Generating realistic synthetic test data using SAS Random Functions

Tom Kari, Tom Kari Consulting
TASS, December 2017

Ottawa Area SAS Users Society

www.OASUS.ca

Generating realistic synthetic test data using SAS Random Functions

- Why would we want to do this?
- American Community Survey (California) 2016 as an example
- SAS Random Number functions
- Using random numbers to generate test data
- Please feel free to ask questions during the presentation!

Testing activities that may require large volumes of data

- Volume testing (can the system process the required volumes of data expected in production?)

Testing activities that may require large volumes of data

- Volume testing (can the system process the required volumes of data expected in production?)
- Performance testing (is the performance of the system at high volumes acceptable?)

Testing activities that may require large volumes of data

- Volume testing (can the system process the required volumes of data expected in production?)
- Performance testing (is the performance of the system at high volumes acceptable?)
- Usability, for analytical systems (when presented with realistic data, can the users effectively use the analytical tools?)

Approaches for obtaining large volumes of data

Replicate the unit and integration test files

Issues:

- Typically hand-crafted to ensure correct system behaviour
- Usually contain small proportions of the domains of the variables
- Frequently concentrated on boundary conditions
- Replication will tend to produce highly skewed datasets
- Won't permit effective assessment of the system

Approaches for obtaining large volumes of data

Use production data from previous cycles, possibly with anonymization

Issues:

- If it's a new system, there isn't any production data
- Current production data may be a poor model for new system behaviour
- Confidentiality: Challenges using the data in documents, training
- Confidentiality: Can't share data with partners, hardware/software vendors, subcontractors

Ottawa Area SAS Users Society

www.OASUS.ca

Approaches for obtaining large volumes of data

	Marst						Total (ALL)
	Married(1)	Living common law(2)	Never Married(3)	Separated(4)	Divorced(5)	Widowed(6)	
	N	N	N	N	N	N	N
Income							
Zero to 5000	7950	2 08	5073	419	1025	941	17416
Greater than 5000, Less than or equal to 10000	608	1	3419	29	747	35	12159
Greater than 10000, Less than or equal to 20000	1 673	2	6380	5	1374	7	22913
Greater than 20000, Less than or equal to 30000	1 073	2	6103	2	1329	12	2 98
Greater than 30000, Less than or equal to 40000	29	2134	5130	86	1074	10	6
Greater than 40000, Less than or equal to 50000	040	1775	157	398	8	85	15 02
Greater than 50000, Less than or equal to 60000		1234	3 04	284		34	11 5
Greater than 60000, Less than or equal to 70000		994	23	214			8 9
Greater than 70000, Less than or equal to 80000	3	751	1899	190	9		02
Greater than 80000, Less than or equal to 90000	2 00	5	1355	120		25	868
Greater than 90000, Less than or equal to 100000	673		1078	109	2	212	9
Greater than 100000, Less than or equal to 120000	31		1471	117	306	277	
Greater than 120000, Less than or equal to 140000	3	3	80	87	168	171	
Greater than 140000, Less than or equal to 160000	7	2	5 4		123	93	6
Greater than 160000, Less than or equal to 180000	97	111	7	23	49	64	1 21
Greater than 180000, Less than or equal to 200000	392		225	20	39	4	33
Greater than 200000, Less than or equal to 220000	262	53	174	9	29	31	58
Greater than 220000, Less than or equal to 240000	106	2	67	7	17	18	241
Greater than 240000, Less than or equal to 260000	131	3	91	13	21	13	292
Greater than 260000, Less than or equal to 280000	42	11	23	5	1	4	86
Total (ALL)	72833	17622	43771	4047	9351	8881	156505

Ottawa Area SAS Users Society

www.OASUS.ca

Approaches for obtaining large volumes of data

Don’t do volume, performance, usability tests

Approaches for obtaining large volumes of data

Don’t do volume, performance, usability tests

Issues:

–RISK!

Ottawa Area SAS Users Society

www.OASUS.ca

Approaches for obtaining large volumes of data

Do your testing on your production data store

Ottawa Area SAS Users Society

www.OASUS.ca

Approaches for obtaining large volumes of data

Do your testing on your production data store

What could possibly go wrong?

Ottawa Area SAS Users Society

www.OASUS.ca

Approaches for obtaining large volumes of data

Do your testing on your production data store

What could possibly go wrong?

Ontario woman discovers she can't trade in her van because 'Fred Flintstone' put a lien on it

NP ALLISON JONES, THE CANADIAN PRESS | May 11, 2017 12:00 AM ET

More from The Canadian Press

Approaches for obtaining large volumes of data

Generate synthetic data that more or less models your expected production data

Issues:

- It is necessary to have some conception of what the production data characteristics will be
- It requires time and funding

Example: 2016 American Community Survey (California)

- Real-world example
- Large volume (39 million records)
- Diverse characteristics of variables
- I'm very familiar with this kind of data
- The audience has passing knowledge of the Canadian Census long form, which is similar

Ottawa Area SAS Users Society

www.OASUS.ca

2016 American Community Survey

Gender

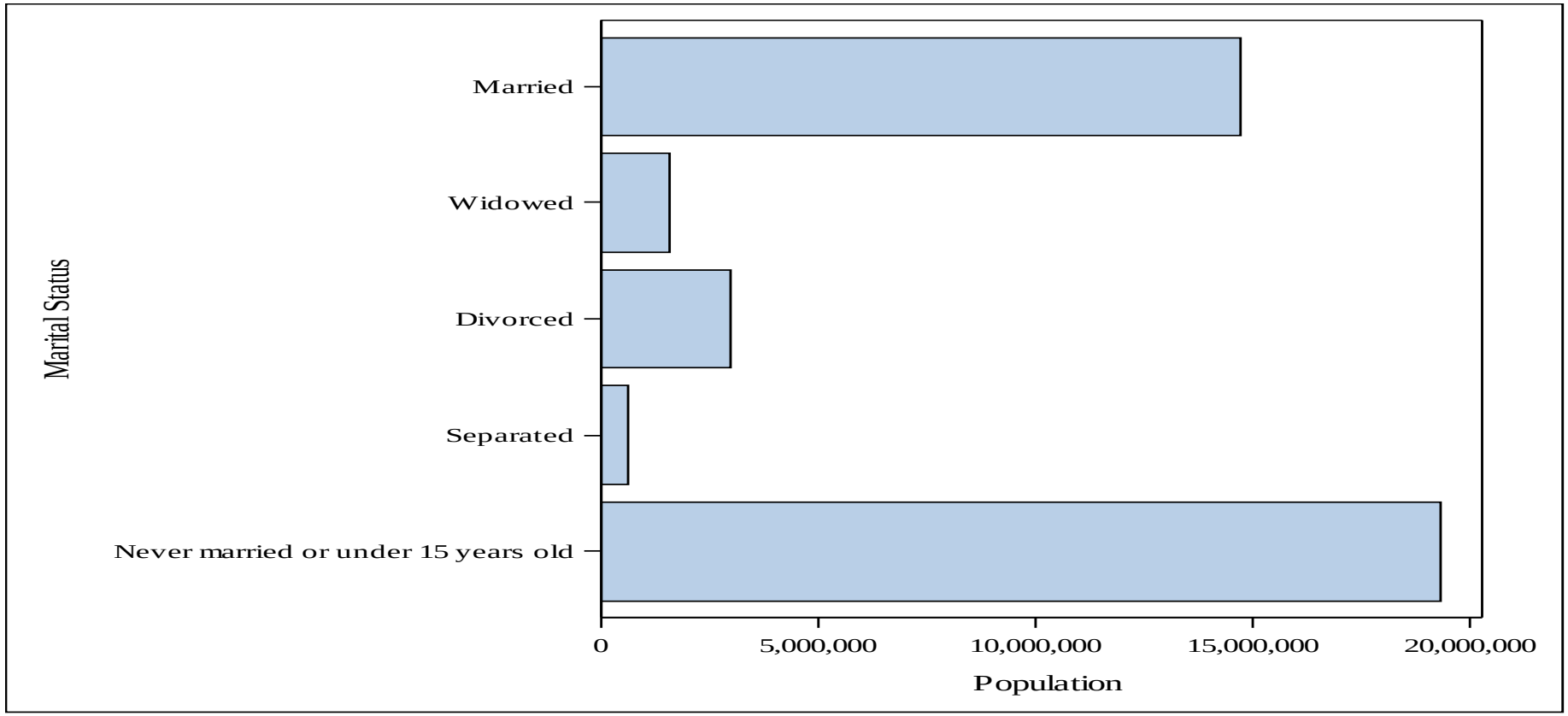
	Count
Female	19,752,605
Male	19,497,412
	39,250,017

