3.000     8.000    11.000    19.000

STATE  MD.      DEL.     W.VA.    KY.      TENN.    MAINE    N.H.     VT.      MASS.    R.I.     CONN.    N.Y.
ACT.   .53758   .50818   .52621   .46405   .46364   .42952   .46582   .41350   .60458   .63814   .53731   .52759

PRED.  .44515   .51032   .47913   .48423   .49378   .38326   .46609   .39298   .57930   .61778   .52417   .50625

STATE  N.J.     PENN.    VA.      N.C.     S.C.     GA.      FLA.     ALA.     MISS.    ARK.     LA.      OKLA.
ACT.   .50397   .51304   .47282   .52106   .51240   .62580   .48485   .57657   .59565   .53861   .63814   .40980

PRED.  .47297   .53789   .39778   .51069   .45944   .66960   .44742   .57680   .59227   .53442   .50536   .46391

STATE  TEXAS    OHIO     MICH.    IND.     ILL.     WIS.     MINN.    IOWA     MO.      N.D.     S.D.     NEB.
ACT.   .50989   .46755   .51066   .44765   .50103   .48135   .50716   .43091   .50704   .44547   .41735   .38473

PRED.  .50364   .45868   .52306   .44270   .49888   .48342   .54598   .46043   .55992   .46112   .48032   .40359

STATE  KANSAS   MONT.    ARIZ.    COL.     IDAHO    WYO.     UTAH     NEV.     N.MEX.   CALIF.   ORE.     WASH.
ACT.   .39262   .48744   .44532   .45794   .46367   .44907   .45189   .51162   .50223   .49725   .47450   .49052

