
11

Predicting Share of Wallet without Survey Data

Appendix 11.A Six Steps

```
libname sq 'c://0-SOW_q';

data simulate_trx;
set sq.AMPECS_data;
call streaminit(12345);
do i=1 to 30212;

x1=rand('binomial',(1/30), 30);
x2=rand('binomial',(1/30), 30);
x3=rand('binomial',(1/30), 30);
x4=rand('binomial',(1/30), 30);
x5=rand('binomial',(3/30), 30);
x6=rand('binomial',(1/30), 30);

d1=AMPECS_Services_DOLLARS;
d2=AMPECS_Communications_DOLLARS;
d3=AMPECS_Entertainment_DOLLARS;
d4=AMPECS_Merchandise_DOLLARS;
d5=AMPECS_Supplies_DOLLARS;
d6=AMPECS_Travel_DOLLARS;

output;
drop i;
end;
run;
```

```

data sq.SOWq_data;
set simulate_trx;
array x(6)    x1-x6;
array d(6)    d1-d6;
array _01x(6)  _01x1-_01x6;
array _01xxd(6) _01xxd1-_01xxd6;
array xd(6)   xd1-xd6;
array xxd(6)  xxd1-xxd6;

do j=1 to 6;
  _01x(j)=0;
  if x(j) ne 0 then _01x(j)=1;
  if d(j) le 0 then d(j)=uniform(12345)*100;
  xd(j)= x(j)*d(j);
  _01xxd(j)= _01x(j)*xd(j);

  sum_x =sum(of x1- x6);
  sum_01x=sum(of _01x1-_01x6);

  _01xxd(j)= _01x(j)*x(j)*d(j);
  xxd(j) = x(j)*x(j)*d(j);

  SUM_catgDOL=sum(of _01xxd1-_01xxd6);
  SUM_trnxDOL=sum(of xxd1- xxd6);
  drop j;
end;

SOW_q=SUM_catgDOL/SUM_trnxDOL;
label
sum_x='TOTAL_TRX'
sum_01x='TOTAL_CATGS'
x1='SERVICES_TRX(prob. expected)'
x2='COMMUNICATIONS_TRX(prob. expected)'
x3='ENTERTAINMENT_TRX(prob. expected)'
x4='MERCHANDISE_TRX(prob. expected)'
x5='SUPPLIES_TRX(prob. expected)'
x6='TRAVEL_TRX(prob. expected)'

```

```
xd1='SERVICES_DOL'  
xd2='COMMUNICATIONS_DOL'  
xd3='ENTERTAINMENT_DOL'  
xd4='MERCHANDISE_DOL'  
xd5='SUPPLIES_DOL'  
xd6='TRAVEL_DOL'  
xxd1='TOTAL SERVICES_DOL(prob. expected)'  
xxd2='TOTAL COMMUNICATIONS_DOL(prob. expected)'  
xxd3='TOTAL ENTERTAINMENT_DOL(prob. expected)'  
xxd4='TOTAL MERCHANDISE_DOL (prob. expected)'  
xxd5='TOTAL SUPPLIES_DOL(prob. expected)'  
xxd6='TOTAL TRAVEL_DOL(prob. expected)'  
  
SUM_catgDOL='SUM of catg-DOLLARS'  
SUM_trnxDOL='SUM of DOLLAR WEIGHTS'  
SOW_q='SOW_q';  
run;
```

Appendix 11.B Seven Steps

```
libname sq 'c://0-SOW_q';  
title2' BS=30000 BAL_TO_LIMIT PAY_AMOUNT_1 PAY_AMOUNT_2  
PAY_AMOUNT_3';  
  
data SSOWq_data;  
set sq.SOWq_data (in = a) sq.SOWq_data (in = b);  
if 0.00< SOW_q< 0.05 then SOW_q=SOW_q*0.00;  
if 0.05<= SOW_q< 0.10 then SOW_q=SOW_q*0.05;  
if 0.10<= SOW_q<0.20 then SOW_q=SOW_q*0.10;  
if 0.20<= SOW_q<0.30 then SOW_q=SOW_q*0.20;  
if 0.30<= SOW_q <0.40 then SOW_q=SOW_q*0.30;  
if 0.40<= SOW_q <0.50 then SOW_q=SOW_q*0.40;  
if 0.50<= SOW_q <0.60 then SOW_q=SOW_q*0.50;
```

```

if 0.60<= SOW_q <0.70 then SOW_q=SOW_q*0.60;
if 0.70<= SOW_q <0.80 then SOW_q=SOW_q*0.70;
if 0.80<= SOW_q <0.85 then SOW_q=SOW_q*0.80;
if 0.85<= SOW_q <0.90 then SOW_q=SOW_q*0.85;
if a then do;
Y = 1;
wt = SOW_q;
end;
if b then do;
Y = 0;
wt = 1 - SOW_q;
end;
run;

