

SAS/OR® Can Help Me Lose Weight?!

AKA, A SAS/OR PROC LP Tutorial

by MAJ Doug McAllaster, US Army
HQ Department of Army - Personnel Staff, Pentagon, Washington DC

ABSTRACT

This paper has two parts. Part one is a tutorial on SAS/OR's (Procedure for Linear Programming) PROC LP. This tutorial uses a seminal prototype, the diet problem, to answer these questions: (1) What is a linear program? (2) How can I model my diet as an LP? (3) What is an objective function? (4) What are constraints? (5) How does SAS/OR calculate a solution? (6) What if I change my mind? Part two is a brief overview of the math programming capabilities of SAS/OR.

PART ONE - PROC LP TUTORIAL

Diet Problem.

Do you remember when eating a balanced diet was a relatively simple problem? We could take a vitamin supplement (pill) and eat some of each of the four food groups: bread, meat, dairy, & fruit/vegetable. That was before we found out that high fat diets are killing us! Well, since we can take a pill to get required vitamins, our efforts here will help us determine a daily diet that gets us enough good stuff without exceeding our allotment of bad stuff. In particular, we have to get enough carbohydrates, protein, and calories without exceeding total fat (or fat calories), cholesterol, or sodium. Of course, we'd like to achieve a diet that meets requirements while not exceeding bad stuff at minimal dollar cost.

Diet Problem Data.

Appendix 1 has the data for our problem. We are going to calculate the best daily diet we can consume from just four servings of food. We must remember what we are trying to calculate here

(called decision variables)! We want to know how many servings of each food. The four foods we consider are: bagels for breakfast (is anything better than good coffee & bagels?), chicken soup for lunch (a perennial favorite), a microwave chicken dinner (we know beef is bad for us), and green beans (just in case we need them). (Note that the author was recently a geographical bachelor living in a college dorm situation with minimal cooking facilities.)

Appendix 1 shows each food as a column header. The first column is for a bowl of chicken soup supplemented with a can of white chicken (there's never enough meat in those things.) The table tells us that the soup costs \$2.99. The soup lunch gives us: 65 grams of carbohydrates, 48 grams of protein, 540 total calories, (of which) 100 (are) fat calories, 10 grams of fat, 110 milligrams (mg) of cholesterol, and 1550 mg of sodium. The last three rows are special rows we will discuss later. We read the other three food columns likewise. The fifth column tells us whether each row is a constraint we must exceed (ge=greater than) or a constraint we must not exceed (le=less than). The last column gives us the total amount of nutrition we exceed (or not) from our diet. Note that the cost\$\$ row has a missing value. That is because this is the number we want to calculate (minimize).

Most math programs can be formulated in the above manner. We have a quantity we want to maximize or minimize, typically cost. We also have multiple constraints we must obey, i.e., to exceed some threshold quantities but to stay below other thresholds.

Refer to appendix 2 for our SAS program and to appendix 3 for its abbreviated output. (I edited the output to shorten its length to the essential pages.)

Basic Diet - Problem1.

The first problem we solve is the basic one in which we exclude the last three special rows. PROC LP answers our question with the output called the (decision) variable summary. PROC LP calculates the number of servings of each food that we should eat in order to minimize our cost while simultaneously meeting our nutritional requirements. We read (under "Activity") that we should eat 0.8176 servings of soup, 0.0 chicken dinners, 2.992 bagel servings, and 2.68 servings of green beans.

PROC LP also calculates tells us (under "Activity") exactly how much of our nutritional requirements we get and at what cost. Recall that we gave simple boundaries to either stay below (ceilings) or to stay above (floors). We see that our diet costs us \$8.03. We get 421 grams of carbohydrates (we needed at least 300 grams). We get just the minimum requirement of protein (50 grams) and of total calories (2000). Likewise, it shows that we do not exceed any of our ceilings on the bad stuff.

Modified Diet - Problem2.

Well, we don't like not having any chicken dinner. We communicate these desires to PROC LP by including a further constrain on our problem formulation. This is the "mineat" (minimum we want to eat) constraint. This equation tells PROC LP that we have a lower bound "lowerbd" requirement of one serving of each food. The next equation "maxeat" is the maximum "upperbd" we want to eat on any day. In words, we're not will to eat more than two of anything except bagels, where our limit is three.

We simply include this constraint in the problem2 data set and send the data set to PROC LP for solution. (On a technical note, PROC LP will send messages to the user if some of the constraints are

cannot be satisfied. The analyst must modify one or more constraints to get a feasible solution.) The output tells us the solution to this new (more constrained) problem. Now we get one soup, one chicken dinner, about 2&1/2 bagels, and 1&1/4 green bean servings. Our modified diet costs us \$9.80. Intuitively, this additional constraint had to cost us at least as much if not more than the previously unconstrained solution.

Integer Diet - Problem3.

