AMPL：数理計画のためのモデリング言語

第 2 章

配合モデルとその他の投入モデル：費用の最小化

配合問題（*diet problem*）を題材に、費用を最小化する線形計画モデルの立式と *AMPL による実装を学ぶ*



AGENDA

本章の内容

2-1 配合問題の線形計画問題

夕食の価格とビタミン供給率から、明示的な LP を手で立式する

2-2 配合問題の AMPL モデル

集合・パラメータ・変数を使い、データ非依存の汎用モデルを書く

2-3 AMPL 配合モデルを使う

データを与えて解き、実行不可能性の診断や整数解も検討する

2-4 ブレンド・経済・スケジューリングへの一般化

投入・産出モデルとして書き直し、幅広い応用へつなげる

SECTION 2-1

$ 配合問題の設定：食品と栄養素

8 種類の調理済み夕食から、1 週間分（＝1 日必要量の 700% 以上）のビタミン要件を満たす最も安い組合せを選ぶ。

記号	食品	価格/パック
BEEF	ビーフ	$3.19
CHK	チキン	$2.59
FISH	魚	$2.29
HAM	ハム	$2.89
MCH	マカロニ&チーズ	$1.89
MTL	ミートローフ	$1.99
SPG	スパゲッティ	$1.99
TUR	七面鳥	$2.49

食品	A	C	B1	B2
BEEF	60	20	10	15
CHK	8	0	20	20
FISH	8	10	15	10
HAM	40	40	35	10
MCH	15	35	15	15
MTL	70	30	15	15
SPG	25	50	25	15
TUR	60	20	15	10

表：1 パック当たりビタミン A・C・B1・B2 の 1 日必要量に対する充足率（%）

≥ 700%
各栄養素について、1 日必要量の 700% 以上を満たす

SECTION 2-1

LP の定式化：目的関数と制約

購入するパック数を X_BEEF, X_CHK, … と表す。総費用とビタミン充足量は、価格・充足率を係数とする 1 次式で書ける。

【目的関数：総費用の最小化】

$$\text{総費用} = 3.19\,X_BEEF + 2.59\,X_CHK + 2.29\,X_FISH + 2.89\,X_HAM + 1.89\,X_MCH + 1.99\,X_MTL + 1.99\,X_SPG + 2.49\,X_TUR$$

【制約の例：ビタミン A の充足率 ≥ 700】

$$60\,X_BEEF + 8\,X_CHK + 8\,X_FISH + 40\,X_HAM + 15\,X_MCH + 70\,X_MTL + 25\,X_SPG + 60\,X_TUR \geq 700$$

- ビタミン C・B1・B2 についても同様の式があり、いずれも 700 以上でなければならない
- 常識的な要件として、非負制約 X_BEEF ≥ 0, X_CHK ≥ 0, … , X_TUR ≥ 0 を加える

4 本
の栄養素制約（A, C, B1, B2）
＋ 8 個の非負制約

SECTION 2-1

明示的な AMPL モデル（diet0.mod）

第 1 章の生産 LP と同様に、変数・目的関数・制約をそのまま書き写せば動くモデルになる。

diet0.mod

```
var Xbeef >= 0; var Xchk >= 0; var Xfish >= 0;
var Xham >= 0; var Xmch >= 0; var Xmtl >= 0;
var Xspg >= 0; var Xtur >= 0;

minimize cost:
  3.19*Xbeef + 2.59*Xchk + 2.29*Xfish + 2.89*Xham +
  1.89*Xmch + 1.99*Xmtl + 1.99*Xspg + 2.49*Xtur;

subject to A:
  60*Xbeef + 8*Xchk + 8*Xfish + 40*Xham +
  15*Xmch + 70*Xmtl + 25*Xspg + 60*Xtur >= 700;
subject to C:
  20*Xbeef + 0*Xchk + 10*Xfish + 40*Xham +
  35*Xmch + 30*Xmtl + 50*Xspg + 20*Xtur >= 700;
subject to B1:
  10*Xbeef + 20*Xchk + 15*Xfish + 35*Xham +
  15*Xmch + 15*Xmtl + 25*Xspg + 15*Xtur >= 700;
subject to B2:
  15*Xbeef + 20*Xchk + 10*Xfish + 10*Xham +
  15*Xmch + 15*Xmtl + 15*Xspg + 10*Xtur >= 700;
```

