Fast Permutation Tests, Especially for Multiple Comparisons and Even When One Sample is Large, that Efficiently Maximize Power Under Conventional Monte Carlo and Allow for Simultaneous Permutation-Style P-Value Adjustments

J.D. Opdyke, DataMineIt

Contents

DataMineIt SM
J.D. Opdyke

1. Goal and Rationale

- **GOAL:**
  Quickly implement many non-parametric permutation tests, even when one sample in a pair is large, with maximum power under conventional Monte Carlo

- **WHY MANY TESTS &/OR ONE LARGE SAMPLE?**
  - **"Parity Testing" in Regulatory Telecom OSS Reports**
  - **Medical studies using MRI data**
  - **clinical trials with large controls and many and smaller studies**
  - **Any multiple comparisons context requiring permutation-style p-value adjustments of permutation test p-values (and thus, computationally intensive nested sampling loops)**

DataMineIt SM
J.D. Opdyke

1. Goal and Rationale

- **WHY CONVENTIONAL MONTE CARLO?**
  - **Faster, more efficient sampling techniques (e.g. various methods of importance sampling) are not always implementable**
  - **when such methods can be implemented but their results are suspect, conventional Monte Carlo can be a useful verification**
- **WHY MAXIMUM POWER?**
  - **best test, all else equal**

DataMineIt SM
J.D. Opdyke

2. Permutation Sampling, Duplicate Samples, & Power

- PROC PLAN, PROC MULTTEST, PROC NPAR1WAY, & PROC TWOSAMPL® can sample without replacement *within* a sample, as required of permutation tests
- None can sample without replacement *across* samples (i.e. none can avoid drawing duplicate samples)
- Duplicate samples → loss of power due to increased variance of estimated p-value

DataMineIt[SM]
J.D. Opdyke

3. Maximizing Power Under Conventional Monte Carlo

- Use “oversampling” to efficiently obtain a unique set of samples (no duplicates)
  a. draw more samples than desired (r)
  b. delete duplicates
  c. randomly select the desired number (T) of samples from the remainder
  d. recall PROC PLAN if fewer than T samples remain
- “Oversampling” preserves the uniform distribution sampling assumption of nonparametric permutation tests

DataMineIt[SM]
J.D. Opdyke

4. Efficiently "Oversample" Based on Expected Runtime

- **Draw just enough "extra" samples (r-T) to minimize expected runtime**

- **Expected Runtime = $g(n_1, n_2, r, T)$ =
PROC PLAN RunTime *
expected # of Calls To PROC PLAN =
$PPRT(r, [n_1+n_2])$ *
$CTPP(r, T, [n_1+n_2]!/[n_1!n_2!])$**

- **Choose optimal r, r*, such that $\partial g/\partial r = 0$**

DataMineIt SM
J.D. Opdyke

4. Efficiently "Oversample" Based on Expected Runtime

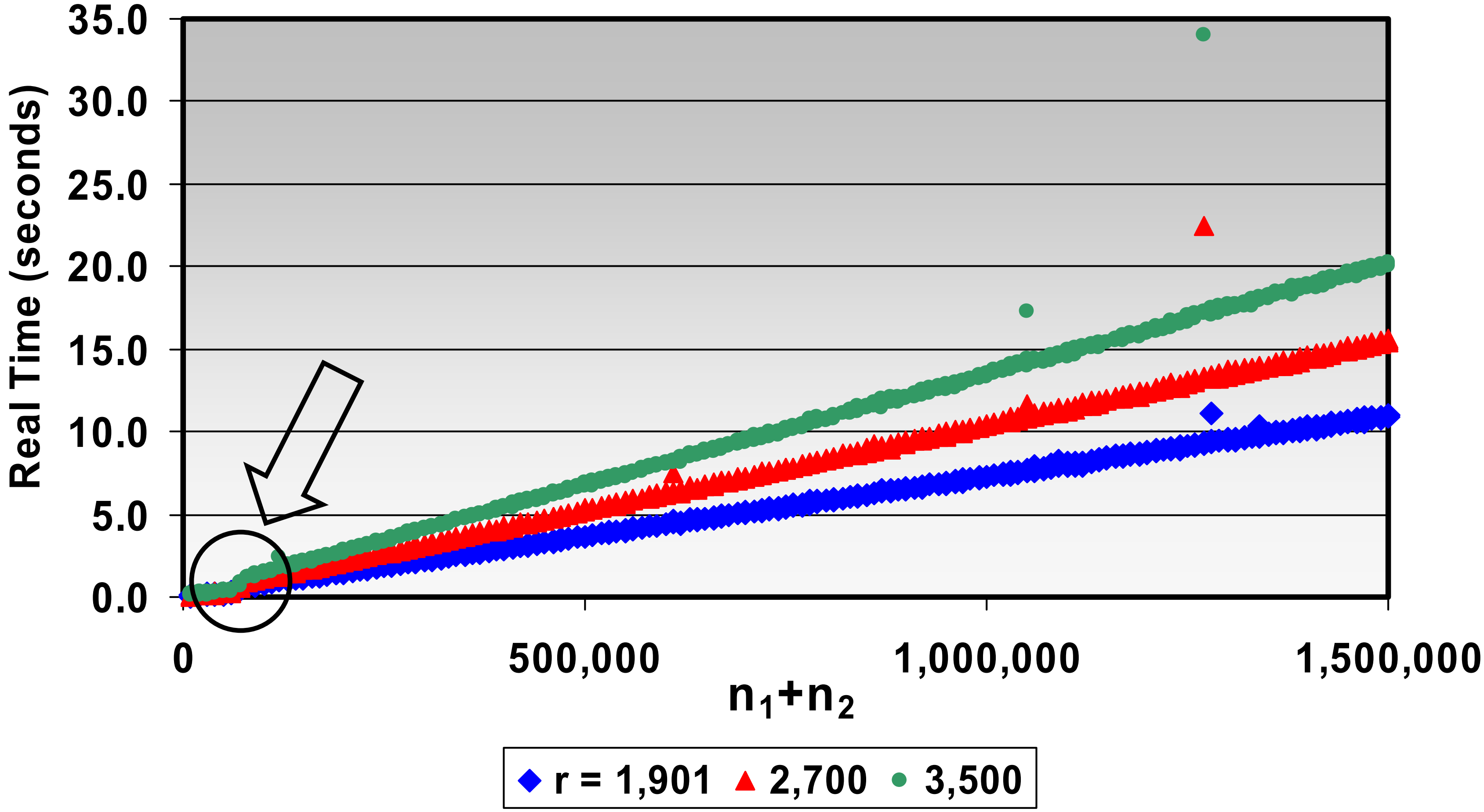
- **PPRT(r,[n_1+n_2]) ≈ β_0 + β_1*(n_1+n_2) + β_2*r + β_3*r *(n_1+n_2)**