PRED.  .38674   .50195   .46282   .45569   .41046   .45201   .37014   .47875   .54236   .51676   .47801   .49584
```

Figure 9: The page for GUESS = .35 of the lucky-guess program, regenerated from the program's own data cards by the Python port of the main program and VOTCOR (supplementary kit, archive/votcor.py) and set on 10-character-per-inch line-printer paper. Layout follows the program's FORMATs; the conventions of the 7090's output routines may differ in small details, such as leading zeros. The "/" of PREDICTIONS is overprinted on the line, as the format's `+` carriage control directs.

D.5 VOTCOR

VOTCOR is an external function with five arguments, `(K1,K2,F,V1,V2)`: two control values, a weight, the prediction and the actual vote. That is the shape of the call in the state-by-state program, `CALL VOTCOR(KAPPA1,KAPPA2,WEIGHT,VOTE,VKACT)` (Appendix B.4). Under the Fortran Monitor System, which this deck's control cards invoke, MAD and FORTRAN II both produced relocatable binary for the same loader, so the FORTRAN program could in principle have called this MAD routine as it stands. In this version, however, the first three arguments are never used, while the comment in Appendix B says that KAPPA1 and KAPPA2 "control subroutine VOTCOR". At least one other version therefore existed.

VOTCOR scores the prediction for eight groups of states, held in its own 8×48 array of zeros and ones. Its dimension vector `A = 2, 1, 48` makes the 384-word block WEIGHT a two-dimensional array of 8 rows and 48 columns. The preset cards place ones at linear locations 1, 54, 111, 170 and 230, which are the first states of the five regions in rows 1 to 5. The code then fills row 8 for all states, row 6 for the Border and Southern states, and row 7 for the rest.

Group	Label	States
1	BORDER	1–5
2	EAST	6–14
3	SOUTH	15–25
4	MIDWEST	26–37

Group	Label	States
5	WEST	38–48
6	(16)	1–5 and 15–25: Border and South
7	(32)	6–14 and 26–48: East, Midwest and West
8	ALL STATES	1–48

Group (32) is exactly the set of 32 states that the 1961 article ranks. For each group g, with d_s the prediction minus the actual vote, it prints:

- **Average error**, $\sum_{s \in g} d_s / n_g$;
- **Mean square root**, the root mean square error $\left(\sum_{s \in g} d_s^2 / n_g \right)^{1/2}$;
- **Predictions**, the number of states in which prediction and result fall on the same side of one half, over the number of states;
- **Correlation**, computed as the square root of r^2;
- **JFK electoral votes**, the number of states the prediction gives to Kennedy.

D.6 What VOTCOR establishes

- **An unweighted correlation.** VOTCOR was written to weight states by their electoral votes. The deck presets ELECT to the 1960 electoral votes of the 48 states, which total 531, and presets the group sizes NUMBER to the electoral totals of the eight groups (41, 133, 125, 153, 79, 166, 365, 531). But the next card overwrites NUMBER with the state counts (5, 9, 11, 12, 11, 16, 32, 48), and the first loop of the routine sets every ELECT to 1. Every statistic is therefore unweighted, and "JFK electoral votes" counts states. The weight argument passed by the state-by-state program is ignored. If the published .82 came from this routine, it is an unweighted product-moment correlation over the 32 states, as Section 10 assumes, and not a weighted one, as Appendix B.7 left open.
- **The sign is lost.** VOTCOR squares the covariance before dividing (`COV=(...).P.2`, where `.P.` is MAD's power operator) and takes the square root of the result, so it reports $|r|$. A prediction that ran opposite to the outcome would have printed as a good one. For the predictions at issue the sign is positive and nothing changes.
- **Accuracy as well as correlation.** The routine reports bias, error and correct calls beside the correlation. The 1961 article quotes only correlations.

D.7 What the run shows

The table gives VOTCOR's output for group (32) at each of the program's 21 corrections, from the port run on the deck's own cards, with the correlation for all 48 states for comparison.

GUESS	Mean error	RMS error	Called correctly	Predicted for Kennedy	r (32 states)	r (48 states)
0.00	-0.085	0.1000	21	1	0.436	0.548
0.05	-0.073	0.0872	21	1	0.532	0.596
0.10	-0.061	0.0746	21	1	0.619	0.644
0.15	-0.049	0.0625	22	2	0.694	0.689
0.20	-0.036	0.0511	24	4	0.755	0.728
0.25	-0.024	0.0410	26	6	0.800	0.759
0.30	-0.012	0.0333	27	9	0.833	0.781
0.35	0.000	0.0301	27	11	0.855	0.793
0.40	+0.012	0.0327	27	13	0.869	0.795
0.45	+0.024	0.0400	26	16	0.876	0.789
0.50	+0.036	0.0499	26	18	0.879	0.777
0.55	+0.048	0.0612	24	20	0.879	0.760
0.60	+0.060	0.0733	21	23	0.877	0.740
0.65	+0.072	0.0858	20	24	0.874	0.719
0.70	+0.084	0.0986	18	26	0.870	0.698
0.75	+0.096	0.1115	18	26	0.865	0.677
0.80	+0.108	0.1246	16	28	0.860	0.656
0.85	+0.120	0.1377	16	28	0.854	0.636
0.90	+0.132	0.1510	16	28	0.849	0.617
0.95	+0.145	0.1642	15	29	0.844	0.600
1.00	+0.157	0.1775	15	29	0.839	0.583

Three findings follow.

- **The continuity benchmark.** At GUESS = 0 the prediction is the 1956 vote. VOTCOR gives .436 for the 32 states, against .39 in the article's table. Public returns give .427 (Section 10).
- **The published optimum is not reproduced.** The article reports a correlation of .83 at a correction of 34 per cent, better than at 33 or 35 per cent and below .80 at 32 and 40 per cent. On the authors' own cards the correlation at .34 is .851. It is .843 at .32, .847 at .33, .855 at .35 and .869 at .40, and it keeps rising to a maximum of .880 at .52, the ratio of least-squares coefficients of Proposition 3. No grid value falls below .80 between .25 and 1. The program's own grid in steps of .05 could not have printed values at .32, .33 or .34, so the published figures come from another run. Weighting by electoral votes, as VOTCOR was first written to do, moves the maximum to .65 (.917) and gives .847 at .34. Using all 48 states

moves it to .39 (.795). No variant we tried reproduces both the location and the sharpness of the published peak.

- **A peak in error, not correlation.** On the 32 states the correction that makes the average error zero is .351, and the one that minimises the root mean square error is .353 (RMS .0301). Both lie close to the published 34 per cent, while the correlation keeps rising. One reading is that the published optimum was found by accuracy, the quantity VOTCOR prints first, and reported with the correlation at that point. The article does not say so, and on this reading too the published correlations at .32 and .40 remain unexplained.

D.8 Quirks

- **A character in column 73.** The second card of FORMAT LIST4 ends with `3X` and the continuation card begins with `X)`. The final X stands in column 73, outside MAD's statement field, so the format reads `3X)` and is well formed.
- **A Hollerith count that includes an apostrophe.** `13H LUCKY GUESS'` counts the apostrophe, as the conversion renders that card column, as part of the heading.
- **Arguments passed as zeros.** The main program passes A, B and C, all zero, as VOTCOR's first three arguments, consistent with their being unused.
- **Preset values overwritten.** Two `VECTOR VALUES NUMBER(1)` cards preset the same block; the second wins.

D.9 A Python port

A statement-by-statement Python port of the main program and VOTCOR is in the supplementary kit (archive/votcor.py), with a renderer for the printed page (archive/render_luckyguess.py). The kit's tests check that the port reads the cards as above, that its group sums reproduce both NUMBER cards from the ELECT values, that its correlation equals the absolute value of the Pearson coefficient for every group, and that its maximum over k lies at the least-squares ratio of Proposition 3 and equals the multiple correlation.

Appendix E. EVAL: the fourfold table in the simulation

The fifth deck is a single subroutine from the simulation program itself, the part of the system that ran the cross-pressure equations of Section 6. It solves the fourfold-table problem of Section 7. It is short, but its COMMON block, the shared memory it uses to talk to the lost main program, describes much of that program's data.

E.1 Identification

The subroutine was supplied as a listing without an archive file name. It opens with the monitor cards `* LABEL` and `* LIST8` and the comment "EVAL SUBROUTINE FOR EVALUATING AND ADJUSTING A 2X2 CONTINGENCY TABLE" (the conversion runs the first two words together). It is FORTRAN II for the IBM 7090. It has no job card and no data, as one would expect of a subroutine deck loaded behind a main program.

E.2 Transcription