Ottawa Area SAS Users Society

www.OASUS.ca

2016 American Community Survey

Marital Status

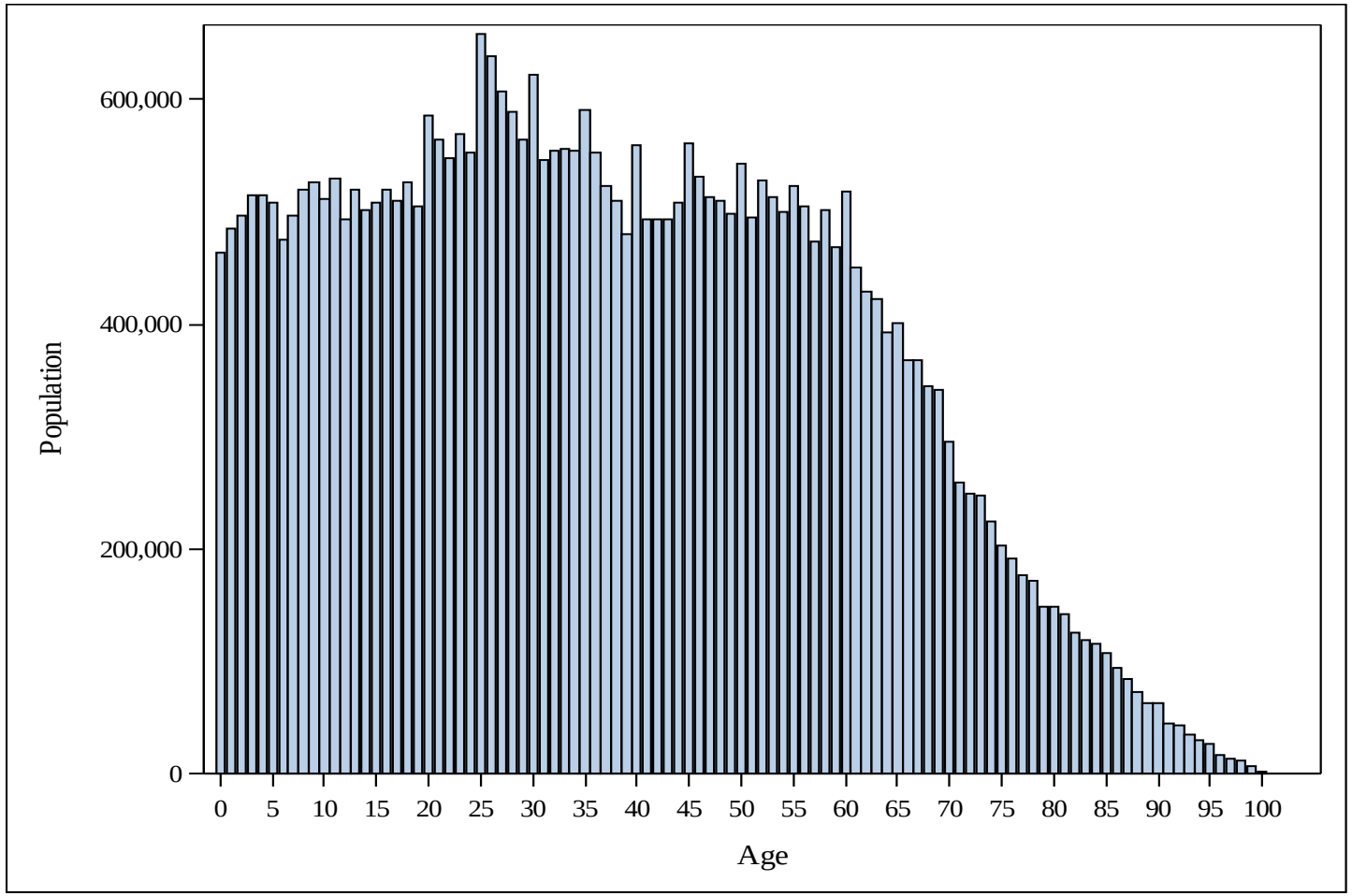


Ottawa Area SAS Users Society

www.OASUS.ca

2016 American Community Survey

Age

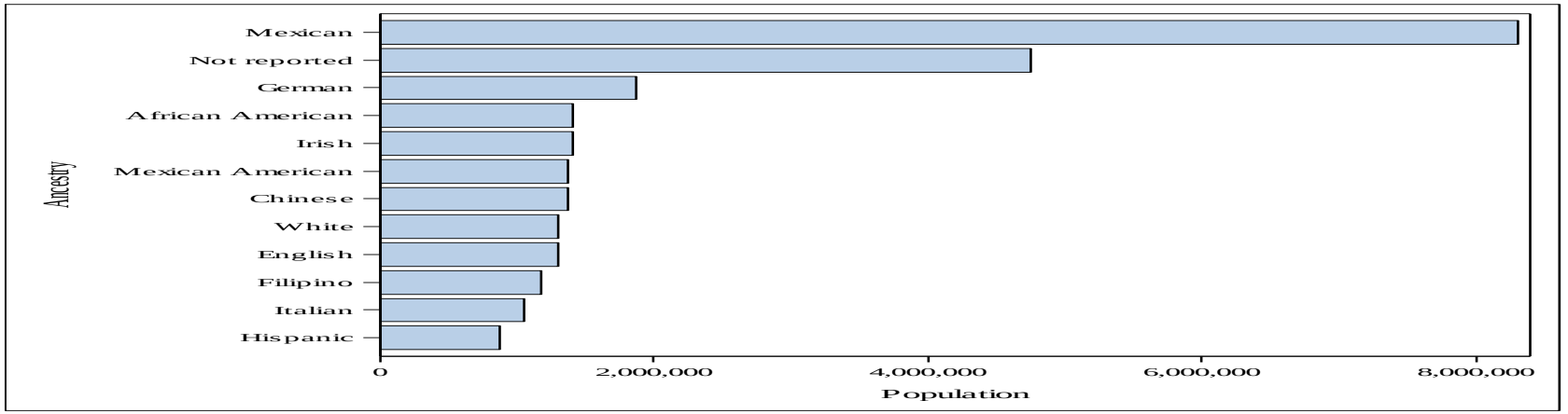


Ottawa Area SAS Users Society

www.OASUS.ca

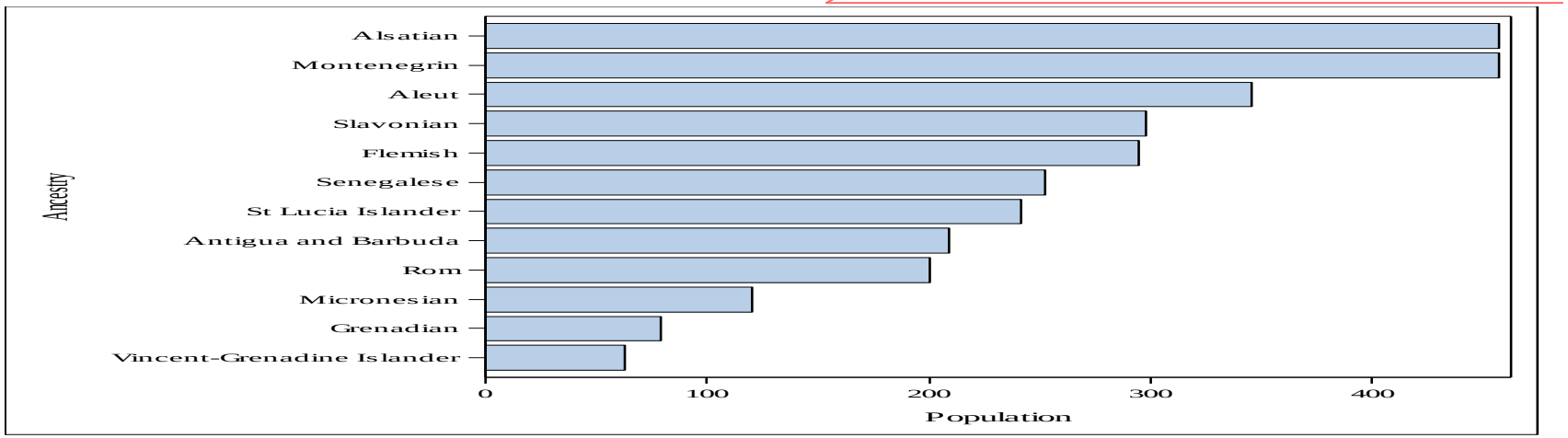
2016 American Community Survey

Ancestry



206 rows omitted

Note change of scale

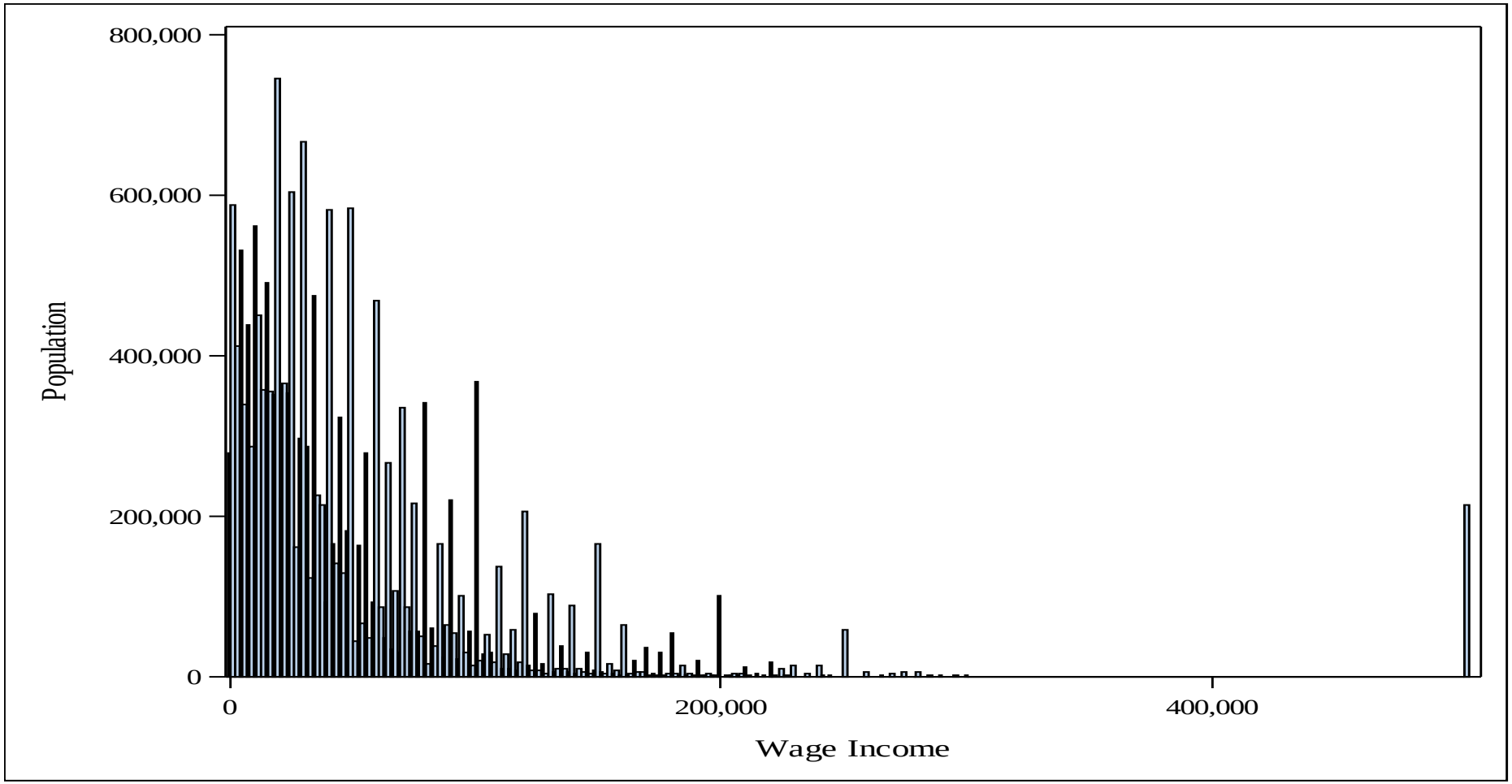


Ottawa Area SAS Users Society

www.OASUS.ca

2016 American Community Survey

Wage Income



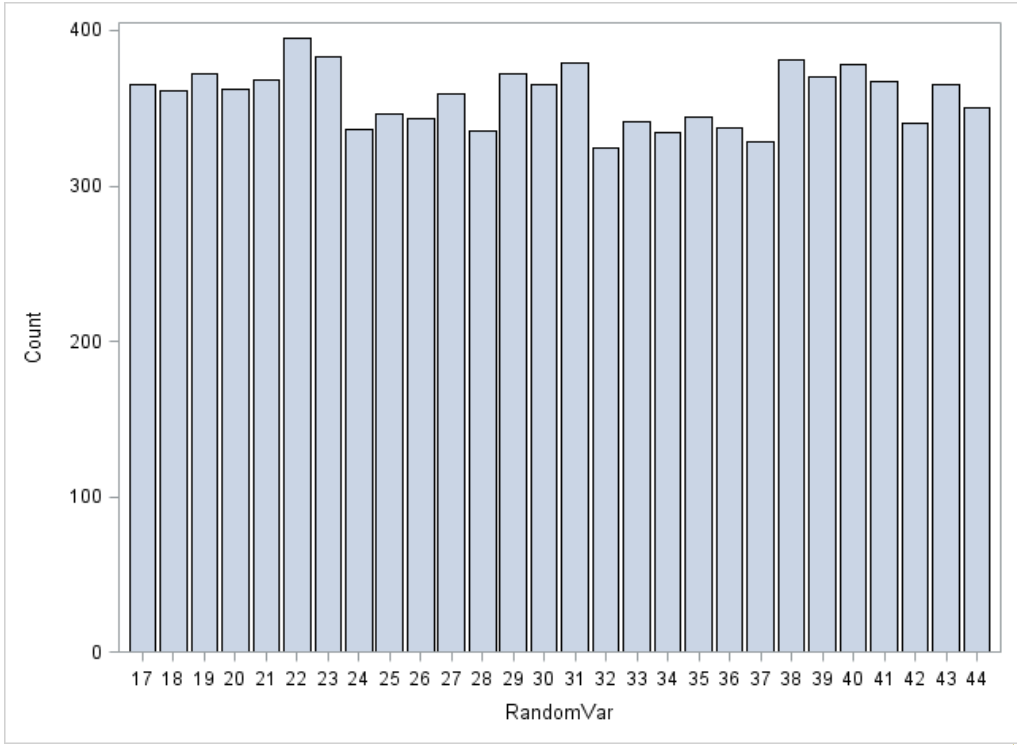
SAS Random Number functions

- The RAND function generates random numbers from a specified distributions
- Available distributions are: Bernoulli, Beta, Binomial, Cauchy, Chi-Square, Erlang, Exponential, F, Gamma, Geometric, Hypergeometric, Lognormal, Negative Binomial, Normal, Poisson, T, Triangular, Uniform, and Weibull
- Also, Tabled option, not a distribution
- The STREAMINIT routine allows you to specify a seed, so the stream of numbers is repeatable
