INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
COL. LINDAVISTA 07738 MÉXICO, DF.
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

METODOLOGÍA Y CÓDIGO FUENTE DE LOS CONTEOS DINÁMICOS

Dr. Alexander Balankin
Dr. Miguel Ángel Martínez Cruz
Lic. Eduardo Virueña Silva

MÉTODO GENERAL

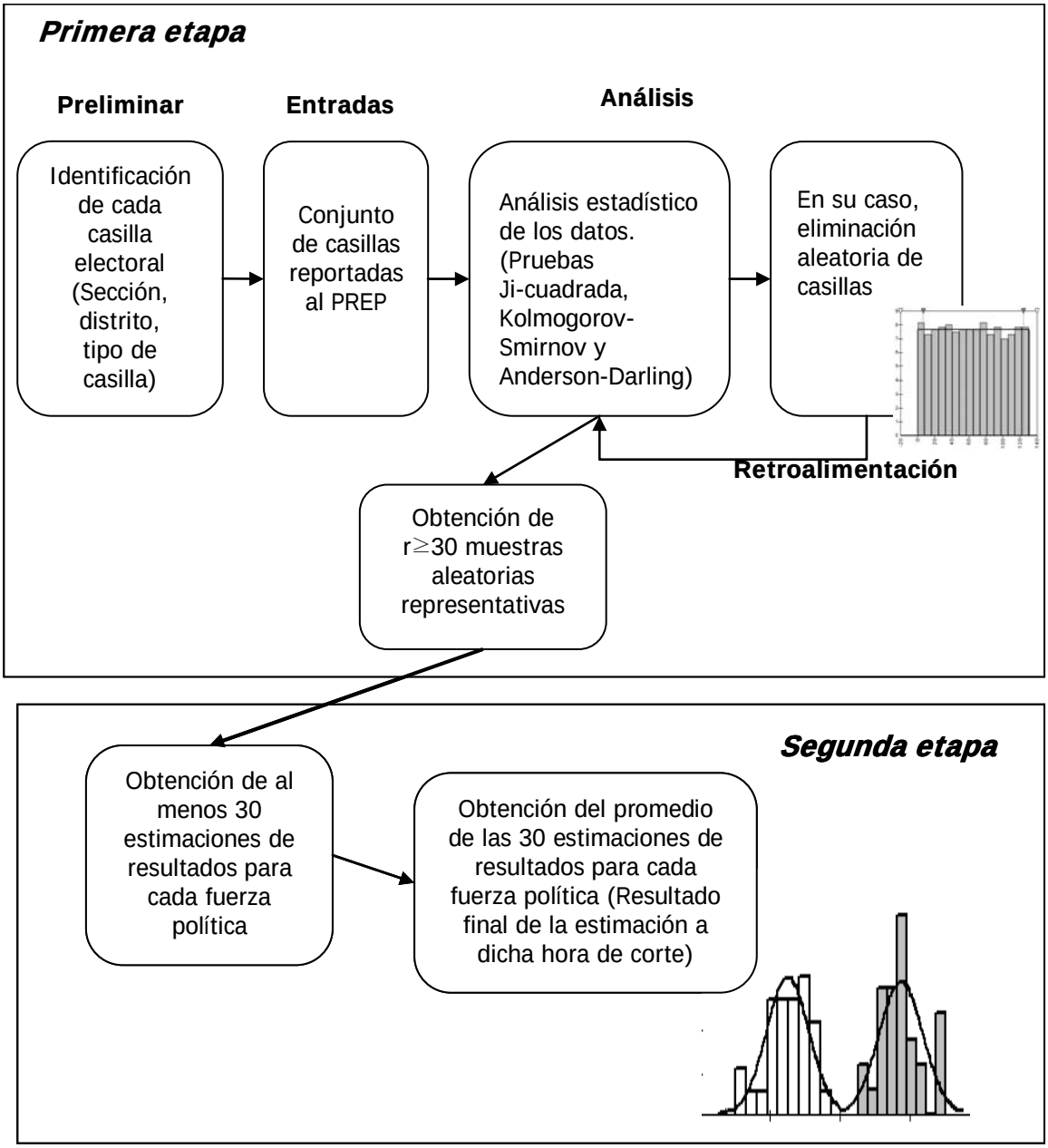
Los Conteos Dinámicos consisten en un conjunto de procedimientos matemáticos, desarrollado por investigadores mexicanos del Instituto Politécnico Nacional, para obtener estimaciones estadísticas representativas, precisas y oportunas, sobre el porcentaje de votos que cada partido político obtiene en una jornada electoral, con base en información parcial del Programa de Resultados Electorales Parciales, PREP (que por sus características, en las primeras horas no es estadísticamente representativo).

En términos generales los Conteos Dinámicos se componen de dos grandes etapas para cada ámbito electoral.

En la primera etapa se selecciona, del conjunto de todas las casillas que reporte el PREP, 30 muestras representativas para dicho ámbito electoral. Cabe aclarar que la selección de casillas se realiza a través de un criterio de bondad de ajuste que discrimina aleatoria e iterativamente una parte de ellas, hasta conformar un conjunto de casillas que se distribuye uniformemente en el ámbito electoral, es decir, en toda el área que éste abarca.

La segunda etapa opera de manera semejante a los conteos rápidos, pero con la ventaja de que se repite al menos 30 veces, de tal modo que el error de estimación resulta menor o igual al de los Conteos Rápidos que se realicen con el mismo tamaño de muestra. Es decir, se hace un proceso típico de estimación de resultados electorales a partir de una muestra representativa de resultados de casillas.

El siguiente diagrama ilustra, en términos generales, las etapas de los conteos dinámicos.



DESARROLLO DE LA METODOLOGÍA

Una estimación de los resultados de una elección cualquiera puede obtenerse a partir de la selección y análisis de los datos provistos por una muestra representativa de las casillas instaladas en el correspondiente ámbito electoral.

En términos generales la operación de los Conteos Dinámicos requiere de dos grandes etapas de trabajo. Una primera consistente en el proceso de selección de muestras representativas de casillas electorales. Y una etapa posterior de generación de múltiples estimaciones de resultados electorales, sustentadas en procedimientos estadísticos convencionales.

1. Selección de muestras representativas de casillas electorales

La selección de muestras representativas es la etapa medular del procedimiento. La calidad de muestras representativas se sustenta en el principio de distribución a lo largo y ancho del ámbito electoral correspondiente (delegación o distrito electoral).

