Stratified Sampling Design using Industry Data

MSDS 6370

James Tsai, Wid Sogata

11/22/2016

1. Introduction

In this paper, we are implementing stratified sampling plan design based on a simulated population of one month's data for an industry. The data includes Sales and Inventory for the complete population for one month. We design a sampling plan for a sample of size 500 to be selected for data collection each month for the next 2 years. The purpose of the data collection is to estimate the total Sales and total Inventory for the entire population each month. The quality of the estimate of total Sales has priority over the estimate of total Inventory although both are important.

2. Population Data

The data is in a CSV format file MSDS_6370_Industry3.csv. The variables are shown in the table below.

UnitID	Unique identifier
Sales	Current month sales for unit
Inventory	Current month inventory for unit

Correlation between Sales and Inventory is 0.82 and there are 9762 records of data.

3. Sampling Method

In designing the sampling plan, we consider that estimates will be formed for two variables, Sales and Inventory. The methods for determining strata and sample allocation to strata are designed to use one variable. Although the correlation coefficient for sales and inventory is high at 0.82, a design based on one probably will not be perfect for the other. First, we will create a sampling design using Sales and form estimates for both Sales and Inventory. On the next step, we create sampling design using Inventory and form estimates for both variables. Using the results from both designs, we will decide which one sampling design is better under the requirement of estimates of total for both.

As part of this design, at the beginning we establish sampling frame from the data. Cumulative square root of the frequency is the chosen method to establish strata boundaries. Due to the characteristic of the data that is heavily skewed, a certainty stratum is also established to ensure highly influential units are included in the sample as self-representing units.

4. Descriptive Statistics

We begin by observing some basic statistics about the Sales and Inventory data. Listed below are the count, mean, standard deviation, minimum value, 25th percentile, 50th percentile, 75th percentile, and maximum value for both Sales and Inventory. We note there are no missing values in the data set, and the data is highly skewed for both Sales and Inventory due to the very large standard deviations and the extreme min and max values.

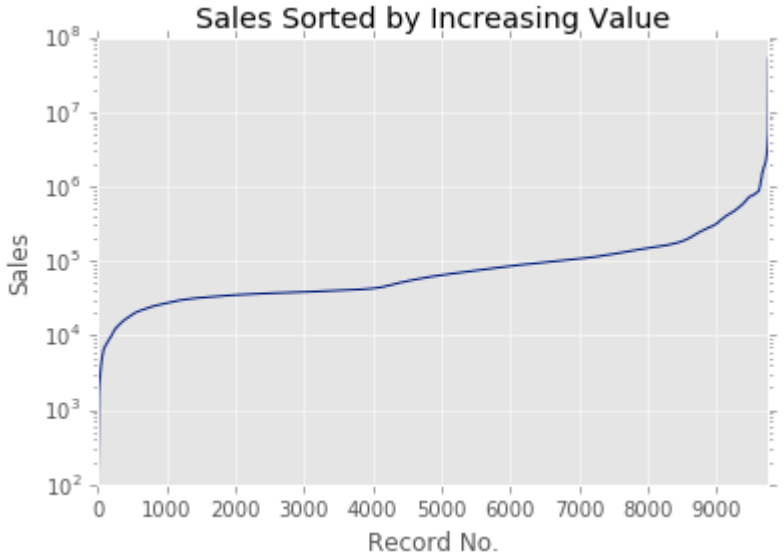
	Sales	Inventory
Count	9,762	9,762
Mean	142,266.99	179,774.11
STD	665,959.08	1,573,357.61
Min	124.43	3.03
25%	36,558.32	22,975.94
50%	62,033.09	55,629.19
75%	116,642.70	136,240.71
Max	52,156,427.82	105,379,552.90

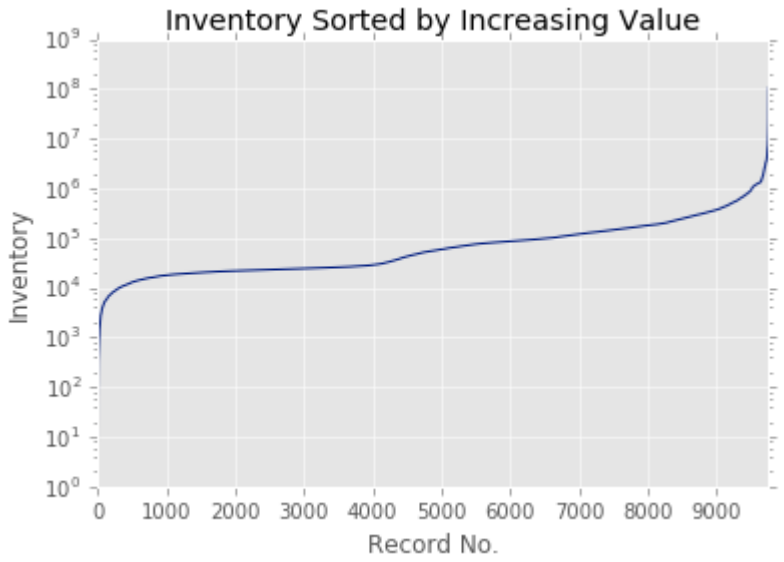
To get a clearer picture of how the data is distributed, we made the following observations by sorting the values from largest to smallest for both Sales and Inventory, and observing most of the values fall within a relatively tight range:

- Values which fall between 1,000 and 10,000:
  - Sales – 66.8%
  - Inventory – 64.1%

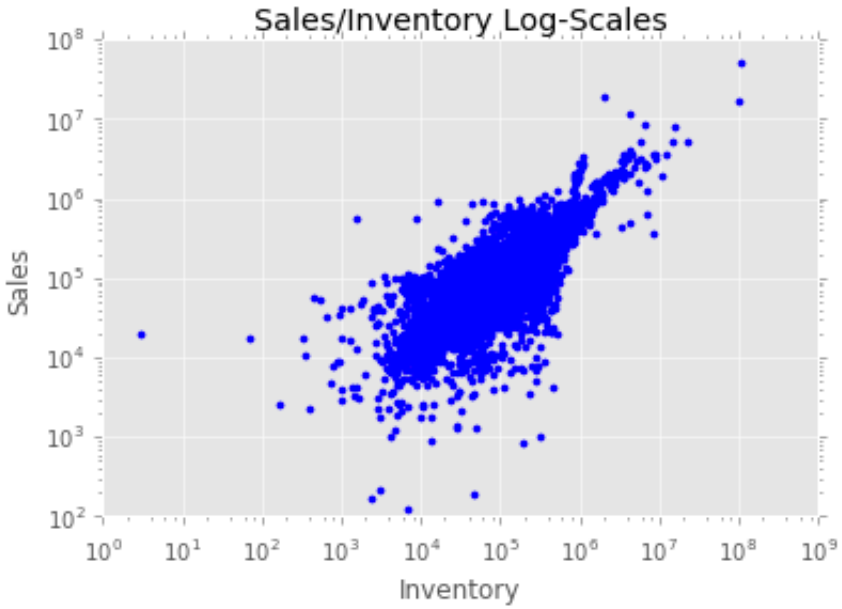
- Values which fall between 10,000 and 100,000:
  - Sales – 29.9%
  - Inventory – 30.2%

We confirm visually with the plots for the sorted Sales and Inventory, showing that the dataset is highly skewed for a relatively small percentage of the values.





Finally, we calculate and plot the correlation between Sales and Inventory using Python/Numpy “corrcoef” function, which calculates the Pearson correlation to 0.82853613. Shown below is a scatter plot of Sales and Inventory data using a log-log scale, which we can visually confirm as being tightly clustered and being positively correlated.