- It is better to use RAND, not the older random number routines (RANUNI etc.)

RAND Uniform

- The simplest option, no parameters, generates a random number uniformly distributed > 0, < 1
- Can be manipulated to generate numbers in a desired range, real or integer

```
%let Repetitions = 10000; /* > 0 */
%let Lower = 17; /* >= 0 */
%let Upper = 44; /* >= Lower */

data RandomNumbers(drop=_:);
  do _i = 1 to &Repetitions;
    RandomVar = floor(&Lower + (rand('uniform') *
                  (&Upper - &Lower + 1)));
    output;
  end;
run;
```



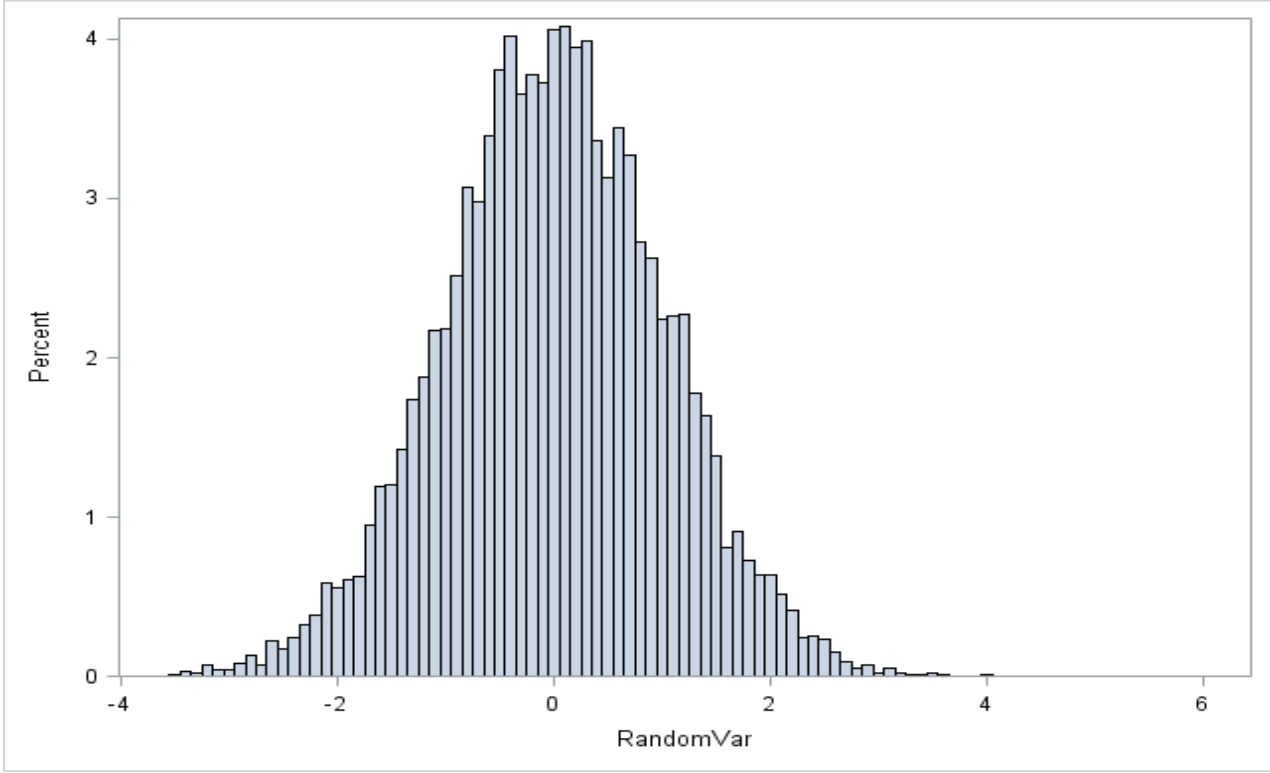
Ottawa Area SAS Users Society

www.OASUS.ca

RAND Normal

- Returns a random number normally distributed as specified by the two parameters: mean(0), and standard Deviation(1)
- Models many natural and statistical phenomena

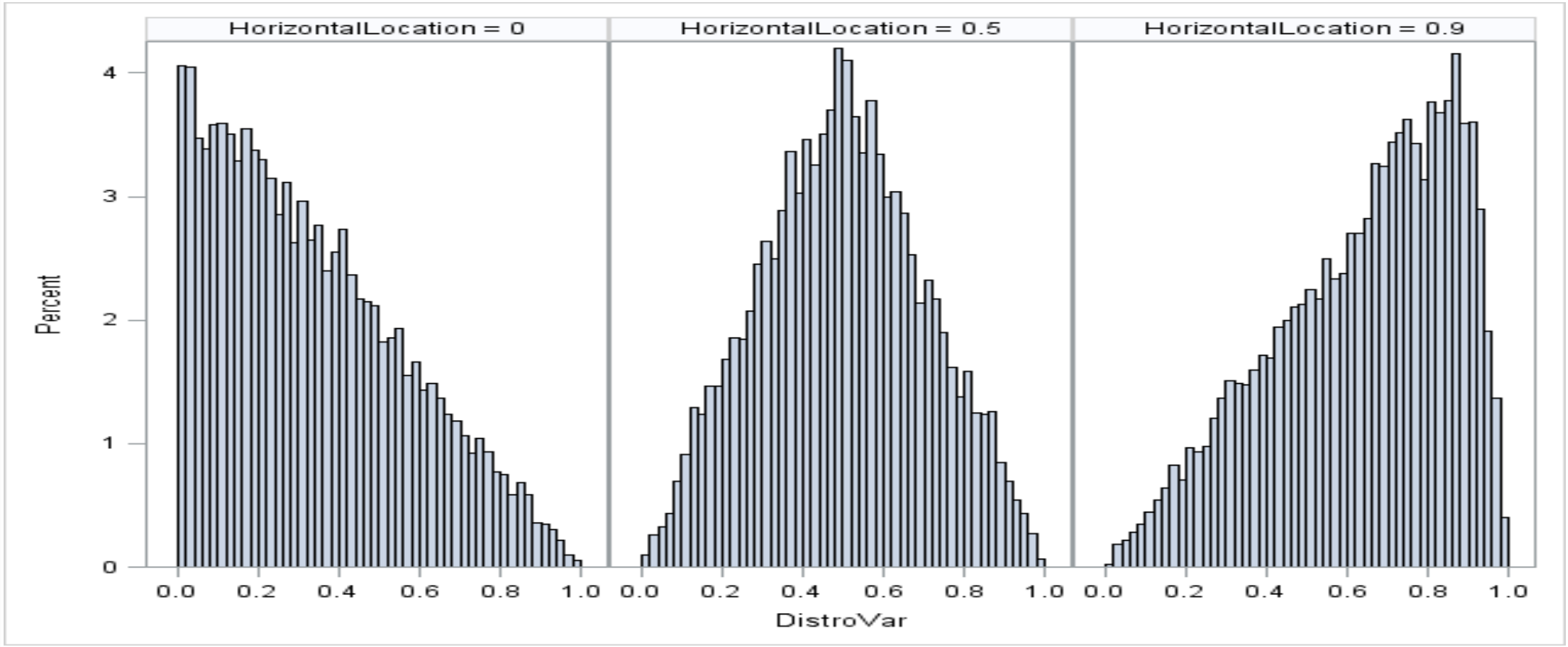
```
data RandomNormal(drop=_:);
  do _i = 1 to 10000;
    RandomVar = rand('normal');
  /* RandomVar = rand('normal', 12, 4); */
    output;
  end;
run;
```



RAND Triangular

- Returns a random number distributed in a triangle (0,1)