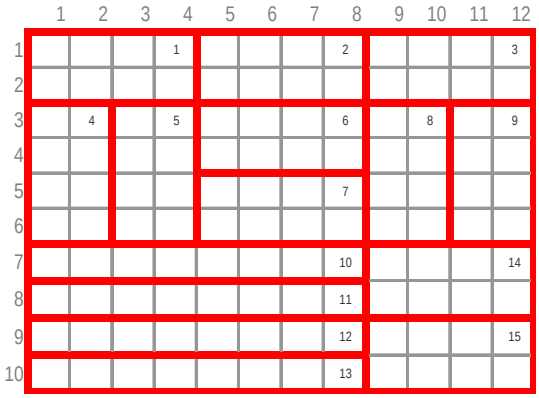
Para tal efecto, se clasifican todas las casillas del ámbito electoral en “m” clases o *bin´s*. Éstos son conjuntos que se conforman con una misma cantidad de casillas electorales, las cuales se presupone están contiguas con base en la perspectiva de que la clave que se les asigna consecutivamente, según el número del distrito y de la sección electoral correspondiente.

Tabla 1. Enumeración de las casillas electorales instaladas en una delegación política hipotética ($N_p = 120$).

Número de casilla	Distrito electoral local	sección	Casilla
1	XXXIV	3119	B
2	XXXIV	3119	C1
3	XXXIV	3120	B
⋮			
118	XXXIV	3160	C4
119	XXXIV	3161	B
120	XXXIV	3161	C1

Para ilustrar lo anterior, supóngase un ejercicio simplificado de un ámbito electoral de sólo 120 casillas, mismas que se organizan en 15 grupos de ocho casillas cada uno, conforme la siguiente ilustración (el número en la celda superior derecha de cada grupo identifica la clave del bin).

Ilustración 1. Conformación de 15 clases o bin´s con 8 casillas electorales instaladas cada una ($N = 120$).



Cabe aclarar que la anterior conformación de las 120 casillas electorales, en 15 clases de 8 casillas cada una, es sólo una posibilidad entre muchas distintas.[1]

De hecho, el sistema informático que opera los Conteos Dinámicos determina de manera dinámica, a partir de la cantidad de casillas que ya han reportado al PREP, en cuántas clases o bin´s debe distribuirse dicho conjunto. En las siguientes ilustraciones las celdas con un valor 1 y sombreadas en color gris corresponden a las casillas electorales que ya habrían reportado resultados.

Particularmente en las primeras horas del PREP puede ocurrir que alguna clase carezca de datos, como se observa en la siguiente ilustración[2]. La necesidad de que en cada clase al menos exista una casilla que haya reportado resultados proviene de la característica deseable de garantizar la dispersión de las unidades en muestra, por todo el espacio del ámbito electoral.

[1] El total de combinaciones posibles se determina con la fórmula. $C_n^N = \frac{N!}{(N-n)!n!}$

Lo importante aquí es que, para garantizar la distribución espacial de la muestra a lo largo de todo el ámbito electoral, sólo resultarían útiles las combinaciones que deriven en grupos de casillas relativamente "compactos". Ya que las combinaciones conformadas por casillas electorales totalmente dispersas podrían provocar que, aun cuando todos los bin´s tengan un número semejante de elementos reportados a una hora determinada, al graficarlos éstos se "apiñen" en determinadas zonas y dejen algunas áreas sin representación.

[2] Para resolver esta situación, el sistema de los conteos dinámicos internamente conformaría una menor cantidad de clases, pero de mayor tamaño (8 clases de 15 casillas, por ejemplo), de manera que cada uno de ellas disponga de al menos una casilla que haya reportado resultados al PREP.

Ilustración 2. Simulación de las casillas que habrían reportado resultados al PREP en las primeras horas de su operación

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	1	0	1	1	1	1
2	0	0	0	0	0	0	1	1	0	0	0	0
3	1	0	0	0	0	1	1	1	0	0	0	0
4	0	0	0	1	0	0	0	1	1	0	0	0
5	0	0	0	0	0	1	1	1	0	1	0	0
6	0	0	1	1	0	1	1	0	1	1	0	0
7	0	0	0	0	1	0	1	0	0	0	1	1
8	0	0	0	0	0	0	0	1	1	0	0	1
9	1	1	0	0	0	1	1	0	0	0	0	1
10	0	0	1	0	0	0	0	1	0	0	0	0

Promedio de casillas reportadas por bin:	2.5
Mínimo de casillas reportadas por bin:	0
Máximo de casillas reportadas por bin:	5

Clave_bin	fx
Bin1	0
Bin2	3
Bin3	4
Bin4	1
Bin5	3
Bin6	4
Bin7	5
Bin8	4
Bin9	0
Bin10	2
Bin11	1
Bin12	4
Bin13	2
Bin14	4
Bin15	1

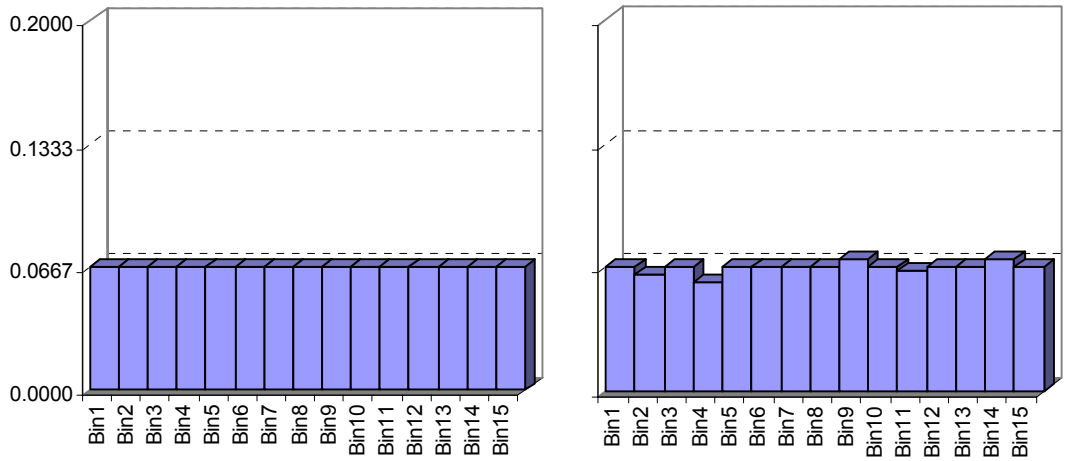
La dispersión espacial es una condición necesaria de las muestras representativas que el sistema de los Conteos Dinámicos selecciona, pero no suficiente. Hace falta también que todas las clases o bin´s sean estratos de tamaño semejante para que algunos de ellos no representen más información que los demás, y por ello pudieran sesgar las estimaciones.