- **CTPP(r, T, [n_1+n_2]!/[n_1!n_2!]) =**

$$\left(\frac{1}{p}\right)=\left(\sum_{j=T}^{r}\left[\frac{\dfrac{(n_1+n_2)!}{n_1!n_2!}}{j!\left(\dfrac{(n_1+n_2)!}{n_1!n_2!}-j\right)!}\sum_{i=0}^{j}\frac{(-1)^i\, j!(j-i)^r}{i!(j-i)!\left(\dfrac{(n_1+n_2)!}{n_1!n_2!}\right)^r}\right]\right)^{-1}$$

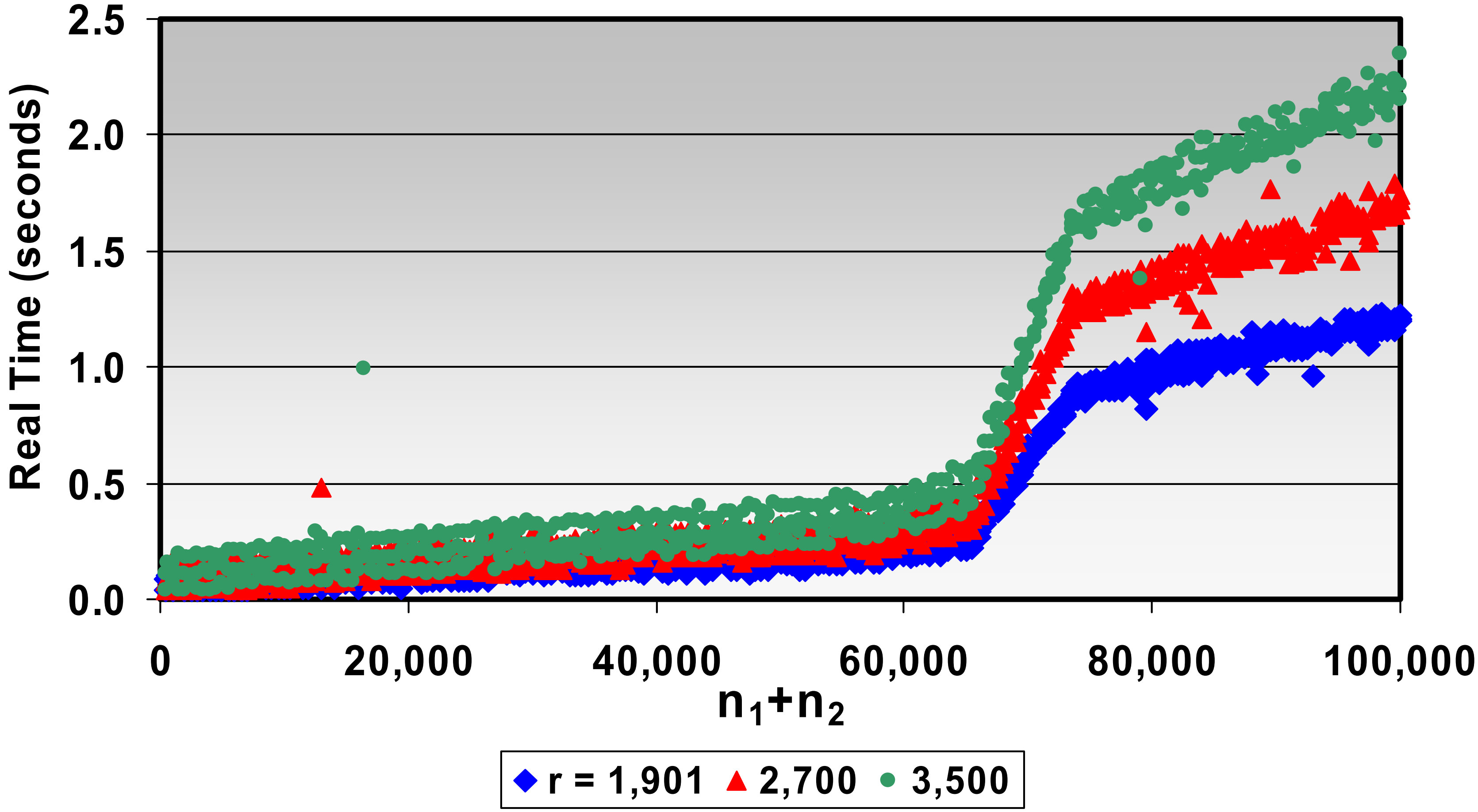
DataMineIt[SM]
J.D. Opdyke