5. Sample Selection Process (Sales)

a. Certainty Stratum

Using the measurement of standard deviation within the data we decided to use the highest 1% of sales as the self-representing stratum. Additionally, this selection of cutoff value is somewhat our judgment call since we are not the experts on the data.

Sales Percentile Removed	Standard Deviation	Total Sales Non-Certainty Strata	Non-Certainty Strata Size	Certainty Stratum Size
0	665,924.97	1,388,810,307	9762	0
1	123,506.08	1,061,749,547	9664	98
2	105,107.01	969,002,984	9566	196
3	90,399.00	896,299,878	9469	293
4	79,278.32	834,784,559	9371	391
5	70,306.29	784,076,862	9273	489

Based on the cutoff of 1% we select the 98 highest sales numbers and assigned them within certainty stratum. We then proceed with the rest of the data for stratification of uncertainty strata.

b. Uncertainty Stratum

To determine the optimum number of bins and strata, we opted for an iterative approach using Python/Pandas/Numpy. The search was limited to 5 to 10 strata after accounting for the certainty stratum. Furthermore, we were given the option of varying the bin sizes to find the best combination. Framing the problem this way, these were the main steps we implemented in the Python code:

- Read the Sales/Inventory data into a Pandas DataFrame.
- Iterate through [20, 25, 30, 35, 40] bins and [5, 6, 7, 8, 9, 10] strata for a total of 30 combinations of bins/strata.
- For each bin/strata combination, calculate the cumulative square root frequency and allocate the stratum accordingly.
- For each of the 30 combinations, create a DataFrame showing the number of bins, strata, and standard deviation for the distances of the stratum. Sort the DataFrame starting with the smallest standard deviation to find the optimum combination.
- For the Sales data, we found the optimum combination to be 35 bins and 5 strata with a standard deviation of 5.03 for the stratum distances.
- For the Inventory data, we found the optimum combination to be 40 bins and 6 strata with a standard deviation of 5.40 for the stratum distances.

Additionally, please refer to the Python notebook we submitted for the actual code.

The following is the stratification result using 35 bins and 5 strata.

bins	f	sqrt(f)	cum(sqrt(f))	stratum
2.9%	1430	37.815341	37.815341	1
5.7%	890	29.832868	67.648209	1
8.6%	810	28.460499	96.108708	1
11.4%	758	27.5318	123.640507	2
14.3%	646	25.41653	149.057037	2
17.1%	507	22.51666	171.573698	2
20.0%	437	20.904545	192.478243	2
22.9%	390	19.748418	212.22666	3
25.7%	353	18.788294	231.014955	3
28.6%	326	18.05547	249.070425	3
31.4%	304	17.435596	266.506021	3
34.3%	285	16.881943	283.387964	3
37.1%	264	16.248077	299.63604	3

40.0%	244	15.620499	315.25654	4
42.9%	223	14.933185	330.189724	4
45.7%	207	14.387495	344.577219	4
48.6%	194	13.928388	358.505607	4
51.4%	180	13.416408	371.922015	4
54.3%	161	12.688578	384.610593	4
57.1%	137	11.7047	396.315292	4
60.0%	118	10.86278	407.178073	4
62.9%	106	10.29563	417.473703	5
65.7%	94	9.69536	427.169063	5
68.6%	80	8.944272	436.113335	5
71.4%	72	8.485281	444.598616	5
74.3%	66	8.124038	452.722654	5
77.1%	60	7.745967	460.468621	5
80.0%	54	7.348469	467.81709	5
82.9%	48	6.928203	474.745294	5
85.7%	44	6.63325	481.378543	5
88.6%	40	6.324555	487.703099	5
91.4%	40	6.324555	494.027654	5
94.3%	37	6.082763	500.110416	5
97.1%	33	5.744563	505.854979	5
100.0%	26	5.09902	510.953999	5
Total	9664			

Distribution of cum$\sqrt{f}$ to 5 strata:

	1	**2**	**3**	**4**	**5**
Boundaries	0-8.6%	8.6-20.0%	20-37.1%	37.1-60.0%	60.0-100%
Length of cum(sqrt(f)) interval	96.1	96.4	107.1	107.6	103.7

Using SAS we summarize the number of members and min max value for each strata from the population.

Obs	stratum	_TYPE_	_FREQ_	min	max
1	1	1	3130	124.43	38659.26
2	2	1	2348	38659.89	73455.95
3	3	1	1922	73478.35	119791.91
4	4	1	1464	119905.18	272061.76
5	5	1	800	272069.68	1367184.71

c. Sample Selection Using Neyman Allocation for Uncertainty Strata

With the design plan of using total sample of 500, since we allocated 98 samples of self-representing stratum, the next step is to select 402 samples out of 9664 available records using Neyman allocation method for each strata. We conducted this method using SAS and hand calculation for verification.

Using SAS:

Obs	stratum	Total	Variance	AllocProportion	SampleSize	ActualProportion
1	1	3130	84709497.96	0.09186	37	0.09204
2	2	2348	130420851.69	0.08551	34	0.08458
3	3	1922	156636558.53	0.07671	31	0.07711
4	4	1464	1522520112.58	0.18216	73	0.18159
5	5	800	48837688089.89	0.56376	227	0.56468

Hand calculation:

Cumulative Sales	Stratum	N_h	S_h	N_h*S_h	n_h*	n_h	Rounded
Up to 90978612.65	1	3130	9202.31	28803230.3	36.94	36.94	37
Up to 121321608.30	2	2348	11417.76	26808900.48	34.38	34.38	34
Up to 181958391.98	3	1922	12512.2	24048448.4	30.84	30.84	31
Up to 242635131.96	4	1464	39006.16	57105018.24	73.24	73.24	73
Up to 424855802.85	5	800	220854.34	176683472	226.60	226.6	227
Total		9664		313449069.4	402	402	402

We arrive at number of samples of 37, 34, 31, 73 and 227 for stratum 1 to 5.

6. Sample Selection Process (Inventory)

a. Certainty Stratum

We repeat the same process for Inventory variable. We also decided to have the cutoff at the highest 1% of inventory number with number of 98 samples.

Inventory Percentile Removed	Standard Deviation	Total Inventory Non-Certainty Strata	Non-Certainty Strata Size	Certainty Stratum Size
0	1,573,277.02	1,388,810,307	9762	0
1	194,408.77	1,061,749,547	9664	98
2	155,519.08	969,002,984	9566	196
3	127,930.39	896,299,878	9469	293
4	110,888.82	834,784,559	9371	391
5	98,799.56	784,076,862	9273	489

b. Uncertainty Stratum

The following is the stratification result using 40 bins and 6 strata.

inventory	f	sqrt(f)	cum(sqrt(f))	stratum
2.5%	1870	43.243497	43.243497	1
5.0%	1266	35.580894	78.82439	1
7.5%	1079	32.848135	111.672526	2
10.0%	662	25.729361	137.401886	2
12.5%	466	21.587033	158.98892	2
15.0%	385	19.621417	178.610336	3
17.5%	347	18.627936	197.238272	3
20.0%	322	17.944358	215.182631	3
22.5%	295	17.175564	232.358195	3
25.0%	263	16.217275	248.57547	3
27.5%	236	15.362291	263.937761	4
30.0%	217	14.73092	278.668681	4
32.5%	200	14.142136	292.810817	4
35.0%	185	13.601471	306.412287	4
37.5%	172	13.114877	319.527164	4
40.0%	160	12.649111	332.176275	4
42.5%	150	12.247449	344.423724	5