```
*     LABEL
*      LIST8
C EVALSUBROUTINE FOR EVALUATING AND ADJUSTING A 2X2 CONTINGENCY TABLE
C
      SUBROUTINE EVAL(INUM)
      DIMENSION R(60),P(60),Q(60),TABLE(2,2),COR(9),TITLE(12),IA(200),
     1JA(200),KA(200),LA(200),XTABLE(200),IMISS(20),CSHIFT(20),II(480),
     2JJ(480),KK(480),LL(480),VOTEK(480,20),VOTEN(480,20)
      COMMON NCLUS,I,J,K,L,R,P,Q,NMISS,IMISS,COR,KAPPA,IEVAL,VK,VN,
     1TABLE,PCOL,QCOL,PROW,QROW,EP1,EP2,TITLE,XTABLE,IA,JA,KA,LA,ALPHA,
     2CSHIFT,II,JJ,KK,LL,VOTEK,VOTEN,LIGHT
C
C     COUNT TIMES EVAL CALLED
C
      IEVAL = IEVAL + 1
C
C     OPTION TO SET TABLE(1,1) EQUAL TO XTABLE AND SKIP COMPUTATION
C
      IF (KAPPA) 20,20,29
   29 TABLE(1,1) = XTABLE(IEVAL)
      IF (IEVAL-1) 30,30,31
   30 PRINT 110
   31 IF (IA(IEVAL)-I) 35,32,35
   32 IF (JA(IEVAL)-J) 35,33,35
   33 IF (KA(IEVAL)-K) 35,34,35
   34 IF (LA(IEVAL)-L) 35,36,35
```

```
   35 PRINT 111, IEVAL,IA(IEVAL),JA(IEVAL),KA(IEVAL),LA(IEVAL),I,J,K,L,
     1TABLE(1,1)
      TABLE(1,1) = 0.0
   36 IF (IEVAL-KAPPA) 15,15,37
   37 PRINT 112, I,J,K,L,IEVAL,KAPPA
      TABLE(1,1) = 0.0
      GO TO 15
C
C     SOLVE QUADRATIC EQUATION
C
   20 A = PCOL * PROW
      B = PROW * QCOL
      C = PCOL * QROW
      D = QCOL * QROW
      CONST = (A + D) + COR(INUM)*(B + C)
      RADIC2 = CONST**2 - 4.0*((COR(INUM) - 1.0)**2) * A * D
      IF (RADIC2) 10, 13, 13
   10 PRINT 104, RADIC2
   11 RADIC2 = 0.0
   13 RADIC = SQRTF(RADIC2)
      EP1 = (CONST - RADIC)/(2.0*(COR(INUM) - 1.0))
      EP2 = (CONST + RADIC)/(2.0*(COR(INUM) - 1.0))
   14 TABLE(1,1) = A + EP1
      TABLE(1,2) = B - EP1
      TABLE(2,1) = C - EP1
      TABLE(2,2) = D + EP1
C
C     IF SENSE SWITCH 5 IS DOWN, PRINT ON-LINE RESUME AND PRESERVE
C          TABLE(1,1) VALUE
C
      IF (LIGHT) 1,5,1
    1 IF (IEVAL-1) 2,2,3
    2 PRINT 100
    3 PRINT 101, I, J, K, L, EP1, EP2, ((TABLE(II,JJ), JJ = 1, 2), II =
     1  1, 2), A, B, C, D
      XTABLE(IEVAL) = TABLE(1,1)
      IA(IEVAL) = I
      JA(IEVAL) = J
      KA(IEVAL) = K
      LA(IEVAL) = L
C
```