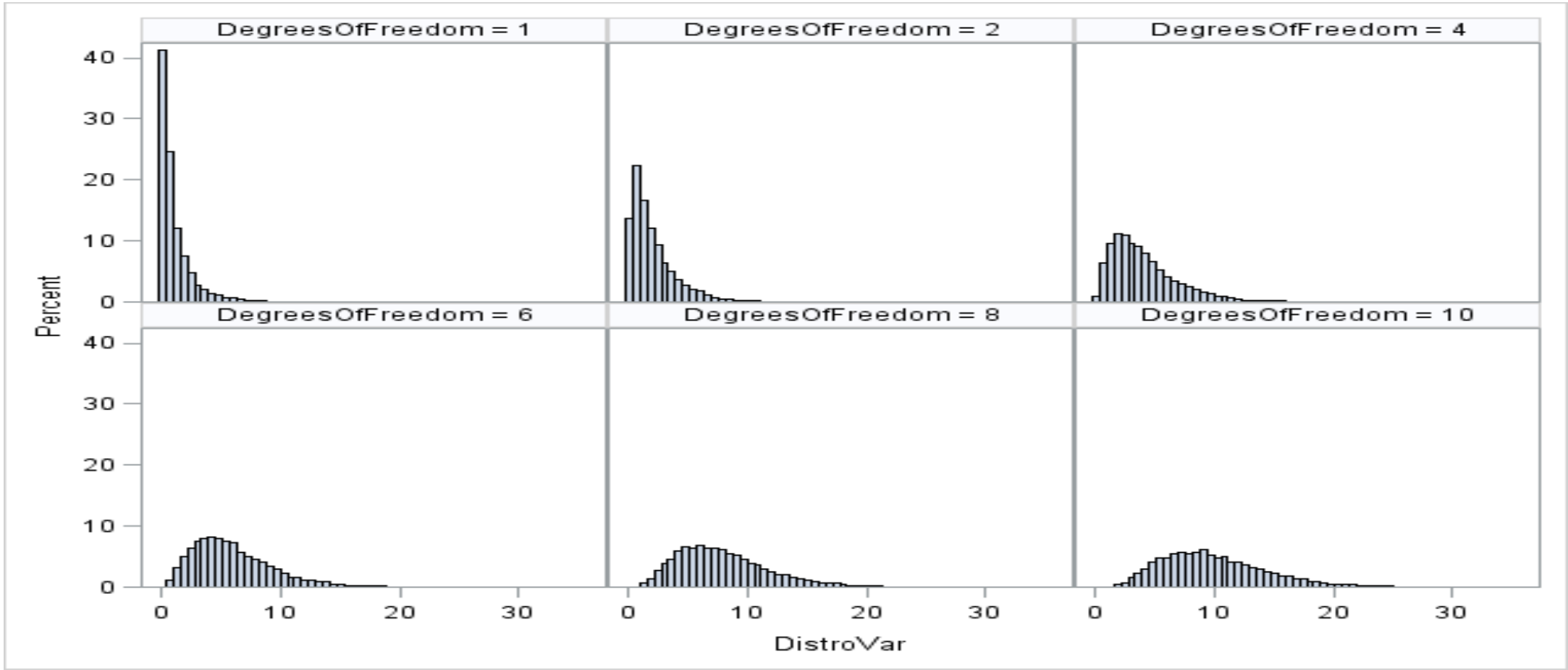
RandomVar = rand('triangle', .5);



RAND Chi-Square

- Returns a random number distributed in the chi-square distribution

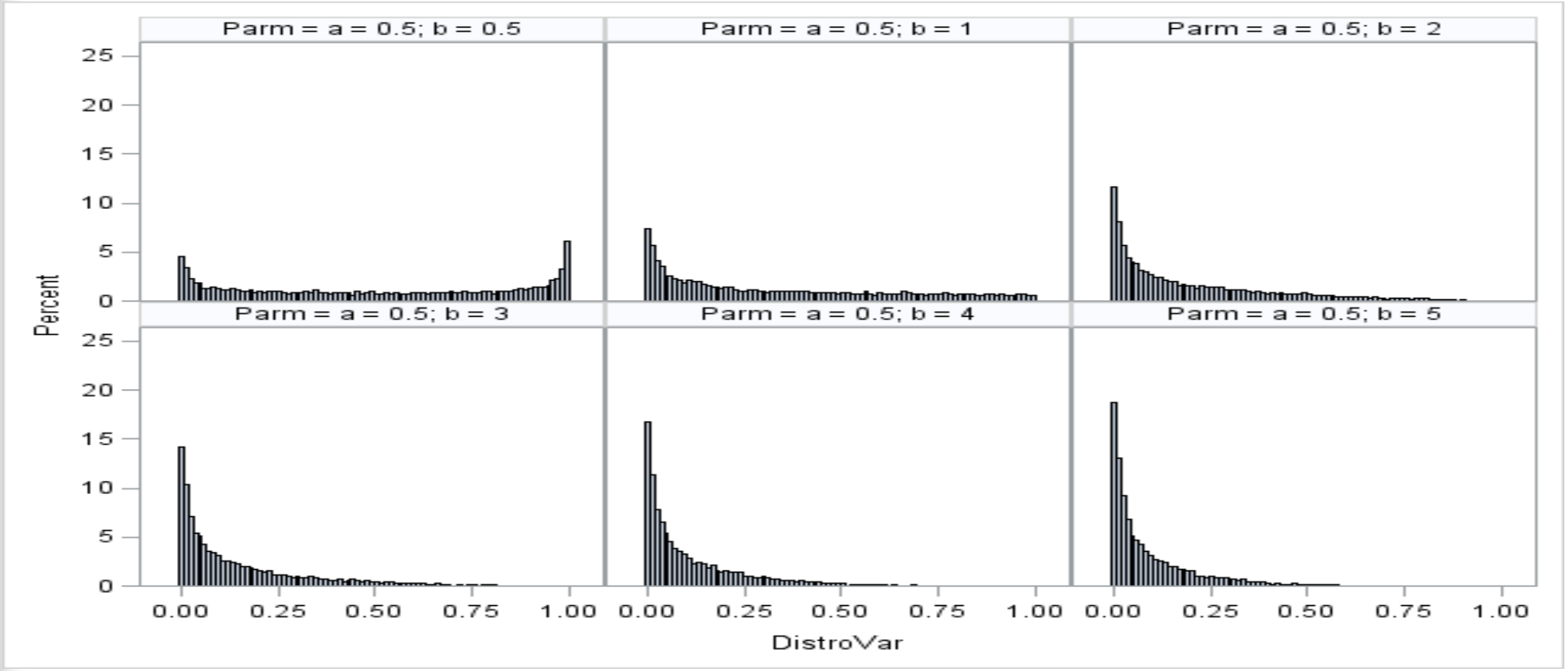
RandomVar = rand('chisquare', 6);



RAND Beta

- Returns a random number distributed in the beta distribution

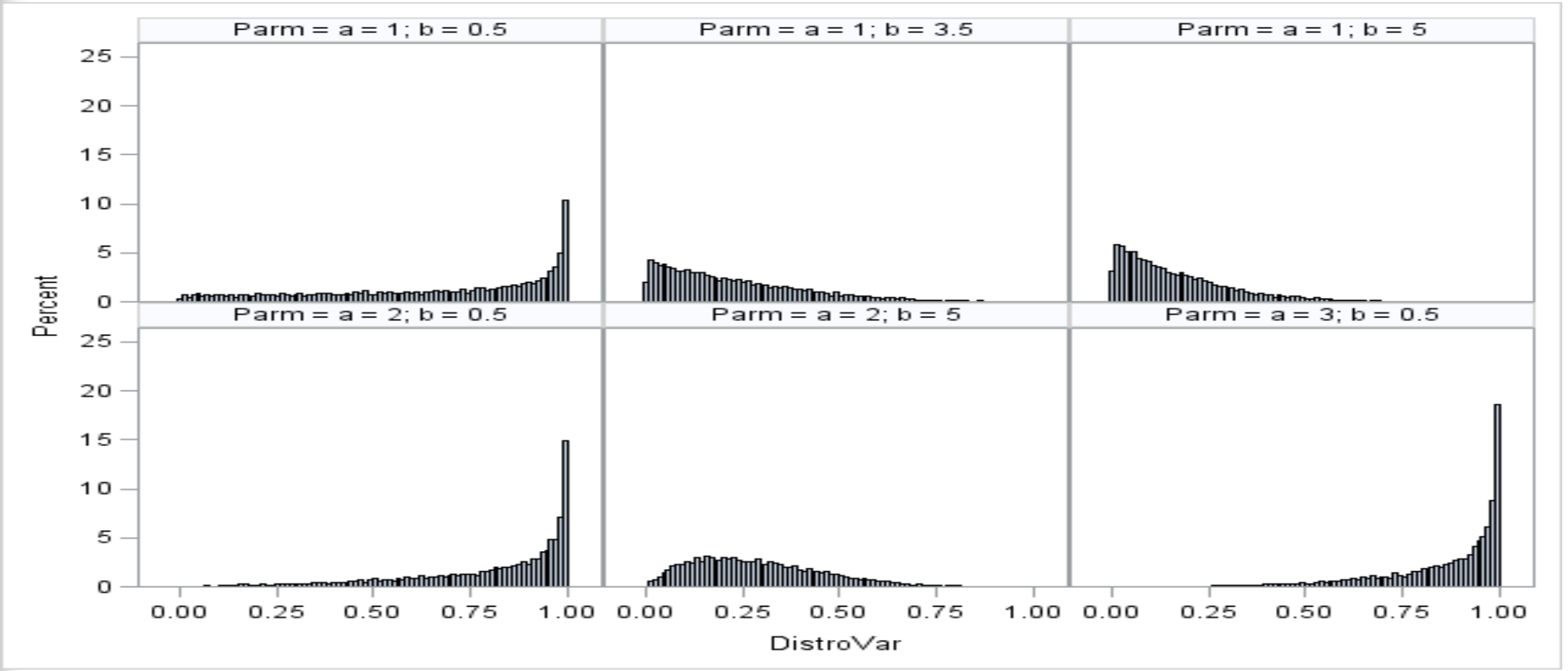
```
RandomVar = rand('beta', 5, .5);
```



