

# 構造推定入門: 講義スライド

応用社会科学RAブートキャンプ

講師: 遠山祐太 (早稲田大学)

最終更新: 2023-08-30 02:05:15

# イントロダクション

# 自己紹介

- 遠山祐太（とおやま ゆうた）
  - ホームページ: <https://yutatoyama.github.io/>
- 経歴：
  - 仙台市生まれ、仙台一高卒
  - 京大経済→東大院修士（公共政策→経済学）→Ph.D留学
  - 米国ノースウェスタン大学で Ph.D. in Economics 取得
  - 帰国後、2018年9月より早稲田大学政治経済学部准教授
  - その他のお仕事：UTEconアドバイザー
- 研究・教育：産業組織論、応用計量経済学、エネルギー・環境経済学

# 今日のプランと参考文献

- 前半（13時半から15時）：
  1. イントロダクション：構造推定とは？（15分）
  2. 離散選択モデルと最尤法の導入（20分）
  3. 事例：きのこの山 VS たけのこの里（25分）
  4. コード：`mlogit`パッケージを使った推定（30分）
- 後半（15時半から17時）：
  1. コード：パッケージを使わずに最尤推定をプログラミング（30分）
  2. 演習：データを用いた実習（60分）

# 本授業の進め方と注意

- 全体的に密度が高く、カバーする量も多めです。
- **講義中の質問を強く推奨します！！**
- スライドタイトルについて：
  - 「【R分析】」：講義パートではサラッとカバー。後ほど説明。
  - 「【参考】」：計量経済学の知識が必要な点。ひとまず割愛してOK

# 参考文献

- 上武・遠山・若森・渡辺 「実証ビジネスエコノミクス」 連載第1回
  - 本講義の内容は連載第1回を書籍向けに拡充したものとなる。
- Croissant "Estimation of Random Utility Models in R: The mlogit Package" *Journal of Statistical Software*
- Train "Discrete Choice Methods with Simulation"
  - 特にChapter 2,3, and 6
- Adams, Clarke, and Quinn "Microeconometrics and Matlab"
- ダハナ・勝又「Rによるマーケティング・データ分析」第5章

# 構造推定アプローチとは？

# 構造推定(Structural Estimation)とは？

- 構造推定: 経済主体の意思決定モデルを活用した実証分析のアプローチ
- 手順
  - Step 1: 分析したい経済現象に関する経済モデルを構築する。
  - Step 2: 経済モデル内のパラメタを、データを用いて推定する。
  - Step 3: 推定したモデルを用いて、**反実仮想 (counterfactual)**シミュレーションを行う。

# 構造推定の例：価格付け

- 問：企業が自社製品の価格付けを検討している。どうすればよいか？
- Step 1：以下の利潤最大化問題を考えよう。

$$\max_p (p - mc)D(p)$$

- $p$ : 価格、 $mc$ : 限界費用(=単位費用),  $D(p)$ : 需要関数
- Step 2: 需要関数  $D(p)$  と 限界費用  $mc$  を知りたい。
  - 線形な需要関数を考えて、(操作変数法などで)推定する
- $$Q_t = \alpha_0 + \alpha_1 \cdot P_t + \beta \cdot income_t + \epsilon_t$$
  - 限界費用は(1) 企業の内部情報、(2) 計量経済学的に推定する (詳細は割愛)
- Step 3: 利潤を最大にするような価格を探す！ (数値計算)

# 価格付け分析の実例

- 例 1 : Ziprecruiter.com の価格付け実験
  - 米国のオンライン求人サイトが、求人企業の月会費の改定を検討。
  - シカゴ大学の研究者と共同して需要測定のためのA/Bテストを実行。
  - 分析から、価格の最適化により収入を約 60 %改善できることが判明。
  - 実際に、価格を 99 ドルから 249 ドルへ改定。
- 例 2 : バッファローのnasneの価格付け
  - ネットワークレコーダーの新製品に関する価格付けの検討。
  - UTEconが、販売予測モデルの構築に携わる。
  - 販売価格29800円が「スイートスポット」にはまり、売れ行きは好調だった。
- 両事例の詳細については[こちら](#)を参照。

# 構造推定の例：企業合併分析

- ・ 問：自動車市場において、トヨタとホンダが合併を検討している。この合併を許すべきか？
- ・ 論点：
  - (-) 寡占化の進行。競争がなくなって、価格が上がるかも。
  - (+) 新企業の効率化。規模の経済、ノウハウの共有など。
- ・ 重要性：大型の企業合併は公正取引委員会により事前審査が求められる。

# 合併シミュレーション

- 構造推定による分析ステップ：
  1. 企業間の競争モデルを構築する（例：ベルトラン競争）
  2. モデルの要素（需要関数・費用関数）をデータから推定する。
  3. 仮に合併が起きた場合にどのような価格付けになるかをシミュレーションする。
- **合併シミュレーション**と呼ばれ、独禁法実務においても活用されている。
  - 例：公取委「平成28年度における主要な企業結合事例」の「出光興産(株)による昭和シェル石油(株)の株式取得及びJXホールディングス(株)による東燃ゼネラル石油(株)の株式取得」
- 学術研究の事例は[こちら](#)の補足資料
  - Fan (2013)
  - Wollmann (2018)