Finally, we decide that we don't want to mess around with left leftovers. That is, we want to consume complete (integer) servings. In other words, we don't want 1/4 or 1/2 servings left over. PROC LP can also handle these difficult integer constraints. Again, we just append another constraint to our problem data set. The row `_id_` `no_left` has a `_type_` integer. This tells PROC LP that only integer solutions are acceptable.

Sending this newly constrained data set to PROC LP gives us an answer. We eat one soup, one chicken dinner, three bagels, and one green beans. This solution costs us \$10.37 and prevents leftovers. We can also see how our nutritional requirements measure up against our requirements.

Review.

We are now in a position to test ourselves against the learning objectives. A linear (math) program allocates scarce resources to competing demands using a technique called optimization. The printout at appendix 3 shows us the formulation (equations) we use to model a diet problem in math terms that computers can solve. The first equation is called the objective function. It is our objective to minimize (or maximize) the total value of this equation subject to the following constraint equations. The constraints are conditions we must meet. We don't really care how SAS/OR calculates the solution. That's the whole point. We let it crunch the numbers. Instead, we learned how to read the output to understand the solution PROC LP gave us. Finally, we learned how we can

modify a problem to include other constraints including simple bounds and integer requirements.

PART TWO - OVERVIEW of SAS/OR

SAS/OR provides the capability to solve many operations research problems. Operations research is concerned with making optimal usage of scarce resources, e.g., raw materials, manpower, time, and money. SAS/OR software has procedures that optimally allocate scarce resources to competing requirements using a technique called math programming. The SAS/OR procedures for math programming are PROCs ASSIGN, TRANS, NETFLOW, LP, & NLP. SAS/OR also has procedures for solving scheduling and other project management type problems.

Math Programming.

Math programming is concerned with allocating scarce resources towards the accomplishment of some objective. OR analysts can optimize (either maximize or minimize) some quantifiable objective that is subject to some constraints. For example, corporations often want to maximize a their total profit subject to payroll, production, and distribution capacities. SAS/OR has math programming procedures that enable an analyst to determine the best way to accomplish a quantifiable objective using a math program. The math program is a statement of a business problem in math (equation) form. In fact, the field of operations research is also called management science.

Personal Optimization.

There are many decision problems that can be formulated as a math program. Even individuals face optimization problems daily. For example, finding the shortest way to work is a decision most of us face. Alternatively, perhaps we are more concerned with finding the cheapest or fastest or safest or least interrupted or environmentally

cleanest way to work. Another important decision problem many of us face in getting to work is whether to purchase or lease a car. After that decision is made, deciding on the most car for our money becomes another decision problem.

Blending problems.

The above tutorial used a classic blending problem (the diet problem) to find the mix of ingredients (servings) to be used to make a single product (diet) so that it meets standards (nutrition) at minimum cost. Although individuals do not perform these calculations to determine their diets, farmers do use math programs to formulate an optimal feed mix for their animal stock.

Business Decision Making.

Now, individuals do not usually analyze their decisions with math programs. However, managers must efficiently use their resources to optimize their business' productivity or else someone else will drive them (no pun intended) out of business. SAS/OR has procedures that contain algorithms to solve these and other important decision problems.

REFERENCES

SAS Institute Inc. (1989), SAS/OR User's Guide, Version 6, First Edition, Cary, NC, SAS Institute Inc.

McAllaster, Doug (1994), Insuring Down-Sizing is Right-Sizing with SAS/OR Software, Proceedings of the Nineteenth Annual SAS Users Group International Conference, Cary NC: SAS Institute Inc. 1994. 1695pp.

McAllaster, Doug (1995), Optimal Distribution of Army Officers with SAS/OR Software, Proceedings of the Twentieth Annual SAS Users Group International Conference, Cary NC: SAS Institute Inc. 1995. 1554pp.

McAllaster, Doug (1996), Optimal Matching Officers to Jobs using SAS/OR Software, Proceedings of the Twenty-First Annual SAS Users Group International Conference, Cary NC: SAS Institute Inc. 1996. 1688pp.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration

AUTHOR CONTACT

Readers may contact me at 6059 Estates Dr, Alexandria, VA 22310. Or via e-mail at mcallast@pentagon-dcsper.army.mil.

Appendix 1 Diet Problem Data

r		soup	dinnr	bagel	beans		reqd
#	_id	chckn	chckn	butrd	green	_type	
1	cost_\$	2.99	2.85	1.3	0.63	min	.
1	carbo_s	65	28	110	14	ge	300
1	protein	48	28	0	4	ge	50
1	tot_cal	540	420	300	245	ge	2000
1	fat_cal	100	110	80	0	le	500
1	tot_fat	10	12	8	0	le	60
1	chlstrl	110	150	40	0	le	500
1	sodium	1550	1590	115	1260	le	5000
2	min_eat	1	1	1	1	lowerbd	.
2	max_eat	2	2	3	2	upperbd	.
3	no_left	1	2	3	4	integer	.