Ottawa Area SAS Users Society

www.OASUS.ca

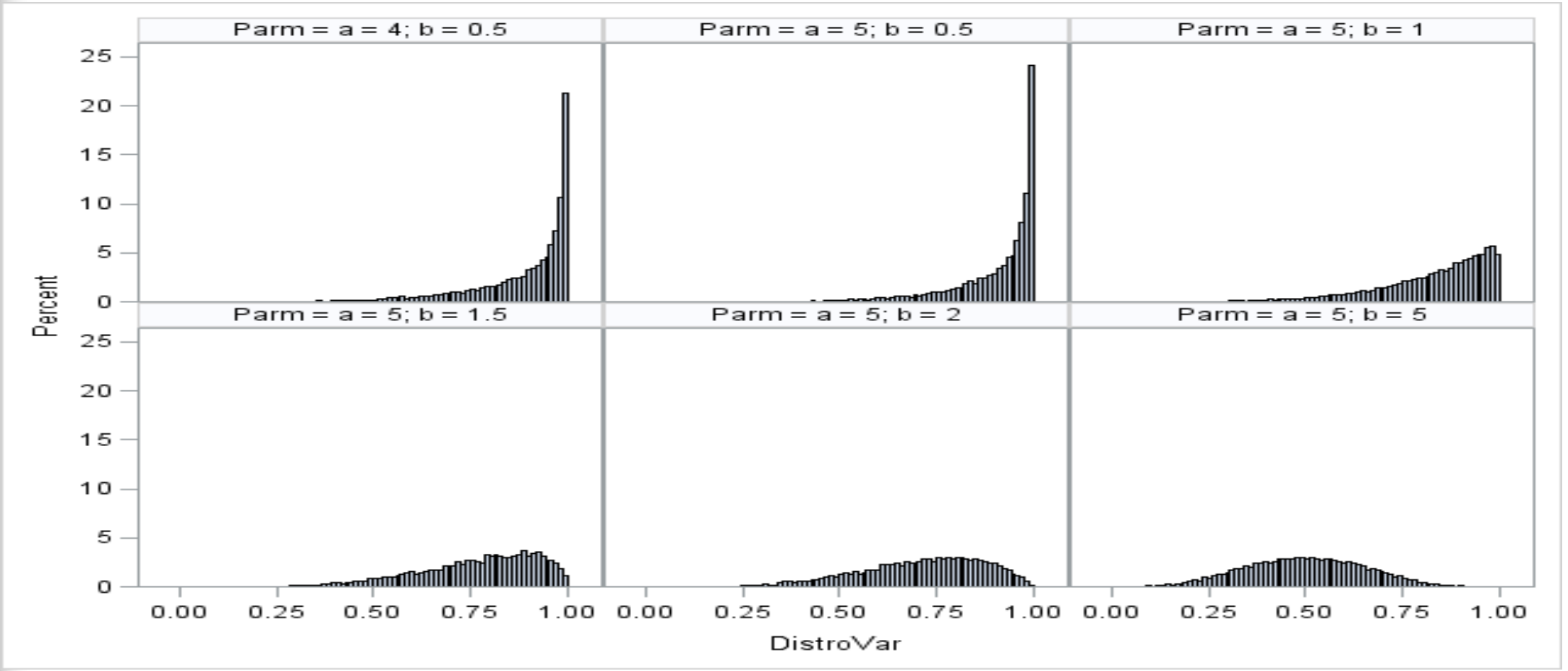
RAND Beta



Ottawa Area SAS Users Society

www.OASUS.ca

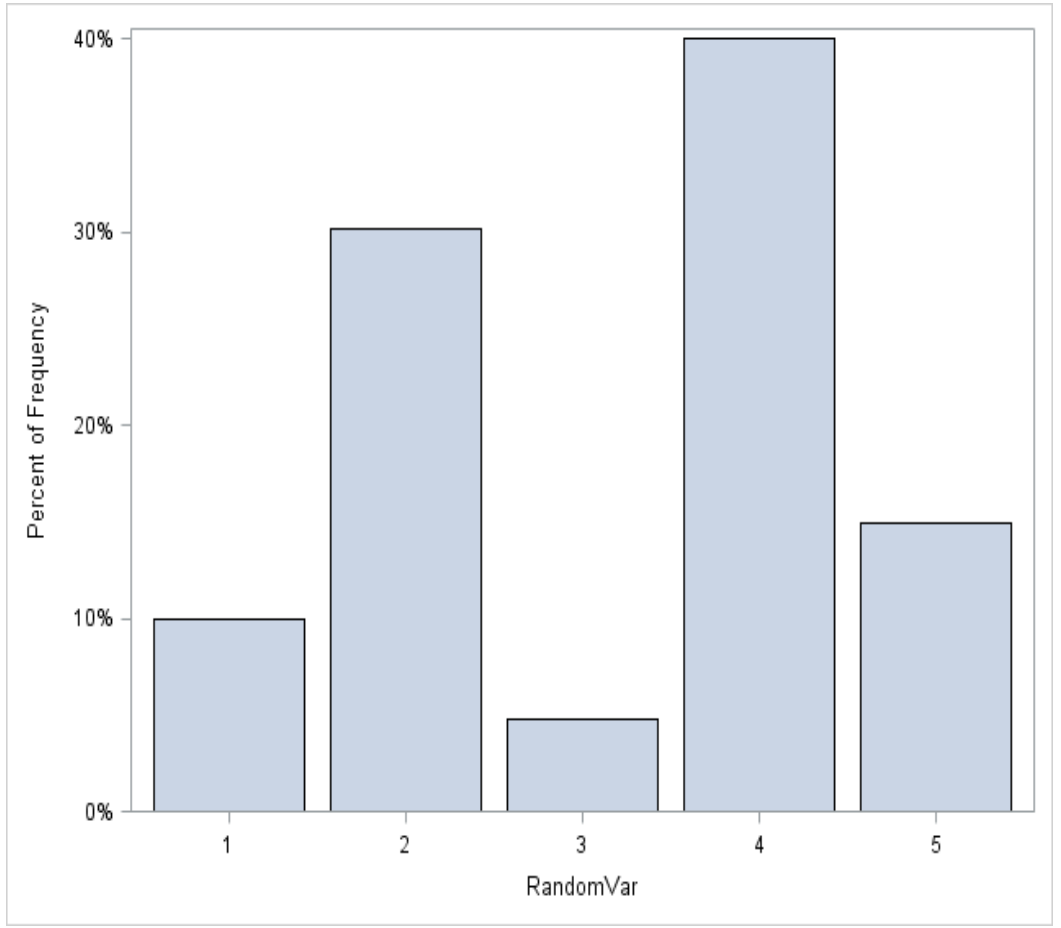
RAND Beta



RAND Tabled

- Returns an integer based on a specified distribution

```
data RandomNumbers(drop=_:);
  do _i = 1 to 10000;
    RandomVar = rand('tabled', .1, .3, .05, .4);
    /* 10% will be 1,
        30% will be 2,
         5% will be 3,
        40% will be 4,
        the remainder (15%) will be 5 */
    output;
  end;
run;
/* Can have up to 32K percentages */
```



STREAMINIT routine

- Allows you to specify a seed, so that the number stream is reproducible

```
/* No input SAS dataset */
data RandomNumbers(drop=_:);
   call streaminit(1704211720);

   do _j = 1 to 15;
      Run_1 = rand('uniform');
      output;
   end;
run;

/* With an input SAS dataset */
data RandomNumbers;
   if _n_ = 1 then
      call streaminit(1704211720);
   set sashelp.class;
   Run_1 = rand('uniform');
run;
```

Generating data: Gender

```
/* Option 1 */

data AgeData(drop=_:);
  do _i = 1 to 39250017;
    RandomVar = rand('uniform');

    if RandomVar <= 19752605 / 39250017
    then Gender = 1; /* Female */
    else Gender = 2; /* Male */
    output;
  end;
run;

/* Option 2 */

data AgeData(drop=_:);
  do _i = 1 to 39250017;
    Gender = rand('tabled', 19752605 / 39250017);
    output;
  end;
run;
```

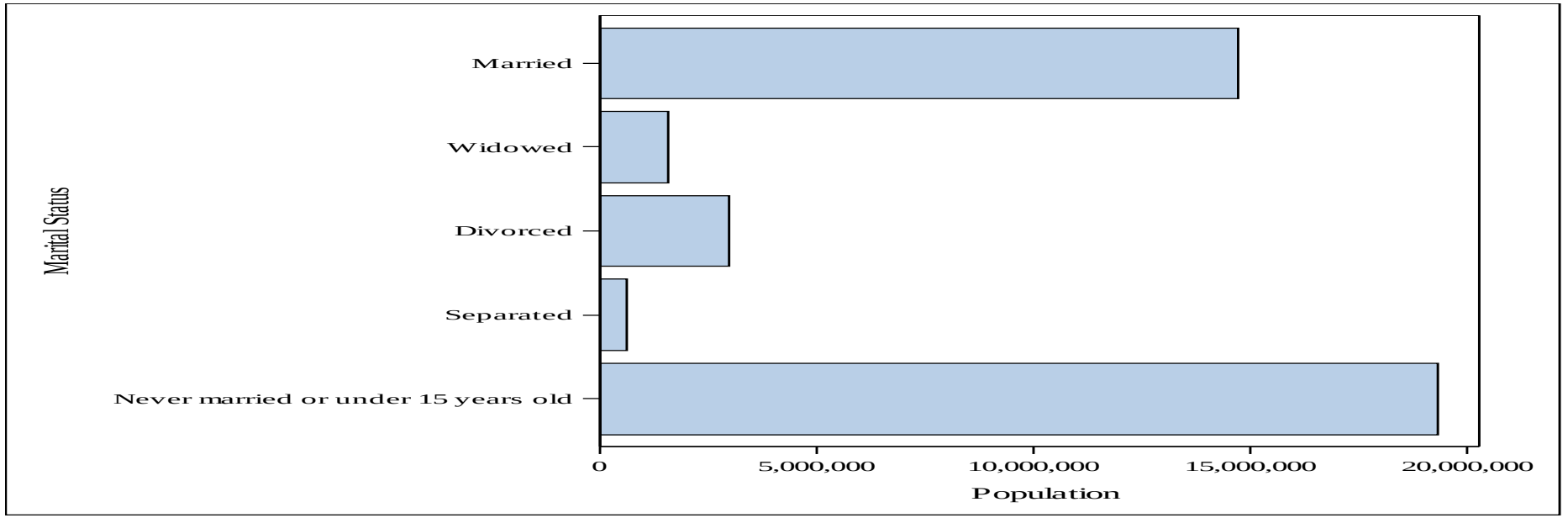
	Count
Female (1)	19,753,788
Male (2)	19,496,229
	39,250,017

Ottawa Area SAS Users Society

www.OASUS.ca

Generating data: Marital Status

```
data MarstData(drop=_:);
  do _i = 1 to 39250017;
    Marst = rand('tabled', 0.375361748, 0.04050276,
            0.0758204, 0.01657612, 0.491738972);
    output;
  end;
run;
```



Generating data: Ancestry

For variables with more than a couple of dozen codes:

1. Get a dataset summarizing counts by code
2. Generate a new dataset, creating approximately but on average slightly more records per code than actual
3. Randomly trim this result down to the exact number of records desired

Generating data: Ancestry

```
/* Get a count by ancestry value */
proc sql noprint;
        create table ACSAncestrySumm as
                select Ancestry, count(*) as AncestryCount
                        from ACSAncestry
                                group by Ancestry
                                        order by Ancestry;
quit;

/* Adjust upwards for disclosure avoidance with small counts */
data ACSAncestrySumm;
        set ACSAncestrySumm;

        if AncestryCount < 20 then
                AncestryCount = 20;
run;
```