# その他の例 (私の研究関連)

- 環境政策：排出権取引の政策ルールを変更したら、どのような均衡が実現するか？
  - 参考：[こちらにFowlie, Reguant, and Ryan \(2016\)の解説記事](#)
- 政治経済学：「もし全員が投票」したら、選挙結果はどのように変化するか？
  - 投票行動モデルを構築し、選好と投票コストを推定。そのうえでシミュレーション
  - [Kawai, Toyama, and Watanabe \(2021\) を参照。](#)

# 構造推定の強み・弱み

- 強み：経済モデルを用いたシミュレーションができる。
  - 例：ある政策の代わりに別の政策が行われたらどうなっていたか？
  - 例：意思決定におけるあるメカニズム・チャネルを捨象した場合、どうなるか？
  - 因果効果・政策効果の推定の一手段
  - 経済的に関心があるアウトカム（特に経済厚生）を見ることができる。
- 弱み：経済モデルに強く依拠している。
  - 仮定が強い。
  - 分析が複雑になることが多い（プログラミングなど）
- 補足資料にて、構造推定アプローチと誘導形アプローチの議論についてもう少し詳しく説明。

# 今回の例：離散選択モデル

- 本レクチャーでは、**離散選択モデル**を題材として構造推定を紹介する。
- 理由 1：応用範囲が非常に広い
  - マーケティング：どの財を購入するか？
  - 交通：通勤経路の選択（電車・バス・自家用車）
  - 労働・教育：どの学校に出願するか？いつ引退するか？
  - 政治経済学：誰に投票するか？
  - 産業組織論：市場に参入するか否か（ゲーム理論的モデル）
- 理由 2：モデル・計量分析・プログラミングともに比較的シンプル
- 理由 3：私が一番好きな経済モデルだから。

# これからの流れ

- 離散選択モデルの導入：モデルと推定方法
- アンケートデータに基づく応用事例：きのこの山VSたけのこの里
- パッケージを用いた離散選択モデルの推定
- パッケージを利用しないモデルの推定
  - 尤度関数の定義
  - 数値最適化による尤度関数の最大化

# 離散選択モデル

# 例：あなたはどのiPhoneを買いますか？

iPhone 14      iPhone SE(第3世代)      iPhone 13



ブルー ミッドナイト グリーン

119,800円(税込)から  
購入 さらに詳しく >

62,800円(税込)から  
購入 さらに詳しく >

107,800円(税込)から\*  
購入 さらに詳しく >

iPhone 14      iPhone SE(第3世代)      iPhone 13

**6.1インチ**  
Super Retina XDRディスプレイ<sup>1</sup>

**4.7インチ**  
Retina HDディスプレイ

**6.1インチ**  
Super Retina XDRディスプレイ<sup>1</sup>

SOS 緊急SOS 衝突事故検出

SOS 緊急SOS

SOS 緊急SOS

 先進的なデュアルカメラシステム  
12MPメイン | 超広角

 先進的なカメラシステム  
12MPメイン  
オートフォーカスに対応したTrueDepth  
フロントカメラ

 デュアルカメラシステム  
12MPメイン | 超広角  
TrueDepthフロントカメラ

2倍 の光学ズームレンジ

-

2倍 の光学ズームレンジ

最大20時間のビデオ再生<sup>2</sup>

最大15時間のビデオ再生<sup>3</sup>

最大19時間のビデオ再生<sup>3</sup>

# 離散選択モデル (Discrete Choice Model)

- 离散選択モデル：「**数多くの選択肢の中からどれを選ぶか**」を表現する数理モデル
- 意思決定者は**複数(有限個)の選択肢**に直面している。
  - iPhoneの例：iPhone 13, 14, SE(3世代), 13 Pro, などなど
- それぞれの選択肢(例：製品)には、付随する**特徴**がある
  - iPhoneの例：価格、ディスプレイサイズ、カメラの画素数、バッテリー持続時間、などなど
- 意思決定者は「どの特徴を重視するか」という**好み**をもっている。
  - 例：値段は気にしない（親が払ってくれるから）、大きい画面が良い、などなど
- 選択肢の特徴を踏まえて、**自分の満足度(効用)が最も高くなる選択**をする

# セットアップ<sup>°</sup>

- 消費者  $i = 1, \dots, N$
- 製品・選択肢  $j \in \mathbf{J} = \{1, \dots, J\}$ 
  - $\mathbf{J}$  を選択肢集合(choice set)と呼ぶ。
- 消費者  $i$  が製品  $j$  から得られる効用

$$u_{ij} = \alpha p_j + \beta X_j + \epsilon_{ij}$$

- $p_j$ : 価格
- $X_j$ : 製品の属性・特徴のベクトル。
- $(\alpha, \beta)$ : 選好パラメタ
- $\epsilon_{ij}$ : 個人かつ製品レベルのランダムな選好ショック

# 簡単な例 1 : iPhoneの購入

- 以下の効用関数を考えよう

$$u_{ij} = -2p_{jt} + 3 \times (\text{画面サイズ}) + 0.5 \times (\text{バッテリー時間})$$

- 以下の三機種を比較しよう

モデル	価格(万円)	画面サイズ(インチ)	バッテリー(時間)	効用
iPhone 14	11.98	6.1	20	4.34
iPhone SE	6.28	4.7	15	<b>9.04</b>
iPhone 13	10.78	6.1	19	6.24