GRAPH 1: PROC PLAN Runtime by r by n_1+n_2



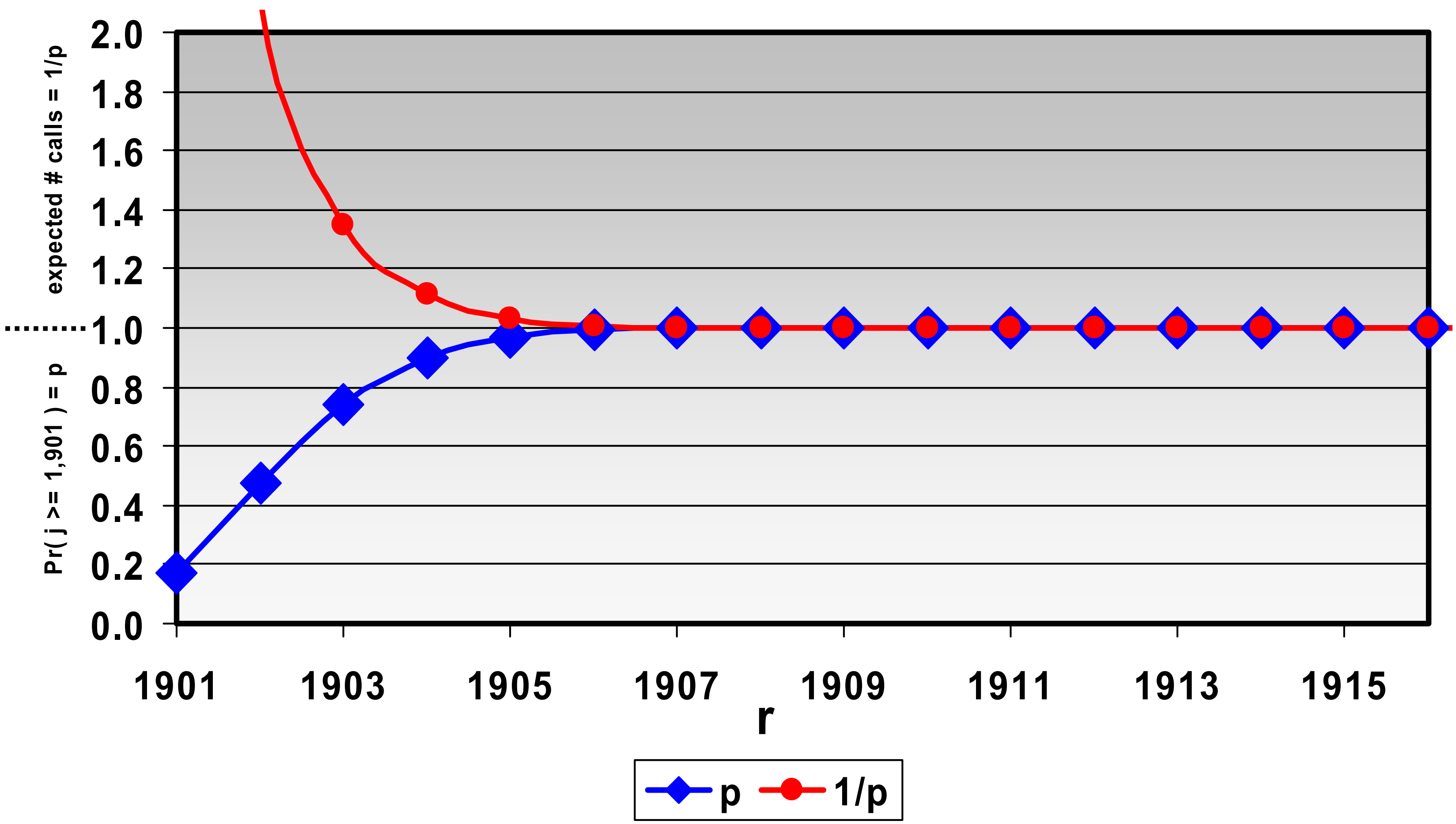
DataMineIt[SM]
J.D. Opdyke

GRAPH 2: PROC PLAN Runtime by r by n_1+n_2



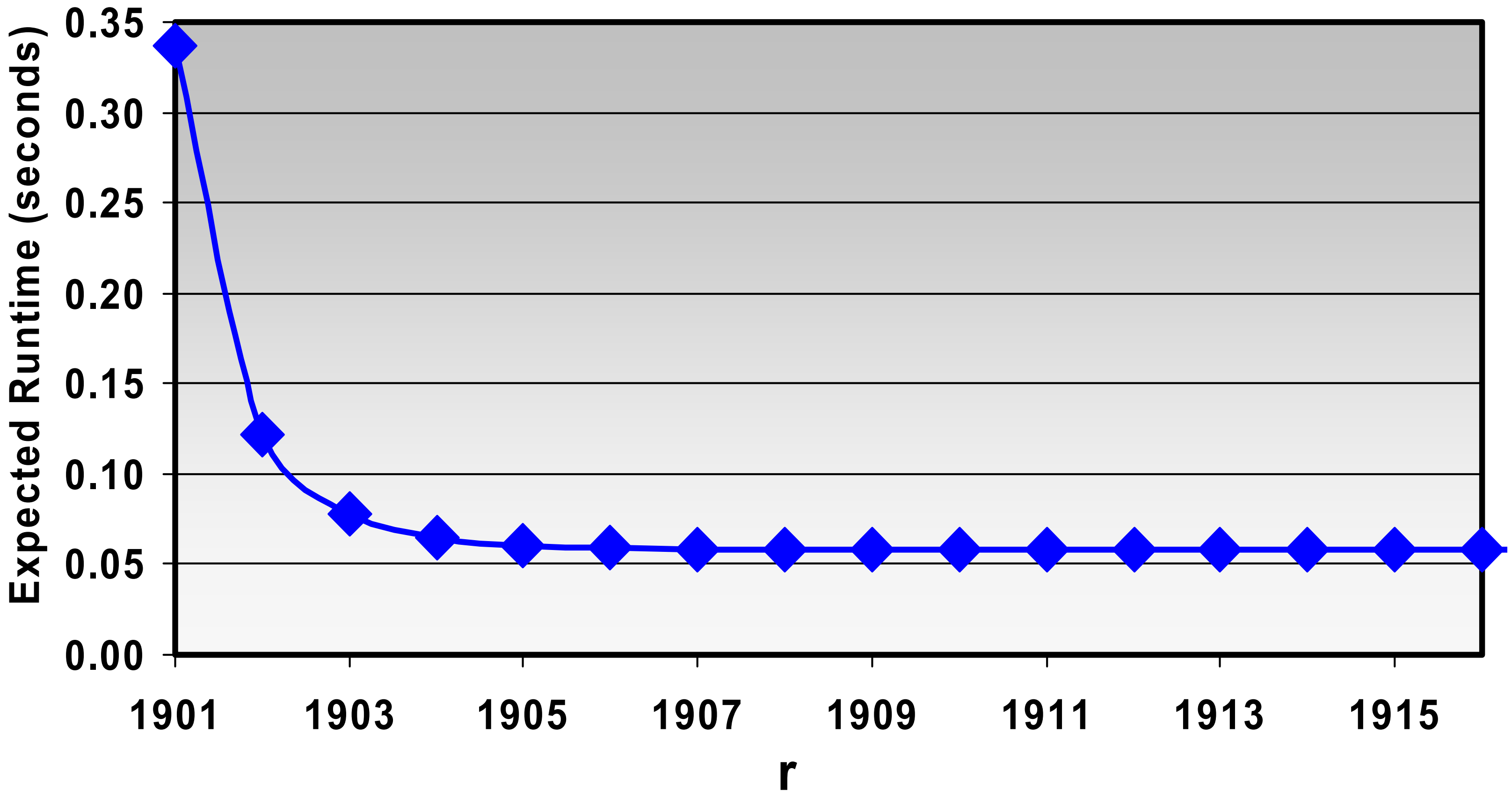
DataMineIt SM
J.D. Opdyke

GRAPH 3: Probability of At Least T Unique Samples (p) & Expected Number of Calls to PROC PLAN (1/p) by r (for n_1=68, n_2=4, and T=1,901)