45.0%	136	11.661904	356.085627	5
47.5%	122	11.045361	367.130988	5
50.0%	111	10.535654	377.666642	5
52.5%	101	10.049876	387.716518	5
55.0%	95	9.746794	397.463312	5
57.5%	88	9.380832	406.844144	5
60.0%	82	9.055385	415.899529	5
62.5%	75	8.660254	424.559783	5
65.0%	69	8.306624	432.866407	6
67.5%	62	7.874008	440.740415	6
70.0%	56	7.483315	448.223729	6
72.5%	52	7.211103	455.434832	6
75.0%	47	6.855655	462.290486	6
77.5%	43	6.557439	468.847925	6
80.0%	38	6.164414	475.012339	6
82.5%	36	6	481.012339	6
85.0%	32	5.656854	486.669193	6
87.5%	27	5.196152	491.865346	6
90.0%	26	5.09902	496.964365	6
92.5%	24	4.898979	501.863345	6
95.0%	23	4.795832	506.659176	6
97.5%	23	4.795832	511.455008	6
100.0%	21	4.582576	516.037583	6
Total	9664			

Distribution of cum$\sqrt{f}$ to 6 strata:

	1	**2**	**3**	**4**	**5**	**6**
Boundaries	0-5.0%	5.0-12.5%	12-5-25.0%	25.0-40.0%	40-62.5%	62.5-100%
Length of cum(sqrt(f)) interval	78.8	80.1	89.7	83.6	92.4	86.8

Using SAS we summarize the number of members and min max value for each stratum from the population.

Obs	stratum	_TYPE_	_FREQ_	_var_	min	max
1	1	1	3136	29785781.85	3.03	24698.98
2	2	1	2207	215871723.09	24699.31	69949.82
3	3	1	1612	149644650.63	69959.40	117747.29
4	4	1	1170	413222041.82	117755.42	187733.78
5	5	1	960	3740386430.18	187958.49	408405.01
6	6	1	579	91289705219.28	409506.68	1564136.35

c. Sample Selection Using Neyman Allocation for Uncertainty Strata

With the design plan of using total sample of 500, since we allocated 98 samples of self-representing stratum, the next step is to select 402 samples out of 9664 available records using Neyman allocation method for each stratum.

Using SAS:

Obs	stratum	Total	Variance	AllocProportion	SampleSize	ActualProportion
1	1	3136	29785781.85	0.05239	21	0.05224
2	2	2207	215871723.09	0.09926	40	0.09950
3	3	1612	149644650.63	0.06036	24	0.05970
4	4	1170	413222041.82	0.07280	30	0.07463
5	5	960	3740386430.18	0.17971	72	0.17910
6	6	579	91289705219.28	0.53548	215	0.53483

Hand Calculation:

Cumulative Sales	Stratum	N_h	S_h	N_h*S_h	n_h*	n_h	Rounded
Up to 24698.98	1	3136	5457.64	17115159.04	21.06	21.06	21
Up to 69949.82	2	2207	14692.57	32426501.99	39.90	39.90	40
Up to 117747.29	3	1612	12232.93	19719483.16	24.26	24.26	24
Up to 187733.78	4	1170	20327.86	23783596.2	29.27	29.27	30
Up to 408405.01	5	960	61158.7	58712352	72.25	72.25	72
Up to 1564136.35	6	579	302141.86	174940136.9	215.26	215.26	215
Total		9664		326697229.3	402	402	402

We arrive at number of samples of 21, 40, 24, 30, 72 and 215 for stratum 1 to 6.

7. Results

a. Sales Variable

Using SAS procedure Surveymeans with all 6 strata that include certainty and uncertainty arrangement we arrived at the following results. As expected with 95% confidence level, the true value of population mean (142, 267) and total sum (1,388,810,3018) are within all estimation range of 95% confidence intervals. The estimation for Inventory for population mean and population total with true value of 179,774 and 1,754,954,823 respectively are also within estimation range.

The SURVEYMEANS Procedure

Data Summary	
Number of Strata	6
Number of Observations	500
Sum of Weights	9762

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
sales	143761	5315.584496	133316.996	154204.880	1403394277	51890736	1301440514	1505348040
inventory	194550	14649	165767.081	223332.643	1899195750	143007275	1618218240	2180173260

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
sales	142921	5317.410267	132473.824	153368.883	1395198250	51908559	1293209469	1497187031
inventory	182745	14332	154585.573	210905.091	1783959931	139911789	1509064363	2058855500

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
sales	144934	5331.699127	134458.806	155410.013	1414849708	52048047	1312586864	1517112552
inventory	181143	14213	153217.269	209068.475	1768316716	138748383	1495706982	2040926451

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
sales	142599	5306.421961	132172.594	153024.473	1392046885	51801291	1290268861	1493824909
inventory	175679	13972	148227.584	203131.013	1714981310	136393870	1446997670	1982964949

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
sales	144414	5331.355073	133938.768	154888.623	1409766496	52044688	1307510251	1512022740
inventory	188551	14391	160276.167	216825.684	1840634135	140483164	1564615941	2116652329

b. Inventory Variable

Now using SAS procedure Surveymeans on Inventory as stratification reference with 7 strata, we arrived at the following results. As expected with 95% confidence level, the true value of population mean and total sum are within all estimation range of 95% confidence intervals. Estimation of mean and total population of variable Sales are also within range in all test sampling.

The SURVEYMEANS Procedure

Data Summary	
Number of Strata	7
Number of Observations	500
Sum of Weights	9762

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
inventory	179125	13292	153008.702	205240.477	1748614243	129756165	1493670947	2003557538
sales	144589	6523.551963	131771.358	157406.145	1411475394	63682914	1286351998	1536598789

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
inventory	180474	13287	154366.977	206580.386	1761784078	129710538	1506930430	2016637726
sales	159172	10585	138375.316	179969.240	1553839777	103329212	1350819830	1756859724

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
inventory	179936	13293	153818.235	206054.124	1756536988	129766385	1501573613	2011500362
sales	142293	6757.048474	129016.852	155569.182	1389064432	65962307	1259462513	1518666350

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
inventory	181608	13294	155487.787	207728.341	1772857917	129777973	1517871774	2027844060
sales	140515	6720.781646	127310.115	153719.931	1371707658	65608270	1242801347	1500613970

Statistics								
Variable	Mean	Std Error of Mean	95% CL for Mean		Sum	Std Error of Sum	95% CL for Sum	
inventory	178558	13287	152451.475	204664.341	1743082298	129709188	1488231302	1997933293
sales	138410	6102.861588	126419.050	150400.703	1351157213	59576135	1234102767	1468211660

8. Conclusion

In this project, we learn about stratification technique of sampling design using optimal allocation. We implement cumulative square root frequency method for computing approximately optimal stratification of sampling units from the entire population. We then use Neyman method to allocate sample to strata based on the strata variances and similar sampling costs in the strata.

Based on 5 sampling plan estimation we conducted, we would recommend to utilize variable Sales for stratification to make estimation for the next 2 years. It is based on our finding that using variable Sales overall smaller standard error of around 5% and narrower confidence interval in comparison to variable Inventory.

In general, we recommend to use stratification based on the variable of interest to get more accurate estimation.