- iPhone SEを購入する！！**

## 簡単な例 2：もし価格をあまり気にしないと？

- 価格の係数が小さい (=価格をあまり気にしない)

$$u_{ij} = -1p_{jt} + 3 \times (\text{画面サイズ}) + 0.5 \times (\text{バッテリー時間})$$

- 以下の三機種を比較しよう

モデル	価格(万円)	画面サイズ(インチ)	バッテリー(時間)	効用
iPhone 14	11.98	6.1	20	16.32
iPhone SE	6.28	4.7	15	15.32
iPhone 13	10.78	6.1	19	<b>17.02</b>

- iPhone 13を購入する！！**
- ポイント：人々の「好み」によって、選択は異なってくる！！

# 離散選択問題の定式化

- 以下のようにノーテーションを定義

$$u_{ij} = V_j + \epsilon_{ij}$$

- $V_j = \alpha p_j + \beta X_j.$
- 消費者  $i$  は最も高い効用が得られる製品・選択肢  $j$  を一つ選ぶ。

$$d_i = \arg \max_{j \in \{1, \dots, J\}} u_{ij}$$

# 離散選択問題から選択確率の導出

- ・消費者は効用の要素を把握した上で選択 -> 決定論的選択
- ・しかしながら、分析者には選好ショック  $\{\epsilon_{ij}\}_{j=1}^J$  は観察できない。(いわゆる誤差項)
- ・モデルの予測として、以下の**選択確率**を考える。

$$P(d_i = j) = \Pr \left( \{\epsilon_{ij}\}_{j=1}^J : V_j + \epsilon_{ij} \geq V_k + \epsilon_{ik} \forall k \neq j \right)$$

- 解釈：選択肢  $j$  から得られる効用が最も大きくなる確率
- ・この選択確率は  $\{\epsilon_{ij}\}_{j=1}^J$  に関する分布の仮定による。
  - 例：ロジットモデル、プロビットモデル

# 多項ロジットモデル

- $\epsilon_{i,j}$  が i.i.d. の第一種極値分布(type I extreme value distribution) に従うと仮定。

$$F(x) = \exp(-\exp(-x))$$

- 分散は  $\pi^2/6$ .
- この分布の元で、選択確率は

$$\Pr(d_i = j | \{p_j, X_j\}_{j=1}^J) = \frac{\exp(\alpha p_j + \beta X_j)}{\sum_{k=1}^J \exp(\alpha p_k + \beta X_k)}$$

- **ロジット確率**とも呼ばれる。
- 導出はTrainのChapter 3を参照。

# モデルがあると何が嬉しいか？

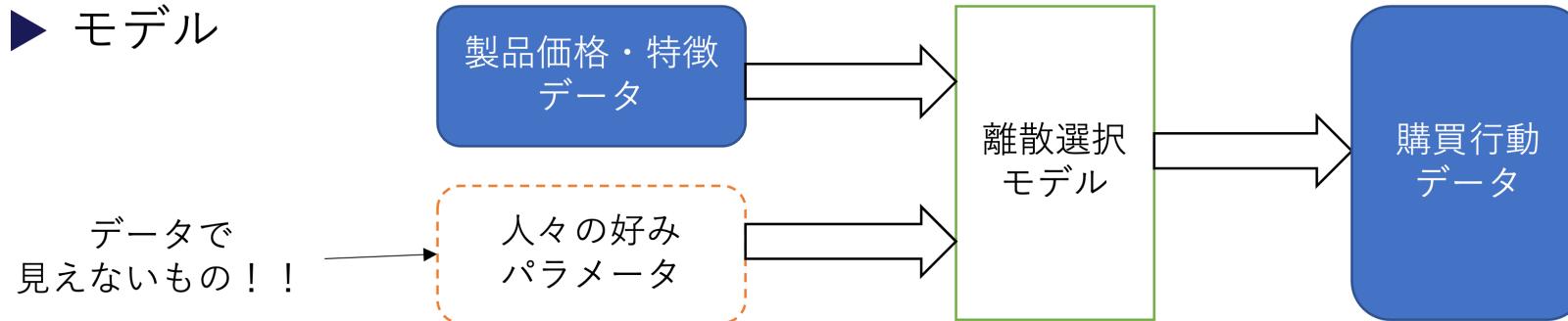
- ロジットモデル再掲

$$\Pr(d_i = j | \{p_j, X_j\}_{j=1}^J) = \frac{\exp(\alpha p_j + \beta X_j)}{\sum_{k=1}^J \exp(\alpha p_k + \beta X_k)}$$