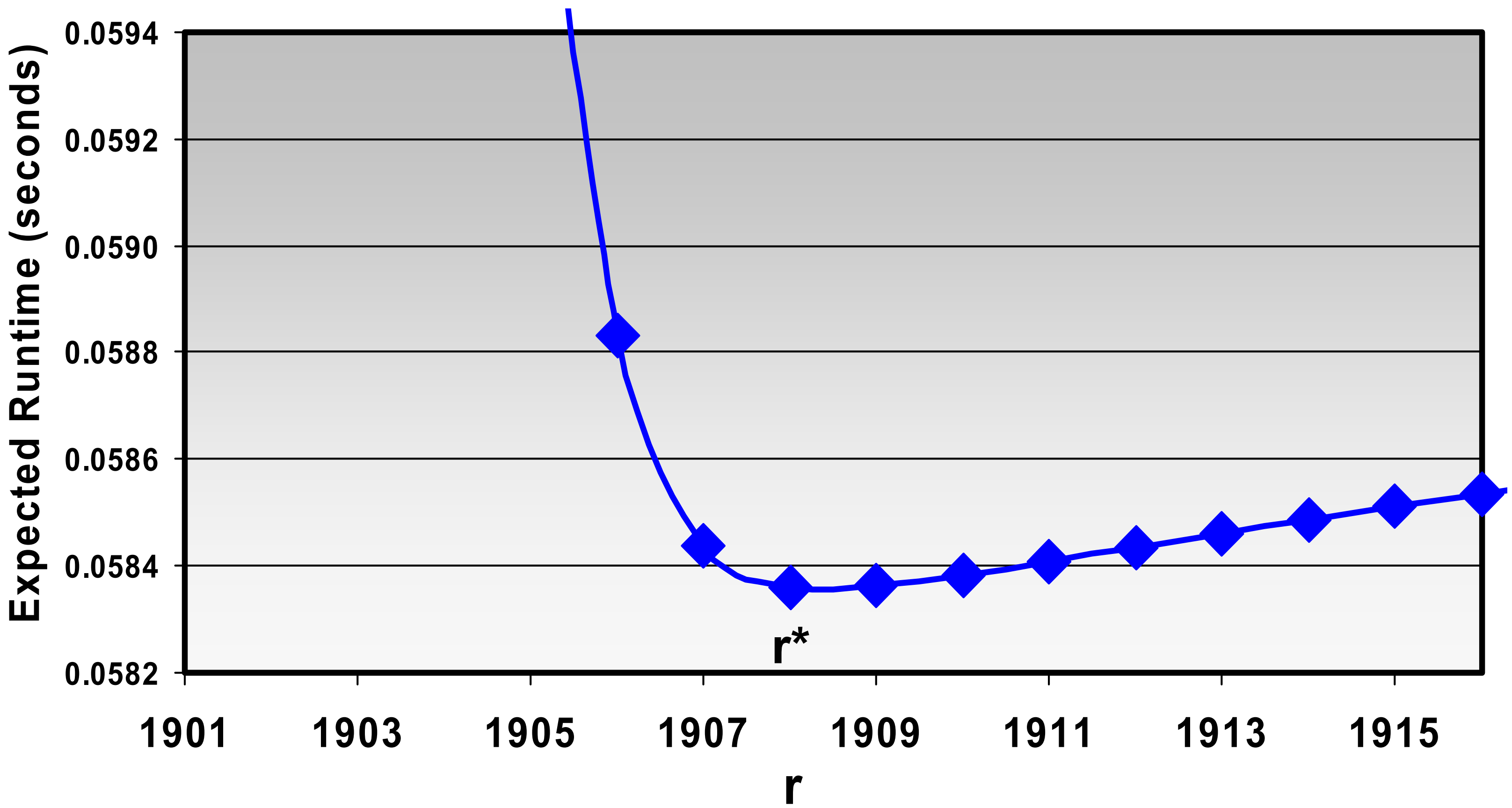
DataMineIt SM
J.D. Opdyke

GRAPH 4: Expected Runtime (1/p * one runtime) by r (for n_1=68, n_2=4, and T=1,901)



DataMineIt SM
J.D. Opdyke

GRAPH 5: Expected Runtime (1/p * one runtime) by r (for n_1=68, n_2=4, and T=1,901)



5. Approximate Optimal r*

- Precision required to numerically calculate r* is too high to do “on the fly” in SAS® for every n_1 & n_2
- However, since $\partial g/\partial r \approx 0$ for r slightly > r*, we can approximate:
  a. Define ranges based on combinations $(n_1+n_2)!/n_1!n_2!$
  b. Pick suboptimal r*s corresponding to each lower bound to obtain largest r* for each range
  c. Runtime of suboptimal r* ≈ runtime of r* because $\partial g/\partial r \approx 0$ for r slightly > r* (see Graphs 4 & 5)

5. Approximate Optimal r*

$C = (n_1+n_2)!/n_1!n_2!$	"low-end" r*	p (lower bound)	1/p (lower bound)
C < 10,626	C	1.0 (assuming C ≥ T)	1.0
10,626 ≤ C < 52,360	2,138	0.997929320330667	1.002074976280530
52,360 ≤ C < 101,270	1,956	0.999058342955471	1.000942544598290
101,270 ≤ C < 521,855	1,934	0.999429717692296	1.000570607715190
521,855 ≤ C < 1,028,790	1,912	0.999726555240808	1.000273519551680
1,028,790 ≤ C < 10,009,125	1,908	0.999512839120371	1.000487398321020
10,009,125 ≤ C < 25,637,001	1,904	0.999961594180711	1.000038407294350
25,637,001 ≤ C < 100,290,905	1,903	0.999944615376581	1.000055387691050
100,290,905 ≤ C < 5,031,771,045	1,902	0.999839691379204	1.000160334323770
5,031,771,045 ≤ C	1,901	0.999641154940541	1.000358973875460

DataMineIt SM
J.D. Opdyke

6. How Much Power Gain…?

- **Permutation test p-values relying on any type of sampling will have actual size level (asl) > α**
- **∴ either p-values or critical value (c_α) should be adjusted**
- **Smaller variance of no replacement sampling (NR) ⇒ smaller asl ⇒ larger c_α^* ⇒ larger power**
- $\sigma^2_{NR} < \sigma^2_{WR}$ ⇒ $asl_{NR} < asl_{WR}$ ⇒ $c_\alpha{}^*{}_{NR} > c_\alpha{}^*{}_{WR}$ ⇒ $power_{NR} > power_{WR}$

DataMineIt SM
J.D. Opdyke

6. How Much Power Gain…?

- σ^2_{WR} is based on the binomial, $\sigma^2_{bin} = n_p pq$
  σ^2_{NR} based on hypergeometric, $\sigma^2_{hyp} = n_p pq(N-n_p)/(N-1)$
  (N = # possible samples, n_p = # permutation samples)

- $\sigma^2_{bin} > \sigma^2_{hyp} \Rightarrow \sigma^2_{WR} > \sigma^2_{NR}$

- $asl_{WR} = \Pr(S \le n_p{*}\alpha \mid p) = \dfrac{1}{n_p}\sum_{i=0}^{n_p}\sum_{k=0}^{\lfloor n_p\alpha\rfloor}\binom{n_p}{i}\left(\dfrac{i}{n_p}\right)^k\left(1-\dfrac{i}{n_p}\right)^{(n_p-k)}$