Generating data: Ancestry

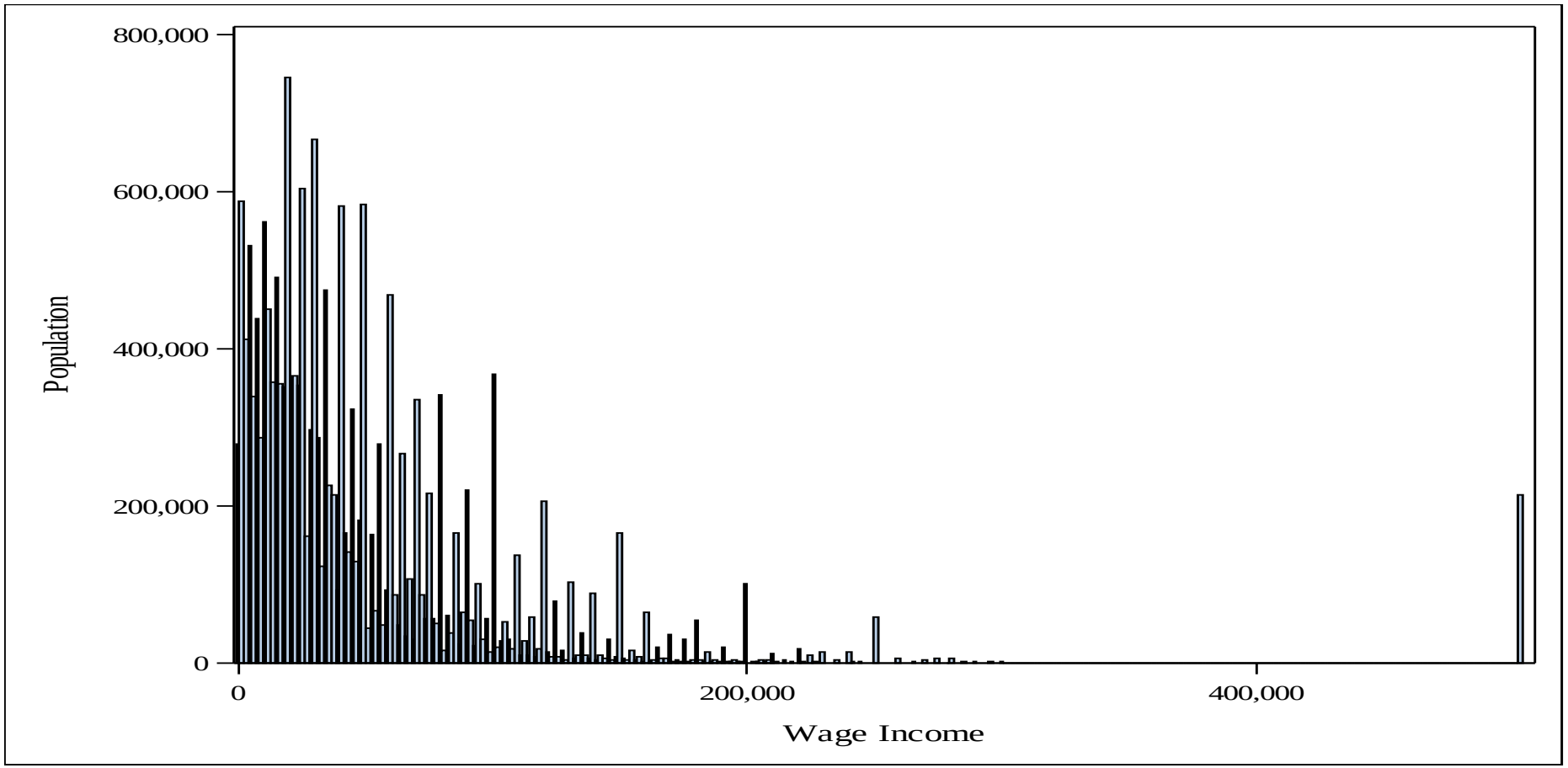
```
/* Generate records, they'll be grouped by ancestry value */
/* Create a few too many records */
data AncestryRecsBig(keep=Ancestry RN);
        set ACSAncestrySumm;
        AncestryCount = int(AncestryCount * (.98 + (rand('uniform')*.07)));

        do _i = 1 to AncestryCount;
                RN = rand('uniform');
                output;
        end;
run;

/* Sort by a random number to randomize the ancestry value sequence */
proc sort data=AncestryRecsBig;
        by RN;
run;

/* Select just the number of records that we want */
data AncestryFinal;
        set AncestryRecsBig(obs=39250017 drop=RN);
run;
NOTE: The data set WORK.ANCESTRYFINAL has 39250017 observations and 1 variables.
```

Generating data: Income

- In this case, we don't really care about the exact values, and there is an enormous number of different values.
- On option is to group by ranges (0-9…90-99, 100-199…900-999, 1000-1999…9000-9999), and generate values as with Ancestry, perturbating the individual values.
- Or we can use random number distributions to generate data directly, if the curve is a good fit.

