

B15

# **DB2 UDB Advanced Analytics for Business Intelligence**

Peter J. Haas



Anaheim, CA

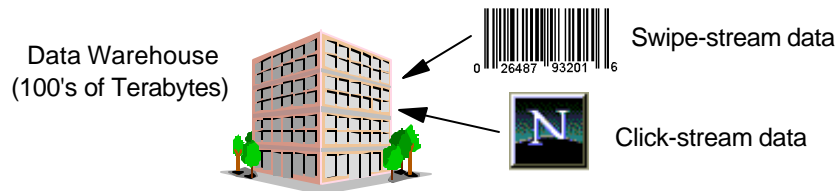
Sept 9 - 13, 2002

© IBM Corporation 2001

- ▶ Our goal is to show how DB2 can be used for some basic data-analytic tasks, and illustrate the surprising power of SQL for data analysis
- ▶ We will introduce the needed basic statistical concepts in a gentle, user-friendly way

# From Data to Knowledge

- The challenge: extracting useful business information from (massive) data (automatically)



- Data analysis via SQL queries
  - processing occurs close to data
  - automatically exploits parallelism
  - can exploit other DB features: incremental maintenance, etc.
  - Can exploit database sampling (see session B16)
- DB2 is surprisingly powerful for analytics!



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ The essence of BI is the extraction of useful business information from one's data, perhaps in an automated manner.
- ▶ These days, the amount of data on hand is often huge, with data warehouses containing 100's of terabytes, and now even petabytes, of data.
- ▶ In this presentation we survey some useful information-extraction techniques that can be performed by means of SQL queries.
- ▶ There are many advantages to using SQL queries, when possible, for data analysis. The user does not need to learn how to use a new statistical package. Processing is more efficient, since the data does not need to be transferred to an application sitting on top of the database. Also, DB2 will automatically choose efficient plans for accessing and manipulating the data, automatically parallelizing computations when possible. And of course, the user benefits from the other database functionality that comes built into DB2, such as incremental maintenance, data security, maintainability, consistency, and recovery.

## Some Different Types of Analyses

- Understanding the overall "shape" of the data
  - summaries
  - pictures
- Detecting outliers
- Detecting dependencies
  - between customers
  - over time
- Statistical modelling
  - for prediction and decision-making
  - functional relationships
  - inference (answering questions)



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ There are a number of useful analyses that we can do:
- ▶ "Descriptive statistics" is concerned with techniques for bringing out interesting patterns and trends in the data, either with numerical summaries or with graphical data visualization methods.
- ▶ Detecting dependencies in the data is important. These include dependencies between customers, between products, between marketing activities and customer behavior, or between customer behavior at different time points.
- ▶ By getting a picture of "normal" trends and variation in the data, we are in a position to identify potential outlier data values (useful for fraud detection, etc.)
- ▶ Finally, we often need to use our data to build statistical models for purposes of prediction (discovering functional relationships), inference (answering questions) and decisionmaking, all in the presence of uncertainty.

## Pertinent Features of DB2 UWO

- Classical aggregation functions
  - SUM, COUNT, AVERAGE, ...
- Statistical functions
  - STDDEV, CORR, REGR\_\*
- OLAP functions
  - ROWNUMBER, RANK, window aggregates, ...
- Other V5-V7 enhancements
  - common table expressions
  - CASE
  - triggers
- Can use SQL to combine tools in new and powerful ways



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ With its new analytical features, DB2 provides a comprehensive and flexible set of tools for accomplishing the foregoing tasks, and hence a powerful framework for data analysis.
- ▶ The DB2 analytical toolbox contains the classical aggregation functions such as sum(), count(), and average(), as well as new statistical functions for correlation and regression analysis.
- ▶ The OLAP functions are powerful in and of themselves, and can be combined with the DB2's statistical functions in many interesting ways.
- ▶ Finally, general enhancements such as common table expressions, random number generation, and logical operators such as the CASE function, extend the range of SQL queries even further.

# CAVEAT

- These queries are not bullet-proof!!
  - not optimized
  - no careful null handling
  - no output formatting
- They **are** educational, though
  - basic functionality
  - tricks

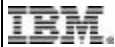


## Classical Summary Statistics

```
VIEW transvw1(country, year, amount)
```

```
select country, year,  
       count(*) as count, sum(amount) as sum,  
       avg(amount) as avg, max(amount) as max,  
       stddev(amount) as stddev  
from transvw1  
group by country, year;
```

COUNTRY	YEAR	COUNT	SUM	AVG	MAX	STDDEV
GERMANY	1998	31	3126.04	100.84	109.75	6.18
GERMANY	1999	24	3549.06	147.87	160.87	7.49
USA	1998	20	4031.20	201.56	249.34	28.91
USA	1999	25	7820.09	312.80	607.98	281.36



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ We now discuss methods for describing our data in order to reveal interesting patterns.
- ▶ Perhaps the simplest way to describe data is to compute summary numbers that describe overall data characteristics. Functions such as count(), sum(), avg(), min(), and max() have been supported since the earliest versions of DB2. More complex aggregates such as stddev() (standard deviation) have been added starting in V5.
- ▶ stddev() is a measure of the variability of the data. The majority of data values are typically within one standard deviation of the mean, and almost all data values are typically within two standard deviations.
- ▶ In the example, USA sales in 1999 are much more variable than sales in other years and in other locations. That is, the amounts in the individual sales transactions vary more widely from their average value of \$312.80.

## Outliers: Credit Card Fraud Detection

- Create a customer card-usage profile table:

```
CREATE VIEW profile(cust_id, avg_amt, sd_amt) AS
  select cust_id, avg(charge_amt), stddev(charge_amt)
  FROM trans
  WHERE date BETWEEN '2002-01-01' and '2002-03-31'
  GROUP BY cust_id;
```

- Detect and flag unusually large charges

```
CREATE TRIGGER big_chrg
AFTER INSERT ON trans
REFERENCING NEW AS newrow FOR EACH ROW MODE DB2SQL
WHEN (newrow.charge_amt > (SELECT avg_amt + 2e0 * sd_amt
                           FROM profile
                           WHERE profile.cust_id =
                               newrow.cust_id))
INSERT INTO big_charges(cust_id, charge_amt)
VALUES(newrow.cust_id, newrow.charge_amt);
```



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

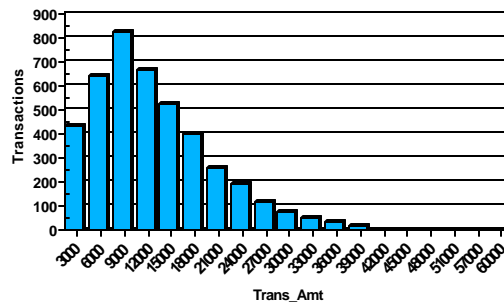
- ▶ One common use for the avg() and stddev() functions is to detect outliers. We illustrate the basic idea in the context of credit card fraud detection.
- ▶ The profile table maintains the average and standard deviation of each customer's card purchases over a reference period.
- ▶ The trigger inserts a customer transaction into the big\_charges table whenever the charge amount is more than two standard deviations above the average.
- ▶ The big\_charges table presumably has triggers that will set off an alarm if, e.g., a customer has more than 3 big charges over a 12 hour period.