```
C     VERIFY THAT ALL ENTRIES ARE BETWEEN ZERO AND ONE
C     IF THEY ARE NOT, ADJUST THEM AND COMMENT ATTACHED PRINTER
C
 5    DO 9 I1=1,2
      DO 9 J1=1,2
      IF (TABLE(I1,J1))6,7,7
    6 PRINT 102, I, J, K, L, I1, J1, TABLE(I1,J1)
      TABLE(I1,J1)=0.0
      GO TO 9
    7 IF (TABLE(I1,J1) - 1.0) 9, 9, 8
    8 PRINT 103, I, J, K, L, I1, J1, TABLE(I1,J1)
      TABLE(I1,J1)=1.0
    9 CONTINUE
   15 RETURN
C
  100 FORMAT (9H0 CELL ID5X17HROOTS OF EQUATION7X26HCONTINGENCY TABLE AD
  1001JUSTED13X28HCONTINGENCY TABLE UNADJUSTED/2X7HI J K L7X15HUSED   NO
  1002T USED7X1HA9X1HB9X1HC9X1HD9X1HA9X1HB9X1HC9X1HD/1X)
  101 FORMAT (2XI1,1XI1,1XI1,1XI1,2X10F10.4)
  102 FORMAT (10H0FOR CELL 4I1,8H, TABLE(I1,1H,I1,3H) =F9.6,33H.  THIS V
  1021ALUE REPLACED BY  0.0000/1X)
  103 FORMAT (10H0FOR CELL 4I1,8H, TABLE(I1,1H,I1,3H) =F9.4,33H.  THIS V
  1031ALUE REPLACED BY  1.0000/1X)
  104 FORMAT (18H0RADICAL SQUARED =F12.5/35H THIS QUANTITY WILL BE SET T
  1041O ZERO./1H0)
  110 FORMAT (46H0TABLE(1,1) SET EQUAL TO XTABLE FOR ALL CELLS.)
  111 FORMAT (12H0FOR XTABLE(I2,15H), CELL ID IS  4I1/14X15H ID SHOULD B
     1E  4I1/13H TABLE(1,1) =F7.5,18H, REPLACED BY 0.0.)
  112 FORMAT (11H0FOR CELL  4I1,9H, IEVAL =I3,9H, KAPPA =I3,30H. TABLE(1
     1,1) SET EQUAL TO 0.0.)
C
      END
```

E.3 What EVAL computes

EVAL receives, through COMMON, the column margin PCOL and row margin PROW of a fourfold table of proportions, with QCOL $= 1 -$ PCOL and QROW $= 1 -$ PROW, and an index INUM that selects one of nine odds ratios, COR(1) to COR(9). It starts from the table that independence would give,

$$A = \text{PCOL} \cdot \text{PROW}, \quad B = \text{PROW} \cdot \text{QCOL}, \quad C = \text{PCOL} \cdot \text{QROW}, \quad D = \text{QCOL} \cdot \text{QROW},$$

and moves every cell by the same amount e: $A+e$, $B-e$, $C-e$, $D+e$. This keeps all four margins. Requiring the odds ratio of the new table to equal θ = COR(INUM), and using $AD = BC$, gives a quadratic in e:

$$(1-\theta)\,e^2 \;+\; \left[(A+D)+\theta\,(B+C)\right]e \;+\; (1-\theta)\,AD \;=\; 0.$$

The code's CONST is the bracketed coefficient and its RADIC2 the discriminant. Of the two roots, EP1 and EP2 = (CONST$\mp$RADIC)/(2($\theta-1$)), EVAL keeps EP1 and sets TABLE(1,1) = A+EP1. The printed resume heads the two roots "USED" and "NOT USED".

This is the problem of Section 7 written in the deviation from independence rather than in the cell itself. With all quantities divided by N, PROW = R/N and PCOL = C/N, and TABLE(1,1) = a/N. The surviving subroutine therefore confirms the formulation of Section 7.1: the authors fixed the margins, imposed an odds ratio and solved a quadratic.

The root is always the feasible one. The two roots multiply to AD and have the same sign. EP1 is the one of smaller magnitude, so the other has magnitude at least $\sqrt{AD}$. For $\theta < 1$ the roots are negative and feasibility requires $|e| \le \min(A, D) \le \sqrt{AD}$; for $\theta > 1$ they are positive and feasibility requires $e \le \min(B, C) \le \sqrt{BC} = \sqrt{AD}$. Only EP1 can qualify, and by Proposition 1 one root does. EP1 is therefore the unique solution of Proposition 1. Over 20,000 random margins and odds ratios, the Python port of EVAL agrees with the reconstruction's solver to within 10^{-14}.

The discriminant is never negative for valid margins. Because $AD = BC$, the arithmetic–geometric mean inequality gives CONST $\ge 2\sqrt{AD} + 2\theta\sqrt{BC} = 2(1+\theta)\sqrt{AD}$. Since $1+\theta \ge |1-\theta|$ for $\theta \ge 0$, CONST$^2 \ge 4(1-\theta)^2 AD$. The branch that prints "RADICAL SQUARED = … THIS QUANTITY WILL BE SET TO ZERO" can only be reached through rounding or margins outside $[0, 1]$. The same holds for the pass that clips each cell to $[0, 1]$: with valid margins the chosen root keeps every cell feasible. Both are defensive checks, of a piece with the guards in the state-by-state program (Appendix B.7).

There is no independence branch. At $\theta = 1$ the quadratic is linear and the formula divides by zero. The independence benchmark of Section 7.3, naive subtraction, cannot be run through EVAL as written; it can only be approached with θ near 1, where the formula loses precision by cancellation.

E.4 The replay option and the sense switch

EVAL counts its calls in IEVAL. When LIGHT is on (the comment ties it to sense switch 5 on the operator's console), it prints a resume of each table, its two roots and the unadjusted table (FORMAT 100 and 101), and it stores TABLE(1,1) with the cell's four-digit code in XTABLE, IA, JA, KA and LA. When KAPPA is positive, a later run skips the computation and replays the stored TABLE(1,1) values, checking each against the cell it is meant for and setting it to zero, with a message, if the codes disagree or the call count passes KAPPA. Two things follow.

- **Only a was needed downstream.** Replay sets TABLE(1,1) and leaves the other three cells as they were. The caller can only have used TABLE(1,1), the cell a of Section 7, the

anti-Catholic 1958 Democrats. This matches the equations of Section 6, which use a alone.

- **Fewer calls than types.** At most 200 values can be replayed, fewer than the 480 types, consistent with Section 7's step being needed only by the types whose equations use a.

E.5 What the COMMON block reveals

The COMMON block lists the main program's shared variables in order. Read against the other decks, it identifies much of what the lost program held.

Variables	Reading
NCLUS, R(60), P(60), Q(60)	One cell's record from the cluster summary tape: the number of clusters and R, P, Q for each, with the same capacity of 60 clusters as Appendices A and C
I, J, K, L	The four-digit code of the cell being processed (region, two cell digits, party)
NMISS, IMISS(20)	A count and list of up to 20 items, plausibly missing clusters
COR(9), KAPPA, IEVAL, LIGHT	The nine odds ratios and the controls of E.4
VK, VN	The Kennedy and Nixon votes of the current type, the V_k and V_n of Section 6
ALPHA	A further parameter, not used by EVAL
II, JJ, KK, LL (480)	The codes of all 480 types
VOTEK(480,20), VOTEN(480,20), CSHIFT(20)	Kennedy and Nixon votes for every type under up to 20 values of a shift parameter

The last row connects the simulation to the state-by-state program. That program reads NJOB sets of 480 votes, each preceded by its CSHIFT (Appendix B.3). The simulation computed up to 20 such sets in one pass, one for each value of the shift parameter, and wrote them out for it. Systematic variation of a parameter was therefore built into the simulation, not only into the reporting. Which parameter CSHIFT shifted is not recorded; ALPHA is one candidate.

The first row supports the inference drawn in Appendix A.6 and Figure 7: the simulation read its P and Q from the cluster summary tape, one cell record at a time. The block also opens with the same ten names, NCLUS through IMISS, as the COMMON block of the cluster-profile program (Appendix C), so the two programs share a template for reading the tape.

The nine odds ratios match the nine cells of Section 6.2 in number, but nothing records which INUM each cell used, or whether all nine were set to the article's 0.6. The 1961 article reports a single value.

E.6 A Python port

A statement-by-statement Python port of EVAL, including the replay and clipping logic and the printed messages, is in the supplementary kit (archive/evalsub.py). The kit's tests check that it agrees with the reconstruction's solver, that it preserves the margins, that the discriminant is never negative for valid margins, that replay restores a stored value and zeroes a mismatched one, and that $\theta = 1$ fails as it would have on the 7090.

Appendix F. GUTM3A: a Guttman scaling program

The sixth deck is a utility for survey analysis, not part of the 1960 simulation. It is included because it comes from the same account and the same people, and because it shows the kind of attitude measurement the group was doing after 1960.

F.1 Identification

The deck was supplied without an archive file name. Its job card carries the account number of Appendices A, C and D, M888-1505, and its first comment reads "GUTM3A REVISED GUTMAN SCALE PROGRAM SAM POPKIN AUGUST 1962". It is FORTRAN II for the IBM 7090 and ends with `CALL EXITM`, a return to the monitor. The date places it after the 1960 campaign, during the work that led to Pool, Abelson and Popkin (1964), and shows that the account was still in use in 1962.

F.2 Transcription