Esta condición de “relativa equidad” en la cantidad de información que cada clase aporta se cumple a través de la valoración de una condición: que la cantidad de casillas incorporadas en cada una de las clases describan una distribución estadística uniforme, según las pruebas de bondad de ajuste *Ji cuadrada, Kolmogorov-Smirnov* y *Anderson- Darling.*

Como se sabe, un conjunto de datos se distribuye de manera uniforme si al graficar las frecuencias de todas sus clases forman aproximadamente un

rectángulo, como se ilustra en las siguientes gráficas (la de la izquierda tiene una distribución perfectamente uniforme; mientras que la otra describe una distribución aproximadamente uniforme). Ambas, al ser valoradas por los estadísticos de bondad de ajuste mencionados, serían consideradas como distribuidas uniformemente.

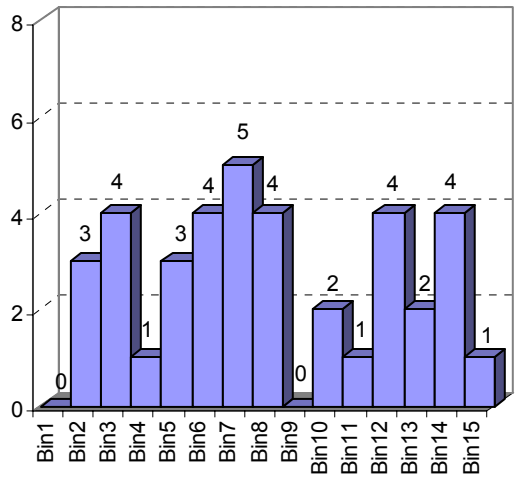
Ilustración 3. Dos conjuntos de datos de una distribución estadística uniforme



Como podría esperarse, los datos reales que se obtendrían al graficar la llegada de paquetes electorales a las sedes distritales (datos del PREP), en una hora particular de corte, generalmente no corresponden a una distribución uniforme. De hecho, los datos que llegan al PREP consecutivamente no son, en sí mismos, representativos y útiles para hacer estimaciones de los resultados finales. Se trata de información sesgada, ya que las casillas electorales más próximas a la sede distrital, las mejor comunicadas o las que presenten menos dificultades en la etapa de cómputo y escrutinio, suelen llegar primero que las demás. Este sesgo se resuelve a través de procesos de selección aleatoria de las casillas que se incorporarán a las muestras representativas de casillas electorales.

Por ejemplo, los datos hipotéticos que anteriormente se mencionaron para una hora temprana de la noche (ilustración 2) muestran la siguiente gráfica.

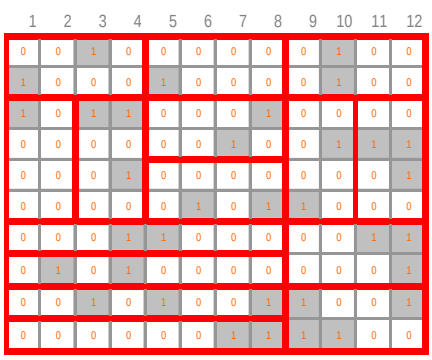
Ilustración 4. Gráfica de barras de las casillas que habrían reportado al PREP a una hora temprana de la noche



Esta situación obligaría a que el sistema de los Conteos Dinámicos conforme menos clases (en el ejemplo manejado, quizá cinco clases de 24 casillas cada una) para que todas tengan al menos un dato reportado. En su defecto, dado que se trataría de una hora temprana, el encargado de operar el sistema informáticos de los Conteos Dinámicos podría determinar esperar a seleccionar las muestras representativas con algún corte posterior, quizá a partir de las 22:00 horas.

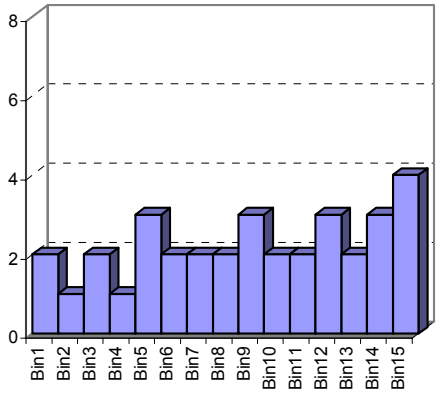
Supóngase que se opta por la segunda alternativa y que a las 22:30 horas 34 de las 120 casillas ya han reportado datos al PREP. Ahora cada una de las 15 clases determinadas de antemano tendría al menos un dato, aunque aparentemente las frecuencias de las 15 clases no describen una distribución uniforme.

Ilustración 5. Simulación de las casillas que habrían reportado resultados al PREP a las 22:30 horas



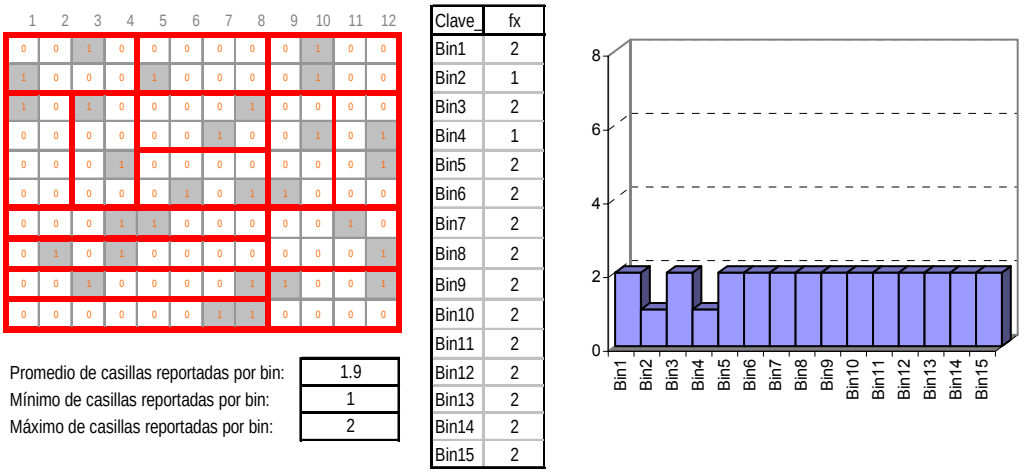
Promedio de casillas reportadas por bin:	2.3
Mínimo de casillas reportadas por bin:	1
Máximo de casillas reportadas por bin:	4