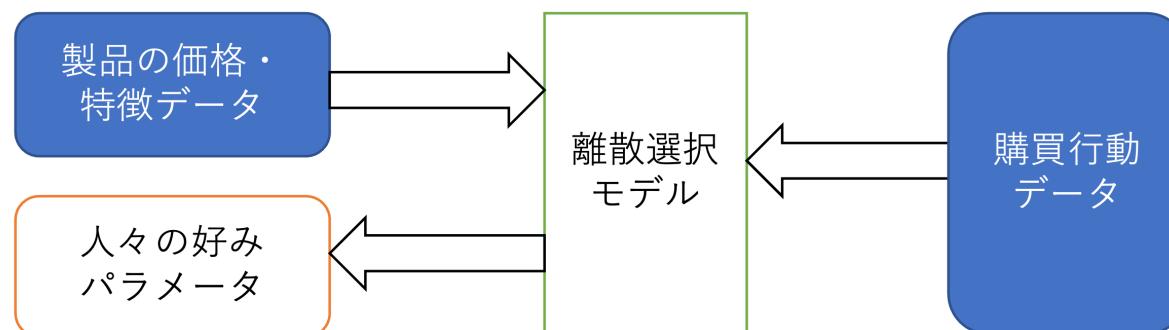
- 例 1：製品の価格  $p_j$  を下げたら需要がどう変化するか？
  - iPhone SEの値下げ -> SEの需要増加
  - 同時に、iPhone 13の需要は減るかも（製品の代替）
- 例 2：新製品を導入したらどうなるか？
  - 例えば、iPhone Max という  $J + 1$  番目の新製品  $\{p_{J+1}, X_{J+1}\}$
  - 新しい製品の需要予測、そして既存の製品への影響を予測できる。

# モデルからデータへ：

## ▶ モデル



## ▶ データ分析：



▶ 「モデルの予測」と「実際の購買行動」が近くなるような「パラメタ」を推定する！！

# 最尤法 (maximum likelihood) による推定

- データ: 各個人  $i$  について  $\{X_j, p_j, d_{ij}\}_{j=1}^J$ 
  - $d_{ij} = 1$  消費者  $i$  が 製品  $j$  を選んだとき 1, それ以外は0
- 尤度関数(likelihood): ある実現値が発生するような確率をパラメタの関数と表したもの。
- 多項ロジットモデルの尤度関数 (パラメタ  $\theta = (\alpha, \beta)$ )

$$L(\theta) = \prod_{i=1}^N \left[ \prod_{j=1}^J (Pr(d_i = j|\theta))^{d_{ij}} \right]$$

- 尤度関数は、「手元にあるデータが、モデルによって生成される確率」と解釈される。
- この確率を最大にするようなパラメタを「良い推定量」とする。
  - 一定の仮定のもとで一致性・漸近正規性・効率性などの理論的性質が担保されている。

# 対数尤度関数の最大化

- 最尤法では、以下の対数尤度を用いる。

$$\log L(\theta) = \sum_{i=1}^N \sum_{j=1}^J d_{ij} \log Pr(d_i = j | \theta)$$

- 対数尤度関数を最大化するようなパラメタ  $\theta$  が最尤推定量となる。
- ポイント：最大化するパラメタは、基本的に数値計算・数値最適化で求める。
  - 非常にシンプルなモデル（例：線形回帰モデル）では解析的に得られる。

**事例：きのこの山 VS たけのこの里**

# きのこの山 VS たけのこの里



# きのこの山 VS たけのこの里

- アンケートデータ&離散選択モデルを用いて、「きのこの山」と「たけのこの里」の需要関数を推定しよう。
- 何が知りたいか?
  - 「きのこの山」と「たけのこの里」はどちらが人気か？
  - 「ブランド価値」と「価格」のトレードオフ -> 需要関数・価格弾力性
  - 価格を変えたときに、需要量・収入はどのように変化するか？

# アンケート(メインクエスチョン)

- あなたはスーパー・マーケットのお菓子コーナーに来てします。目の前に「きのこの山」と「たけのこの里」が並んでいます。「きのこの山」と「たけのこの里」の一箱あたり税込価格が、以下の組み合わせで与えられるとき、あなたにとって最も望ましい選択肢を一つ選んでください。
- 価格の組み合わせ：
  - (きのこの山、たけのこの里) = (200円、200円)
  - (きのこの山、たけのこの里) = (180円、200円)
  - (きのこの山、たけのこの里) = (200円、170円)
  - (きのこの山、たけのこの里) = (220円、200円)
  - (きのこの山、たけのこの里) = (190円、210円)
- 選択肢：
  - きのこの山を買う。
  - たけのこの里を買う。
  - どちらも買わない。

# レクチャーで使うデータ

- 2023年春学期「産業組織論」@早稲田大学政治経済学部で行ったアンケート結果を利用。
- アンケートでは上述の選択肢問題に加えて、消費者属性についての設問も。
- トピックス・レクチャーの実習(後半パート)において、RAブーストキャンプ参加者に回答してもらったアンケートデータを使った演習を行う。

# 結果を見る前に：選択型コンジョイント分析

- 今から見していく分析を一般に、**選択型コンジョイント分析**と呼ぶ。
- 仮想状況における選択に関する質問から、消費者の需要・選好を推定することが可能。
- マーケティング、環境経済学などにおいて幅広く使われている手法。
- 限界点：あくまで仮想の状況であるので「実際の行動」を反映するか否かは要注意。
- マーケティングにおけるコンジョイント分析の解説として、Allenby et al "Economic foundations of conjoint analysis" *Handbook of the Economics of Marketing*