Generating data: Income

Highly distorted because of high reporting levels at 000's, 0000's

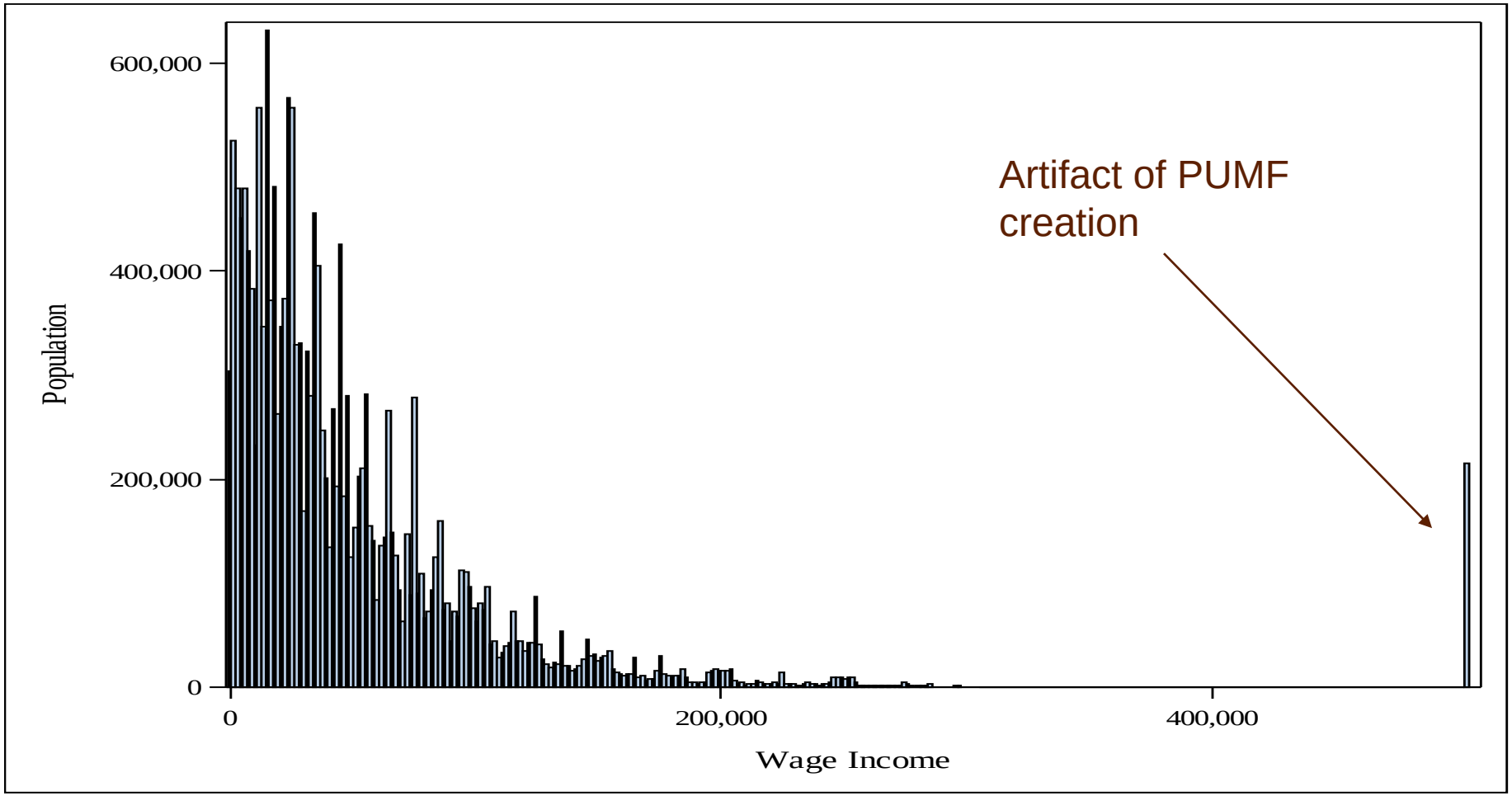


Ottawa Area SAS Users Society

www.OASUS.ca

Generating data: Income

Smoothed to represent real data

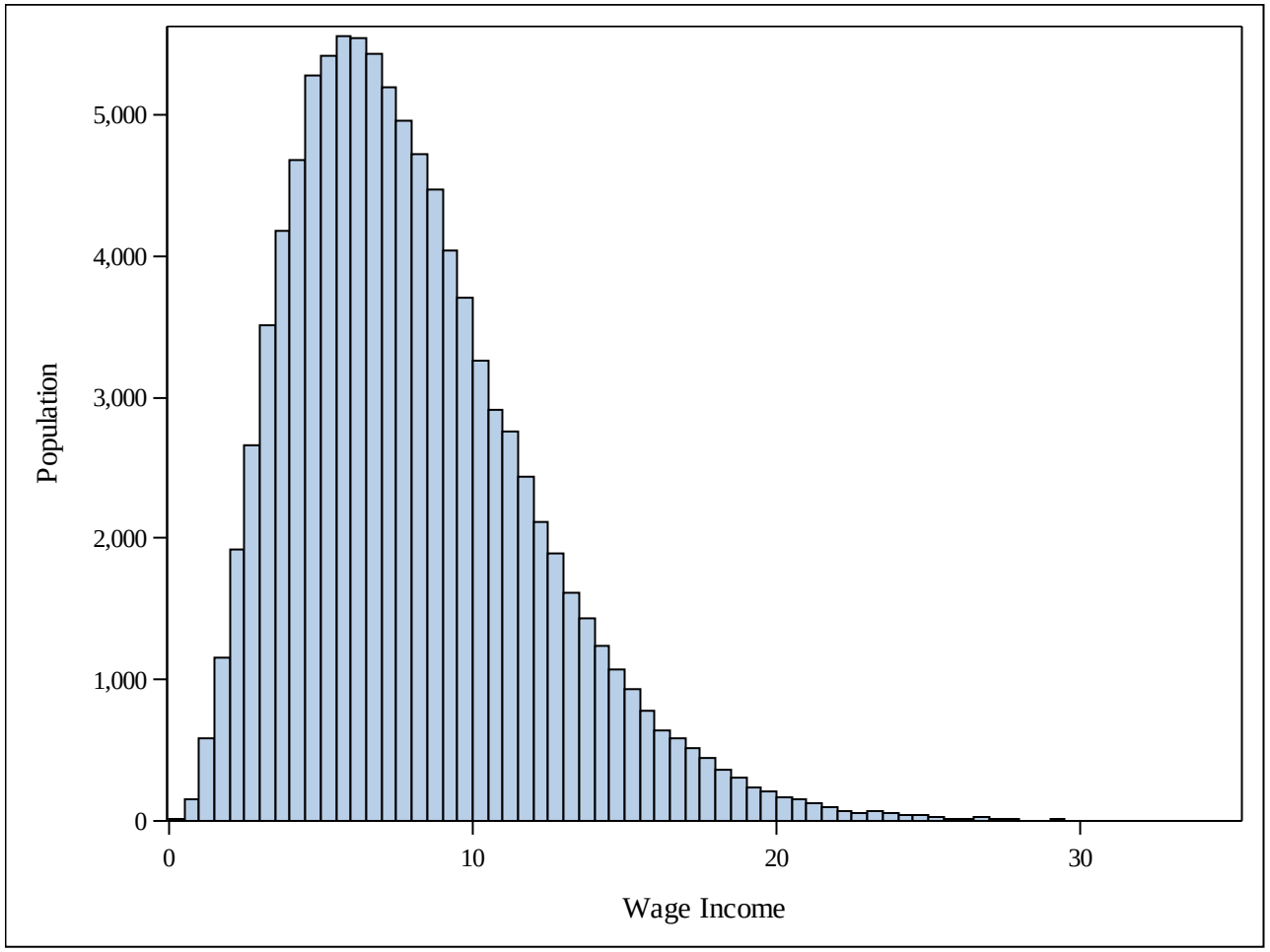


Ottawa Area SAS Users Society

www.OASUS.ca

Income: Step 1

```
data WagPData(drop=_:);
    do _i = 1 to 100000;
        WagP = rand('chisq', 8);
        output;
    end;
run;
```

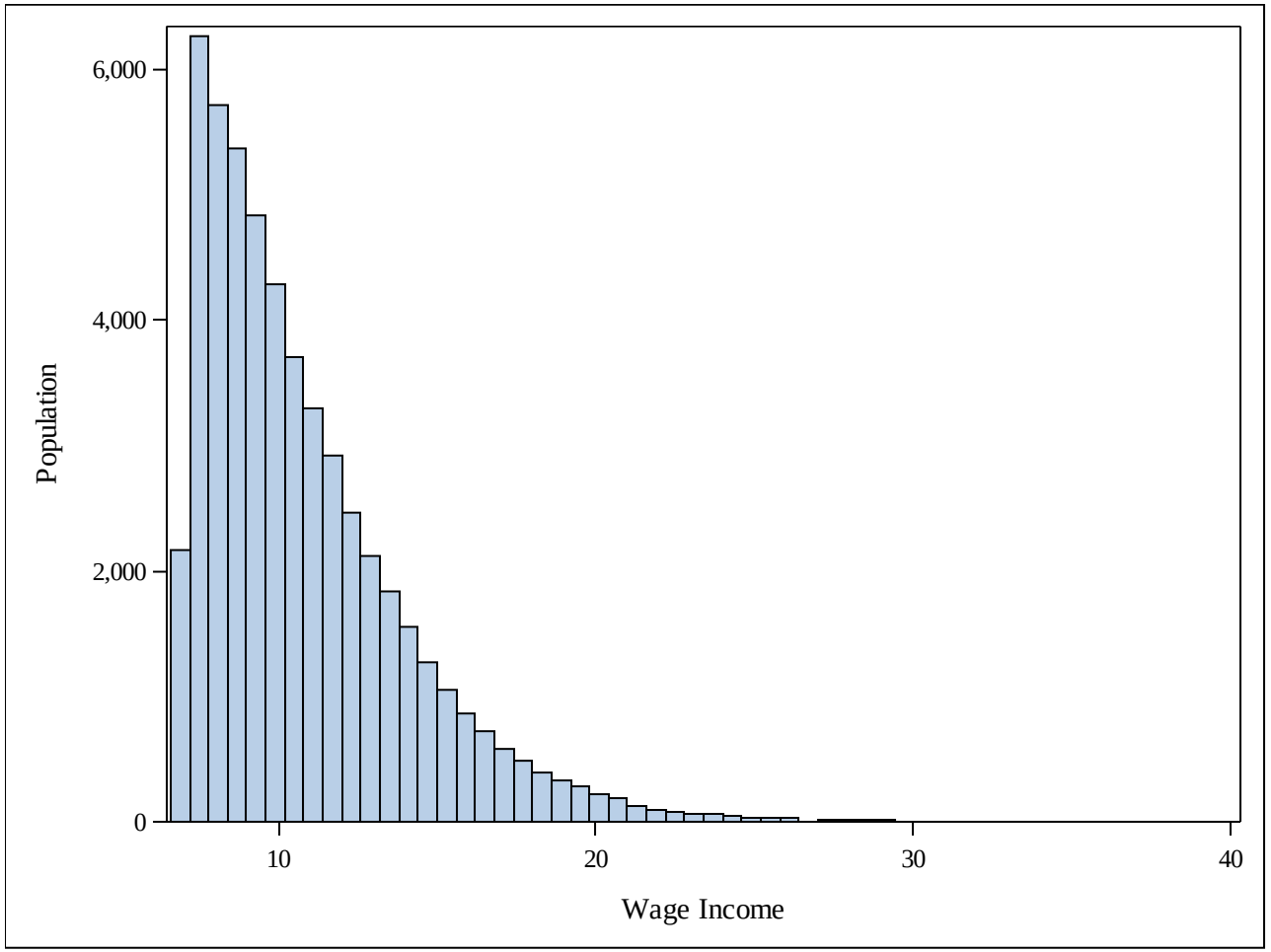


Ottawa Area SAS Users Society

www.OASUS.ca

Income: Step 2

```
data WagPData(drop=_:);
    do _i = 1 to 100000;
        WagP = rand('chisq', 8);
        if WagP > 7
        then output;
    end;
run;

/* max(WagP) is around 35 */
```

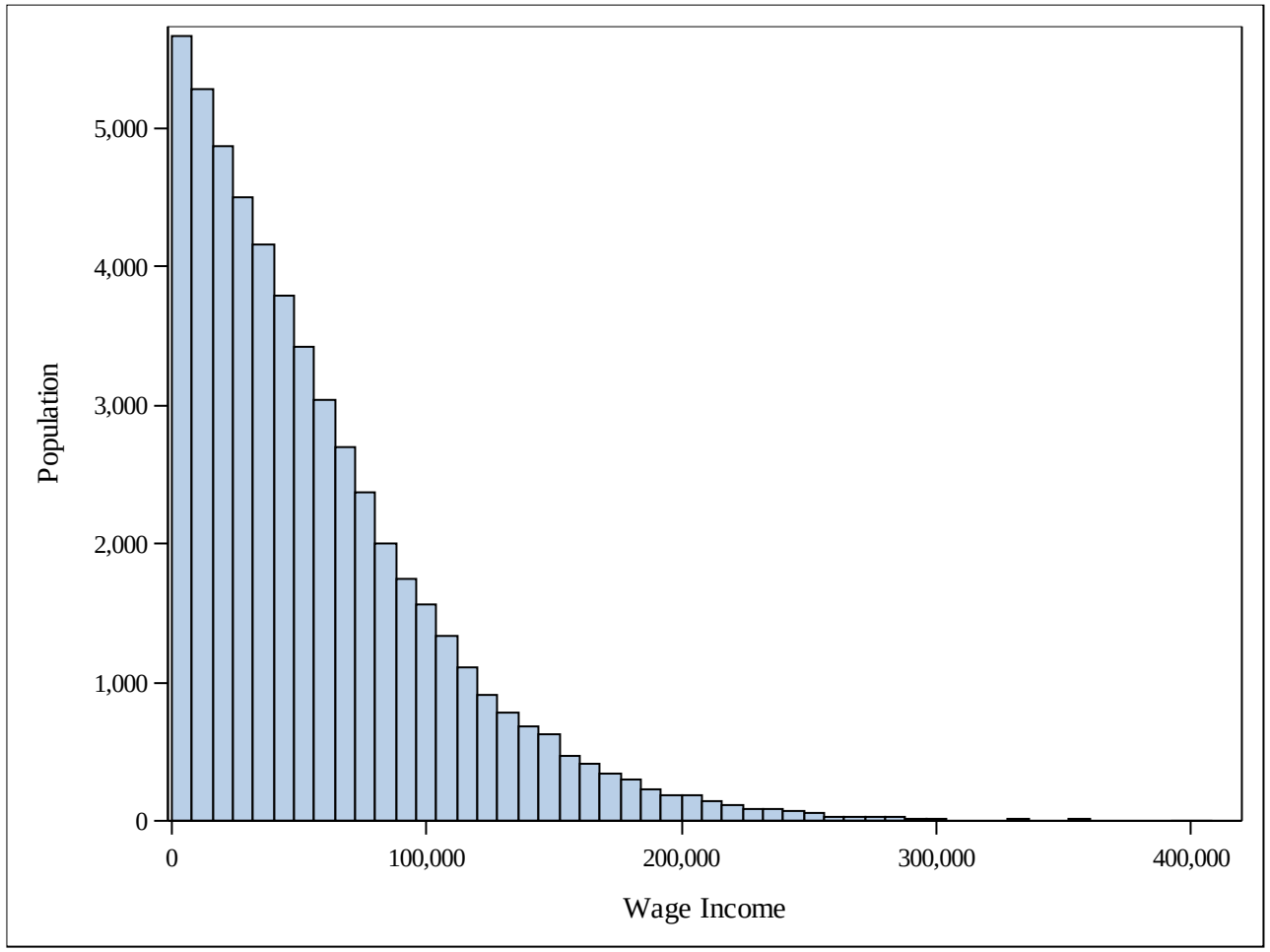


Ottawa Area SAS Users Society

www.OASUS.ca

Income: Step 3

```
data WagPData(drop=_:);
    do _i = 1 to 100000;
        WagP = rand('chisq', 8);
        if WagP > 7 & WagP < 35
        then do;
            WagP = (WagP - 7) * 15000;
            output;
        end;
    end;
run;
```