```
*M888-1505,FMS,RESULT,5,5,10000,2000
*     LIST8
*     LABEL
CGUTM3A REVISED GUTMAN SCALE PROGRAM SAM POPKIN AUGUST 1962
      DIMENSION REP  (25),NQUEST(25),ICOL(10,25),NPOS(10,25),NPLUS(10,10
     1,25),POSN(13,25),ILABEL(64),NFREQ(64,25),ITITLE(13,25),IWGHT(6),
     2TRIX(13,64),RATIO(13),PROD(13),NCOL(960),LWGHT(6)
      READ  99,((TRIX(I,J),I=1,13),J=1,64)
  99  FORMAT(13F5.0)
      READ 100,NCOLS,NPEOPL,NSCALS,ITAPE
 100  FORMAT(2I4,2I2)
C     NCOLS=COLUMNS OF INFORMATION PERSON
C     NPEOPL=NUMBER OF PEOPLE
C     NSCALS=NUMBER OF SCALES
C     ITAPE IS NUMBER OF TAPE ON WHICH CARDS ARE LOCATED
      DO 200 K=1,NSCALS
      READ 101,NQUEST(K),(ITITLE(I,K),I=1,13)
 101  FORMAT(I2,13A6)
      JJ=NQUEST(K)
      READ 102,(ICOL(J,K),NPOS(J,K),(NPLUS(I,J,K),I=1,9),J=1,JJ)
 102  FORMAT(6(I3,10I1))
      DO 200  J=1,13
 200  POSN(J,K)=0.0
      DO 2 I=1,64
      DO 2 K=1,25
 2    NFREQ(I,K)=0
```