# 【R分析】下準備

```
rm(list = ls())
library("tidyverse")
library("knitr")
library("mlogit") # ロジットモデル推定のためのパッケージ
library("stargazer") # 推定結果の表作成
library("optimx") # 数値最適化のためのパッケージ
```

# 【R分析】データの読み込み

```
data <- readr::read_csv("data/KTSurvey_Spring2023_cleaned.csv")
```

# 変数一覧

変数	説明
ID	回答者のID
experience	きのこの山・たけのこの里を最後にいつ食べたか？
Q1	(きのこの山、たけのこの里) = (200円、200円)
Q2	(きのこの山、たけのこの里) = (180円、200円)
Q3	(きのこの山、たけのこの里) = (200円、170円)
Q4	(きのこの山、たけのこの里) = (220円、200円)
Q5	(きのこの山、たけのこの里) = (190円、210円)
age	年齢
gender	性別
region	出身地
familyhouse	実家暮らしか 一人暮らしか

# 【R分析】アンケートのメインの結果

各問における選択肢のシェアを計算する。

```
# 回答者数
N <- length(data$ID)

# 必要なデータを抽出
data %>%
  select(ID, Q1, Q2, Q3, Q4, Q5) %>%
  gather(key = Q, value = choice, Q1, Q2, Q3, Q4, Q5) -> datafig

# データ整形
datafig %>%
  mutate( Q = ifelse(Q == "Q1", "Q1: (200円, 200円)", Q),
         Q = ifelse(Q == "Q2", "Q2: (180円, 200円)", Q),
         Q = ifelse(Q == "Q3", "Q3: (200円, 170円)", Q),
         Q = ifelse(Q == "Q4", "Q4: (220円, 200円)", Q),
         Q = ifelse(Q == "Q5", "Q5: (190円, 210円)", Q) ) -> datafig
```

# 【R分析】シェアの計算

```
datafig %>%
  group_by(Q, choice) %>%
  tally() %>%
  mutate( n = n/N) %>%
  pivot_wider( id_cols = "Q", names_from = "choice", values_from = "n") %>%
  knitr::kable( digits = 2) -> tab
```

# 選択肢のシェア

tab

Q	1: きのこの山を買う	2: たけのこの里を買う	3: どちらも買わない
Q1: (200円, 200円)	0.30	0.50	0.20
Q2: (180円, 200円)	0.58	0.24	0.17
Q3: (200円, 170円)	0.12	0.76	0.12
Q4: (220円, 200円)	0.09	0.61	0.30
Q5: (190円, 210円)	0.54	0.21	0.26

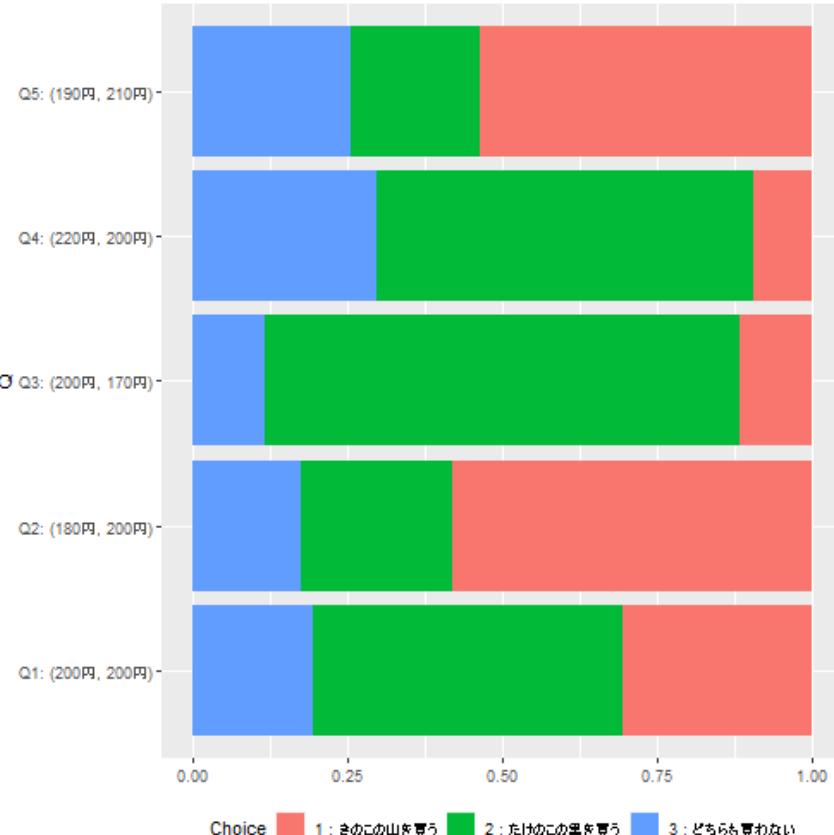
# 【R分析】 グラフで出力

帯グラフで結果を出力する。

```
p <- ggplot() +  
  geom_bar(data = datafig, aes(x = Q, fill = as.factor(choice) ), position = "fill" ) +  
  coord_flip() +  
  theme(legend.position="bottom", axis.title.x=element_blank())+  
  scale_fill_discrete(name = "Choice")
```

```
plot(p)
```