## (Equi-Width) Histograms

equi-width histogram for transaction amounts (20 buckets):

```
WITH dt as (SELECT t.transid, sum(amount) as trans_amt,
  case
    when (sum(amount)-0)/((60000-0)/20) < 0 then 0
    when (sum(amount)-0)/((60000-0)/20) > 19 then 19
    else int((sum(amount)-0)/((60000-0)/20))
  end as bucket
  FROM trans t, transitem ti WHERE t.transid=ti.transid
  GROUP BY t.transid)
SELECT bucket, count(bucket) as height, (bucket+1) *
(60000-0)/20 as max_amt FROM dt GROUP BY bucket;
```

BUCKET	HEIGHT	MAX_AMT
0	435	3000
1	645	6000
2	830	9000
3	669	12000
4	533	15000
5	405	18000
6	265	21000
7	192	24000
8	123	27000
9	82	30000
10	55	33000
11	35	36000
12	22	39000
13	7	42000
14	7	45000
15	1	48000



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ This example shows how the CASE statement can be used to create a histogram, that is, a graphical representation of the distribution of transaction amounts.
- ▶ The idea is to assign each data value to one of 20 buckets. The histogram displays the number of data values in each bucket as the height of the associated bar.
- ▶ The histogram is called "equi-width" because the range of data values assigned to a bucket (upper value minus lower value) is the same for each bucket.

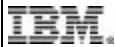
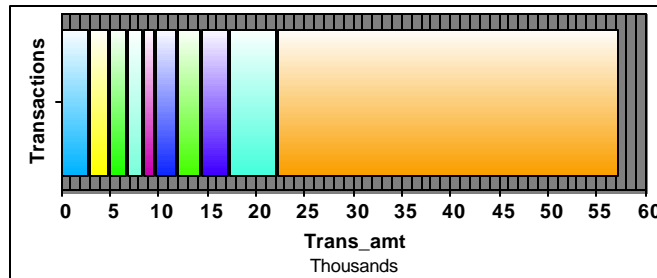


# Quantiles (Equi-Height Histograms)

equi-height histogram for transaction amounts (10 buckets):

```
WITH dt as
  (SELECT t.transid, sum(amount) as trans_amt,
    rownumber() over (order by sum(amount)) * 10 /
    (select count(distinct transid)+1
     from stars.transitem) as bucket
   FROM stars.trans t, stars.transitem ti
   WHERE t.transid=ti.transid GROUP BY t.transid
  )
SELECT bucket, count(bucket) as b_count, max(trans_amt) as
part_value
FROM dt GROUP BY bucket;
```

BUCKET	B_COUNT	PART_VALUE
0	430	2957.54
1	431	5094.14
2	431	6873.05
3	431	8429.81
4	431	9793.69
5	431	12019.40
6	431	14468.20
7	431	17355.26
8	431	22215.92
9	431	57360.41



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

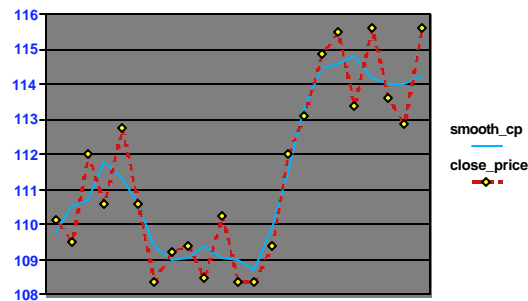
- ▶ An equi-height histogram has bucket boundaries chosen so that each bucket contains approximately the same number of data points.
- ▶ In this example, there are 10 buckets, so that 10% of the data points fall in each bucket. The internal bucket boundaries are often referred to as the 0.1, 0.2, ... , 0.9 *quantiles* of the data distribution, or as 10th, 20th, ... ,90th *percentiles*. For example, the 10th percentile for the example data set is \$2957.54, that is, 10% of transactions have a dollar value that is less than this number.
- ▶ In effect, the query computes the total sales amount for each of N transactions as tran\_amt, sorts the amounts in increasing order, and assigns the number 1 to the smallest transaction, 2 to the next largest transaction, etc., using the rownumber() function. Dividing these numbers by N---which is computed as count(distinct transid)---and rounding to the nearest integer produces the desired bucket number for each transaction.

# Smoothed Time Series

Three-day running-mean smoothed average of IBM stock prices:

```
SELECT date, symbol, close_price,  
       avg(close_price) OVER (order by date rows between 1  
                             preceding and 1 following) AS smooth_cp FROM stocktab  
WHERE symbol = 'IBM' and date between '1999-08-01' and  
       '1999-09-01' ;
```

DATE	SYMBOL	CLOSE_PRICE	SMOOTH_CP
08/02/1999	IBM	110.125	109.8125
08/03/1999	IBM	109.500	110.5416
08/04/1999	IBM	112.000	110.7083
08/05/1999	IBM	110.625	111.7916
08/06/1999	IBM	112.750	111.3333
08/09/1999	IBM	110.625	110.5833
08/10/1999	IBM	108.375	109.4166
08/11/1999	IBM	109.250	109.0000
08/12/1999	IBM	109.375	109.0416
08/13/1999	IBM	108.500	109.3750
08/16/1999	IBM	110.250	109.0416
08/17/1999	IBM	108.375	109.0000
08/18/1999	IBM	108.375	108.7083
08/19/1999	IBM	109.375	109.9166
08/20/1999	IBM	112.000	111.5000
08/23/1999	IBM	113.125	113.3333
08/24/1999	IBM	114.875	114.5000
08/25/1999	IBM	115.500	114.5833
08/26/1999	IBM	113.375	114.8333
08/27/1999	IBM	115.625	114.2083
08/30/1999	IBM	113.625	114.0416
08/31/1999	IBM	112.875	114.0416
09/01/1999	IBM	115.625	114.2500



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

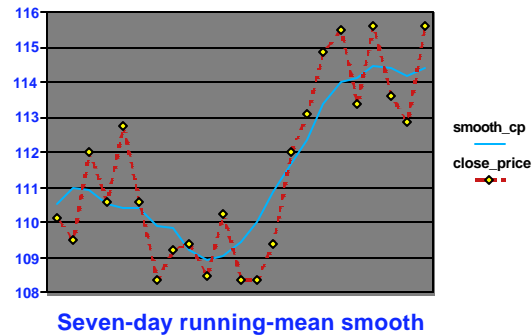
- ▶ Here the moving-window version of the avg() function is used to smooth a data time series in order to reveal underlying trends. Such smoothing is sometimes called "nonparametric regression", because we are fitting a smooth function to the data, but this function does not have the form  $y = ax^2$  or any other such parametric representation.
- ▶ The OVER clause defines the window (containing the points to be averaged) as the current data point plus the point preceding and the point following the current data point, when the points are ordered by date.