Appendix 2 Diet Problem SAS program

```

data rawdata;
  infile ssdata firstobs=3;
  input rnumber _id_ $ soup ckndn bagel gbean _type_ $ _rhs_; run;
proc print data=rawdata;
  title 'problem set input data'; run;
data problem1;
  drop rnumber;
  set rawdata;
  if rnumber<2; run;
proc lp data=problem1;
  title 'problem1 -- linear - no minimum servings'; run;
data problem2;
  drop rnumber;
  set rawdata;
  if rnumber<3; run;
proc lp data=problem2;
  title 'problem2 -- linear - with min & max servings'; run;
data problem3;
  drop rnumber;
  set rawdata;
  if rnumber<4; run;
proc lp data=problem3;
  title 'problem3 -- integer- with min & max servings'; run;

```

Appendix 3 SAS program output

problem set input data

OBS	RNUMBER	_ID_	SOUP	CKNDN	BAGEL	GBEAN	_TYPE_	_RHS_
1	1	cost_\$	2.99	2.85	1.3	0.63	min	.
2	1	carbo_s	65.00	28.00	110.0	14.00	ge	300
3	1	protein	48.00	28.00	0.0	4.00	ge	50
4	1	tot_cal	540.00	420.00	300.0	245.00	ge	2000
5	1	fat_cal	100.00	110.00	80.0	0.00	le	500
6	1	tot_fat	10.00	12.00	8.0	0.00	le	60
7	1	chlstrl	110.00	150.00	40.0	0.00	le	500
8	1	sodium	1550.00	1590.00	115.0	1260.00	le	5000
9	2	min_eat	1.00	1.00	1.0	1.00	lowerbd	.
10	2	max_eat	2.00	2.00	3.0	2.00	upperbd	.
11	3	no_left	1.00	2.00	3.0	4.00	integer	.

Appendix 3 SAS program output

problem1 -- linear - _no minimum servings

LINEAR PROGRAMMING PROCEDURE

VARIABLE SUMMARY

Variable Col Name	Status	Type	Price	Activity	Reduced Cost
1 SOUP	BASIC	NON-NEG	2.99	0.817605	0.000000
2 CKNDN		NON-NEG	2.85	0.000000	0.948363
3 BAGEL	BASIC	NON-NEG	1.3	2.999177	0.000000
4 GBEAN	BASIC	NON-NEG	0.63	2.688735	0.000000

CONSTRAINT SUMMARY

Constraint Row Name	Type	S/S Col	Rhs	Activity	Dual Activity
1 cost_\$	OBJECT			8.037473	.
2 carbo_s	GE	5	300	420.696	0.000000
3 protein	GE	6	50	50.000000	0.026436
4 tot_cal	GE	7	2000	2000.000	0.004510
5 fat_cal	LE	8	500	321.695	0.000000
6 tot_fat	LE	9	60	32.169466	0.000000
7 chlstrl	LE	10	500	209.904	0.000000
8 sodium_	LE	11	5000	5000.000	-0.000461

Appendix 3 SAS program output

problem2 -- linear - with min & max servings

VARIABLE SUMMARY

Variable Col Name	Status	Type	Price	Activity	Reduced Cost
1 SOUP		UPLOWBD	2.99	1.000000	1.147159
2 CKNDN		UPLOWBD	2.85	1.000000	1.558994
3 BAGEL	BASIC	UPLOWBD	1.3	2.443222	0.000000
4 GBEAN	BASIC	UPLOWBD	0.63	1.253198	0.000000

CONSTRAINT SUMMARY

Constraint Row Name	Type	S/S Col	Rhs	Activity	Dual Activity
1 cost_\$	OBJECT			9.805703	.
2 carbo_s	GE	5	300	379.299	0.000000
3 protein	GE	6	50	81.012792	0.000000
4 tot_cal	GE	7	2000	2000.000	0.004475
5 fat_cal	LE	8	500	405.458	0.000000
6 tot_fat	LE	9	60	41.545773	0.000000
7 chlstr1	LE	10	500	357.729	0.000000
8 sodium_	LE	11	5000	5000.000	-0.000370

Appendix 3 SAS program output

problem3 -- integer- with min & max servings

VARIABLE SUMMARY

Variable Col Name	Status	Type	Price	Activity	Reduced Cost
1 SOUP		INTEGER	2.99	1.000000	2.990000
2 CKNDN		INTEGER	2.85	1.000000	2.850000
3 BAGEL		INTEGER	1.3	3.000000	1.300000
4 GBEAN		INTEGER	0.63	1.000000	0.630000

CONSTRAINT SUMMARY

Constraint Row Name	Type	S/S Col	Rhs	Activity	Dual Activity
1 cost_\$	OBJECT			10.370000	.
2 carbo_s	GE	5	300	437.000	0.000000
3 protein	GE	6	50	80.000000	0.000000
4 tot_cal	GE	7	2000	2105.000	0.000000
5 fat_cal	LE	8	500	450.000	0.000000
6 tot_fat	LE	9	60	46.000000	0.000000
7 chlstr1	LE	10	500	380.000	0.000000
8 sodium_	LE	11	5000	4745.000	0.000000