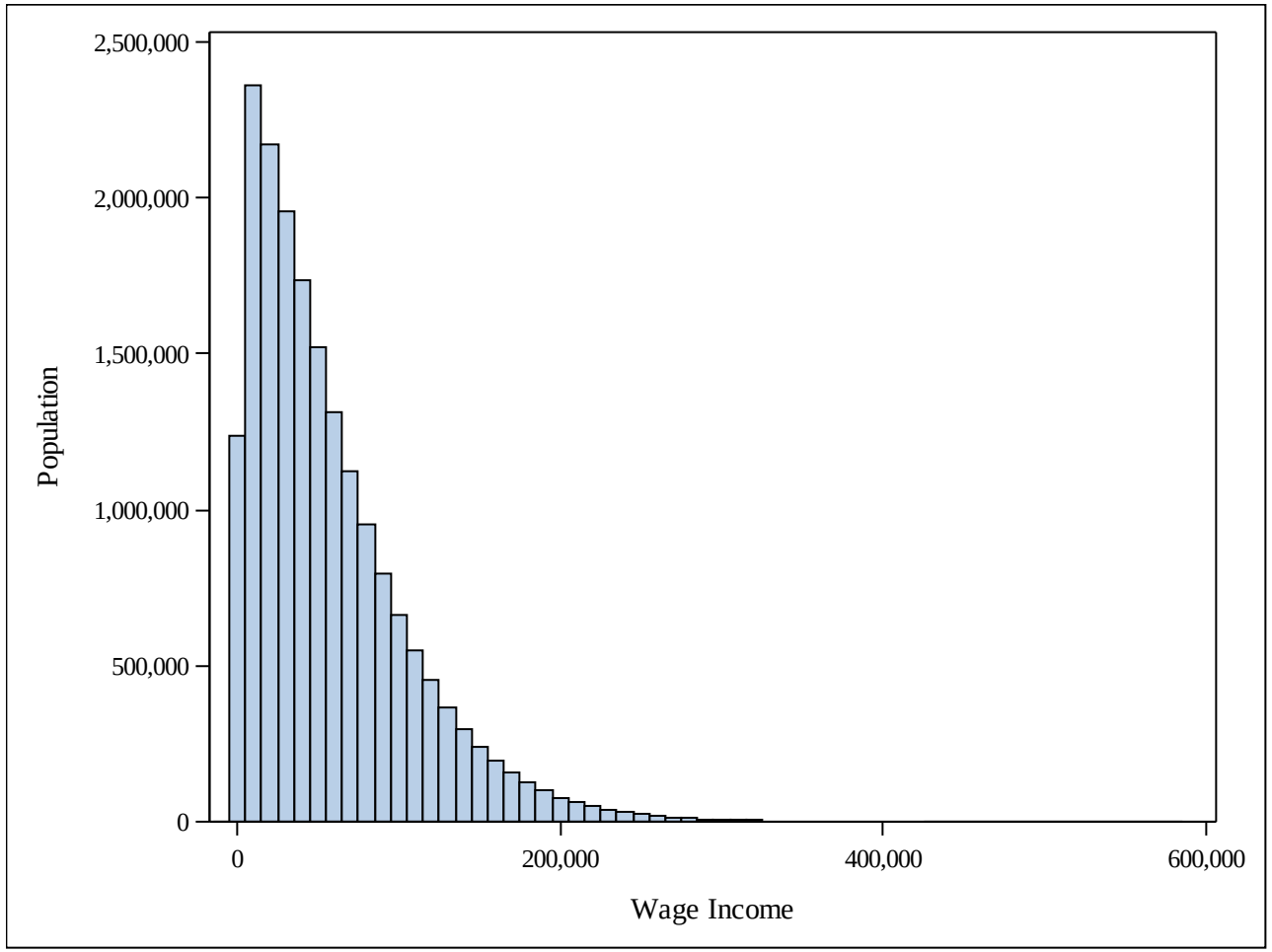


Ottawa Area SAS Users Society

www.OASUS.ca

Income: Step 4

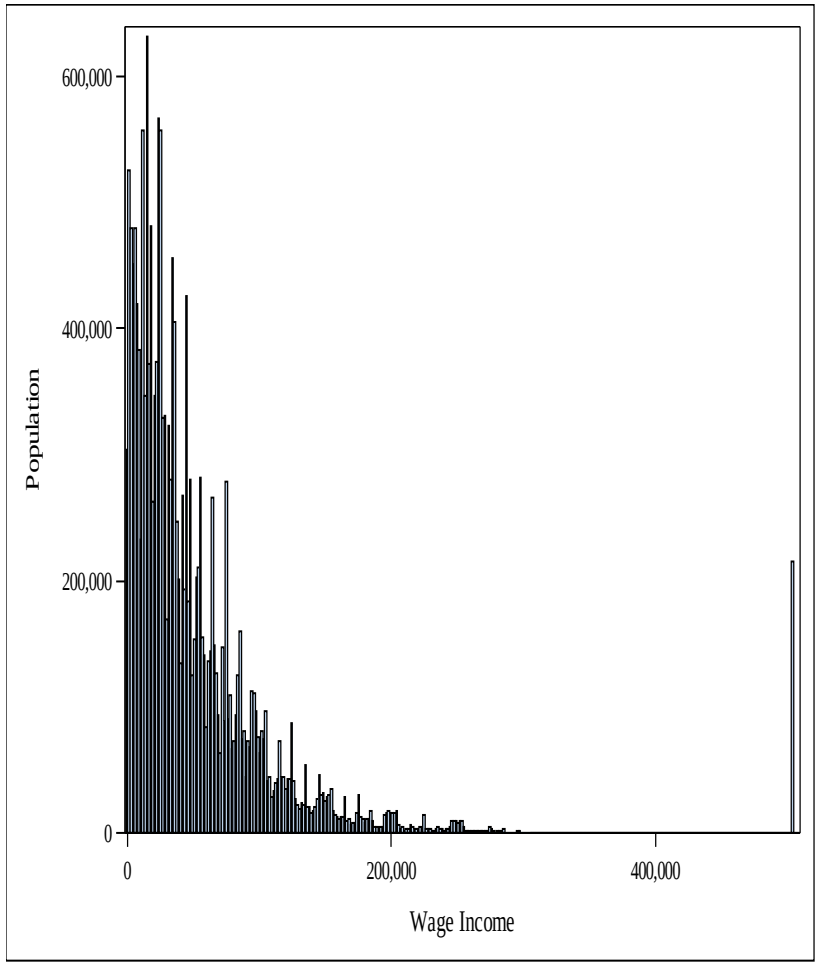
```
/* When we ran 100,000 iterations,
   we got 53,491 records */

data WagPData(drop=_:);
    /* To get desired 18640615 records,
       run 18640615 * 100,000 / 53,491
cycles */
    do _i = 1 to 34848134;
        WagP = rand('chisq', 8);
        if WagP > 7 & WagP < 35
        then do;
            WagP = int((WagP - 7) * 15000);
            output;
        end;
    end;
run;
```

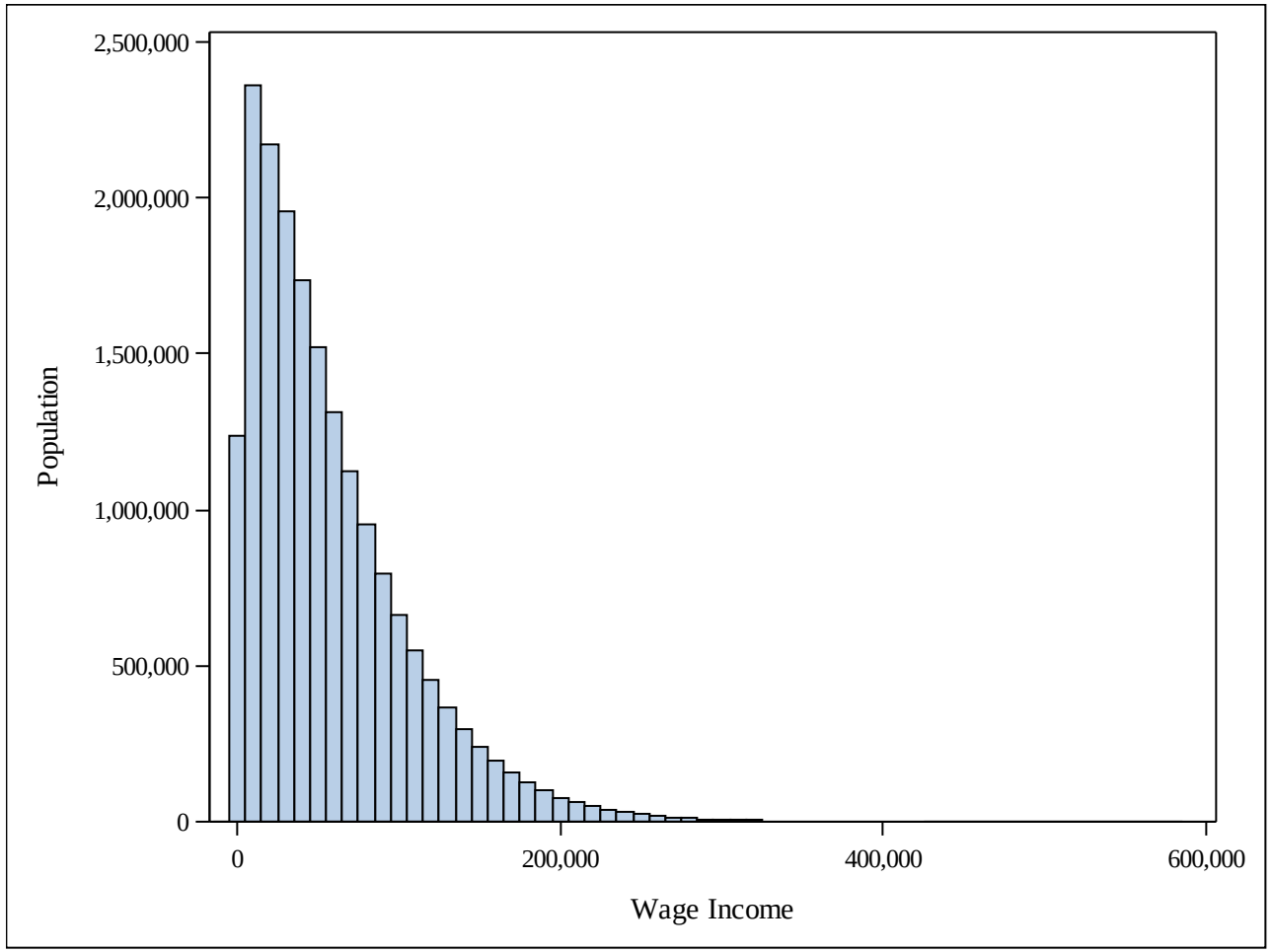


Ottawa Area SAS Users Society

www.OASUS.ca

Income: Complete





Generating data: Age

Can either follow the process for Ancestry, or for Income

Final thoughts

Be prepared to do some additional cleaning, particularly at the high and low end. Watch for negative values where forbidden.

These methods will produce good-looking univariate results. Correlations in multivariate results won't appear. In my experience, this hasn't been a problem.

Ottawa Area SAS Users Society

www.OASUS.ca

So get out there and generate some data!

Thank you for your attention and participation.

Tom