# Smoothed Time Series

Seven-day running-mean smoothed average of IBM stock prices:

```
SELECT date, symbol, close_price,
       avg(close_price) over (order by date rows between 3
                             preceding and 3 following) as smooth_cp
FROM stocktab
WHERE symbol = 'IBM' and date between '1999-08-01' and
    '1999-09-01' ;
```

DATE	SYMBOL	CLOSE_PRICE	SMOOTH_CP
08/02/1999	IBM	110.125	109.8125
08/03/1999	IBM	109.500	110.5416
08/04/1999	IBM	112.000	110.7083
08/05/1999	IBM	110.625	111.7916
08/06/1999	IBM	112.750	111.3333
08/09/1999	IBM	110.625	110.5833
08/10/1999	IBM	108.375	109.4166
08/11/1999	IBM	109.250	109.0000
08/12/1999	IBM	109.375	109.0416
08/13/1999	IBM	108.500	109.3750
08/16/1999	IBM	110.250	109.0416
08/17/1999	IBM	108.375	109.0000
08/18/1999	IBM	108.375	108.7083
08/19/1999	IBM	109.375	109.9166
08/20/1999	IBM	112.000	111.5000
08/23/1999	IBM	113.125	113.3333
08/24/1999	IBM	114.875	114.5000
08/25/1999	IBM	115.500	114.5833
08/26/1999	IBM	113.375	114.8333
08/27/1999	IBM	115.625	114.2083
08/30/1999	IBM	113.625	114.0416
08/31/1999	IBM	112.875	114.0416
09/01/1999	IBM	115.625	114.2500



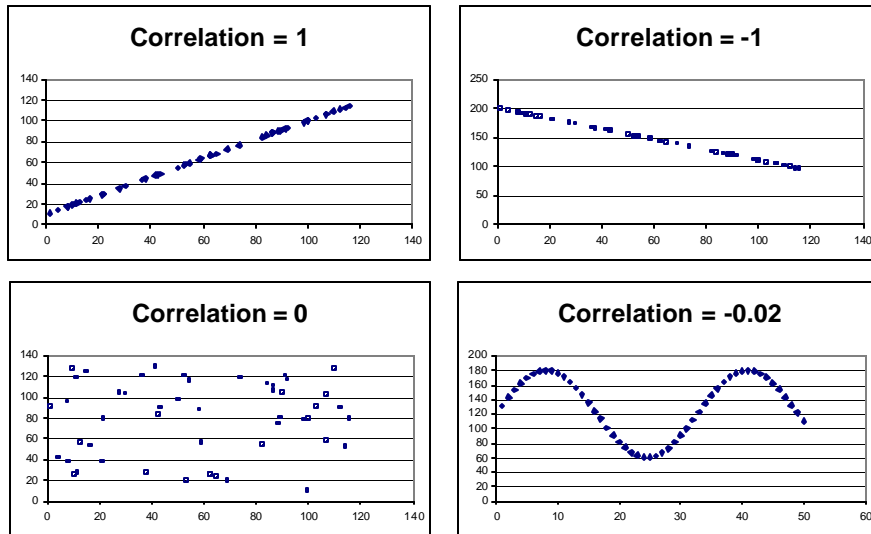
© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ This query is the same as the last one, but now we average over the current data point along with the three points that precede and the three points that follow the current data point.
- ▶ Use of the wider window (7 points vs 3 points) results in a greater degree of smoothing in the fitted (solid blue) curve.

# Detecting Dependencies: Correlation

- Correlation coefficient: measures strength of linear relationship



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ The correlation functions introduced in V6 of DB2 UWO permit detection of linear relationships between two variables.
- ▶ In particular, the correlation coefficient measures the strength of such a linear relationship. A value of 1 (resp., -1) indicated a perfect positive (resp., negative) linear relationship between two attributes, while a value of 0 indicates no apparent linear relationship.
- ▶ In the lower right plot, there is a perfect relationship between the two attributes, but the correlation coefficient has a low value because the relationship is not linear.

## Correlation in DB2

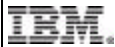
Sales regions where income and purchases are not aligned:

```
VIEW transvw2(country, state, annual_purchases, income)

SELECT country, state,
       correlation(annual_purchases, income) AS correlation
FROM transvw2
GROUP BY country, state
having abs(correlation(annual_purchases, income)) > 0.10;
```

COUNTRY	STATE	CORRELATION
USA	AK	0.78
USA	AL	0.68
USA	DE	-0.30 <=
USA	GA	0.14
USA	KS	0.69
USA	LA	0.48

Can also display covariance (= "unnormalized" correlation)



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ The correlation() function computes the correlation coefficient between two attributes. Each argument to the function need not, of course, be simply a column name, but can be an arithmetic expression involving one or more columns.
- ▶ This query detects anomalous regions where higher income led to apparently lower purchases, as indicated by the negative correlation for the state of Delaware (DE) in the USA. Such a region might merit closer investigation to see what is going on.
- ▶ Note the use of the HAVING clause to restrict the output to cases of "significant" correlation.
- ▶ DB2 also provides a covariance() function, which can be viewed as an "unnormalized" correlation. That is, the covariance is dependent on the unit of measurement and can take on arbitrarily large or small values, rather than being constrained to lie between 0 and 1 as is the case with correlation. Covariance() is usually used to compute other quantities of more direct interest.

## Another Use for Correlation

Customers with similar buying habits:

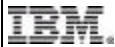
```
VIEW transvw3(custid, prodid, amount)
```

total amount purchased  
over all transactions

```
SELECT a.custid as custid1, b.custid as custid2,  
       corr(a.amount, b.amount) as corr  
FROM transvw3 a, transvw3 b  
WHERE a.prodid = b.prodid and a.custid < b.custid  
GROUP BY a.custid, b.custid  
HAVING corr(a.amount, b.amount) >= 0.5  
       and count(*) > 100  
ORDER BY corr desc;
```

CUSTID1	CUSTID2	CORR
2300	6823	0.99
1071	2300	0.85
1223	4539	0.83
1010	1071	0.78
1010	2300	0.72
1071	6823	0.65

2300 — 1010  
| |  
6823 — 1071  
  
4539 — 1223



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ This query identifies customers with apparently similar buying habits. Whenever customer 2300 bought a large amount of a given product, then customer 6823 also tended to buy a large amount.
- ▶ The HAVING clause restricts the output to cases of high positive correlation, and to cases where there are at least 100 products involved (that is, at least 100 data points are used to compute the correlation).
- ▶ The correlation values can be used informally to cluster customers into similar groups. Here we have identified two clusters.

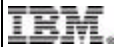
# Autocorrelation

Correlate yearly sales with sales from previous years

```
VIEW transvw4(pgname, year, total_sales)

WITH dt (pgname, year, sales_0, sales_1, sales_2) AS
  (SELECT pgname, year, total_sales,
    max(total_sales) over
      (partition by pgname order by year rows between 1 preceding
        and 1 preceding),
    max(total_sales) over
      (partition by pgname order by year rows between 2 preceding
        and 2 preceding)
    FROM transvw4
  )
SELECT pgname, correlation(sales_0,sales_1)*100 as "correlation1(%)",
  correlation(sales_0,sales_2)*100 as "correlation2(%)",
FROM dt GROUP BY pgname;
```

PGNAME	correlation1(%)	correlation2(%)
antibiotics	-2.83	-29.07
camcorder	-54.78	20.75
coats	-25.40	-1.68
vcr	59.39	33.17



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

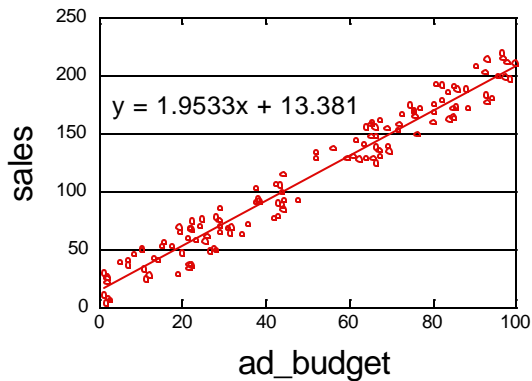
- ▶ It is frequently of interest to study the dependence between current and previous customer behavior. This query computes the correlation between sales for a given year (sales\_0) with both sales from the previous year (sales\_1) and sales from two years before (sales\_2).
- ▶ The moving-window version of the max() function is used in a somewhat tricky way to compute the sales\_1 and sales\_2 columns in the table dt. Note that the "max" is actually computed only for a single value, since the window is of length 1. We could just as well have used the avg(), sum(), or min() functions instead of max(). (But max and min tend to have slightly more stable numerical behavior in general than sum or avg.)