- $asl_{NR} = \Pr(S \le n_p{*}\alpha \mid p) = \dfrac{1}{n_p}\sum_{S=0}^{N\ by\ \frac{N}{n_p}}\ \sum_{k=0}^{\lfloor n_p\alpha\rfloor}\dfrac{\binom{S}{k}\binom{N-S}{n_p-k}}{\binom{N}{n_p}}$

6. How Much Power Gain…?

- if (asl / α) is essentially constant close to $\alpha = 0.05$, then
  $c_\alpha{}^* \times (asl / \alpha) = \alpha$
  $c_\alpha{}^* = \alpha^2 / asl$

- $\therefore\ c_\alpha{}^*{}_{WR} = \alpha^2 / asl_{WR}$ and $c_\alpha{}^*{}_{NR} = \alpha^2 / asl_{NR}$

- power can only be obtained via simulation, but by CLT we know that asymptotically:

  power = $1 - \Phi\left(z_\alpha - \frac{\delta\sqrt{n}}{\sigma} \right)$ where δ = effect, $z_\alpha = \Phi^{-1}(1-\alpha)$

- $\text{power}_{NR} \approx 1 - \Phi\left(z_{c^*_{\alpha_{NR}}} - \frac{\delta\sqrt{n}}{\sigma} \right)$, $\text{power}_{WR} \approx 1 - \Phi\left(z_{c^*_{\alpha_{WR}}} - \frac{\delta\sqrt{n}}{\sigma} \right)$

6. How Much Power Gain…?

# Permutation Samples, n_p	# Possible Samples, N	asl_{NR}	asl_{WR}	$c_{\alpha}^{*}{}_{NR}$	$c_{\alpha}^{*}{}_{WR}$
1,000	1,000	0.05100	0.05144	0.04902	0.04859
1,000	2,000	0.05122	0.05144	0.04880	0.04859
1,000	5,000	0.05136	0.05144	0.04868	0.04859
1,000	10,000	0.05140	0.05144	0.04863	0.04859
1,901	1,901	0.05050	0.05073	0.04951	0.04927
1,901	3,802	0.05062	0.05073	0.04939	0.04927
1,901	9,505	0.05069	0.05073	0.04932	0.04927
5,000	5,000	0.05020	0.05028	0.04980	0.04971
5,000	10,000	0.05024	0.05028	0.04976	0.04971