APPENDIX

SAS Code

```
ODS GRAPHICS OFF;

data in_sales;
infile 'D:\Dropbox\SMU\Sampling\Project\MSDS_6370_Industry3_sales.csv'
dlm=',' firstobs=2;
input UnitID sales inventory stratum;
run;

data in_sales_cert;
infile
'D:\Dropbox\SMU\Sampling\Project\MSDS_6370_Industry3_sales_cert.csv'
dlm=',' firstobs=2;
input UnitID sales inventory stratum;
run;

data in_inv;
infile 'D:\Dropbox\SMU\Sampling\Project\MSDS_6370_Industry3_inv.csv'
dlm=',' firstobs=2;
input UnitID sales inventory stratum;
run;

data in_inv_cert;
infile
'D:\Dropbox\SMU\Sampling\Project\MSDS_6370_Industry3_inv_cert.csv'
dlm=',' firstobs=2;
input UnitID sales inventory stratum;
run;


proc summary data=in_sales;
class stratum;
var sales;
output out=in_sales_sum
 var = _var_ min=min max=max;
run;

data in_sales_sum;
set in_sales_sum;
   if (_TYPE_ = 0) then delete;
run;

proc print data = in_sales_sum;
run;

proc surveyselect data=in_sales n=402 out=in_sales_neyman;
strata stratum / alloc=neyman
      var = in_sales_sum
      nosample;
```

```
run;

proc print data = in_sales_neyman;
run;

data in_sales_all;
set in_sales in_sales_cert;
run;


proc surveyselect data=in_sales_all method = srs out = all_1
      sampsize = (37, 34, 31, 73, 227, 98) seed=91110;
strata stratum;
title "Select All 1";
run;

proc surveyselect data=in_sales_all method = srs out = all_2
      sampsize = (37, 34, 31, 73, 227, 98) seed=91111;
strata stratum;
title "Select All 2";
run;

proc surveyselect data=in_sales_all method = srs out = all_3
      sampsize = (37, 34, 31, 73, 227, 98) seed=91112;
strata stratum;
title "Select All 3";
run;

proc surveyselect data=in_sales_all method = srs out = all_4
      sampsize = (37, 34, 31, 73, 227, 98) seed=91113;
strata stratum;
title "Select All 4";
run;

proc surveyselect data=in_sales_all method = srs out = all_5
      sampsize = (37, 34, 31, 73, 227, 98) seed=91114;
strata stratum;
title "Select All 5";
run;

title 'PROC SURVEYMEANS with Weights';
proc surveymeans data=all_1 mean clm sum clsum total=500;
 var sales inventory;
 strata stratum;
 weight SamplingWeight;

 title "Stats Result 1";
run;

proc surveymeans data=all_2 mean clm sum clsum total=500;
 var sales inventory;
 strata stratum;
 weight SamplingWeight;
 title "Stats Result 2";
run;

proc surveymeans data=all_3 mean clm sum clsum total=500;
```

```
 var sales inventory;
 strata stratum;
 weight SamplingWeight;
 title "Stats Result 3";
run;

proc surveymeans data=all_4 mean clm sum clsum total=500;
 var sales inventory;
 strata stratum;
 weight SamplingWeight;
 title "Stats Result 4";
run;

proc surveymeans data=all_5 mean clm sum clsum total=500;
 var sales inventory;
 strata stratum;
 weight SamplingWeight;
 title "Stats Result 5";
run;


proc summary data=in_inv;
class stratum;
var inventory;
output out=in_inv_sum
/* sum=cum_sum n=n mean=cum_mean std=pop_std var=var */
 var = _var_ min=min max=max;
run;

data in_inv_sum;
set in_inv_sum;
   if (_TYPE_ = 0) then delete;
run;

proc print data = in_inv_sum;
run;

proc surveyselect data=in_inv n=402 out=in_inv_neyman;
strata stratum / alloc=neyman
      var = in_inv_sum
      nosample;
run;

proc print data = in_inv_neyman;
run;

data in_inv_all;
set in_inv in_inv_cert;
run;


proc surveyselect data=in_inv_all method = srs out = all_1_inv
      sampsize = (21, 40, 24, 30, 72, 215, 98) seed=11116;
strata stratum;
title "Select All 1";
run;
```

```
proc surveyselect data=in_inv_all method = srs out = all_2_inv
      sampsize = (21, 40, 24, 30, 72, 215, 98) seed=11117;
strata stratum;
title "Select All 2";
run;

proc surveyselect data=in_inv_all method = srs out = all_3_inv
      sampsize = (21, 40, 24, 30, 72, 215, 98) seed=11118;
strata stratum;
title "Select All 3";
run;

proc surveyselect data=in_inv_all method = srs out = all_4_inv
      sampsize = (21, 40, 24, 30, 72, 215, 98) seed=11119;
strata stratum;
title "Select All 4";
run;

proc surveyselect data=in_inv_all method = srs out = all_5_inv
      sampsize = (21, 40, 24, 30, 72, 215, 98) seed=11110;
strata stratum;
title "Select All 5";
run;

title 'PROC SURVEYMEANS with Weights';
proc surveymeans data=all_1_inv mean clm sum clsum total=500;
 var inventory sales;
 strata stratum;
 weight SamplingWeight;

 title "Stats Result 1";
run;

proc surveymeans data=all_2_inv mean clm sum clsum total=500;
 var inventory sales;
 strata stratum;
 weight SamplingWeight;
 title "Stats Result 2";
run;

proc surveymeans data=all_3_inv mean clm sum clsum total=500;
 var inventory sales;
 strata stratum;
 weight SamplingWeight;
 title "Stats Result 3";
run;

proc surveymeans data=all_4_inv mean clm sum clsum total=500;
 var inventory sales;
 strata stratum;
 weight SamplingWeight;
 title "Stats Result 4";
run;

proc surveymeans data=all_5_inv mean clm sum clsum total=500;
 var inventory sales;
 strata stratum;
```

```
 weight SamplingWeight;
 title "Stats Result 5";
run;
```