## Least-Squares Fit (Linear Regression)

- Fits a line of the form  $y = ax + b$  from (x,y) pairs
- Ex: effect of advertising budget on 1999 sales

```
SELECT      y      x
regr_count(sales, ad_budget) AS num_cities,
regr_slope(sales, ad_budget) AS a,
regr_icpt(sales, ad_budget) AS b
FROM ad_camp;
```

num_cities	a	b
126	1.9533	13.381



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ The regression suite of functions was introduced into V6 of DB2 UWO. The functions take a set of  $(x_i, y_i)$  data pairs and choose numbers (a,b) to fit a line of the form  $y = ax + b$  that minimizes the sum of the squared errors, where the  $i$ th error is  $e_i = y_i - (ax_i + b)$ .
- ▶ Note that the y value is the first argument and x is the second argument to each regression function.
- ▶ `regr_count()` returns the number of (x,y) pairs used to fit the regression line. A pair is only used if **both** x and y are non-null.



## Fitting Other Types of Curves

- $y = ax^2 + b$

```
SELECT regr_slope(y, x*x) AS a,
       regr_icpt(y, x*x) AS b
```

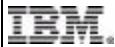
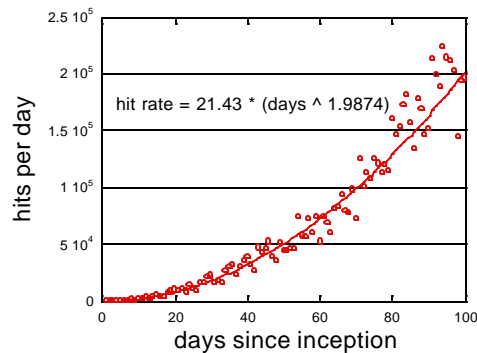
- $y = bx^a$

- $\log y = a \log x + \log b$

```
SELECT regr_slope(log(y), log(x)) AS a,
       exp(regr_icpt(log(y), log(x))) AS b ...
```

```
SELECT
  regr_count(hits,days) as num_days
  regr_slope(log(hits),log(days)) AS a,
  exp(regr_icpt(log(hits),log(days))) AS b
FROM traffic_data;
```

NUM_DAYS	A	B
100	1.9874	21.4302



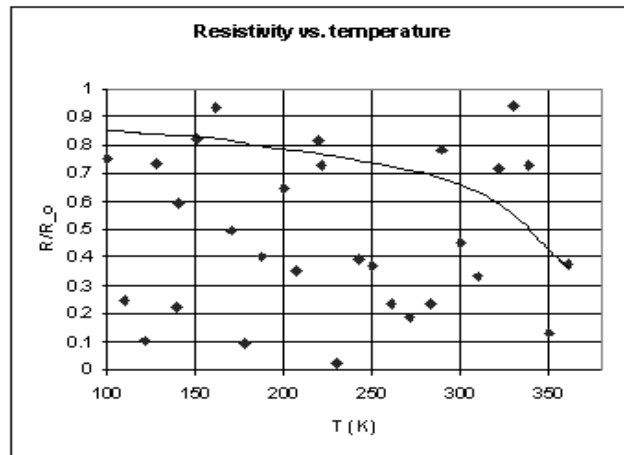
© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ By choosing the arguments to the regression functions to be arithmetic expressions involving the input columns, a variety of curves can be fitted (not just straight lines).
- ▶ The second example plots the daily hit rate at a hypothetical new web site. Since the hit rate rises rapidly at first, a power curve is a reasonable model to try and fit to the data.

## Quality of Fit

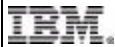
- Want diagnostics!
- especially in automated environment



From L. Kovar, "Band Structure in Germanium, My #\$\$%"

*Ann. Improbable Research*, 7(3), 2001

<http://www.improb.com/airchives/paperair/volume7/v7i3/germanium-7-3.html>



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

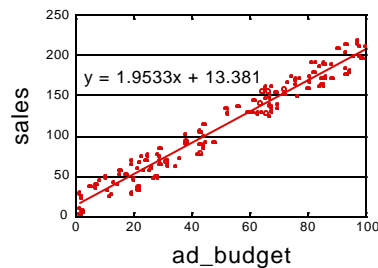
- ▶ As can be seen from this example, the fit of a curve can be quite poor! (The article in which this plot appeared is actually a piece of science humor: a research report written in the style of an extremely disgruntled undergraduate physics major.)
- ▶ It is important to have some measure of goodness-of-fit, especially if the fitting is being done in an automatic-processing environment, so that nobody is actually looking at a plot of the data and fitted curve.

## Quality of Fit - Continued

- R-Squared
  - roughly, the square of the correlation of x and y
  - proportion of y-variation explained by the model

```
SELECT
  regr_count(sales, ad_budget) AS num_cities,
  regr_slope(sales, ad_budget) AS a,
  regr_icpt(sales, ad_budget) AS b,
  regr_r2(sales, ad_budget) as r-squared
FROM ad_camp;
```

num_cities	a	b	r-squared
128	1.9533	13.381	0.95917



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ A standard diagnostic statistic is  $R^2$ , which is essentially the square of the correlation coefficient between x and y.
- ▶  $R^2$  also can be interpreted as the proportion of variation in the y values that is explained by the variation in the x values (as opposed to variation due to randomness or to other variables not included in the model).
- ▶ The function `regr_r2()` computes this quantity automatically.
- ▶ The fit in the example is extremely good, with  $R^2 = 0.96$

## Quality of Fit for Nonlinear Curves

- **Incorrect:** Compute R-Squared for transformed data

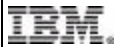
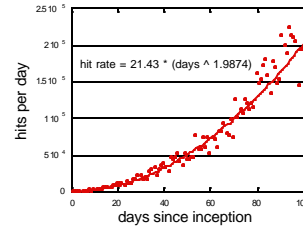
```
select regr_r2(log(hits),log(days))as r2 from traffic_data;
```

**r2: 0.9912**

- **Correct:** Compute R-Squared for original data

```
with coeffs(a,b) as
(select regr_slope(log(hits),log(days)) as a,
      exp(regr_icpt(log(hits),log(days))) as b
 from traffic_data),
residuals(days,hits,error) as
(select t.days, t.hits, t.hits - c.b * power(t.days,c.a)
 from traffic_data t, coeffs c)
select 1e0 - (sum(error*error)/regr_syy(hits,days)) as r2
from residuals;
```

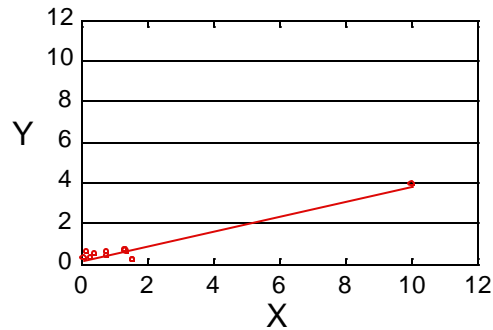
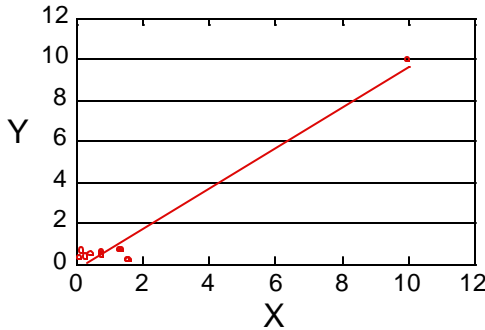
**r2: 0.9554**



- ▶ When using the regression functions to fit a nonlinear curve, some care has to be taken when evaluating the quality of the fit using the R-squared statistic.
- ▶ R-Squared must be evaluated not for the linear fit of the transformed data, but for the nonlinear fit of the original data. The displayed query shows how to do this, by first computing a table of residuals (differences between exact and predicted y-values) for the nonlinear fit.
- ▶ As can be seen, using the built-in R2 function gives an overoptimistic view of the fit---this is typical.