Clave_	fx
Bin1	2
Bin2	1
Bin3	2
Bin4	1
Bin5	3
Bin6	2
Bin7	2
Bin8	2
Bin9	3
Bin10	2
Bin11	2
Bin12	3
Bin13	2
Bin14	3
Bin15	4



Para resolver esta situación, el sistema informático de los Conteos Dinámicos selecciona aleatoriamente una casilla electoral de las clases que tienen más datos reportados que el promedio, y la quitan del conjunto correspondiente. A continuación el sistema procede a evaluar si con ello los datos ahora sí se distribuyen de manera uniforme. Si se logró tal cometido el sistema selecciona dicha muestra; si no, aleatoriamente selecciona otra casilla de las clases con más observaciones, la elimina del conjunto y vuelve a evaluar. Este ciclo concluye hasta lograr uniformidad en los datos. En este momento el sistema “guarda” dicho conjunto y lo convierte en una muestra aleatoria válida.

Ilustración 6. Simulación de una posible eliminación de casillas realizada aleatoriamente por el sistema de los CD para lograr una distribución uniforme de los datos



1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	1	1	0	0	0
0	0	0	1	1	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1	1	0	0	1
0	0	0	0	0	0	1	1	0	0	0	0

Promedio de casillas reportadas por bin:	1.9
Mínimo de casillas reportadas por bin:	1
Máximo de casillas reportadas por bin:	2

Clave	fx
Bin1	2
Bin2	1
Bin3	2
Bin4	1
Bin5	2
Bin6	2
Bin7	2
Bin8	2
Bin9	2
Bin10	2
Bin11	2
Bin12	2
Bin13	2
Bin14	2
Bin15	2

El sistema no se detiene al obtener esta primera muestra de casillas electorales distribuidas uniformemente en todos los bin´s (según el ejercicio hipotético desarrollado de n = 28 elementos).

Empieza de nuevo en múltiples ocasiones, en cada una de las cuales elimina distintas casillas, ya que opera aleatoriamente. Así, obtiene al menos 30 muestras aleatorias representativas distintas (cada una de ellas debe tener al menos 50% de elementos diferentes con el fin de reducir la correlación entre las “r” muestras seleccionadas).

2. Generación de estimaciones de tendencias de resultados electorales.

Una vez que se cuenta con r ≥ 30 muestras representativas de casillas electorales que han reportado información al PREP, se procede a analizar dicha información para obtener estimaciones de los resultados finales de la elección que se trate.

Sin pérdida de generalidad, supóngase que se trata de la elección del Jefe Delegacional en Milpa Alta[3]. En esta delegación la primera de las 30 o más muestras se expresa, según los datos de la ilustración 6, a través de una matriz de 28 filas y $\ell+1$ columnas.

Designamos con la letra L al conjunto de los ℓ "candidatos" que contienden en una delegación (los primeros $\ell-1$ datos corresponden a los partidos políticos y candidaturas comunes, y los votos nulos ocupan la última posición ℓ).

A continuación se procede mediante una serie de operaciones matemáticas convencionales en cualquier estimación de porcentajes. Se trata de aspectos bien conocidos, por ejemplo, que el conjunto de datos que integran la muestra describen una *distribución estadística* cualquiera con media μ^l y desviación estándar σ^l.

2.1. Resumen de conceptos y fórmulas generales de estimación estadística.

Una vez que se conozca la votación de todas y cada una de las casillas instaladas, el valor real de dichos parámetros puede calcularse mediante las siguientes expresiones:

$$\mu^l = \frac{1}{N}\sum_{i=1}^{N} x_i^l \qquad (1)$$

$$\sigma^l = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i^l - \mu^l)^2} \qquad (2)$$

[3] En 2006 en esta delegación se instalaron 123 casillas electorales, un dato muy semejante a las 120 que se ilustraron en el apartado previo.

El total de votos emitidos en la elección del jefe delegacional de la delegación Milpa Alta puede comprobarse mediante la siguiente expresión:

$$V = N \sum_{l=1}^{L} \mu^l \,. \qquad (3)$$

Una estimación puntual del valor del μ^l, con la información disponible ($n < N$) antes de terminar el conteo oficial, se obtiene a través del estimador de μ^l

$$\bar{x}_n^{\,l} = \frac{1}{N} \sum_{i=1}^{n} x_i^l \,, \qquad (4)$$

La variabilidad de la muestra se obtiene a través del estimador:

$$\sigma_n^l(n) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i^l - \bar{x}^{\,l})^2} \,. \qquad (5)$$

Debido a que $\bar{x}_n^{\,l}$ varía de muestra en muestra, no es desde luego, igual a la media de la población (μ^l). Ello porque hay un error de la muestra.

Entonces es necesario proporcionar una estimación de intervalo en torno al cual refleje nuestro juicio acerca del alcance de este error muestral. El tamaño del intervalo dependerá de qué tan confiados queramos estar que el intervalo contenga la media de la población verdadera y desconocida.

El estimador $\bar{x}_n^{\,l} = \frac{1}{N} \sum_{i=1}^{n} x_i^l$ es *insesgado* debido a que su esperanza es igual al valor verdadero del parámetro μ^l.