Python Code

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('data/MSDS_6370_Industry3.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9762 entries, 0 to 9761
Data columns (total 3 columns):
UnitID       9762 non-null int64
sales        9762 non-null float64
inventory    9762 non-null float64
dtypes: float64(2), int64(1)
memory usage: 228.9 KB
```

In [3]:
```
df.drop('UnitID', axis=1, inplace=True)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9762 entries, 0 to 9761
Data columns (total 2 columns):
sales        9762 non-null float64
inventory    9762 non-null float64
dtypes: float64(2)
memory usage: 152.6 KB
```

In [5]:
```
df.describe()
```

Out[5]:

	sales	inventory
count	9.762000e+03	9.762000e+03
mean	1.422670e+05	1.797741e+05
std	6.659591e+05	1.573358e+06
min	1.244302e+02	3.029167e+00
25%	3.655832e+04	2.297594e+04
50%	6.203309e+04	5.562919e+04
75%	1.166427e+05	1.362407e+05
max	5.215643e+07	1.053796e+08

In [6]:
```
df.inventory.max()
```

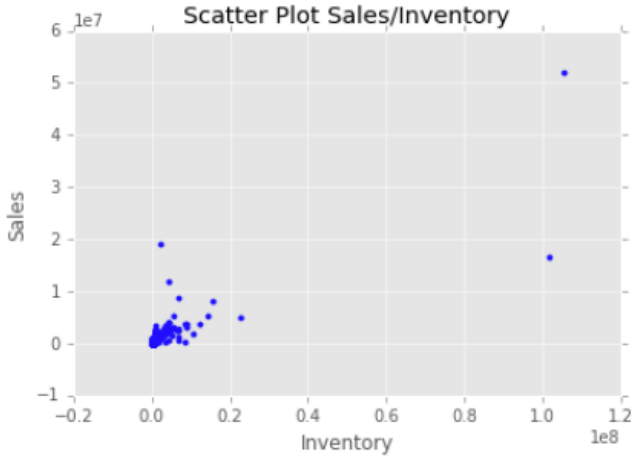
Out[6]:
```
105379552.90000001
```

In [7]:
```
# Pearson product-moment correlation coefficients sales/inventory
np.corrcoef(df.sales, df.inventory)
```

Out[7]:
```
array([[ 1.        ,  0.82853613],
       [ 0.82853613,  1.        ]])
```

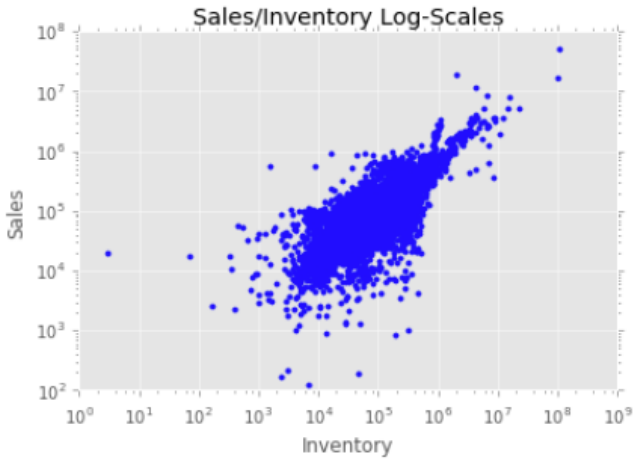
In [8]:
```
plt.style.use('ggplot')
plt.scatter(df.inventory, df.sales, s=10, color='b')
plt.xlabel('Inventory')
plt.ylabel('Sales')
plt.title('Scatter Plot Sales/Inventory')
```

Out[8]:
```
<matplotlib.text.Text at 0x117bac890>
```



In [9]:
```
plt.style.use('ggplot')
plt.scatter(df.inventory, df.sales, s=10, color='b')
plt.xlabel('Inventory')
plt.ylabel('Sales')
plt.xscale('log')
plt.yscale('log')
plt.title('Sales/Inventory Log-Scales')
```

Out[9]: <matplotlib.text.Text at 0x11a40ebd0>



In [10]:
```
# returns the stratum given the value, cutoff, and max number of strata
def calc_stratum(value, cutoff, max_strata):
    num = value/cutoff
    if num >= max_strata:
        return max_strata
    else:
        return int(num)+1
```

In [11]:
```
# returns DataFrame with counts and cumulative frequency calculations for sale
s
def calc_cum_freq_sales(_df, _bins, _stratum, _print):
    buckets = np.linspace(0, _df.sales.sum(), _bins+1)
    groups = _df.groupby(pd.cut(_df.cum_sales, buckets))
    g1 = groups.count()
    df2 = g1.reset_index(drop=True)
    if _print:
        df2
    df2['sqrt'] = np.sqrt(df2.sales)
    df2['sqrt_cumulative'] = df2.sqrt.cumsum()
    c1 = df2.sqrt_cumulative.max()/_stratum
    df2['bin'] = df2.sqrt_cumulative.apply(lambda x: calc_stratum(x, c1, _stra
tum))
    return df2
```

In [12]:
```
# returns DataFrame with counts and cumulative frequency calculations for inve
ntory
def calc_cum_freq_inventory(_df, _bins, _stratum, _print):
    buckets = np.linspace(0, _df.inventory.sum(), _bins+1)
    groups = _df.groupby(pd.cut(_df.cum_inventory, buckets))
    g1 = groups.count()
    df2 = g1.reset_index(drop=True)
    if _print:
        df2
    df2['sqrt'] = np.sqrt(df2.inventory)
    df2['sqrt_cumulative'] = df2.sqrt.cumsum()
    c1 = df2.sqrt_cumulative.max()/_stratum
    df2['bin'] = df2.sqrt_cumulative.apply(lambda x: calc_stratum(x, c1, _stra
tum))
    return df2
```

In [13]:
```
# returns the overall standard deviation for all the strata
def calc_cum_freq_sd(_df, _print):
    b = _df.bin.min()
    s0 = _df[_df.bin==b]['sqrt_cumulative'].max()
    g1 = _df.groupby(_df.bin)
    s1 = g1.sqrt_cumulative.max()
    #print s1
    s2 = s1.diff()
    s3 = s2.dropna()
    if not np.isnan(s0):
        s3[1] = s0
    if _print:
        print s3.sort_index()
    # don't use Bessel's correction in SD calculation
    return s3.std(ddof=0)
```

In [14]:
```
# returns sales total for each stratum
def calc_cum_sales(_df, _df_cf):
    my_values = []
    i = 0
    s = 1
    for j in _df_cf.groupby(_df_cf.bin).sum()['sales']:
        j = i + j
        tot = _df.sales[i:j].sum()
        sd = _df.sales[i:j].std(ddof=0)
        values = [s, tot, sd]
        print tot, sd
        my_values.append(values)
        i = j
        s = s+1
    df_cs = pd.DataFrame(my_values, columns=['stratum','total_sales','sd'])
    return df_cs
```

In [15]:
```
# returns inventory total for each stratum
def calc_cum_inventory(_df, _df_cf):
    my_values = []
    i = 0
    s = 1
    for j in _df_cf.groupby(_df_cf.bin).sum()['inventory']:
        j = i + j
        tot = _df.inventory[i:j].sum()
        sd = _df.inventory[i:j].std(ddof=0)
        values = [s, tot, sd]
        print tot, sd
        values = [s, tot, sd]
        my_values.append(values)
        i = j
        s = s+1
    df_cs = pd.DataFrame(my_values, columns=
['stratum','total_inventory','sd'])
    return df_cs
```

ptimum Non-Certainty Bins/Strata Combination for Sales

In [16]:
```
# create sales DataFrame
df_sales = df.sort_values(by='sales')
df_sales['cum_sales'] = df_sales.sales.cumsum()
df_sales.drop(['inventory'], axis=1, inplace=True)
df_sales = df_sales.reset_index(drop=True)
print 'Standard deviation before Certainty Stratum: %f' % df_sales.sales.std(d
dof=0)
n1 = len(df_sales)
print 'Number of records: %d' % n1
```

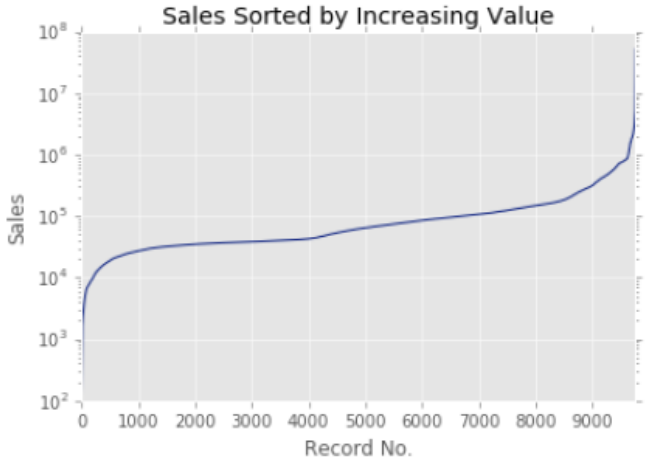
```
Standard deviation before Certainty Stratum: 665924.970455
Number of records: 9762
```

In [17]:
```
print(plt.style.available)
```