SECTION 2-1

解いてみる：単調な最適解

amplsession

```
ampl: model diet0.mod;
ampl: solve;
MINOS 5.5: optimal solution found.
6 iterations, objective 88.2

ampl: display Xbeef,Xchk,Xfish,Xham,
  Xmch,Xmtl,Xspg,Xtur;
Xbeef = 0
Xchk = 0
Xfish = 0
Xham = 0
Xmch = 46.6667
Xmtl = -3.69159e-18
Xspg = -4.05347e-16
Xtur = 0
```

$88.20
マカロニ&チーズ 46 2/3 パックのみを購入

- ビタミン A・B1・B2 は、ちょうど 700%（15% × 46 2/3）しか満たしていない
- ビタミン C は必要量よりずっと多く供給される
- 極小の負の値（Xmtl, Xspg）は 0 とみなしてよい
- 制約を = に変えると多様化するが、費用は $89.99 に上昇

SECTION 2-2

一般モデルへ：集合の宣言

- 受け入れられる食事にするには、線形計画問題に大幅な修正が必要
- 食品・栄養素の集合はもちろん、制約や上下限の性質も変えたくなる
- 個別のデータファイルと組み合わせられる汎用モデルに依拠すれば、こうした変更ははるかに容易になる（第 1 章の生産モデルと同じ考え方）
- このモデルは「栄養素」と「食品」という 2 つの集合を扱う

diet.mod

```
set NUTR;
set FOOD;
```

```
NUTR ： 栄養素の集合
   （例：A, C, B1, B2）

FOOD ： 食品の集合
   （例：BEEF, CHK, …）
```

SECTION 2-2

モデルが必要とする数値：パラメータ

diet.mod

```
param cost {FOOD} > 0;

param f_min {FOOD} >= 0;
param f_max {j in FOOD} >= f_min[j];

param n_min {NUTR} >= 0;
param n_max {i in NUTR} >= n_min[i];

param amt {NUTR,FOOD} >= 0;
```

`cost{FOOD}`

各食品 1 パックの費用（正）

`f_min / f_max`

各食品を買うパック数の下限・上限

`n_min / n_max`

各栄養素の摂取量の下限・上限

`amt{NUTR,FOOD}`

amt[i,j] = 食品 j の 1 パックに含まれる栄養素 i の量

f_max の宣言では、上限が対応する下限以上でなければならないと述べるため、FOOD を走るダミー添字 j が必要。

SECTION 2-2

決定変数・目的関数・制約

diet.mod

```
var Buy {j in FOOD} >= f_min[j], <= f_max[j];

minimize Total_Cost:  sum {j in FOOD} cost[j] * Buy[j];

subject to Diet {i in NUTR}:
   n_min[i] <= sum {j in FOOD} amt[i,j] * Buy[j] <= n_max[i];
```

- Buy[j] ：食品 j を買うパック数（決定変数）。f_min[j] と f_max[j] の間に収まる
- 目的関数 Total_Cost ：全食品 j にわたる cost[j]×Buy[j] の合計を最小化
- 制約 Diet[i] ：栄養素 i の総供給量（amt[i,j]×Buy[j] の総和）が n_min[i] と n_max[i] の「二重不等式」を満たす

SECTION 2-2

完全な配合モデル（図 2-1）

diet.mod

```
set NUTR;
set FOOD;

param cost {FOOD} > 0;
param f_min {FOOD} >= 0;
param f_max {j in FOOD} >= f_min[j];

param n_min {NUTR} >= 0;
param n_max {i in NUTR} >= n_min[i];

param amt {NUTR,FOOD} >= 0;

var Buy {j in FOOD} >= f_min[j], <= f_max[j];

minimize Total_Cost:  sum {j in FOOD} cost[j] * Buy[j];

subject to Diet {i in NUTR}:
   n_min[i] <= sum {j in FOOD} amt[i,j] * Buy[j] <= n_max[i];
```

SECTION 2-3

D データを与える（diet.dat）

diet.dat

```
set NUTR := A B1 B2 C ;
set FOOD := BEEF CHK FISH HAM MCH MTL SPG TUR ;

param:   cost  f_min  f_max :=
  BEEF   3.19    0     100
  CHK    2.59    0     100
  FISH   2.29    0     100
   ...    ...    ...   ...
  TUR    2.49    0     100 ;

param:   n_min  n_max :=
   A      700   10000
   ...     ...    ...  ;

param amt (tr):
         A    C   B1   B2 :=
   BEEF  60   20   10   15
   ...   ...  ...  ...  ... ;
```

amt (tr) の (tr) は
表の「転置」を表す：

列 = 第 1 添字（栄養素）
行 = 第 2 添字（食品）

- f_min / n_min はもとの要件どおり
- f_max / n_max は当面、最適解に影響しない大きな値
- モデル側を param amt {FOOD,NUTR} と変えることもできる（その場合 amt[j,i] と参照）