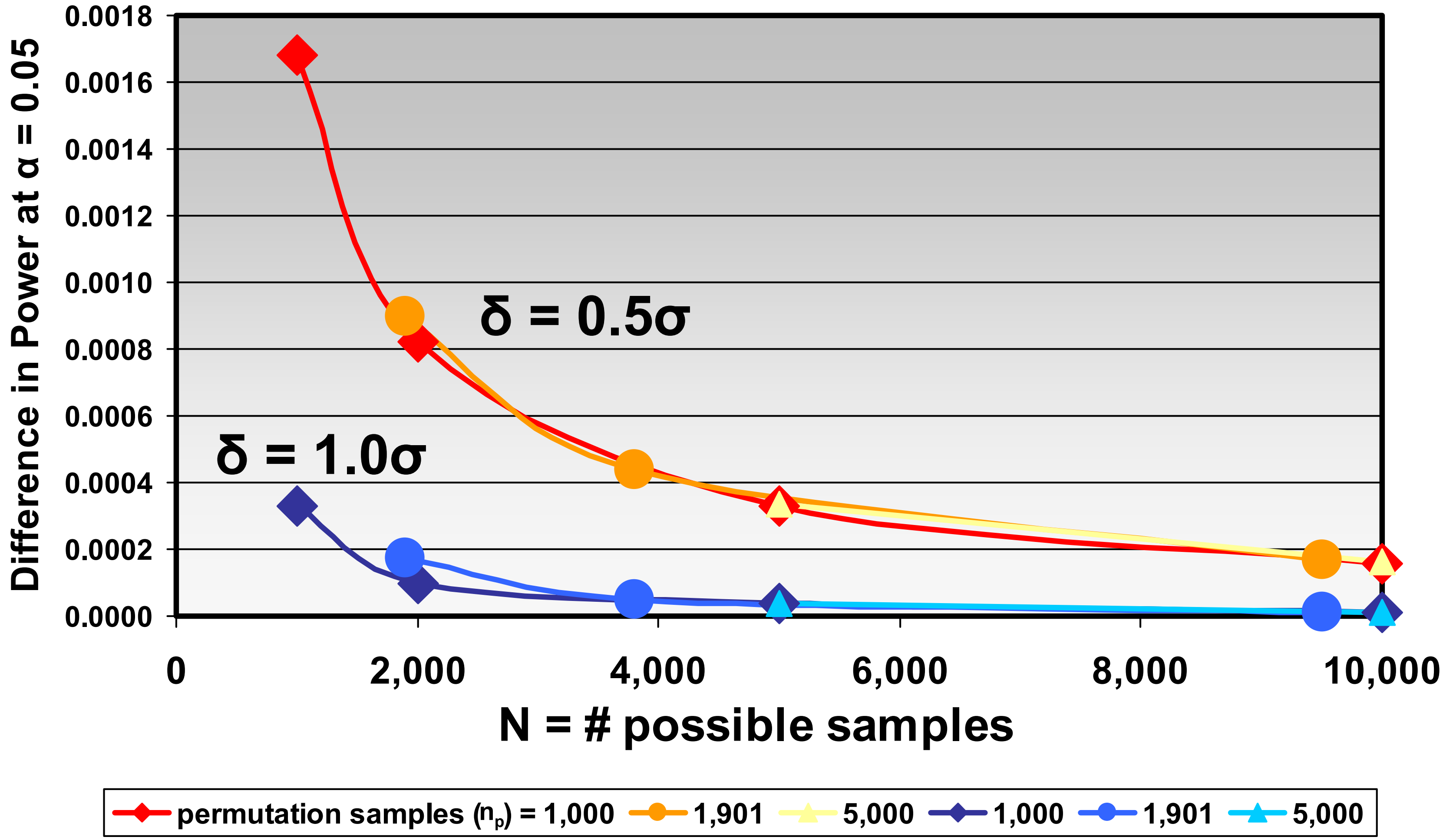
DataMineIt SM
J.D. Opdyke

6. How Much Power Gain…?

# Permutation Samples, n_p	# Possible Samples, N	$Power_{NR}$ $\delta=0.5\sigma$	$Power_{WR}$ $\delta=0.5\sigma$	$\Delta Power$ $\delta=0.5\sigma$	$Power_{NR}$ $\delta = \sigma$	$Power_{WR}$ $\delta = \sigma$	$\Delta Power$ $\delta = \sigma$
1,000	1,000	0.53093	0.52925	0.00168	0.96483	0.96450	0.00033
1,000	2,000	0.58483	0.58401	0.00082	0.98147	0.98137	0.00010
1,000	5,000	0.58434	0.58401	0.00033	0.98141	0.98137	0.00004
1,000	10,000	0.63373	0.63357	0.00016	0.99040	0.99039	0.00001
1,901	1,901	0.53283	0.53193	0.00090	0.96519	0.96502	0.00017
1,901	3,802	0.58708	0.58664	0.00044	0.98173	0.98168	0.00005