# アンケートのメインの結果



# アンケート結果からのポイント

Q	1: きのこの山を買う	2: たけのこの里を買う	3: どちらも買わない
Q1: (200円, 200円)	0.30	0.50	0.20
Q2: (180円, 200円)	0.58	0.24	0.17
Q3: (200円, 170円)	0.12	0.76	0.12
Q4: (220円, 200円)	0.09	0.61	0.30
Q5: (190円, 210円)	0.54	0.21	0.26

- ポイント1：価格が上がるとシェアが下がる→価格に反応
- ポイント2：価格差が大きくても選ぶ人もいる→きのこ／たけのこのブランド価値
- 離散選択モデルを使って、より詳しく分析してみよう！

# 離散選択モデル：多項ロジットモデル

- 消費者  $i$  が設問  $k$  において、選択肢  $j \in \{Kinoko, Takenoko, outside\}$  から得る効用

$$U_{i,k,Kinoko} = \alpha_{Kinoko} - \beta \cdot p_{k,Kinoko} + \epsilon_{i,k,Kinoko}$$

$$U_{i,k,Takenoko} = \alpha_{Takenoko} - \beta \cdot p_{k,Takenoko} + \epsilon_{i,k,Takenoko}$$

$$U_{i,k,outside} = \epsilon_{i,k,outside}$$

- $\alpha_{Kinoko}, \alpha_{Takenoko}$ : きのこ・たけのこ自体への好み
- $p_{k,j}$  設問  $k$  における選択肢  $j$  の価格
- $\epsilon_{i,k,j}$ : ランダムな選好ショック
- 補足：「何も買わない」オプションを「アウトサイド・グッズ」と呼ぶ。
- 推定するパラメタは  $\alpha_{Kinoko}, \alpha_{Takenoko}, \beta$  の3つ。
  - 全消費者で共通のパラメタとする。後ほど拡張（ランダム係数ロジット）

# 多項ロジットモデルにおける選択確率

- 各選択肢の選択確率は

$$P_k(j|\theta) = \frac{\exp(\alpha_j - \beta \cdot p_{j,k})}{1 + \exp(\alpha_{Kinoko} - \beta \cdot p_{Kinoko,k}) + \exp(\alpha_{Takenoko} - \beta \cdot p_{Takenoko,k})}$$

- 尤度関数は

$$L(\theta) = \prod_{i=1}^N \prod_{k=1}^5 P_k(j=y_{i,k}|\theta)$$

- 対数尤度関数を、パラメタ  $\theta$  に関して最大化する。
  - `mlogit` パッケージを用いる。
  - 自分でプログラムを書く。

# 【R分析】推定：mlogitパッケージ

- mlogitパッケージ：Rにおける離散選択モデル推定のためのパッケージ
- ホームページは[こちら](#)
- 様々な手法をカバー
  - 多項ロジット:[[モデル](#)] [[例](#)]
  - ランダム係数ロジット: [[モデル](#)] [[例](#)]
- (個人的な感想として) パッケージで用いるためのデータ加工が若干独特。
  - データ加工の詳細は[こちら](#)

# 【R分析】 mlogitパッケージのためのデータ加工

まずは推定用にデータを加工する。

```
# きのこ・たけのこを食べたことない人をDropする。
data %>%
  filter( experience != "4 : 食べたことがない") -> data_for_estimation

# データをLong形式に加工する。
# 各行は「回答者ID-設問」という単位になる。
data_for_estimation %>%
  gather(key = "occasion", value = choice, starts_with("Q")) -> data_for_estimation
```

# 【R分析】価格データの作成

- データにマージする、各設問・各選択肢の価格情報を準備する。

```
pricedata <-  
  data.frame( occasion = c("Q1", "Q2", "Q3", "Q4", "Q5") ,  
             price_0 = numeric(5) ,  
             price_1 = c(200, 180, 200, 220, 190) ,  
             price_2 = c(200, 200, 170, 200, 210) )  
pricedata <- as_tibble(pricedata)
```

- 注意点
  - `price_0` という形で、変数名\_選択肢番号 という定義になっている。
  - 0はアウトサイドグッズ、1はきのこ、2はたけのこ
  - 後ほど`mlogit`で利用するため（後述）

# 【R分析】データをマージ

価格データをアンケート結果データに結合する。

```
data_for_estimation %>%
  left_join(pricedata) %>%
  arrange(ID, occasion) -> data_for_estimation
```

# 【R分析】きのこの山・たけのこの里ダミー変数

- ダミー変数を作成する。

```
data_for_estimation %>%
  mutate( Kinoko_0 = 0,
         Kinoko_1 = 1,
         Kinoko_2 = 0,
         Takenoko_0 = 0,
         Takenoko_1 = 0,
         Takenoko_2 = 1    ) %>%
  arrange(ID, occasion) -> data_for_estimation
```

- 1は「きのこ」、2は「たけのこ」、0は「どちらも買わない」なので、kinoko\_1はきのこを買うことから1、takenoko\_2はたけのこを買うことから1となる。

