



United States Steel Corporation

# Auto-Swag

Application of Random Selection without Replacement.

Golden Horseshoe (GHSUG) SAS Users Group

April 25 2008

*Jim Burkhardt, P.Eng.*

Process Technology Engineer

Best Practices

U.S. Steel Canada

© 2008 United States Steel Corporation





# Automate the selection of GHSUG SAS SWAG winners

## What is SWAG?

### **Wikipedia Definition (abridged)**

**Promotional items** or **promotional products** used in marketing and communication programs.

Usually imprinted with a company's name, logo or slogan, and given away at trade shows, conferences, and as part of guerilla marketing campaigns.



# Automate the selection of SAS SWAG winners

Why Bother?

# The challenge!



# Automate the selection of SAS SWAG winners

It ain't Gucci, but !!

## SAS Swag Check-List

- Camp Chair
- Travel Bag
- Toque
- Backpack
- Thermos
- Mugs (various)
- Golf Shirt
- T-Shirt
- Lunch Bag
- Paper Clip dispenser
- Yoyo
- Pens (various)
- Memory Sticks
- CD Case
- LED Key chain
- Magnetic Clip
- Sunglass Visor Clip





# Today's Presentation

## Outline

- Create a Test Data Set
- Random Selection
- Develop the Macro
- Verify the Macro
- Apply the Macro → Who gets the swag?



# Requirements

## What do we know?

- “Selection without replacement”
- The number of prizes (draws)
- The number of people entered in the draw
- An input file containing one observation per entrant.



# Development

## Create a test dataset

One  
Observation  
per valid  
entry

```
data test_input_list,  
  length lastname $30 firstname $30.;  
  input lastname firstname;  
  cards;  
      Hong Barry  
      McCulloch Craig  
      Burkhardt Jim  
      .... and more ....  
      ;  
run;
```



# Development

## Define the macro

Macro Name  
Input Dataset  
No of Draws

```
%macro GHSUG_Auto_Swag(  
    input_datset=,  
    no_of_draws=      );  
    ..... Macro statements .....  
%mend;
```



# Development

Add  
Unique  
Index to  
input file

## Assign Unique Index

```
data attendance_list;  
    ticket_no+1;  
    set &input_dataset end=lastrec;  
    if lastrec then do;  
        call symput('Total_no_tickets', ticket_no);  
    end;  
run;
```



# Development

## Assign Unique Index

Add  
Unique  
Index to  
input file

```
data attendance_list;  
  ticket_no+1;  
  set &input_dataset end=lastrec;  
  if lastrec then do;  
    call symput('Total_no_tickets', ticket_no);  
  end;  
run;
```



```
retain ticket_no 0;
```

```
ticket_no=ticket_no+1;
```



# Development

## Determine size of selection 'pool'

Store Size in  
Macro Variable

```
data attendance_list;  
  ticket_no+1;  
  set test_input_list end=lastrec;  
  if lastrec then do;  
    call symput('Total_no_tickets', ticket_no);  
  end;  
run;
```



## Development – Random Selection

### Random Uniform Distribution Function RANUNI()

- Constant Probability Density
- Continuous function.
- Output scaled  $0 \rightarrow 1$
- Generated number sequence controlled by 'seed' value
- Selection with Replacement

Function  
Properties



## Development – Random Selection

### More About Seed Values (too much information)

**Positive seeds** – creates repetitive data stream:

- Important in process simulation where complex logic is employed.

**Negative seeds** – creates independent random data stream:

- Important where multiple similar but independent distributions are required.

**Zero seeds** – creates unrepeatable data stream:

- Most often used.

Selection using  
Random Uniform  
Distribution

'Ranuni()'