1,901	9,505	0.63628	0.63611	0.00017	0.99058	0.99056	0.00001
5,000	5,000	0.58864	0.58830	0.00034	0.98191	0.98187	0.00004
5,000	10,000	0.63787	0.63771	0.00016	0.99068	0.99067	0.00001

DataMineIt SM
J.D. Opdyke

GRAPH 6: Permutation Sampling With vs. Without Replacement: Approximate Difference in Power at $\alpha = 0.05$ by N by n_p by δ



7. … At What Cost?

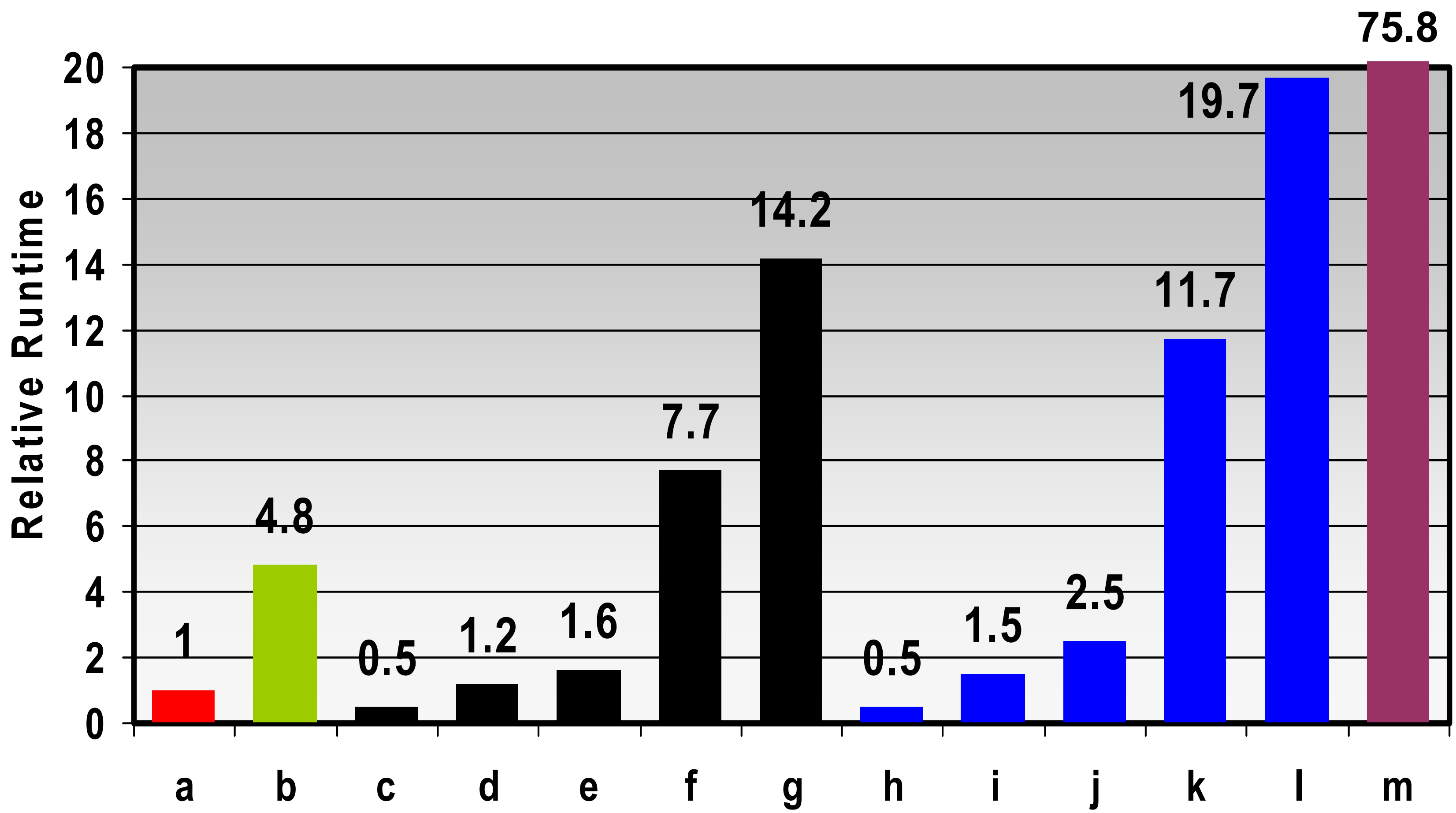
- Even for small N, n_p, and δ, approximate power gains from NR sampling are relatively small
- However, runtime cost also is small – typically less than 1%
- ∴ use NR permutation sampling unless cost of 1% of runtime is high and cost of Type II error is low