# 【R分析】選択に関する変数

- 各設問での選択に関する変数choiceを再定義する。

```
data_for_estimation %>%  
  mutate( choice = case_when( choice == "1 : きのこの山を買う" ~ 1,  
                             choice == "2 : たけのこの里を買う" ~ 2,  
                             choice == "3 : どちらも買わない" ~ 0 )) -> data_for_estimation
```

# 【R分析】選択の状況に関する変数の定義

- 現在のデータの各行は、各個人が各設問においてどう行動したかを示している。
- 変数choiceidは、このような「選択の状況」を示すIDとなっている。
  - 構築としては、単純に上から下まで連番を振ればよい。

```
data_for_estimation$choiceid <- 1:nrow(data_for_estimation)
```

# 【R分析】 mlogit用のデータセット

- mlogit専用のデータ形式に変更する。

```
datalogit <- dfidx(data = as.data.frame(data_for_estimation),  
                    choice = "choice",  
                    varying = 9:17,  
                    sep = "_",  
                    idx = list(c("choiceid", "ID")),  
                    idnames = c("chid", "alt"),  
                    opposite = c("price"))
```

# 【R分析】各引数の説明

- `data`: データセット。
- `choice`: 各選択状況（各行）における選択を示す変数
- `varying`: データセットにおいて、選択肢ごとに異なる値をとる変数。今回は9-17列目。
- `sep`: 上のvaryingな変数は、`変数名_選択肢`となっており、この変数名と選択肢を分離する記号を指定する。
- `idx`: 各選択状況を示す変数(`choiceid`)と、各個人を示す変数(`ID`)を指定する。
- `idnames`: 新しいデータにおいて、各選択状況を示す変数と選択肢をしめす変数の名前
- `opposite`: 効用関数において符号がマイナスにはいるような変数の指定。今回は`price`.

# 【R分析】新しいデータセットにおけるインデックス情報

```
head(datalogit$idx)
```

```
## ~~~ indexes ~~~~  
##   chid ID alt  
## 1     1  1   0  
## 2     1  1   1  
## 3     1  1   2  
## 4     2  1   0  
## 5     2  1   1  
## 6     2  1   2  
## indexes:  1, 1, 2
```

# 【R分析】多項ロジットモデルの推定

- 多項ロジットモデルの推定を行う。

```
multilogit <- mlogit( formula = choice ~ price + Kinoko + Takenoko | 0,  
                      data = datalogit)
```

- 引数：
  - `formula`: 効用の定式化を指定。今回は、価格、きのこダミー、たけのこダミー。なお、|の後の0はひとまず無視してOK.
  - `data`: 上で作成したデータセット
- `stargazer`で推定結果をレポート

```
stargazer::stargazer(multilogit, type = "text")
```

# 多項ロジットモデルの推定結果

```
##  
## =====  
## Dependent variable:  
## -----  
## choice  
## -----  
## price          0.056***  
##                  (0.004)  
##  
## Kinoko        11.583***  
##                  (0.700)  
##  
## Takenoko     11.950***  
##                  (0.710)  
##  
## -----  
## Observations      1,205  
## Log Likelihood   -1,099.179  
## =====  
## Note:           *p<0.1; **p<0.05; ***p<0.01
```

# 離散選択モデル：ランダム係数ロジットモデル

- もし消費者間で異なった選好パラメタを持っていたらどうなるか？

$$U_{i,k,Kinoko} = \alpha_{i,Kinoko} - \beta_i p_{k,Kinoko} + \epsilon_{i,k,Kinoko}$$

$$U_{i,k,Takenoko} = \alpha_{i,Takenoko} - \beta_i p_{k,Takenoko} + \epsilon_{i,k,Takenoko}$$

$$U_{i,k,outside} = \epsilon_{i,k,other}$$

- ここで、
  - $\alpha_{i,j} \sim N(\theta_j, \sigma_j^2)$  for  $j \in \{ \text{Kinoko}, \text{Takenoko} \}$
  - $\beta_i \sim \log N(\theta_\beta, \sigma_\beta^2)$ .
- これらの分布のパラメタを推定する。
  - 技術的な詳細は補足資料を参照(少し難しめ)

# 【R分析】ランダム係数ロジットモデルの推定

```
rcdclogit <- mlogit(choice ~ price + Kinoko + Takenoko | 0,  
                      data = datalogit,  
                      panel = TRUE,  
                      rpar = c(price = "ln", Kinoko = "n", Takenoko = "n") ,  
                      R = 50,  
                      correlation = FALSE)
```

- 引数：
  - `panel`: 今回は各消費者について複数回の選択を観察しているのでパネルデータ構造となっている。
  - `rpar`: ランダム係数の定式化。`ln`は対数正規分布、`n`は正規分布
  - `R`: モンテカルロ積分における乱数のドロー数。多いほうが望ましいが、計算時間の関係上ひとまず50としておく。(要追加説明)
  - `correlation`: ランダム係数の間の相関構造。多くの場合は独立(つまり `FALSE`)

```
stargazer::stargazer(multilogit, rcdclogit, type="text")
```