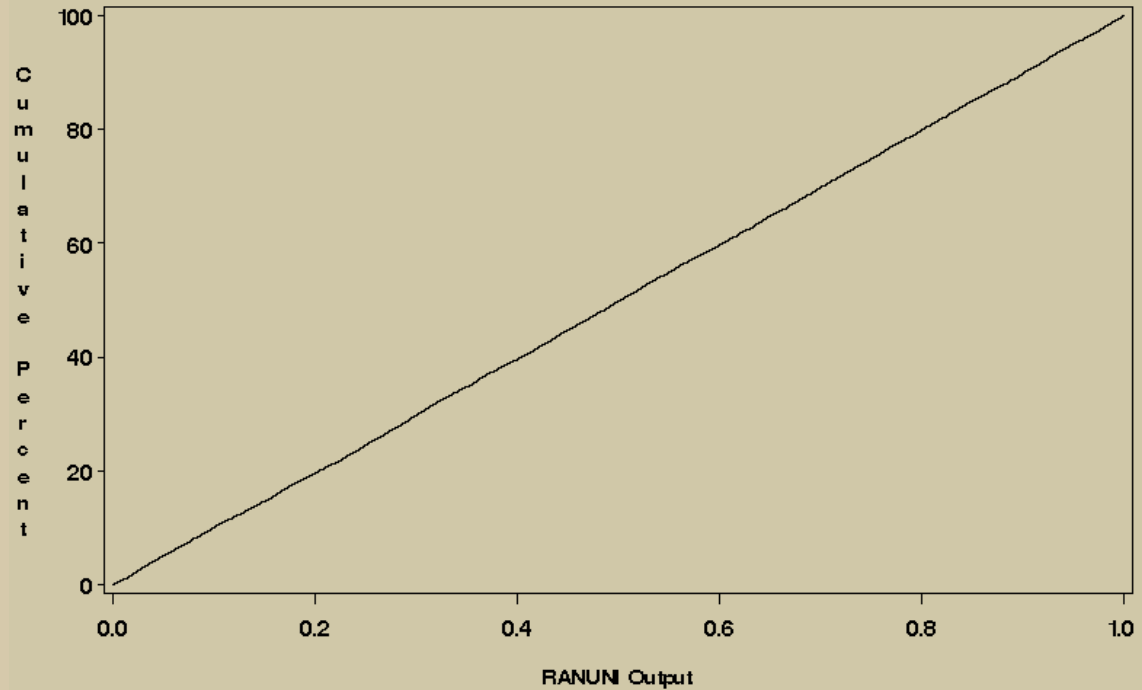
# Development – Random Selection

## Function RANUNI (10000 Draws)

Random  
Uniform  
Distribution  
Scaled 0-1  
Must be  
transformed

### Uniform Distribution – Function RANUNI

Cumulative Distribution Function  
Sample Size – 10000





# Development – Random Selection

## Function RANUNI – Scaling

Scaled to represent the pool of tickets dispensed

$$\text{RANUNI}(0) \times \text{Total\_No\_Tickets}$$

Use Seed=0 to trigger random data stream

Transform  
Random  
Uniform  
Distribution



## Development – Random Selection

### Function RANUNI – Discrete Function

Transform  
Random  
Uniform  
Distribution

Change to a Discrete function

**`int( RANUNI(0) x Total_No_Tickets ) + 1`**



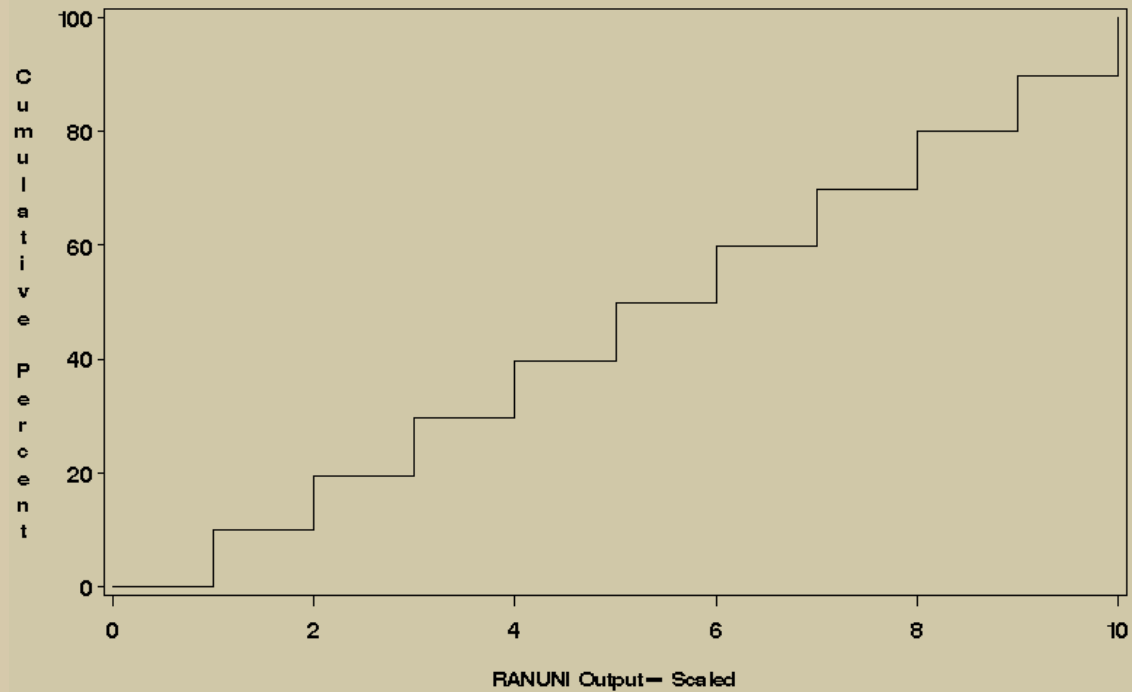
# Development Verification

## Discrete and Scaled Function RANUNI (10,000 Draws)

Transformed  
Random  
Uniform  
Distribution  
Example using  
10000 draws