DataMineIt℠
J.D. Opdyke

8. Speed Premium for Multiple Comparisons

- For multiple study groups per control group (and/or multiple comparisons), code merges each group of original data to the PROC PLAN sampling output separately
- Separate merges avoids multiple outputting of control records for each corresponding study group (or each multiply-compared group)
- ⇒ Huge runtime savings on sorts/merges of large datasets (see Graph 7)
- PROC MULTTEST, PROC NPAR1WAY, and PROC TWOSAMPL® do not have this option (Graph 7)

GRAPH 7: Relative Start-to-Finish Runtime (T=1,901)



Graph 7 Key

DataMineIt[SM]
J.D. Opdyke

Column	Method
a	PROC PLAN with "oversampling"
b	Cytel's PROC TWOSAMPL®
c	PROC NPAR1WAY, study/control=1, $(n_1+n_2)<10^4$
d	PROC NPAR1WAY, study/control>1, $(n_1+n_2)<10^4$
e	PROC NPAR1WAY, study/control=1, $10^4<(n_1+n_2)<10^5$
f	PROC NPAR1WAY, study/control=1, $10^5<(n_1+n_2)<1.5*10^6$
g	PROC NPAR1WAY, study/control=1, $10^6<(n_1+n_2)<1.5*10^7$
h	PROC MULTTEST, study/control=1, $(n_1+n_2)<10^4$
i	PROC MULTTEST, study/control>1, $(n_1+n_2)<10^4$
j	PROC MULTTEST, study/control=1, $10^4<(n_1+n_2)<10^5$
k	PROC MULTTEST, study/control=1, $10^5<(n_1+n_2)<1.5*10^6$
l	PROC MULTTEST, study/control=1, $10^6<(n_1+n_2)<1.5*10^7$
m	Looping in SAS® (see Jackson affidavit)

DataMineIt SM
J.D. Opdyke

9. Increased Power for Permutation-Style P-Value Adj

- Take a single step resampling method adjustment
- No-replacement sampling $\Rightarrow$ increased power from:
  - ❖ a) smaller variance of each p_i^*

$$\Rightarrow \quad \min_{1 \le j \le k} p^*_{j_{NR}} \text{ is stochastically larger than } \min_{1 \le j \le k} p^*_{j_{WR}}$$

$$\Rightarrow \Pr\left(\min_{1 \le j \le k} p_{j_{NR}} \le p_i \mid H_0^C \right) < \Pr\left(\min_{1 \le j \le k} p_{j_{WR}} \le p_i \mid H_0^C \right)$$

$$\Rightarrow \quad \tilde{p}_{i_{NR_{a)}}} < \tilde{p}_{i_{WR_{a)}}} \Rightarrow \quad power_{NR_{a)}} > power_{WR_{a)}}$$

DataMineIt SM
J.D. Opdyke

9. Increased Power for Permutation-Style P-Value Adj

❖ **b) previous Monte Carlo error p-value adjustment**

$$\Rightarrow \quad p_{i_{NR}} < p_{i_{WR}}$$

$$\Rightarrow \quad \Pr\left(\min_{1 \le j \le k} p_j \le p_{i_{NR}} \mid H_0^C \right) < \Pr\left(\min_{1 \le j \le k} p_j \le p_{i_{WR}} \mid H_0^C \right)$$

$$\Rightarrow \quad \tilde{p}_{i_{NR_{b)}}} < \tilde{p}_{i_{WR_{b)}}}$$

$$\Rightarrow \quad power_{NR_{b)}} > power_{WR_{b)}}$$

DataMineIt[SM]
J.D. Opdyke

9. Increased Power for Permutation-Style P-Value Adj

- ∴ use NR sampling for both permutation tests and permutation-style p-value adjustments to maximize power gain

$$\Pr\left(\min_{1\le j\le k} p_{j_{NR}} \le p_{i_{NR}} \mid H_0^C\right) < \Pr\left(\min_{1\le j\le k} p_{j_{WR}} \le p_{i_{WR}} \mid H_0^C\right)$$

$$\Rightarrow \quad \tilde{p}_{i_{NR}} < \tilde{p}_{i_{WR}}$$

$$\Rightarrow \quad power_{NR} > power_{WR}$$

- Same rationale applies to stepwise adjustments

Contact:

- J.D. Opdyke, President

DataMineIt SM

When You Need to Know SM

jdopdyke@datamineit.com

www.datamineit.com

203-249-4837