## Influence of Individual Data Points

- Some data are more important than others



- Measure of influence: HAT diagonal

- $h_i = (m_{x2} - 2m_x x_i + x_i^2) / s_{xx}$ 
  - $m_x = \text{avg}(x_1, \dots, x_n)$
  - $m_{x2} = \text{avg}(x_1^2, \dots, x_n^2)$
  - $s_{xx} = (x_1 - m_x)^2 + \dots + (x_n - m_x)^2$



© IBM Corporation 2001

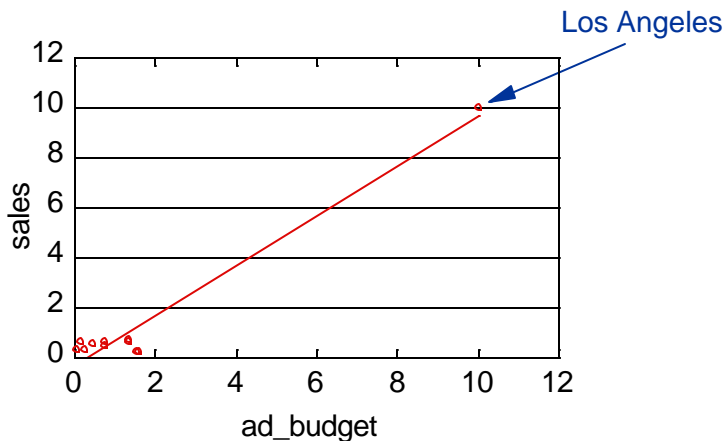
DB2 and Business Intelligence Technical Conference

- As can be seen in this example, certain data points can have a strong influence on the fitted line because the x value is located far from the center of the x values. Changing the y value of the rightmost point in our example completely changes the slope of the regression line.
- Some standard diagnostic statistics used to detect such points are the "HAT matrix" diagonal entries, whose formula is given in the slide.

## HAT Diagonal Computation

```
WITH stats(mx, mx2, sxx) AS
(SELECT regr_avgx(sales,ad_budget),
      regr_avgx(sales,ad_budget*ad_budget), regr_sxx(sales,ad_budget)
 FROM cal_ad_camp
 )
SELECT d.label as city, (s.mx2 - 2 * s.mx * d.x + d.x * d.x) / s.sxx AS
hat
FROM xy_data d, stats s
ORDER BY hat DESC;
```

city	hat
-----	-----
Los Angeles	0.9644
Boonville	0.1222
Grass Valley	0.1195
Yreka	0.1154
Gilroy	0.1099
Lemoore	0.1011
Hilmar	0.1011
Mendocino	0.0923
Turlock	0.0922
Morgan Hill	0.0910
Truckee	0.0910



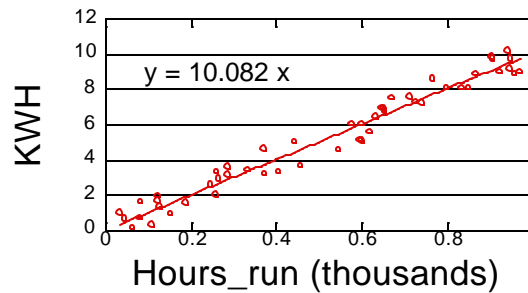
© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ This query computes the HAT diagonals. Note that we write, for example, `regr_avgx(y,x*x)` rather than `avg(x*x)`. We do this for two reasons. First, the function `avg(x*x)` may include `x` values for which the corresponding `y` value is `NULL`, so that the point `(x,y)` was not used in the regression. Secondly, the regression functions are designed to work together, efficiently computing all the necessary statistics in a single pass through the data.
- ▶ Also note that the function `regr_sxx()` is available to conveniently compute the quantity `sxx`. Analogous functions `regr_syy()` and `regr_sxy()` are available. These functions are provided because they occur frequently in the formulas for the various diagnostic statistics used in regression.
- ▶ As expected, the HAT statistic for the rightmost point is almost 10 times larger than the HAT statistics for the other points.

## Regression through the origin

- Fit a line of the form  $y = ax$ 
  - $a = (x_1y_1 + \dots + x_ny_n) / (x_1^2 + \dots + x_n^2)$
  - recall `regr_sxx`, `regr_sxy`
    - `regr_sxx`:  $(x_1 - m_x)^2 + \dots + (x_n - m_x)^2$
    - here  $m_x = \text{avg}(x_1, \dots, x_n)$  as before
  - a trick:
    - $(x_1^2 + \dots + x_n^2) = \text{regr\_sxx} + n m_x^2$
    - $(x_1y_1 + \dots + x_ny_n) = \text{regr\_sxy} + n m_x m_y$



```
SELECT
  regr_count(kwh, hours_run) as num_machines,
  (regr_sxy(kwh, hours_run) + regr_count(kwh, hours_run) *
   regr_avgx(kwh, hours_run) * regr_avgy(kwh, hours_run))
  / (regr_sxx(kwh, hours_run) + regr_count(kwh, hours_run) *
     regr_avgx(kwh, hours_run) * regr_avgx(kwh, hours_run))
  as a;
FROM power_data;
```

num_machines	a
50	10.082



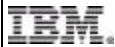
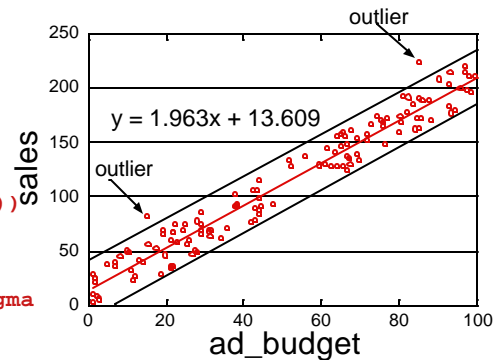
- Sometimes we know in advance that the fitted line **MUST** go through the origin. In our example, we know that no electricity is used when all of the machines are shut off.
- In this case, we need to use a different formula for the slope  $a$ , which involves both sums of  $x_i^2$  terms and sums of  $x_iy_i$  products. The trick shown in the slide can be used to express these sums in terms of quantities that can be computed using functions `regr_sxx()` and `regr_sxy()`.