### Transformed Uniform Distribution — Function RANUNI

Cumulative Distribution Function  
Sample Size — 10000 / 10 Discrete Values





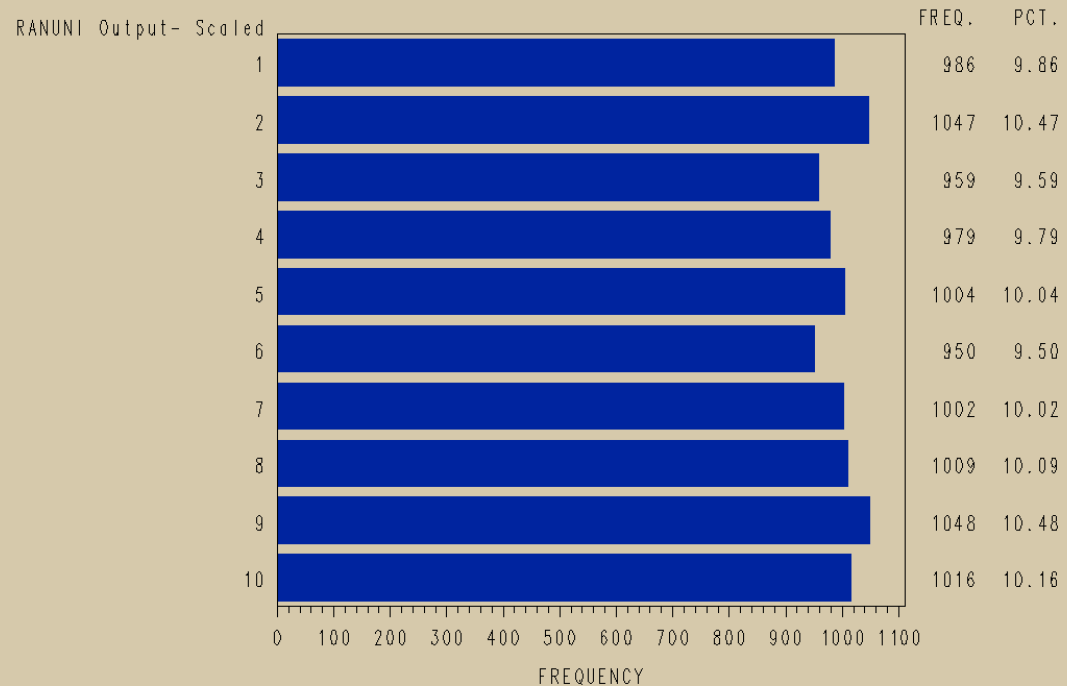
# Development Verification

Transformed  
Random  
Uniform  
Distribution

Sample Size of  
10000 – begins  
approaches the  
perfect uniform  
distribution

## Transformed and Scaled

Uniform Distribution — Transformed RANUNI (0–10)  
Sample Size - 10000





# Development – Draw the Winning Numbers

Define Status  
Array to enable  
Selection  
without  
replacement

## Main Drawing Logic

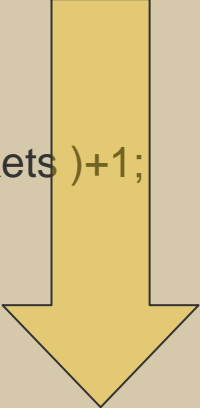
```
data draw;  
    array status{ &total_no_tickets } $1. _temporary_;  
  
do while ( no_drawn < &no_of_draws );  
  
    Ticket_no=int( ranuni( 0 ) * &total_no_tickets )+1;  
  
    if status{ ticket_no } = " then do;  
        status{ ticket_no }='Y';  
        no_drawn+1;  
        output;  
        end;  
  
    end;  
  
    keep no_drawn ticket_no;  
run;
```