```
[u'seaborn-darkgrid', u'seaborn-notebook', u'classic', u'seaborn-ticks', u'gr
ayscale', u'bmh', u'seaborn-talk', u'dark_background', u'ggplot', u'fivethirt
yeight', u'seaborn-colorblind', u'seaborn-deep', u'seaborn-whitegrid', u'seab
orn-bright', u'seaborn-poster', u'seaborn-muted', u'seaborn-paper', u'seaborn
-white', u'seaborn-pastel', u'seaborn-dark', u'seaborn-dark-palette']
```

In [18]:
```
plt.style.use('seaborn-dark-palette')
df_sales.sales.plot()
plt.yscale('log')
plt.ylabel('Sales')
plt.xlabel('Record No.')
plt.title('Sales Sorted by Increasing Value')
plt.xticks([0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000])
```

Out[18]:
```
([<matplotlib.axis.XTick at 0x11ad0add0>,
  <matplotlib.axis.XTick at 0x11ad2aa10>,
  <matplotlib.axis.XTick at 0x11b147950>,
  <matplotlib.axis.XTick at 0x11b147fd0>,
  <matplotlib.axis.XTick at 0x11b1e8790>,
  <matplotlib.axis.XTick at 0x11b1e8f10>,
  <matplotlib.axis.XTick at 0x11adcc950>,
  <matplotlib.axis.XTick at 0x11b204990>,
  <matplotlib.axis.XTick at 0x11b204e90>,
  <matplotlib.axis.XTick at 0x11b176650>],
 <a list of 10 Text xticklabel objects>)
```



In [19]:
```
# 66.8% of the Sales fall between 10^4 and 10^5
# 29.9% of the Sales fall between 10^5 and 10^6
print 'Number of records between 10^1 and 10^2: %d' % len(df_sales[(df_sales.s
ales >= 1e1) & (df_sales.sales < 1e2)])
print 'Number of records between 10^2 and 10^3: %d' % len(df_sales[(df_sales.s
ales >= 1e2) & (df_sales.sales < 1e3)])
print 'Number of records between 10^3 and 10^4: %d' % len(df_sales[(df_sales.s
ales >= 1e3) & (df_sales.sales < 1e4)])
print 'Number of records between 10^4 and 10^5: %d' % len(df_sales[(df_sales.s
ales >= 1e4) & (df_sales.sales < 1e5)])
print 'Number of records between 10^5 and 10^6: %d' % len(df_sales[(df_sales.s
ales >= 1e5) & (df_sales.sales < 1e6)])
print 'Number of records between 10^6 and 10^7: %d' % len(df_sales[(df_sales.s
ales >= 1e6) & (df_sales.sales < 1e7)])
print 'Number of records between 10^7 and 10^8: %d' % len(df_sales[(df_sales.s
ales >= 1e7) & (df_sales.sales < 1e8)])
```

```
Number of records between 10^1 and 10^2: 0
Number of records between 10^2 and 10^3: 7
Number of records between 10^3 and 10^4: 191
Number of records between 10^4 and 10^5: 6524
Number of records between 10^5 and 10^6: 2915
Number of records between 10^6 and 10^7: 121
Number of records between 10^7 and 10^8: 4
```

In [20]:
```
# test for how the standard deviation changes by removing 1% to 5% quantile sa
les data
print '0%','sd:',df_sales.sales.std(ddof=0),'\t', 'count:',len(df_sales)
print '1%','sd:',df_sales[df_sales.sales < df_sales.sales.quantile(.99)].sales
d(ddof=0),'\t', 'count:',len(df_sales[df_sales.sales < df_sales.sales.quantile
99)])
print '2%','sd:',df_sales[df_sales.sales < df_sales.sales.quantile(.98)].sales
d(ddof=0),'\t', 'count:',len(df_sales[df_sales.sales < df_sales.sales.quantile
98)])
print '3%','sd:',df_sales[df_sales.sales < df_sales.sales.quantile(.97)].sales
d(ddof=0),'\t', 'count:',len(df_sales[df_sales.sales < df_sales.sales.quantile
97)])
print '4%','sd:',df_sales[df_sales.sales < df_sales.sales.quantile(.96)].sales
d(ddof=0),'\t', 'count:',len(df_sales[df_sales.sales < df_sales.sales.quantile
96)])
print '5%','sd:',df_sales[df_sales.sales < df_sales.sales.quantile(.95)].sales
d(ddof=0),'\t', 'count:',len(df_sales[df_sales.sales < df_sales.sales.quantile
95)])
```

```
0% sd: 665924.970455    count: 9762
1% sd: 150003.654494    count: 9664
2% sd: 123310.546229    count: 9566
3% sd: 104921.333229    count: 9469
4% sd: 90012.2594549    count: 9371
5% sd: 78876.7637276    count: 9273
```

In [21]:
```
df[df.sales < df.sales.quantile(.95)].sales.std(ddof=0)
```

Out[21]:
```
78876.76372759535
```

In [22]:
```
# drop all the records above 99% Quantile to get Non-Certainty Strata
df_sales = df_sales[df_sales.sales < df_sales.sales.quantile(.99)]
print 'Standard deviation after Certainty Stratum: %f' % df_sales.sales.std(dd
of=0)
n2 = len(df_sales)
print 'Number of records: %d' % n2
```

```
Standard deviation after Certainty Stratum: 150003.654494
Number of records: 9664
```

In [23]:
```
print 'Total   Sales 99%% Quantile: %f' % df_sales.sales.sum()
print 'Average Sales 99%% Quantile: %f' % df_sales.sales.mean()
```

```
Total   Sales 99% Quantile: 1061749547.744449
Average Sales 99% Quantile: 109866.468103
```

In [24]:
```
# number of records dropped
print 'Number of records removed: %d' % (n1-n2)
```

```
Number of records removed: 98
```

In [25]:
```
df_sales.describe()
```

Out[25]:

	sales	cum_sales
count	9.664000e+03	9.664000e+03
mean	1.098665e+05	2.483819e+08
std	1.500114e+05	2.352828e+08
min	1.244302e+02	1.244302e+02
25%	3.644921e+04	6.417081e+07
50%	6.096108e+04	1.688346e+08
75%	1.134662e+05	3.765527e+08
max	1.367185e+06	1.061750e+09

In [26]:
```
# bins and strata combinations to try
bins = [20, 25, 30, 35, 40]
strata = [5, 6, 7, 8, 9, 10]

# define stats
stats = []

for bin in bins:
    for stratum in strata:
        df_cum_freq = calc_cum_freq_sales(df_sales, bin, stratum, False)
        #print df_cum_freq
        sd = calc_cum_freq_sd(df_cum_freq, False)
        stat = [bin, stratum, sd]
        stats.append(stat)

# accumulated stats
df_stats = pd.DataFrame(stats, columns=['bins','strata','sd'])
df_stats = df_stats.sort_values(by='sd')
```

In [27]:
```
# top 5 stats for sales with lowest standard deviation
df_stats.head()
```

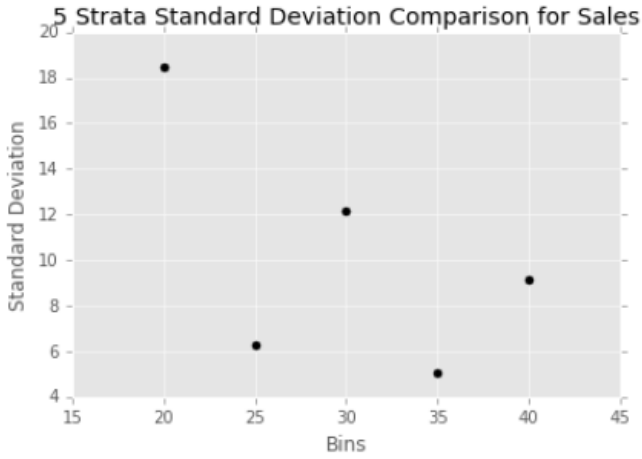
Out[27]:

	bins	strata	sd
18	35	5	5.033852
6	25	5	6.271787
11	25	10	6.427128
16	30	9	6.632302
26	40	7	7.159204

In [28]:
```
df_stats.to_excel('data/sales.xlsx', index=False)
```

In [29]:
```
# Plot variation of Standard Deviation for 10 Strata
df_stats_5 = df_stats[df_stats.strata==5]
plt.style.use('ggplot')
plt.scatter(df_stats_5.bins, df_stats_5.sd, s=25, color='k')
plt.xlabel('Bins')
plt.ylabel('Standard Deviation')
plt.title('5 Strata Standard Deviation Comparison for Sales')
```

Out[29]: <matplotlib.text.Text at 0x11abaf610>



In [30]:
```
# DataFrame with lowest standard deviation among the stratum
df_sales_cf = calc_cum_freq_sales(df_sales, 35, 5, True)
df_sales_cf
```

Out[30]:

	sales	cum_sales	sqrt	sqrt_cumulative	bin
0	1430	1430	37.815341	37.815341	1
1	890	890	29.832868	67.648209	1
2	810	810	28.460499	96.108708	1
3	758	758	27.531800	123.640507	2
4	646	646	25.416530	149.057037	2
5	507	507	22.516660	171.573698	2
6	437	437	20.904545	192.478243	2
7	390	390	19.748418	212.226660	3
8	353	353	18.788294	231.014955	3
9	326	326	18.055470	249.070425	3
10	304	304	17.435596	266.506021	3
11	285	285	16.881943	283.387964	3
12	264	264	16.248077	299.636040	3
13	244	244	15.620499	315.256540	4
14	223	223	14.933185	330.189724	4
15	207	207	14.387495	344.577219	4
16	194	194	13.928388	358.505607	4
17	180	180	13.416408	371.922015	4
18	161	161	12.688578	384.610593	4
19	137	137	11.704700	396.315292	4
20	118	118	10.862780	407.178073	4
21	106	106	10.295630	417.473703	5
22	94	94	9.695360	427.169063	5
23	80	80	8.944272	436.113335	5
24	72	72	8.485281	444.598616	5
25	66	66	8.124038	452.722654	5
26	60	60	7.745967	460.468621	5
27	54	54	7.348469	467.817090	5
28	48	48	6.928203	474.745294	5
29	44	44	6.633250	481.378543	5
30	40	40	6.324555	487.703099	5
31	40	40	6.324555	494.027654	5

	sales	cum_sales	sqrt	sqrt_cumulative	bin
32	37	37	6.082763	500.110416	5
33	33	33	5.744563	505.854979	5
34	26	26	5.099020	510.953999	5

In [31]:
```
# Show the stratum sizes for sales
calc_cum_freq_sd(df_sales_cf, True)
```