De acuerdo al *Teorema del Limite Central*, en el muestreo aleatorio simple con reemplazo de muestras grandes ($n \geq 30$) los promedios muéstrales $\bar{x}^{l}_{j_n}$ se *distribuyen normalmente* con media

$$\mu^{l}_{\bar{x}} = \frac{1}{N^n} \sum_{j_n=1}^{N} \bar{x}^{l}_{j_n} = \mu^{l} \qquad (6)$$

y desviación estándar

$$\sigma^{l}_{\bar{x}} = \sqrt{\frac{1}{N^n} \sum_{j_n=1}^{N^n} \left(\bar{x}^{l}_{j_n} - \mu^{l}_{\bar{x}} \right)^2} = \frac{s}{\sqrt{n}} . \qquad (7)$$

En el muestreo aleatorio simple sin reemplazo de muestras grandes ($n \geq 30$) los promedios muéstrales $\bar{x}^{l}_{j_n}$ se *distribuyen normalmente* con media

$$\mu^{l}_{\bar{x}} = \frac{1}{C^N_n} \sum_{j_n=1}^{C^N_n} \bar{x}^{l}_{j_n} = \mu^{l} , \qquad (8)$$

y desviación estándar

$$\sigma^{l}_{\bar{x}} = \sqrt{\frac{1}{C^N_n} \sum_{j_n=1}^{C^N_n} \left(\bar{x}^{l}_{n_j} - \mu^{l}_{\bar{x}} \right)^2} = \frac{s}{\sqrt{n}} \sqrt{\frac{N-n}{N-1}} , \qquad (9)$$

donde la expresión para calcular el número de combinaciones es:

$$C^N_n = \frac{N!}{(N-n)!n!} .$$

Como también se sabe, en una distribución normal según la desigualdad de Chebyshev, aproximadamente 95% de los datos están situados a una distancia inferior a dos desviaciones estándar de la media. De lo anterior se deduce que:

$$P\left(\mu - 1.96\sigma_{\bar{x}} < \bar{x} < \mu + 1.96\sigma_{\bar{x}}\right) = 0.95\,, \qquad (10)$$

$$0.95 = P(\bar{x} < \mu + 1.96\sigma_{\bar{x}}) - P(\bar{x} < \mu - 1.96\sigma_{\bar{x}}) = P(\mu > \bar{x} - 1.96\sigma_{\bar{x}}) - P(\mu > \bar{x} + 1.96\sigma_{\bar{x}})$$

$$P\left(\bar{x} - 1.96\sigma_{\bar{x}} < \mu < \bar{x} + 1.96\sigma_{\bar{x}}\right) = 0.95\,. \qquad (11)$$

Por tanto, la fórmula (11) nos da un intervalo de valores tal que la probabilidad de que la media de la población μ^l esté contenida en él es de 0.95. Este tipo de intervalos se llaman ***intervalos de confianza*** de un parámetro poblacional.

En estadística, un *intervalo de confianza* para un parámetro poblacional es el intervalo entre dos números con una probabilidad asociada (P) que es generado a partir de una muestra aleatoria de una población, de tal manera que si el muestreo se repitiera numerosas veces y el *intervalo de confianza* fuera calculado de la misma manera, la proporción P del *intervalo de confianza contendría* el mismo parámetro poblacional en cuestión. Es importante subrayar que el intervalo de confianza (nivel de significancia) es un valor de certeza que fija el investigador "*a priori*".

El ***nivel de confianza*** del intervalo es la probabilidad de que éste contenga al parámetro poblacional. Se denota por $(1-\alpha)$, donde α corresponde al ***nivel de significancia*** expresado en términos de probabilidad. El *nivel de significancia* representa áreas de riesgo o confianza en la distribución normal. En las ecuaciones (10) y (11), el nivel de confianza es del 95% (*nivel de significancia* $\alpha = 0.05$). Con un nivel de confianza del $(1-\alpha)$ admitimos que la diferencia entre la estimación x_{pn}^l para la media poblacional a partir de la muestra y su valor real μ_p^l es menor que el ***error máximo admisible*** (Er).

Para una *muestra con remplazo* grande ($n \geq 30$) el ***error máximo de estimación*** con nivel de confianza $(1-\alpha)$ es la mitad de la longitud del intervalo,

$$E = Z_{\alpha/2}\frac{s}{\sqrt{n}}, \qquad (12)$$

donde $Z_{\alpha/2}$ es el número de desviaciones estándar normales el cual puede obtenerse de una tabla; por ejemplo $Z_{0.05/2} = 1.96$.

De este modo, el ***error máximo de estimación*** depende de tres factores. El primero es el ***nivel de confianza*** deseado. El segundo factor es la ***desviación estándar de la población*** (2), desconocido antes del conteo total de votos. El tercero es el ***tamaño de la muestra*** (conforme ésta aumenta, el error de la muestra se ve reducido y el intervalo se volverá más pequeño). Así, para un nivel de confianza dado, un error muestral más pequeño tendrá un "costo" en términos de un tamaño de muestra más grande. Similarmente, para un error de la muestra dado, un nivel de confianza más alto tendrá un "costo" en términos de un tamaño muestral más grande.

Para una *muestra sin remplazo* grande ($n \geq 30$) la desviación estándar muestral depende del tamaño relativo de la muestra (n/N) de acuerdo a la relación (9). Así, el ***error máximo de estimación*** con nivel de confianza $(1-\alpha)$ es

$$Er = \pm\frac{Z_{\alpha/2}}{2}\frac{s}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}}, \qquad (13)$$

Cuando $n/N \leq 0.05$ las estimaciones (12) y (10) para muestras con y sin remplazo, respectivamente, prácticamente coinciden.

Para ***muestras pequeñas*** ($n<30$), por regla general, la distribución muestral no sigue la distribución normal, sino la denominada ***t de Student***. En este caso, el *error máximo admisible* para nivel de confianza $(1-\alpha)$ se calcula como

$$Er = \pm \frac{t_{n-1,\alpha/2}}{2} \sigma_{\bar{x}}^{l}, \qquad (14)$$

donde $t_{n-1,\alpha/2}$ es el número de desviaciones estándar de *distribución de Student* y la desviación estándar de los promedios muéstrales ($\sigma_{\bar{x}}^{l}$) esta definida para los muestreos con y sin reemplazo por las fórmulas (7) y (9), respectivamente.

2.2. Ejemplo de estimaciones de resultados electorales para Milpa Alta.

En el caso de los *Conteos Rápidos* se seleccionan una muestra aleatoria simple sin reemplazo de tamaño n . El porcentaje de votos obtenidos por el candidato "l" se estima como

$$\bar{x}^{l} = \frac{1}{v}\left(\frac{1}{n}\sum_{i=1}^{n} v_{il}\right), \qquad (15)$$

donde v_{il} es número de votos por el candidato "l" en la casilla "i" y

$$v = \sum_{i=1}^{n} \sum_{l=1}^{L} v_{il} \qquad (16)$$

es el número total de votos emitidos en la muestra de casillas de tamaño n .