# Development – Draw the Winning Numbers

## Main Drawing Logic

```
data draw;  
    array status{ &total_no_tickets } $1. _temporary_;  
  
do while ( no_drawn < &no_of_draws );  
  
    Ticket_no=int( ranuni( 0 ) * &total_no_tickets )+1;  
  
    if status{ ticket_no } = " then do;  
        status{ ticket_no }='Y';  
        no_drawn+1;  
        output;  
        end;  
  
    end;  
  
    keep no_drawn ticket_no;  
run;
```



```
retain varname1-varname10 $1.;  
array status{n} $1. varname1-varname10;  
drop varname1-varname10;
```

Define Status  
Array to enable  
Selection  
without  
replacement



# Development – Draw the Winning Numbers

## Main Drawing Logic

```
data draw;  
  array status{ &total_no_tickets } $1. _temporary_;  
  
  do while ( no_drawn < &no_of_draws );  
  
    Ticket_no=int( ranuni( 0 ) * &total_no_tickets )+1;  
  
    if status{ ticket_no } = " then do;  
      status{ ticket_no }='Y';  
      no_drawn+1;  
      output;  
      end;  
  
  end;  
  
  keep no_drawn ticket_no;  
run;
```

Use Do While Logic to determine when enough samples have been drawn.



# Development – Draw the Winning Numbers

## Main Drawing Logic

```
data draw;
  array status{ &total_no_tickets } $1. _temporary_;

  do while ( no_drawn < &no_of_draws );

    Ticket_no=int( ranuni( 0 ) * &total_no_tickets )+1;

    if status{ ticket_no } = " " then do;
      status{ ticket_no }='Y';
      no_drawn+1;
      output;
    end;

  end;

  keep no_drawn ticket_no;
run;
```

Generate  
Random number



# Development – Draw the Winning Numbers

## Main Drawing Logic

```
data draw;  
  array status{ &total_no_tickets } $1. _temporary_;  
  
  do while ( no_drawn < &no_of_draws );  
  
    Ticket_no=int( ranuni( 0 ) * &total_no_tickets )+1;  
  
    if status{ ticket_no } = ' ' then do;  
      status{ ticket_no } = 'Y';  
      no_drawn + 1;  
      output;  
      end;  
  
    end;  
  
    keep no_drawn ticket_no;  
run;
```

Check Status  
array element for  
prior selection

Set array  
element flag

Integrate the  
no\_drawn

Output the  
successful draw



# Development – Identify the Winners

Create a results table using the draw information

Publish the Winners

## Associate Ticket Numbers with Drawn Winners

```
proc SQL noprint;

    create table winners as

        select a.no_drawn,
               b.*
        from draw as a
             inner join
             attendance_list as b
             on a.ticket_no=b.ticket_no
        order by a.no_drawn;

quit;

title1 'The Prize Winners Are:';
proc print data=winners noobs;
run;
```



# Evaluation and Verification

## Does this technique produce an unbiased Result?

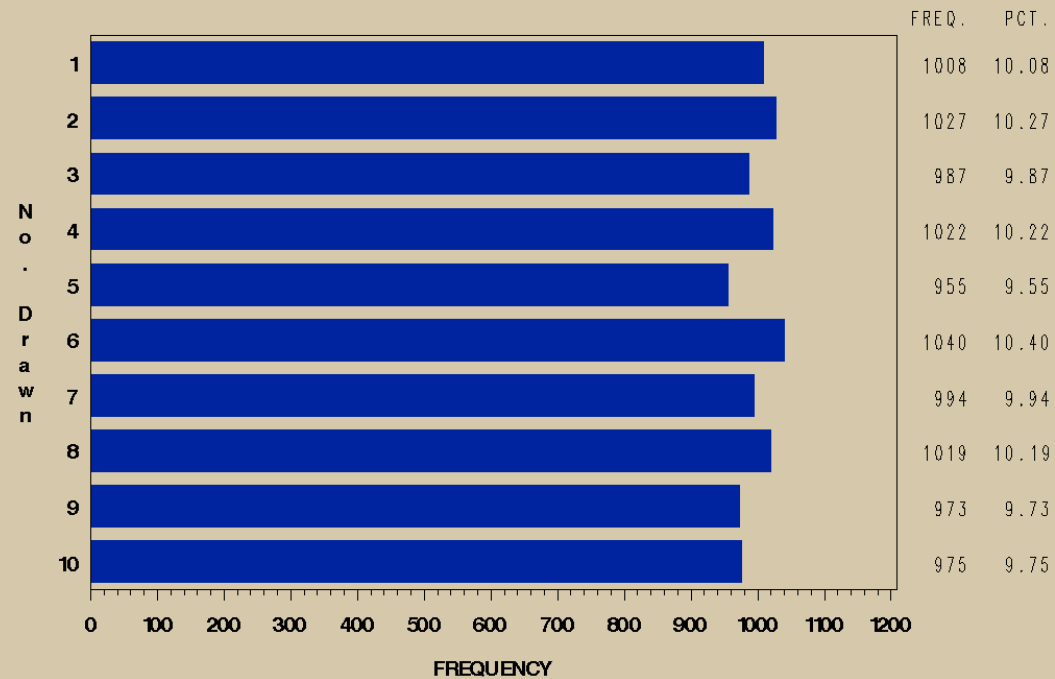
Remember the Ontario Lottery Corporation!