```
bin
1     96.108708
2     96.369535
3    107.157798
4    107.542033
5    103.775926
Name: sqrt_cumulative, dtype: float64
```

Out[31]: 5.033852384267736

Cumulative Sales and SD for each Stratum (needed for Neyman calculation)

In [32]:
```
# Calculate Cumulative Sales for each Stratum
df_cum_sales = calc_cum_sales(df_sales, df_sales_cf)
df_cum_sales
```

```
90978612.6484 9202.30591858
121321608.298 11417.7627485
181958391.982 12512.1965252
242635131.962 39006.155153
424855802.854 220854.343357
```

Out[32]:

	stratum	total_sales	sd
0	1	9.097861e+07	9202.305919
1	2	1.213216e+08	11417.762748
2	3	1.819584e+08	12512.196525
3	4	2.426351e+08	39006.155153
4	5	4.248558e+08	220854.343357

Optimum Non-Certainty Bins/Strata Combination for Inventory

In [35]:
```
print 'Number of records between 10^1 and 10^2: %d' % len(df_inventory[(df_inv
entory.inventory >= 1e1) & (df_inventory.inventory < 1e2)])
print 'Number of records between 10^2 and 10^3: %d' % len(df_inventory[(df_inv
entory.inventory >= 1e2) & (df_inventory.inventory < 1e3)])
print 'Number of records between 10^3 and 10^4: %d' % len(df_inventory[(df_inv
entory.inventory >= 1e3) & (df_inventory.inventory < 1e4)])
print 'Number of records between 10^4 and 10^5: %d' % len(df_inventory[(df_inv
entory.inventory >= 1e4) & (df_inventory.inventory < 1e5)])
print 'Number of records between 10^5 and 10^6: %d' % len(df_inventory[(df_inv
entory.inventory >= 1e5) & (df_inventory.inventory < 1e6)])
print 'Number of records between 10^6 and 10^7: %d' % len(df_inventory[(df_inv
entory.inventory >= 1e6) & (df_inventory.inventory < 1e7)])
print 'Number of records between 10^7 and 10^8: %d' % len(df_inventory[(df_inv
entory.inventory >= 1e7) & (df_inventory.inventory < 1e8)])
```

```
Number of records between 10^1 and 10^2: 1
Number of records between 10^2 and 10^3: 13
Number of records between 10^3 and 10^4: 303
Number of records between 10^4 and 10^5: 6258
Number of records between 10^5 and 10^6: 2944
Number of records between 10^6 and 10^7: 235
Number of records between 10^7 and 10^8: 5
```

In [36]:
```
# test for how the standard deviation changes by removing 1% to 5% quantile sa
les data
print '0%','sd:',df_inventory.inventory.std(ddof=0),'\t', 'count:',len(df_inve
ntory)
print '1%','sd:',df_inventory[df_inventory.inventory <
df_inventory.inventory.quantile(.99)].inventory.std(ddof=0),'\t', 'count:',len
_inventory[df_inventory.inventory < df_inventory.inventory.quantile(.99)])
print '2%','sd:',df_inventory[df_inventory.inventory <
df_inventory.inventory.quantile(.98)].inventory.std(ddof=0),'\t', 'count:',len
_inventory[df_inventory.inventory < df_inventory.inventory.quantile(.98)])
print '3%','sd:',df_inventory[df_inventory.inventory <
df_inventory.inventory.quantile(.97)].inventory.std(ddof=0),'\t', 'count:',len
_inventory[df_inventory.inventory < df_inventory.inventory.quantile(.97)])
print '4%','sd:',df_inventory[df_inventory.inventory <
df_inventory.inventory.quantile(.96)].inventory.std(ddof=0),'\t', 'count:',len
_inventory[df_inventory.inventory < df_inventory.inventory.quantile(.96)])
print '5%','sd:',df_inventory[df_inventory.inventory <
df_inventory.inventory.quantile(.95)].inventory.std(ddof=0),'\t', 'count:',len
_inventory[df_inventory.inventory < df_inventory.inventory.quantile(.95)])
```

```
0% sd: 1573277.01959    count: 9762
1% sd: 194408.767718    count: 9664
2% sd: 155519.075332    count: 9566
3% sd: 127930.392719    count: 9469
4% sd: 110888.824825    count: 9371
5% sd: 98799.5644788    count: 9273
```

In [37]:
```
# drop all the records above 99% Quantile to get Non-Certainty Strata
df_inventory = df_inventory[df_inventory.inventory < df_inventory.inventory.qu
antile(.99)]
print 'Standard deviation after Certainty Stratum: %f' % df_inventory.inventor
y.std()
n2 = len(df_inventory)
print 'Number of records: %d' % n2
```

```
Standard deviation after Certainty Stratum: 194418.826899
Number of records: 9664
```

In [38]:
```
print 'Total   Inventory 99%% Quantile: %f' % df_inventory.inventory.sum()
print 'Average Inventory 99%% Quantile: %f' % df_inventory.inventory.mean()
```

```
Total   Inventory 99% Quantile: 1170126497.274873
Average Inventory 99% Quantile: 121080.970331
```

In [39]:
```
# number of records dropped
print 'Number of records removed: %d' % (n1-n2)
```

```
Number of records removed: 98
```

In [40]:
```
# bins and strata combinations to try
bins = [20, 25, 30, 35, 40]
strata = [5, 6, 7, 8, 9, 10]

# define stats
stats = []

for bin in bins:
    for stratum in strata:
        df_cum_freq = calc_cum_freq_inventory(df_inventory, bin, stratum, Fals
e)
        sd = calc_cum_freq_sd(df_cum_freq, False)
        stat = [bin, stratum, sd]
        stats.append(stat)