```

```

PROC LOGISTIC data = SSOWq_data nosimple des outest=coef;
model Y =
BAL_TO_LIMIT PAY_AMOUNT_1 PAY_AMOUNT_2 PAY_AMOUNT_3;
weight wt;
run;

```

```

PROC SCORE data=SSOWq_data predict type=parms score=coef out=score;
var BAL_TO_LIMIT PAY_AMOUNT_1 PAY_AMOUNT_2 PAY_AMOUNT_3;
run;

```

```

data score;
set score;
estimate=Y2;
label estimate='estimate';
wtt=1;
run;

```

```

data notdot;
set score;
if estimate ne .;
PROC MEANS data=notdot sum noprint; var wtt;
output out=samsize (keep=samsize) sum=samsize;
run;

```

```

data scoresam (drop=samsize);
set samsize score;
retain n;
if _n_=1 then n=samsize;
if _n_=1 then delete;
run;

```

```

PROC SORT data=scoresam; by descending estimate;
run;

```

```

data score;
set scoresam;
if estimate ne . then cum_n+wtt;
if estimate = . then dec=.;
else dec=floor(cum_n*10/(n+1));
prob_hat=exp(estimate)/(1 + exp(estimate));
run;

```

```

/* Bootstrapping score data */
data score (drop = i sample_size);
choice = int(ranuni(36830)*n) + 1;
set score point = choice nobs = n;
i+1;
sample_size=30000;
if i = sample_size + 1 then stop;
run;
/* End of bootstrapping */

```

```

PROC TABULATE data=score missing;
class dec;
var Y SOW_q;
table dec all, (Y*(mean*f=5.3 (n sum)*f=6.0) (SOW_q)*((mean min max)*f=5.3));
weight wt;
run;

```

```

PROC SUMMARY data=score missing;
class dec;
var SOW_q wtt;

```

```
output out=sum_dec sum=sum_can sum_wt;
```

```
data sum_dec;  
set sum_dec;  
avg_can=sum_can/sum_wt;  
run;
```

```
data avg_rr;  
set sum_dec;  
if dec=.;  
keep avg_can;  
run;
```

```
data sum_dec1;  
set sum_dec;  
if dec=. or dec=10 then delete;  
cum_n +sum_wt;  
r =sum_can;  
cum_r +sum_can;  
cum_rr=(cum_r/cum_n)*100;  
avg_cann=avg_can*100;  
run;
```

```
data avg_rr;  
set sum_dec1;  
if dec=9;  
keep avg_can;  
avg_can=cum_rr/100;  
run;
```

```
data scoresam;  
set avg_rr sum_dec1;  
retain n;  
if _n_=1 then n=avg_can;  
if _n_=1 then delete;  
lift=(cum_rr/n);  
if dec=0 then decc='top';  
if dec=1 then decc='2';
```

```
if dec=2 then decc='3';
if dec=3 then decc='4';
if dec=4 then decc='5';
if dec=5 then decc='6';
if dec=6 then decc='7';
if dec=7 then decc='8';
if dec=8 then decc='9';
if dec=9 then decc='bottom';
if dec ne .;
run;
```

```
PROC PRINT data=scoresam d split='*' noobs;
var decc sum_wt r avg_cann cum_rr lift;
label decc='DECILE'
      sum_wt ='NUMBER OF*ACCOUNTS'
      r ='NUMBER OF*ACCOUNTS'
      cum_r ='CUM No. CUSTOMERS w/*SOW_q'
      avg_cann ='MEAN*SOW_q (%)'
      cum_rr ='C U M*SOW_q (%)'
      lift ='C U M*LIFT (%)';
sum sum_wt r;
format sum_wt r cum_n cum_r comma10.;
format avg_cann cum_rr 4.2;
format lift 3.0;
run;
```