## Outliers II: Effective Ad Campaigns

- Identify unusually effective campaigns, controlling for ad budget
- Use sigma, the standard deviation about the regression line

```
WITH dt(a, b, sigma) AS
  (SELECT
    regr_slope(sales,ad_budget),
    regr_icpt(sales,ad_budget),
    sqrt((regr_syy(sales,ad_budget)
      - (regr_sxy(sales,ad_budget)
        *regr_sxy(sales,ad_budget)
        /regr_sxx(sales,ad_budget)))
      / (regr_count(sales,ad_budget) - 2))
  FROM ad_camp)
SELECT city, ad_budget, sales
FROM ad_campx ac, dt
WHERE sales > a*ad_budget + b + 2e0*sigma
ORDER BY ad_budget;
```

CITY	BUDGET	SALES
-----	-----	-----
Fresno	15.26	82.00
San Diego	84.99	223.81



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ Here is a second example of outlier detection. We want to identify cities in which the ad campaign is particularly effective. We need to control for the amount of money spend on the ad campaign.
- ▶ To identify outliers, we first define the "normal" relation between advertising budget and resulting sales by fitting a regression line.
- ▶ We then identify as outliers those points that lie more than two standard deviations above or below the regression line, where "standard deviation" is defined in a manner appropriate for regression.
- ▶ The displayed query computes the "regression standard deviation" sigma in terms of the built-in regression functions.



# Multiple Linear Regression

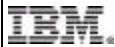
- Fit a line of the form  $y = b_0 + b_1x_1 + \dots + b_nx_n$ 
  - ex:  $y = a_1z + a_2z^2 + b$
- To fit: write in matrix form and solve "normal equations"

$$y = Xb + e$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \dots \\ b_k \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_n \end{bmatrix}$$

Find b's that minimize  $\sum e_i^2$

given      choose      minimize



- ▶ The linear regression methodology can be extended to the case in which there are 2 or more predictor (x) variables.
- ▶ As before, the x values can be functions of the original observations, as in the example.
- ▶ To compute the coefficients  $b_0, \dots, b_n$ , we now have to solve a system of linear equations called the "normal" equations.

## Multiple Linear Regression, Continued

- Compute  $b$  by solving "normal equations":  $[X^T X]b = [X^T y]$
- Multiple regression in DB2 --- a simple approach
  - compute entries of  $[X^T X]$  and  $[X^T y]$  using SQL queries
    - `SELECT x1*x1, x1*x2, x2*x2, x1*y, x2*y ...`
    - matrices are incrementally maintainable
  - solve for  $b$  by feeding entries into a UDF that solves equations (e.g., by Gaussian elimination)
- Research prototype (two  $x$  variables) developed at Almaden
  - Good for fixed problems with changing data
- Issues for an industrial-strength solution
  - general equation-solving UDF
  - robustness to "difficult data"
  - diagnostic statistics ( $R^2$ , etc.)



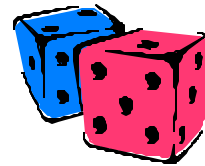
© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ Although the normal equations cannot be solve directly using an SQL query, DB2 can nonetheless be used to solve the fitting problem. A prototype solution has been developed at Almaden Research Center by Kevin Beyer.
- ▶ The idea is to use an SQL query to compute the linear coefficients in the normal equation (the entries of the matrix  $[X^T X]$ ) as well as the constants on the right-hand side (the entries of the vector  $[X^T y]$ ). These coefficients are fed into a UDF that does the actual equation solving (currently by Gaussian elimination).
- ▶ The foregoing approach can be used to solve a wide variety of curve-fitting problems, and is particularly appropriate for a fixed problem in which the data is constantly changing. A full-scale solution requires full generalization of the UDF and being able to deal robustly with ill-conditioned data, e.g., data for which the slope of the line is extremely steep.

# From Description to Modeling

- Scenario 1: business decisions
  - need to make predictions/inferences
  - view data as sample from real world
  - need to distinguish real effects from luck-of-the-draw
- Scenario 2: quick analysis of massive data
  - data is small sample from large database
  - need to assess validity of sample-based computations



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ The discussion so far has focused on the problem of finding interesting patterns in the data.
- ▶ Typically, though, the reason that we are analyzing the data is because we need to make a business decision. To support this decision we need to predict, e.g., customer behavior, or make an inference (answer a question) about our customers.
- ▶ In this case, we need to develop a statistical model of some mechanism out in the world (e.g., customer response to advertising), so that we can do a "what-if?" analysis for alternative scenarios.
- ▶ This means that we no longer view our data as our complete "universe" of information. Rather, we now view our data as being a sample of information generated by the real-world mechanism that we are studying.
- ▶ Since we now view our data as a sample, we need to distinguish meaningful effects from random, luck-of-the-draw fluctuations.
- ▶ In a related scenario, our database might be our entire information universe, but it might be so big that we need to work with a sample of the data. We now need to decide whether conclusions drawn from the sample are likely to be the same as the conclusions that we would have drawn if we were able to look at all of the data.

## Significance of Regression Fit

- View data as a sample
  - $y_i = ax_i + b + \text{error}_i$
- Real effect of x on y, or luck-of-the draw?
- Look at F statistic (with 1 and n-2 "degrees of freedom")
  - measures (y variability caused by x) / (unexplained y variability)
  - if  $a = 0$ , then F should take on "small" values
- A statistical test:
  - let f be observed value of F
  - compute  $\text{Prob}(F \geq f)$  assuming that  $a = 0$
  - suppose that f is so big that  $\text{Prob}(F \geq f)$  is very small
    - unlikely to see this if  $a = 0$
    - therefore, effect of x on y is statistically significant



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- Here we apply the modeling-based view to linear regression, and assume that the data was generated from the model given in the slide. I.e., the y's depend linearly on the x's, but our observations of the y's are clouded by random noise (the error terms, which are assumed random).
- Given a fitted curve, it may be possible that the y values don't depend on the x's at all, but by sheer bad luck the random fluctuations in our data have led us to incorrectly fit a line with a nonzero slope, so that we incorrectly assume a functional dependence of y on x. Thus we need to determine if y really depends on x, or just seems to depend on x because of luck-of-the-draw.
- The classical approach is to compute a statistic that should be small if, in fact  $a=0$ , and whose distribution is known when  $a=0$ . Suppose that the observed value of the statistic is so large so that the probability of seeing this value is very small if, in fact,  $a=0$ . Then we reject the "null hypothesis" that  $a=0$  and assume that the effect of x on y is real. Otherwise, we do not reject the null hypothesis; i.e., our model is not "statistically meaningful".
- For the regression problem at hand, the "F statistic" is usually used for this purpose.

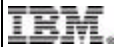
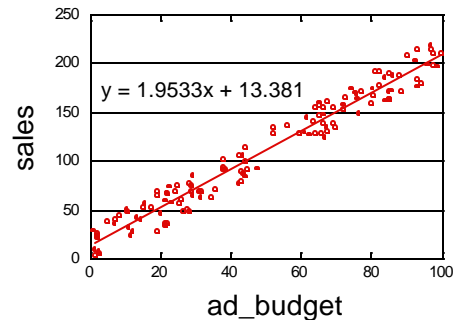
## Significance of Fit, Continued

```
WITH dt(num_cities, a, b, sxx, sigma2) AS
(SELECT
  regr_count(sales,ad_budget),
  regr_slope(sales,ad_budget),
  regr_icpt(sales,ad_budget),
  regr_sxx(sales,ad_budget),
  (regr_syy(sales,ad_budget)
   - (regr_sxy(sales,ad_budget)*regr_sxy(sales,ad_budget)
      /regr_sxx(sales,ad_budget)))
  / (regr_count(sales,ad_budget) - 2)
FROM ad_camp
)
SELECT num_cities, a, b,
  ((a*a*sxx)/sigma2) AS F
FROM dt;
```

num_cities	a	b	F
128	1.9533	13.381	2959.83

Prob( $F_{1,126} > 2959.83$ )  $\ll 0.01$

Caveat: errors need to be iid normal



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ This query shows how to compute the F statistic in terms of the DB2 regression functions.
- ▶ This analysis assumes that the errors in our observations are independent and identically distributed (iid) according to a normal distribution with mean 0. This assumption is often either directly satisfied, or indirectly satisfied after an appropriate transformation of the data.
- ▶ The intermediate constant sigma2 in the query is actually an estimate of the common variance of the errors (i.e., the variance of the foregoing normal distribution).
- ▶ In our example,  $F = 2960$ . If, in fact,  $a=0$ , the probability of seeing such an F value is much less than 1%, so we can assume that our model is statistically significant, and that we are not just fitting a line to random noise.