```
      DO 7 MOB=1,NPEOPL
      READ INPUT TAPE ITAPE,103,(NCOL(L),L=1,NCOLS)
 103  FORMAT(80I1)
      DO 6 K=1,NSCALS
      IWGHT=1
      ISCORE=0
      LSCORE=0
      JJ=NQUEST(K)
      DO 5 J=1,JJ
      JK=ICOL(J,K)
      II=NPOS(J,K)
      DO 4 I=1,II
      IF(NCOL(JK)-NPLUS(I,J,K))4,3,4
 3    ISCORE=ISCORE+IWGHT
      POSN(J,K)=POSN(J,K)+1.
      LSCORE=LSCORE+1
      GO TO 5
 4    CONTINUE
 5    IWGHT=2*IWGHT
      IX=K+NCOLS
      NCOL(IX)=LSCORE
 6    NFREQ(ISCORE+1,K)=NFREQ(ISCORE+1,K)+1
      NCOL(584)=NCOL(129)
      NCOL(585)=NCOL(130)
      PUNCH 1000,(NCOL(IR),IR=1,4),(NCOL(IR),IR=561,585)
1000  FORMAT(4I1,2H0825I1)
 7    CONTINUE
      IF(ITAPE-4)8,9,8
 8    REWIND ITAPE
 9    DO 10 I=1,64
 10   ILABEL(I)=I-1
      DO 11 J=1,6
      LWGHT(J)=2**(J-1)
 11   IWGHT(J)=2**(J)
      A=FLOATF(NPEOPL)*.01
      B=FLOATF(NPEOPL)*.05
      DO 21 K =1,NSCALS
      JJ=NQUEST(K)
      DO 1 J=1,JJ
 1    POSN(J+6,K)=FLOATF(NPEOPL)-POSN(J,K)
      PRINT 104
```