10000 Macro Runs

I won 10.08% of the time or lost 89.92% of the time!

### Test for Bias — Winner Burkhardt

10000 Independent Runs





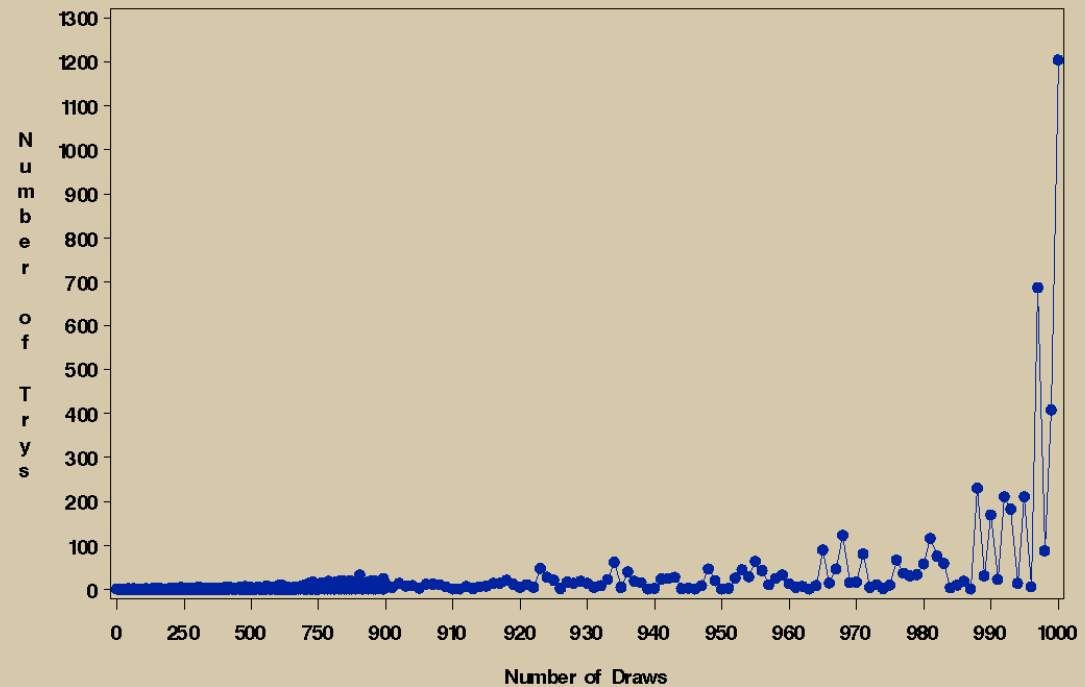
# Evaluation and Verification

Is the macro efficient with large size datasets?

Efficiency drops as the number drawn approaches the 100% of the population size.

## Number of Trys required for a Successful Draw

1000 Draws from a Population of 1000





# Evaluation and Verification

Time to draw 100% of the population (worst case).

Population Size	Time to Draw
10	0.01 sec
100	0.01 sec
1000	0.03 sec
10000	0.04 sec
100000	0.10 sec
1000000	9.85 sec

Standard issue PC

Efficiency drops for very large datasets that require 100% drawn



## The Final Step

Questions?

Who will win the  
SWAG Today!



## References and thanks

- SAS Online Doc V9
  - Macros
  - Data step
- Craig and Barry for supplying the Swag.

Email: [jim.burkhardt@hamiltonsteel.ca](mailto:jim.burkhardt@hamiltonsteel.ca)