## Inference: Effectiveness of Ad Campaign

- An experiment
  - Campaign run in City B in January (8 stores)
  - No campaign in "control" City A (9 stores)
  - Monthly sales computed in stores in both cities for February
    - `VIEW feb_sales(city,store_id,sales)`
- Did campaign result in increased sales?
- Classical test: 2-sample t test
  - restrictive normality assumption
  - restrictive equal-variance assumption
- Wilcoxon Rank Test
  - a more modern "nonparametric" procedure
  - avoids restrictive assumptions



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ Here is an example of statistical inference. Suppose that we want to decide whether a certain advertising campaign for a product is "effective", that is, increases sales. We conduct an advertising campaign in January in City B, which has 8 stores. In City A, which has the same demographics as City B, we do not run a campaign. We then look at the total February sales for the product in each store.
- ▶ A classical procedure for deciding whether sales in City B are actually higher is to use a "two-sample t test". This procedure, which is given in most elementary textbooks, requires that total February sales are normally distributed for each city, and that the store-to-store variability of February sales is the same for both cities.
- ▶ A more modern "nonparametric" procedure called the Wilcoxon Rank Test avoids these restrictive assumptions.

## Wilcoxon Rank Test

- Test statistic: sum of ranks of City B in combined ranking

```
WITH ranked_sales(city, ranks) AS
(SELECT city, rank() over (order by sales)
 FROM feb_sales
 )
SELECT sum(ranks) as W
FROM ranked_sales WHERE city = 'B'
```

- Test if W is significantly > than expected value (assuming no diff.)
  - expected value:  $[(n_A n_B) + n_B(n_B + 1)] / 2$   
( $n_x$  = number of stores in City X)
- example:  $n_A = 9$ ,  $n_B = 8$ , and  $W = 94$ 
  - expected value = 72
  - $\text{Prob}(W > 93) = 2\%$  (from tables) if no real difference in sales



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ The Wilcoxon statistic  $W$  is computed as follows. Combine the 17 stores together and rank them in order of increasing sales. Then sum the ranks of those stores that are in City B. (Our query assumes that there are no duplicate sales totals---the `dense_rank()` function can be used to deal with ties.)
- ▶ We can compute the expected value of this sum assuming that there is no real difference in sales between the cities. If the observed value of  $W$  is much higher than this expected value, then it is reasonable to assume that the advertising campaign was indeed effective.
- ▶ The Wilcoxon statistic can be computed using the `RANK` function.
- ▶ For our example, the expected value is 72, and the observed value of  $W$  is 94. Using readily-available tables (see, e.g., the book of Lindgren), we find that the probability of seeing a value 22 units above the mean---when there is no real difference in sales between the cities---is only 1%. So we conclude that the advertising campaign is in fact effective, and the differences are not due to luck-of-the-draw.

## Inference: Test for Independence

- Is there a relationship between operating system and DB product?
- Contingency-table analysis (number of users)

	Sybase	Oracle	
Linux	120	80	200
Unix	45	95	140
Windows	30	12	42
	195	187	382

- A lesser-known "maximum likelihood"  $\chi^2$  test for independence
- test statistic ( $r$  rows and  $c$  columns):

$$\begin{aligned}
 X = & 2n \log(n) \\
 & + [2n_{11} \log(n_{11}) + \dots + 2n_{rc} \log(n_{rc})] \\
 & - [2n_{1+} \log(n_{1+}) + \dots + 2n_{r+} \log(n_{r+})] \\
 & - [2n_{+1} \log(n_{+1}) + \dots + 2n_{+c} \log(n_{+c})]
 \end{aligned}$$

$n_{ij}$ : # in cell  $(i,j)$   
 $n_{i+}$ : row  $i$  sum  
 $n_{+j}$ : column  $j$  sum  
 $n$ : total # users



© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ Another common type of analysis is testing for independence between two "categorical" (non-numeric) attributes. For example, we may wish to know whether there is a relationship between a user's operating system and database product.
- ▶ Suppose that we take a survey of 382 users. The results of this (hypothetical) survey are laid out in a "contingency table".
- ▶ As usual, we want a test statistic that is small if the two attributes are truly independent and large otherwise. Most statistics textbooks discuss "Pearson's Chi-square statistic". A lesser-known, but equally useful chi-square statistic is the "maximum likelihood" statistic whose formula is given on the slide. Note that the sign of each term in the sum depends on the level of aggregation, e.g., totals for individual cells and the grand total are positive while row-sum and column-sum terms are negative.
- ▶ Use of the chi-square statistic is only valid if each cell frequency  $n_{ij}$  is large enough. Typically, each  $n_{ij}$  should be at least 5.



## Test for Independence, Continued

```
WITH c_table(os, db, n, g1, g2) AS
(SELECT os, db, count(*), 2e0*( 0.5e0-grouping(os)), 2e0*(0.5e0-grouping(db))
FROM survey
GROUP BY CUBE(os,db))
SELECT sum(g1*g2*2e0*n*log(n)) as X
FROM c_table
```

x	os	db	n	g1	g2
34.114	Linux	SYB	120	1.0	1.0
	Linux	-	200	1.0	-1.0
	-	SYB	195	-1.0	1.0
	-	-	382	-1.0	-1.0
...					

c\_table

- If data is truly independent:
  - X should be close to 0
  - X has  $\chi^2$  distribution with  $(r-1)(c-1)$  degrees of freedom
- Computer example:  $r = 3, c = 2$ 
  - if independent,  $\text{Prob}(X > 34.114) < 0.001\%$



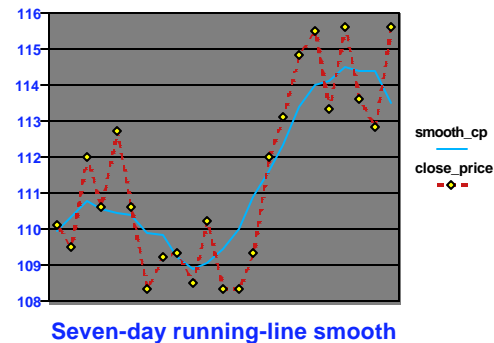
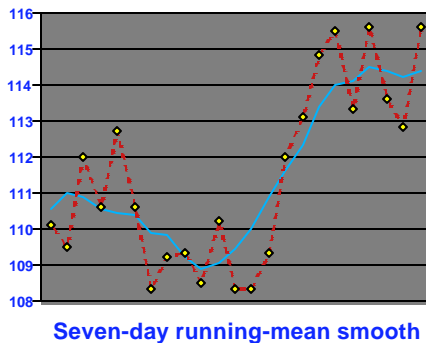
© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ In DB2, all of the entries in a contingency table can be simultaneously computed using the CUBE operator. Some sample entries in the resulting c\_table are given at right.
- ▶ To compute the chi-square statistic, we use DB2's grouping() function in a tricky way. For a given row in a CUBE table, grouping(column)=1 if the column has been "aggregated away", and grouping(column)=0 otherwise. For example, the second row in c\_table represents the number of Linux users, which is obtained by summing over the db attribute. For this row grouping(db)=1 since we have summed over the db attribute, and grouping(os)=0. In our query we have rescaled the value of the grouping function to be either -1 or +1, thereby obtaining the derived attributes g1 and g2. By multiplying each term in the overall sum by g1\*g2, the signs of the terms come out exactly as we need them.
- ▶ For our example, the value of the chi-square statistic turns out to be 34.114. If os and db are truly independent, the probability of seeing such a large value is less than 0.001%. (We do a lookup in standard tables or do a standard numerical computation.) We can therefore assume that there is dependence between os and db.