El error de estimación de porcentaje de votos obtenidos por el candidato con nivel de confianza de 95% de acuerdo a la relación (13) es

$$Er^{l} = 1.96\frac{\bar{x}^{l}\, s^{l}}{2\sqrt{n}}\sqrt{1-\frac{n}{N}}\,. \qquad (17)$$

A partir de las r ≥ 30 muestras representativas seleccionadas previamente, la ecuación (17) puede utilizarse para estimar de resultados electorales con un error máximo definido para el intervalo de confianza de 95%.

Los promedios de las 30 o más muestras conforman una distribución normal con media

$$\mu_{\bar{m}}^{l} = \frac{1}{n_m}\sum_{m=1}^{n_m}\bar{x}_m^{l} \cong \mu_{\bar{x}}^{l} = \mu^{l} \qquad (18)$$

y desviación estándar

$$\sigma_{\bar{m}}^{l} = \sqrt{\frac{1}{n_m}\sum_{m=1}^{n_m}\left(\bar{x}_m^{l} - \mu_{\bar{m}}^{l}\right)^2} < \frac{s}{\sqrt{n}}\,. \qquad (19)$$

Por ejemplo, la Figura 2 muestra el resultado de practicar los conteos dinámicos 30 veces en la elección propuesta como ejemplo, correspondientes al primer y segundo lugar.

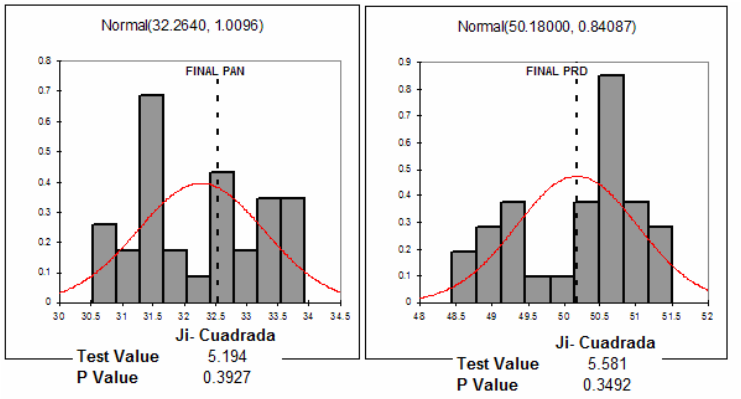


Figura 2. 30 Resultados de 30 procedimientos de conteos dinámicos a las 23:00hrs para la elección de delegado en Milpa Alta (datos ficticios).

El error máximo de estimación para el intervalo de confianza de 95% en caso de 30 conteos dinámicos es

$$Er_p^l < 1.96 \frac{\bar{x}_m^l \sigma_m^l}{2} \sqrt{\frac{C_{30}^{N_m} - 1}{C_{30}^{N_m} - 30}} \qquad (20)$$

donde N_m es numero de casillas computadas en el PREP y

$$C_n^N = \binom{N_m}{30} = \frac{N_m!}{(N_m - 30)!30!}$$

CÓDIGO FUENTE

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <map>
#include <vector>
#include <fstream>
#include <sstream>
#include <set>
#include "votos.h"
#include "voto.h"
#include "base.h"
#include "particion.h"
#include <math.h>

#ifdef  DIPUTADOS
  #define Base BaseDip
  #define Voto VotoDip
  #define BaseNom "base_diputados.txt"
  #define NUMENT  42
  #define pref "DP"
  #define Candidato_comun "Candidato_comun_dip.txt"

#endif

#ifdef  DELEGADOS
  #define Base BaseDel
  #define Voto VotoDel
  #define BaseNom "base_delegacion.txt"
  #define NUMENT  20
  #define pref "JD"
  #define Candidato_comun "Candidato_comun_jd.txt"
#endif

using namespace std;

/*
  Codificación de las candidaturas en común
  - Cada partido está asociado a un bit de un número entero
        bit 0 :    1 : PAN
```

```
        bit 1 :    2 : PRI
        bit 2 :    4 : PRD
        bit 3 :    8 : verde
        bit 4 :   16 : PT
        bit 5 :   32 : Convergencia
        bit 6 :   64 : PANAL
        bit 7 :  128 : PSD

    Por ejemplo, la alianza PRD Convergencia se codifica
    con los bits 2 y 5:  4 + 32 = 36

*/
int LeerCoal(int* coal, int tamano, const char* nombre)
{
    int c;
    string s;
    ifstream dat;

    for (int e=0; e<tamano; ++e)
      coal[e]= 0;

    dat.open(nombre);
    if (!dat)
       return EXIT_FAILURE;

    while ( (c=dat.get()) != EOF )
     {
      if (c==' ' || c=='\t')
        continue;

      dat.putback(c);
      getline(dat,s);
      if (c=='#')
        continue;

      istringstream is(s);

      int n;
      is >> n;
      int p= 0;
      do {
          string part;
          is >> part;
          if (part=="pan")
```

```
            p|= 1;
          else
          if (part=="pri")
            p|= 2;
          else
          if (part=="prd")
            p|= 4;
          else
          if (part=="pt")
            p|= 8;
          else
          if (part=="verde")
            p|= 16;
          else
          if (part=="conv")
            p|= 32;
          else
          if (part=="panal")
            p|= 64;
          else
          if (part=="psd")
            p|= 128;
          else
            {
             cerr << "Partido desconocido en alianza: " << part << endl;
             exit(EXIT_FAILURE);
            }
          c= is.peek();
       } while (c!=EOF);
       coal[n]= p;
      }
    return EXIT_SUCCESS;
}

/*
   Dados los votos de una casilla, y el código de una
   Candidatura común, esta función calcula la suma de los votos
   de la candidatura
*/
int SumaCoal(Votos& v, int cod)
{
   int p= 0;
   int s= 0;
   while (cod)
```

```
    {
     if (cod & 1)
        s+= v[p];
     cod>>=1;
     p++;
    }
  return s;
}


const char* nombrep[] = { "nulos", "pan", "pri", "prd", "pt", "verde", "conv",
                          "panal", "psd", "comun", "Candidato_comun", "nulos" };

/*
   Esta función es el "techo" del logaritmo de base dos de n
*/
int Log2(int n)
{
   int L= 1;
   while (n)
      {
       n/= 2;
       ++L;
      }
   return L;
}