# accumulated stats
df_stats = pd.DataFrame(stats, columns=['bins','strata','sd'])
df_stats = df_stats.sort_values(by='sd')
```

In [41]:
```
# top 5 stats for inventory with lowest standard deviation
df_stats.head()
```

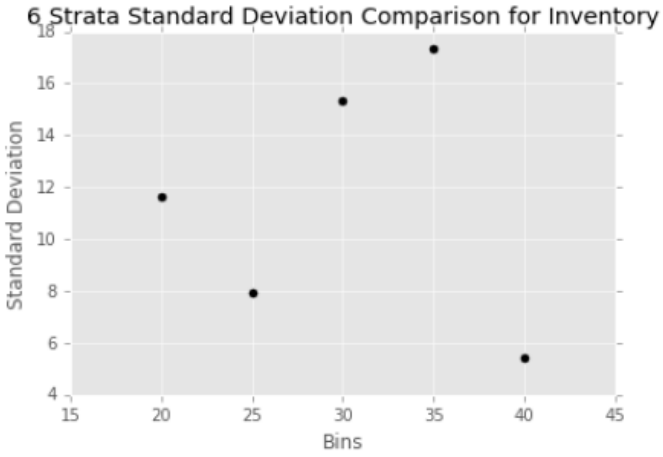
Out[41]:

	bins	strata	sd
25	40	6	5.399333
11	25	10	5.406470
16	30	9	5.739920
23	35	10	6.081633
12	30	5	6.442616

In [42]:
```
df_stats.to_excel('data/inventory_sd.xlsx', index=False)
```

In [43]:
```
# Plot variation of Standard Deviation for 7 Strata
df_stats_6 = df_stats[df_stats.strata==6]
plt.style.use('ggplot')
plt.scatter(df_stats_6.bins, df_stats_6.sd, s=25, color='k')
plt.xlabel('Bins')
plt.ylabel('Standard Deviation')
plt.title('6 Strata Standard Deviation Comparison for Inventory')
```

Out[43]:
```
<matplotlib.text.Text at 0x11b6e4310>
```

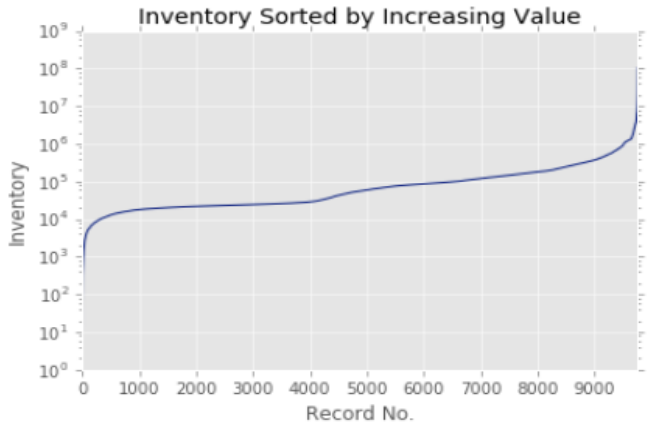


In [33]:
```
# create inventory DataFrame
df_inventory = df.sort_values(by='inventory')
df_inventory['cum_inventory'] = df_inventory.inventory.cumsum()
df_inventory.drop(['sales'], axis=1, inplace=True)
df_inventory = df_inventory.reset_index(drop=True)
print 'Standard deviation before Certainty Stratum: %f' % df_inventory.invento
ry.std(ddof=0)
n1 = len(df_inventory)
print 'Number of records: %d' % n1
```

```
Standard deviation before Certainty Stratum: 1573277.019591
Number of records: 9762
```

In [34]:
```
plt.style.use('seaborn-dark-palette')
df_inventory.inventory.plot()
plt.yscale('log')
plt.ylabel('Inventory')
plt.xlabel('Record No.')
plt.title('Inventory Sorted by Increasing Value')
plt.xticks([0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000])
```

Out[34]:
```
([<matplotlib.axis.XTick at 0x11b22e1d0>,
  <matplotlib.axis.XTick at 0x11b22edd0>,
  <matplotlib.axis.XTick at 0x11b371ad0>,
  <matplotlib.axis.XTick at 0x11b5b2190>,
  <matplotlib.axis.XTick at 0x11b5b2910>,
  <matplotlib.axis.XTick at 0x11b5bc0d0>,
  <matplotlib.axis.XTick at 0x11b341910>,
  <matplotlib.axis.XTick at 0x11b5cbb10>,
  <matplotlib.axis.XTick at 0x11b5ad050>,
  <matplotlib.axis.XTick at 0x11b5ad7d0>],
 <a list of 10 Text xticklabel objects>)
```



In [44]:
```
# DataFrame with lowest standard deviation among the stratum for inventory
df_inventory_cf = calc_cum_freq_inventory(df_inventory, 40, 6, True)
df_inventory_cf
```

Out[44]:

	inventory	cum_inventory	sqrt	sqrt_cumulative	bin
0	1870	1870	43.243497	43.243497	1
1	1266	1266	35.580894	78.824390	1
2	1079	1079	32.848135	111.672526	2
3	662	662	25.729361	137.401886	2
4	466	466	21.587033	158.988920	2
5	385	385	19.621417	178.610336	3
6	347	347	18.627936	197.238272	3
7	322	322	17.944358	215.182631	3
8	295	295	17.175564	232.358195	3
9	263	263	16.217275	248.575470	3
10	236	236	15.362291	263.937761	4
11	217	217	14.730920	278.668681	4
12	200	200	14.142136	292.810817	4
13	185	185	13.601471	306.412287	4
14	172	172	13.114877	319.527164	4
15	160	160	12.649111	332.176275	4
16	150	150	12.247449	344.423724	5
17	136	136	11.661904	356.085627	5
18	122	122	11.045361	367.130988	5
19	111	111	10.535654	377.666642	5
20	101	101	10.049876	387.716518	5
21	95	95	9.746794	397.463312	5
22	88	88	9.380832	406.844144	5
23	82	82	9.055385	415.899529	5
24	75	75	8.660254	424.559783	5
25	69	69	8.306624	432.866407	6
26	62	62	7.874008	440.740415	6
27	56	56	7.483315	448.223729	6
28	52	52	7.211103	455.434832	6
29	47	47	6.855655	462.290486	6
30	43	43	6.557439	468.847925	6
31	38	38	6.164414	475.012339	6

	inventory	cum_inventory	sqrt	sqrt_cumulative	bin
32	36	36	6.000000	481.012339	6
33	32	32	5.656854	486.669193	6
34	27	27	5.196152	491.865346	6
35	26	26	5.099020	496.964365	6
36	24	24	4.898979	501.863345	6
37	23	23	4.795832	506.659176	6
38	23	23	4.795832	511.455008	6
39	21	21	4.582576	516.037583	6

In [45]:
```
# Show the stratum sizes for inventory
calc_cum_freq_sd(df_inventory_cf, True)
```

```
bin
1    78.824390
2    80.164529
3    89.586550
4    83.600805
5    92.383508
6    91.477801
Name: sqrt_cumulative, dtype: float64
```

Out[45]:
```
5.399333412899119
```

Cumulative Inventory for each Stratum (needed for Neyman calculation)

In [46]:
```
# Calculate Cumulative Inventory for each Stratum
df_cum_inventory = calc_cum_inventory(df_inventory, df_inventory_cf)
df_cum_inventory
```

```
58503771.1329 5456.76496079
87727306.1114 14689.2447321
146268074.851 12229.1381121
175462178.175 20319.1747025
263035760.084 61126.8369403
439129406.921 301880.833003
```

Out[46]:

	stratum	total_inventory	sd
0	1	5.850377e+07	5456.764961
1	2	8.772731e+07	14689.244732
2	3	1.462681e+08	12229.138112
3	4	1.754622e+08	20319.174703
4	5	2.630358e+08	61126.836940
5	6	4.391294e+08	301880.833003