# Combining Regression and Windowing

- Running-line estimator (Hastie & Tibshirani, 1990)
  - fits a line  $y = ax + b_i$  to local neighborhood of each  $(x_i, y_i)$  point
  - smoothed  $y$  value is given by  $y_i^{\text{smooth}} = ax_i + b_i$
  - better behavior at endpoints, better statistical properties



IBM

© IBM Corporation 2001

DB2 and Business Intelligence Technical Conference

- ▶ The regression functions can be combined with windowing to compute a "running-line" estimator. Rather than just obtaining a smoothed  $y$  value by averaging the true  $y$  value with some of its neighbors, as we did before with the running mean estimator, we now fit a regression line to  $y$  and its neighbors, and use the  $y$ -value of the fitted line as the smoothed  $y$  value.
- ▶ The running-line estimator is known to behave more nicely than the running-mean estimator.
- ▶ Note how the running-line estimator captures the trend in the rightmost data values more accurately than does the running mean estimator.
- ▶ The running-line and running-mean estimators can be viewed as "nonparametric regression estimators": we fit a curve to the data, but we don't assume a specific form for the curve (linear, quadratic, exponential) *a priori*.

## Regression and Windowing: Cont'd

- Would like to execute the following query:

```
WITH dt(day,date,symbol,close_price) AS
  (SELECT cast(row_number() over (order by date) as float),
    date, symbol, close_price
  FROM stocks WHERE symbol = 'XYZ' and
    date between
    '1999-08-01' and '1999-09-01'
  )
SELECT date, symbol, close_price,
  day * (regr_slope(close_price,day)
    over (order by day rows between
      3 preceding and 3 following))
  + regr_icpt(close_price,day) over (order by day
    rows between 3 preceding and 3 following)
  AS smooth_cp
FROM dt;
```

- Doesn't quite work yet
  - Work-around by expanding:  $\text{regr\_slope}(y,x) = \text{covar}(y,x) / \text{var}(x)$  etc.

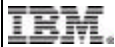


- ▶ Ideally, we would use the displayed query to compute the running-line estimator.
- ▶ Note that we use the rownumber() function to create the x values for our regression, and that we cast the rownumber to a float to ensure good numerical behavior.
- ▶ Unfortunately, this query doesn't quite work, because the regression functions are not completely compatible with windowing yet. (They will be in the future).
- ▶ We can still run the query by expressing the regression functions in terms of the variance(), covariance(), and avg() functions, which ARE compatible with windowing.

## A Regression and Windowing Query

- Final query (assumes no NULLs):

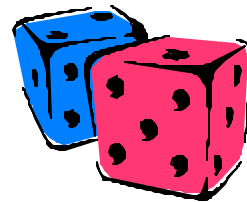
```
with
dt(day,date,symbol,close_price) as
  (select cast(row_number() over (order by date) as real),
    date, symbol, close_price from stocks
  ),
ddt(day,date,symbol,close_price,slope,avgx,avgy) as
  (select day, date, symbol, close_price,
    covar(close_price,day)
      over (order by day rows between 3 preceding and 3 following) /
    var(day)
      over (order by day rows between 3 preceding and 3 following),
    avg(day)
      over (order by day rows between 3 preceding and 3 following),
    avg(close_price)
      over (order by day rows between 3 preceding and 3 following)
    from dt
  )
select date, symbol, close_price,
  day * slope + (avgy-slope*avgx) as smooth_cp
from ddt;
```



- ▶ Here is the resulting query, which is somewhat more cumbersome than the original, but still does the job.
- ▶ This version of the query does not check for NULL x or y values. The query will work as written if, e.g., it is applied to a table expression obtained from the original table by removing all rows having a NULL x or y value.

# Taming Massive Data: Sampling

- Sampling for
  - auditing or "fuzzy exploration"
  - quick approximate answers to aggregation queries
  - making analytics and datamining scalable
- Technology challenges
  - generating a representative sample efficiently
  - estimating an aggregate
  - assessing precision of estimate
- Sampling in DB2 --- Present and Future
  - Go to session B16



© IBM Corporation 2001

DB2 and Business Intelligence Technical  
Conference

- ▶ Sampling is a tool that can often be used to "tame" massive data.
- ▶ One common application is to produce a representative subset of the rows in a table for purposes auditing or "fuzzy exploration". The latter concept refers to the fact that by simply getting one's hands on the raw data, one can often discover interesting features without having to compose an explicit query.
- ▶ Another common application of sampling is to obtain quick approximate answers to aggregation queries (SUM queries, COUNT queries, etc.).
- ▶ Finally, sampling is essential in scaling analytical and data mining algorithms to handle vast amounts of data.
- ▶ The key challenges, then, are how to obtain a representative sample, how to use the sample to estimate a value of an aggregate, and how to assess the precision of this estimate.
- ▶ See my talk in session B16 for an in depth discussion of current and future sampling capabilities of DB2.

# Selected References

- **Introductory Statistics**

- Larry Gonick, et al : *The Cartoon Guide to Statistics*, HarperCollins, 1994
- Jeffrey Clark and Douglas A. Downing: *Forgotten Statistics : A Self-Teaching Refresher Course*, Barrons, 1996
- Lloyd R. Jaisingh: *Statistics for the Utterly Confused*, McGraw-Hill, 2000

- **Advanced Statistics**

- T. J. Hastie and R. J. Tibshirani: *Generalized Additive Models*, Chapman & Hall/CRC, 1999
- B. W. Lindgren: *Statistical Theory*, 3rd Ed., MacMillan, 1976
- R. G. Miller: *Beyond ANOVA, Basics of Applied Statistics*, Wiley, 1986
- R. H. Myers: *Classical and Modern Regression with Applications*, 2nd Ed., Duxbury, 1990
- C.-E. Sarndal, B. Swenson, and J. Wretman: *Model Assisted Survey Sampling*, Springer-Verlag, 1992



# Many Thanks To...

- Kevin Beyer
- Guy Lohman
- Eric Louie
- Bob Lyle
- Hamid Pirahesh
- Ashutosh Singh
- Markos Zaharioudakis
- ...



## For More Information

- Go to: [www.almaden.ibm.com/cs/people/peterh](http://www.almaden.ibm.com/cs/people/peterh)
  - Latest version of this talk
  - CLI files for executing the queries in this talk
- Forthcoming Redbook on BI in DB2