/*
  Esta función forma el nombre de los archivos de salida,
  que son de la forma:
       JDaaaammddhhmmss.csv  para jefes delegacionales
       DPaaaammddhhmmss.csv  para diputados.
  donde:

       aaaa= aÃ±o, mm= mes,  dd= dÃ-a, hh= hora,
       mm= minuto, ss= segundo  en el que estÃ¡ siendo
       realizado el conteo dinÃ¡mico.
*/
string Nombre(const char* pre, time_t* time)
{
   char Time[128];
   tm*  tmx= localtime(time);
   strftime(Time, sizeof(Time), "%Y%m%d-%H%M%S", tmx);
```

```
   ostringstream nombre;
   nombre << pre << Time << ".csv";
   return nombre.str();
}


/*
  Esta función es una tabla de Xi Cuadrada
*/
double XiMax(int n)
{
   switch (n)
     {
      case  5:  return 0.711;
      case  6:  return 1.145;
      case  7:  return 1.635;
      case  8:  return 2.167;
      case  9:  return 2.733;
      case 10:  return 3.325;
      case 11:  return 3.940;
      case 12:  return 4.575;
      case 13:  return 5.226;
      default:  return 5.892;
     }
}

/*
   Una función para calcular el promedio de n datos.
*/
double prom(int n, double* dat)
{
  double pr= 0;
  for (int k=0; k<n; ++k)
     pr+= dat[k];
  return pr/n;
}

/*
   Una función para calcular la desviación estándar de n datos.
*/
double sdev(int n, double* dat)
{
  double s= 0;
  double p= prom(n,dat);
```

```
  for (int k=0; k<n; ++k)
      {
       double t= dat[k]- p;
       s+= t*t;
      }
  return sqrt(s/(n-1));
}

/*
   Una función para calcular el mínimo de n datos.
*/
double min(int n, double* dat)
{
  double mi= dat[0];
  for (int k=1; k<n; ++k)
     if (dat[k]<mi)
        mi= dat[k];
  return mi;
}

/*
   Una función para calcular el máximo de n datos.
*/
double max(int n, double* dat)
{
  double ma= dat[0];
  for (int k=1; k<n; ++k)
     if (dat[k]>ma)
        ma= dat[k];
  return ma;
}


/*

*/
int main(int nargs, char* args[])
{
   const int MAX= Votos::MAX;
   const int MAXCON= 30;

   if (nargs<2)                // archivos de datos.
      {
```

```
    cerr << "forma de uso:\n\t" << args[0] << " arch1.txt arch2.txt ... archN.txt\n";
    return EXIT_FAILURE;
   }


  srand(time(NULL));            // semilla de números aleatorios
  time_t Time;                  // cómputo de la hora del conteo.
  time(&Time);


  ifstream f(BaseNom);          // base de datos de las casilla
  if (f==NULL)
     {
      cerr << "Error al leer las casillas: " << BaseNom << endl;
      return EXIT_FAILURE;
     }

  set<int> caspar[NUMENT];      // casillas participantes en el conteo
  int            cnt[NUMENT];   // número de casillas por entidad.
  map<string,int>  idx[NUMENT];   // mapeo nombre de casilla -> código
  vector<Votos>    inv[NUMENT];   // votos de cada casilla.

  int coal[NUMENT];             // candidato común para cada entidad.
  LeerCoal(coal, NUMENT, Candidato_comun);  // lectura del archivo de
Candidato_comunes

  while(!f.eof())               // lectura de la base de datos
    {                         // de las casillas (proporcionada
     string  linea;             // por el instituto electoral
     f >> linea;
     if (linea.size() > 0)
        {
         Base     B(linea);
         int ent= B.Entidad();
         idx[ent].insert(make_pair(linea, cnt[ent]++));
         inv[ent].push_back(Votos());    // número de votos= 0
        }
    }
  f.close();


  for (int d=0; d<NUMENT; ++d)    // inicio de los codificadores de
     cnt[d]= 0;                 // las casillas.
```

```
for(int fi=1; fi<nargs; fi++)   // lectura de los votos...
  {
   ifstream prep;
   prep.open(args[fi]);
   if (!prep)
     cerr << "no fue posible abrir el archivo " << args[fi] << endl;
   else
     {
      cerr << "Leyendo archivo:\n\t" << args[fi] << endl;
      while (!prep.eof())
        {
         string linea;
         prep >> linea;
         if (linea.size() > 0)
           {
            Voto   V(linea);
            string nom=  V.Casilla();
            int entidad= V.Entidad();
            if (idx[entidad].find(nom)==idx[entidad].end())
              {
               cerr << "ERROR: casilla sin registro: " << nom << endl;
              //  exit(1);
              }
            else
              {          // los votos no se suman, sólo se leen...
               int id= idx[entidad][nom];
               caspar[entidad].insert(id);
               inv[entidad][id][V.Partido()]= V.Votos();
              }
           }
        }
      prep.close();
     }
  }

// Empiezan los conteos dinámicos...

ofstream out (Nombre(pref, &Time).c_str());

for (int ent=0; ent<NUMENT; ++ent) // se hace conteo por cada entidad...
 if (!caspar[ent].empty())
   {
    double       emin;
    vector <int> Par;
```

```
int         tamprep= caspar[ent].size();

cerr << "Entidad= " << ent << " card: " << cnt[ent] << endl;

Par.clear();
for (set<int>::iterator k=caspar[ent].begin(); k!=caspar[ent].end(); ++k)
  Par.push_back(*k);


Votos  sum[MAXCON];
double pords[MAXCON][MAX];
int    caspart[MAXCON];
int    sumavot[MAXCON];
int    n= 0;        // número de conteos que pudieron realizarse

for (int conteo=0; conteo<MAXCON; ++conteo)
 {
  int  casins= cnt[ent];         // número de casillas instaladas.
  int  casviv= Par.size();       // número de casillas "vivas"
  int  numclases= Log2(casviv);  // tamaño de la partición.

  double paso= double(casins)/double(numclases); // ancho de la clase
  Particion P(numclases, paso, Par);     // partición inicial.
  double Xi, h;

  bool Pmal= true;
  while(Pmal)
   {
    if (numclases<=4)           // no aceptamos particiones con
      {                         // 4 clases o menos.
       cerr << "ERROR:  Muy pocas clases; ent= " << ent << endl;
       break;
      }
    if (P.TieneVacias())        // tampoco particiones con
     {                          // clases vacías...
      cerr << "AVISO: Hay clases vacias, intentaremos con "
           << --numclases << " clases;  ent= "<< ent << endl;
/*
      // si requerimos un análisis
      // detallado de la partición,
      // usamos este segmento  de código...
       cerr << "Entidad:," << ent
           << ",Participan," << P.size()
           << ",de," << tamprep << ',' << P.size() / double(tamprep)
```