```
104  FORMAT(115H1FREQUENCY COUNTS AND SCALABILITY TESTS FOR GUTMAN SCAL
    1E SAM POPKIN (SEE ARTICLE BY ROBERT FORD, POQ, FALL 1950)       )
     IPAGE = 1
     PRINT 105,(ITITLE(I,K),I=1,13),IPAGE
105  FORMAT(1H013A6,20X4HPAGEI4)
     PRINT 106
106  FORMAT(1H010X71HQUESTION COLUMN* WEIGHT +RESPONSES -RESPONSES TOTA
    1L POSITIVE VALUES     )
     JJ=NQUEST(K)
     DO 12 J=1,JJ
     II=NPOS(J,K)
12   PRINT 107,J,ICOL(J,K),LWGHT(J),POSN(J,K),POSN(J+6,K),NPEOPL,(NPLUS
    1(I,J,K),I=1,II)
107  FORMAT(1H 17XI1,4XI3,6XI2,6XF5.0,6XF5.0,2XI4,I2,8 (1H,I1))
     PRINT 108
108  FORMAT(1H030X54H* COLUMNS ARE NUMBERED CONSECUTIVELY FROM CARD TO
    1CARD)
     PRINT 109
109  FORMAT(1H030X24HSCORES AND FREQUENCIES**    /1H0,5X8(13HSCORE  NO.
    1     ))
     IMAX=2**JJ
     PRINT 110,(ILABEL(I),NFREQ(I,K),I=1,IMAX)
110  FORMAT(1H05X2I5,3X2I5,3X2I5,3X2I5,3X2I5,3X2I5,3X2I5,3X2I5/)
     PRINT 111
111  FORMAT(1H030X61H** INDIVIDUALS SCORE IS SUM OF WEIGHTS OF POSITIVE
    1 RESPONSES. )
     PRINT 112
112  FORMAT(1H010X28HSCORES THAT SCALE PERFECTLY /1H 10X48HSCORE  NO. P
    1ERCENT QUESTIONS ANSWERED POSITIVELY)
     PRCENT=FLOATF(NFREQ(1,K))/A
     LLL=0
     PRINT 113,ILABEL(1),NFREQ(1,K),PRCENT,LLL
113  FORMAT(1H013XI2,I5,4XF4.1,I5,5(1H,I1))
     ISUM = NFREQ(1,K)
     NFREQ(1,K)=0
     DO13  J=1,JJ
     L=IWGHT(J)
     PRCENT=FLOATF(NFREQ(L,K))/A
     ISUM=ISUM+ NFREQ(L,K)
     PRINT 113,ILABEL(L),NFREQ(L,K),PRCENT,(N,N=1,J)
13   NFREQ(L,K)=0
```

```
     PRCENT=FLOATF(ISUM)/A
     PRINT 114,ISUM,PRCENT
114  FORMAT(1H 16X4H----4X4H----/1H 15XI5,4XF4.1)
     PRINT 104
     IPAGE = 2
     PRINT 105,(ITITLE(I,K),I=1,13),IPAGE
     PRINT 115
115  FORMAT(30HOTEST FOR RANDOMNESS OF ERRORS)
     IO=0
     DO 15 L=1,IMAX
     IF(FLOATF(NFREQ(L,K))-B)15,14,14
14   PRCENT=FLOATF(NFREQ(L,K))/A
     PRINT 116,PRCENT,ILABEL(L)
116  FORMAT(1H0F4.1,31HPERCENT OF SURVEY HAVE SCORE OF I2)
     IO=1
15   CONTINUE
     IF(IO)16,16,17
16   PRINT 117
117  FORMAT(55HONO IMPERFECT SCORE WAS OBTAINED BY 5 PERCENT OF SURVEY)
17   DO 19 J=1,JJ
     PROD(J)=0.
     PROD(J+6)=0.
     DO 18 L=1,IMAX
     PROD(J)=PROD(J)+FLOATF(NFREQ(L,K))*TRIX(J+1,L)
18   PROD(J+6)=PROD(J+6)+FLOATF(NFREQ(L,K))*TRIX(J+7,L)
     RATIO(J)=PROD(J)/POSN(J,K)
19   RATIO(J+6)=PROD(J+6)/POSN(J+6,K)
     PRINT 118
118  FORMAT(////92H0QUESTION +RESPONSES+ERRORS* +RATIO* *-RESPONSES-ERR
    1ORS -RATIO  TOTAL ERRORS REPRODUCIBILITY)
     ACCUM=0.
     DO 20 J=1,JJ
     P=PROD(J)+PROD(J+6)
     ACCUM=ACCUM+P
     Q=1.-P/FLOATF(NPEOPL)
20   PRINT 119,J,POSN(J,K),PROD(J),RATIO(J),POSN(J+6,K),PROD(J+6),RATIO
    1(J+6),P,Q
119  FORMAT(1H0I8,6XF5.0,3XF4.0,2XF6.3,8XF5.0,F7.0,1XF6.3,8XF6.0,5XF6.3
    1)
     PRINT 120
120  FORMAT(67H0*ERROR OCCURS WHEN CHANGE IN ANSWER WOULD IMPROVE REPRO
```