SECTION 2-3

モデル + データを解く／拡張の要件

amplsession

```
ampl: model diet.mod;
ampl: data diet.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
6 iterations, objective 88.2

ampl: display Buy;
Buy [*] :=
BEEF   0     MCH 46.6667
  CHK  0     MTL -1.1e-16
FISH   0     SPG -1.3e-16
  HAM  0     TUR  0
;
```

結果は先ほどと同じ（objective 88.2）。より受け入れやすい食事にするため、次の要件を加えたい：

- 多様性：各食品を 2〜10 パックの範囲で含める
- ナトリウム：総量 40,000 mg を超えない
- カロリー：総量 16,000〜24,000 の範囲
- → 新しいデータ diet2.dat に反映して再実行

SECTION 2-3

実行不可能問題を診断する

amplsession

```
ampl: model diet.mod;
ampl: data diet2.dat;
ampl: solve;
MINOS 5.5: infeasible problem.
```

- B2 の body が lb（700）ちょうど → 供給不足
- NA の body が ub（40000）に到達 → 上限が効いている
- → B2 の要求を下げるか、ナトリウム上限を緩めるかの選択

```
display Diet.lb, Diet.body, Diet.ub;
```

	lb	body	ub
A	700	1993.09	20000
B1	700	841.09	20000
B2	700	601.09	20000
C	700	1272.55	20000
CAL	16000	17222.9	24000
NA	0	40000	40000

let 文で修正

```
ampl: let n_max["NA"] := 50000;
ampl: solve;
MINOS 5.5: optimal solution found.
5 iterations, objective 118.0594032
```

SECTION 2-3

整数個のパックで最良解を求める

- 連続最適解はビーフ 5.36、スパゲッティ 9.31 パックなど端数を含む
- ビーフ 6・スパゲッティ 10 に切り上げるとナトリウム上限に違反
- ビーフ 5・スパゲッティ 9 に切り下げるとビタミン B2 が不足

$118.06 → $119.30
連続最適解 → 整数最適解（+$1.24）

dieti.mod

```
var Buy {j in FOOD} integer
   >= f_min[j], <= f_max[j];
```

amplsession

```
ampl: option solver cplex;
ampl: solve;
CPLEX: optimal integer solution;
  objective 119.3
```

- 整数性は追加の制約なので、費用が上がるのは当然
- しかし食事の内容は大きく変化：3 種類の食品の量がそれぞれ 2 パック以上変わる
- 整数計画などの「離散的」な制限は、一般に解を見つけにくくする（第 20 章で詳述）

SECTION 2-4

一般化：投入産出モデル（図 2-4）

blend.mod

```
set INPUT;   set OUTPUT;

param cost {INPUT} > 0;
param in_min/in_max {INPUT};
param out_min/out_max {OUTPUT};
param io {OUTPUT,INPUT} >= 0;

var X {j in INPUT} >= in_min[j],
                    <= in_max[j];
minimize Total_Cost:
  sum {j in INPUT} cost[j] * X[j];
```

ブレンド

原料（飼料成分・原油留分・石炭）を混合し、ナトリウムやオクタン価などの品質を満たす

経済

生産活動（投入）を組み合わせ、需要（産出）を最小費用で満たす（経済均衡と関係）

スケジューリング

勤務スケジュール（投入）で、各日の必要労働時間（産出）を満たす

前提となる仮定： 連続性（X[j] は範囲内の任意値）／ 費用の線形性／ 収量の線形性
（例：鉛の添加とオクタン価の関係のように、非線形な場合は最適化がより難しくなる）

まとめ

1. **配合問題は費用最小化 LP の典型例**

   利益最大化モデルと異なり、制約は「品質」の量に対する下限が中心になる

2. **集合とパラメータで汎用モデルを構築**

   NUTR・FOOD などの集合と param 宣言により、データを差し替えるだけで様々な問題を解ける

3. **限界値で実行不可能性を診断できる**

   Diet.lb / Diet.body / Diet.ub を調べ、どの制約が効いているかを特定し、let 文でデータを修正できる

4. **整数性は解を大きく変え、専用ソルバーが必要**

   四捨五入は容易に実行不可能になる。CPLEX など整数計画に対応したソルバーで解く

5. **ブレンド・経済・スケジューリングへ一般化**

   投入・産出モデルとして書き直せば、幅広い最小費用配分問題に応用できる

次章：*等式制約とその他の LP の話題*