```
                << endl;
              cerr << P;
              cerr << endl;
*/
              // si la partición tiene clases vacías, se intenta
              // hacer el "ancho" de las clases más grande.
              // i. e., se reduce el número de clases y con esto
              // se calcula una nueva partición, con la que se intenta...
              paso= double(casins)/double(numclases);
              Particion Q(numclases, paso, P);
              P= Q;
              continue;
             }
//           Así, si sabemos ya que el número de clases es >4
//           y la partición no tiene clases vacías, verificamos
//           si la partición cumple con Xi cuadrada de las tablas.

            h= double(P.size()) / double(numclases);
            Xi = P.Xi(h);
            if (Xi >= XiMax(numclases))
              {              // EliminaciÃ³n de casillas por
               P.Ajuste(h);     // encima de la altura mÃ¡xima
               casviv= P.size();    // se actualiza el nÃºmero de
                              // casillas "vivas"
               numclases= Log2(casviv);   // se recalcula la particiÃ³n.
               paso= double(casins)/double(numclases);
               Particion Q(numclases, paso, P);
               P= Q;
              }
            else
              Pmal= false;    // en este punto, tenemos una
                        // partición adecuada para el conteo...
           }

         if (Pmal)            // se reportan las fallas...
           {
            cerr << "Falla conteo: Xi=" << Xi << " XiMax= " << XiMax(numclases) <<
endl;
            continue;
           }
//       una vez encontrada la muestra, se procesa.
          out << endl << endl;
//       reporte de la partición que va a ser usada.
          out << "Entidad,"   << ent  << ",conteo," << n << ",num casillas," << casins <<endl;
```

```
          out << "participan," << P.size() << ",casillas,de," << tamprep << ',' << P.size() /
double(tamprep)
              << endl;
          out << "Altura:," << h << ",Xi2:," << Xi << endl;
          out << P;
          out << endl;

//        cÃ¡lculo de las sumas de la particiÃ³n (conteo dinÃ¡mico)
          for (int t=P.begin(); t>=0; t=++P )
              {
               inv[ent][t].suma(coal[ent]);
               sum[n]+= inv[ent][t];
              }
          caspart[n]= P.size();
          sumavot[n]= sum[n].suma(coal[ent]);

//        cÃ¡lculo de sus promedios...
          double prom[MAX];
          for(int p=0; p<MAX; ++p)
              prom[p]= 0.0;
          for (int t=P.begin(); t>=0; t=++P)
             {
              for(int p=0; p<MAX; ++p)
                 prom[p]+= inv[ent][t].pc(p);
             }
          int nc= P.size();
          for(int p=0; p<MAX; ++p)
               prom[p]/= nc;

//        cÃ¡lculo de sus desviaciones...
          double dvst[MAX];
          for(int p=0; p<MAX; ++p)
              dvst[p]= 0.0;
          for (int t=P.begin(); t>=0; t=++P)
             {
              for(int p=0; p<MAX; ++p)
                 {
                  double s= inv[ent][t].pc(p) - prom[p];
                  dvst[p]+= s*s;
                 }
             }
          for(int p=0; p<MAX; ++p)
              pords[n][p]= sqrt(dvst[p]/nc);
//        el conteo pudo hacerse, se cuenta.
```

```
        ++n;
       }


// reporte de los conteos.
      if (n==0)    // si no pudieron hacerse conteos, se reporta.
        cerr << "ERROR: La entidad= " << ent << " no genera conteos\n";
      else
       {
        out << endl;
        out << "Sumas,Entidad=," << ent <<endl;
        out << "No. de Suma,No. Cas.,";

        for (int p=1; p<MAX; ++p)
           out << ',' << nombrep[p];
        out << ",,";
        for (int p=1; p<MAX; ++p)
           out << ',' << nombrep[p];
        out << endl;


        for (int conteo=0; conteo<n; ++conteo)
           {
            out << conteo << ',' << caspart[conteo] << ','
               << sum[conteo];
            out << ",,";
            for (int p=1; p<MAX; ++p)
               out << ',' << pords[conteo][p];
            out << endl;
           }
        out << endl;
// por último se hacen algunos cálculos estadísticos sobre los
// conteos:  promedios, desviaciones, máximos, mínimos.
        double dat[MAXCON];
        double ma[MAX],maxcas;
        double mi[MAX],mincas;
        double pr[MAX],procas;
        double sd[MAX],sdvcas;

        out << "Conteos," << n << endl;
        out << "Entidad,";
        out << ent <<endl;

        for (int p=0; p<MAX; ++p)
```

```
        {
         for (int conteo=0; conteo<n; ++conteo)
            dat[conteo]= sum[conteo][p] / double(sumavot[conteo]);
         pr[p]= prom(n,dat);
         sd[p]= sdev(n,dat);
         ma[p]= max(n,dat);
         mi[p]= min(n,dat);
        }

      for (int conteo=0; conteo<n; ++conteo)
         dat[conteo]= caspart[conteo];
      procas= prom(n,dat);
      sdvcas= sdev(n,dat);
      maxcas= max(n,dat);
      mincas= min(n,dat);

      out << "Promedio," << procas << ',';
      for (int p=1; p<MAX; ++p)
        out << ',' << pr[p];
      out << ",,";

      for (int p=1; p<MAX; ++p)
       {
        for (int conteo=0; conteo<n; ++conteo)
           dat[conteo]= pords[conteo][p];
        out << ',' << prom(n,dat);
       }
      out << endl;

//    reporte del conteo dinÃ¡mico.
      out << "Desv Est," << sdvcas << ',';
      for (int p=1; p<MAX; ++p)
        out << ',' << sd[p];
      out << ",," << sd[0];
      out << endl;

      out << "Maximos,";
      out << maxcas << ',';
      for (int p=1; p<MAX; ++p)
         out << ',' << ma[p];
      out << ",," << ma[0];
      out << endl;

      out << "Minimos,";
```

```
            out << mincas << ',';
            for (int p=1; p<MAX; ++p)
                out << ',' << mi[p];
            out << ",," << mi[0];
            out << endl;
           }
        }

    out.close();
    return EXIT_SUCCESS;
}
```