```
     1DUCIBILITY.  /34H **RATIO OF ERRORS TO RESPONSES   )
      SUMPS=NPEOPL*JJ
      REP  (K)=1.-ACCUM/SUMPS
 21   PRINT 121,JJ,NPEOPL,SUMPS,ACCUM,REP(K)
 121  FORMAT(5HOWITHI2,13HQUESTIONS ANDI4,26H PEOPLE THERE ARE TOTAL OF
     1F6.0,9HRESPONSES/1H F6.0,54HACTUAL MISTAKES GIVE COEFFICIENT OF RE
     2PRODUCIBILITY OF  F5.4)
      PRINT 104
      PRINT 122
 122  FORMAT(//34HOSUMMARY OF SCALING JUST COMPLETED///40X5HTITLE35X30HC
     1OEFFICIENT OF REPRODUCIBILITY)
      PRINT 123,((ITITLE(I,K),I=1,13),REP(K),K=1,NSCALS)
 123  FORMAT(1HO13A6,F5.4)
      CALL EXITM
      END
```

F.3 Guttman scaling

A set of yes-or-no items forms a Guttman scale when the items can be ordered from easiest to hardest so that anyone who endorses a harder item also endorses every easier one. A respondent's answers can then be reproduced from a single number, the count of items endorsed. The coefficient of reproducibility measures how nearly a set of items achieves this:

$$\text{REP} \;=\; 1 \;-\; \frac{\text{errors}}{\text{respondents} \times \text{items}},$$

where an error is a response that would have to change for the respondent's pattern to become a perfect one. Values of 0.9 and above were conventionally taken to show a scale. The headings cite Robert Ford's rapid scoring procedure (*Public Opinion Quarterly*, Fall 1950), which assigns errors to each response pattern from a table rather than by inspection.

F.4 What the program does

1. **Read the error table** TRIX: 64 cards of 13 numbers, one card for each response pattern of up to six items. For pattern L, entries 2–7 give the errors charged to positive answers on items 1–6, and entries 8–13 those charged to negative answers. The first entry is not used. This is Ford's table, supplied as data; it is not in the deck.
2. **Read the controls**: columns per respondent (NCOLS), respondents (NPEOPL), scales (NSCALS) and the tape holding the respondent records. For each scale it reads the number of items, a title and, for each item, the card column that holds it and the codes that count as a positive answer.
3. **Score every respondent** on every scale. A positive answer to item j adds 2^{j-1} to a pattern code, so the code identifies the response pattern exactly, and adds 1 to a simple count, the

scale score. The program tallies the patterns and stores each scale score after the respondent's data, in column NCOLS + k.

4. **Punch one card per respondent**: an identifier from columns 1–4, the deck code 08, and 25 single-digit values from columns 561–585, with columns 129 and 130 copied into 584 and 585.
5. **Report each scale on two pages**: the items, with their columns, positive and negative counts and positive codes; the frequency of every response pattern; the perfect patterns (no items, item 1, items 1–2, items 1–3, and so on) with their percentages; every imperfect pattern held by at least 5 per cent of respondents, as a "test for randomness of errors"; and, item by item, the errors on positive and negative answers, the ratio of errors to responses, and the coefficient of reproducibility.
6. **Summarise** the coefficient of reproducibility of every scale on a final page.

F.5 What it shows

- **Items in order of difficulty.** The perfect patterns are items 1 to j endorsed and the rest not, so the user had to enter a scale's items from the most to the least often endorsed.
- **One study, hard-coded.** The punched output assumes that a respondent's record fills 560 columns, seven cards of 80, so that the scale scores land in columns 561 onward; columns 129 and 130 are carried along by name. The program was revised for a particular survey rather than written as a general tool, as its name, a third revision, suggests.
- **Scales as data.** The output is a new deck of cards with a scale score per respondent, ready for further tabulation. Scaling items into a single attitude measure is the respondent-level counterpart of the issue clusters of Section 4, which grouped questions by how they sorted voter types (Section 4.3). Nothing in the deck ties it to the 1960 matrix, which it postdates.

F.6 Quirks

- **An array used as a scalar.** IWGHT is declared with six elements but used without a subscript in the scoring loop, where FORTRAN II takes its first element. It is later refilled with powers of two to index the perfect patterns. The two uses do not overlap, so the program works.
- **The spelling GUTMAN**, for Guttman, in the title and the headings.
- **Limits.** Six items per scale (64 patterns) and 25 scales.
- **A tape left in place.** The respondent tape is rewound at the end unless it is tape 4.

The program is not ported. Its results depend on Ford's error table, which was read from data cards that do not survive.