# 推定結果(ロジットとランダム係数ロジット)

```
##  
## =====  
## choice  
## (1) (2)  
## -----  
## price 0.056*** -1.832***  
## (0.004) (0.072)  
##  
## Kinoko 11.583*** 33.590***  
## (0.700) (2.388)  
##  
## Takenoko 11.950*** 34.474***  
## (0.710) (2.406)  
##  
## sd.price 0.131***  
## (0.010)  
##  
## sd.Kinoko 3.096***  
## (0.337)  
##  
## sd.Takenoko 3.629***  
## (0.370)  
##
```

# 推定したランダム係数の分布

- まず、 $\alpha_{i,Kinoko} - \alpha_{i,Takenoko}$  という選好パラメタの差をプロットする。
- 平均及び分散は、

$$E[\alpha_{i,Kinoko} - \alpha_{i,Takenoko}] = E[\alpha_{i,Kinoko}] - E[\alpha_{i,Takenoko}]$$

$$\text{Var}[\alpha_{i,Kinoko} - \alpha_{i,Takenoko}] = \text{Var}[\alpha_{i,Kinoko}] + \text{Var}[\alpha_{i,Takenoko}]$$

ここで、 $\alpha_{i,Kinoko}$  と  $\alpha_{i,Takenoko}$  は独立と仮定している。

# 【R分析】ランダム係数の分布

- `mlogit::rpar`関数を使うと、ランダム係数の分布パラメタを取得できる。

```
dist_Kinoko = rpar(rcdclogit, 'Kinoko')
dist_Takenoko = rpar(rcdclogit, 'Takenoko')
dist_price = rpar(rcdclogit, 'price')
```

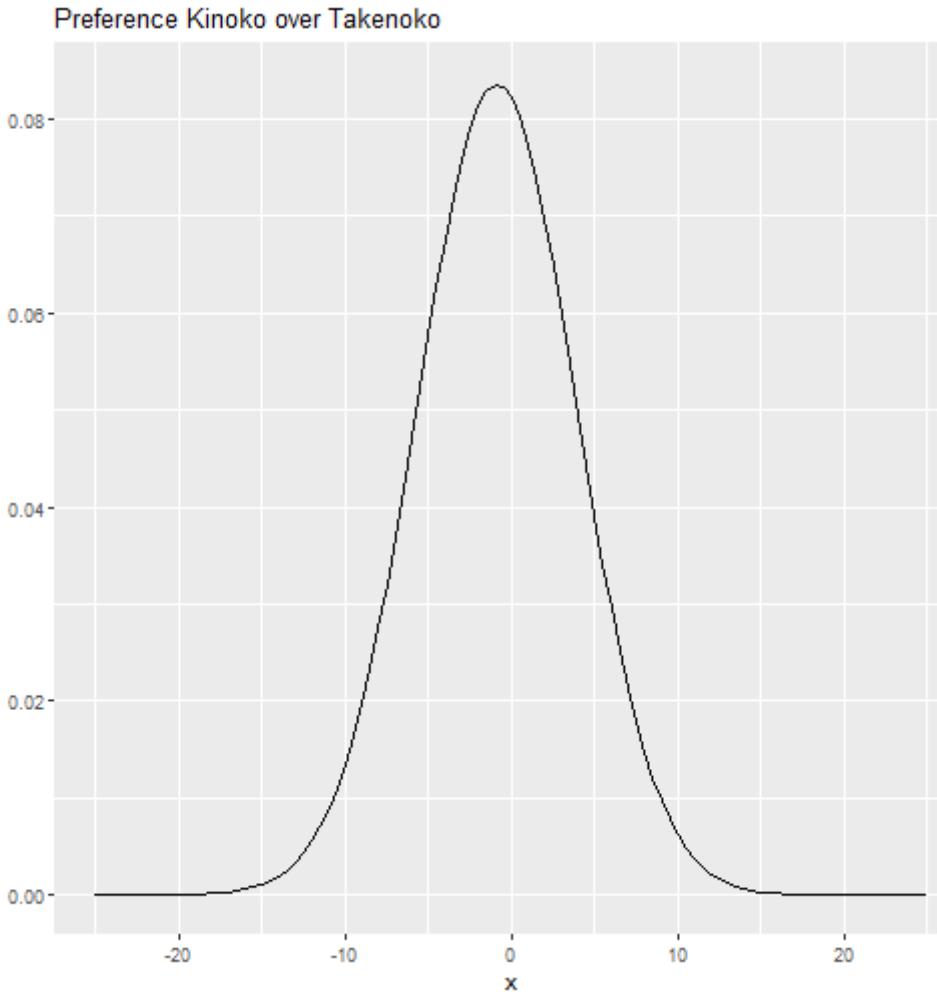
# 【R分析】ランダム係数の分布

- 推定値から、 $\beta_{i,Kinoko} - \beta_{i,Takenoko}$  の分布プロットを作成する。

```
p1 <- ggplot(data = data.frame(x = c(-25, 25)), aes(x)) +
  stat_function(fun = dnorm, n = 101,
                args = list(mean = dist_Kinoko$mean -dist_Takenoko$mean ,
                            sd = sqrt(dist_Kinoko$sigma^2 + dist_Takenoko$sigma^2) ) ) +
  ylab("") +
  ggtitle("Preference Kinoko over Takenoko")
```

```
plot(p1)
```

# ランダム係数の分布： $\beta_{i,Kinoko} - \beta_{i,Takenoko}$



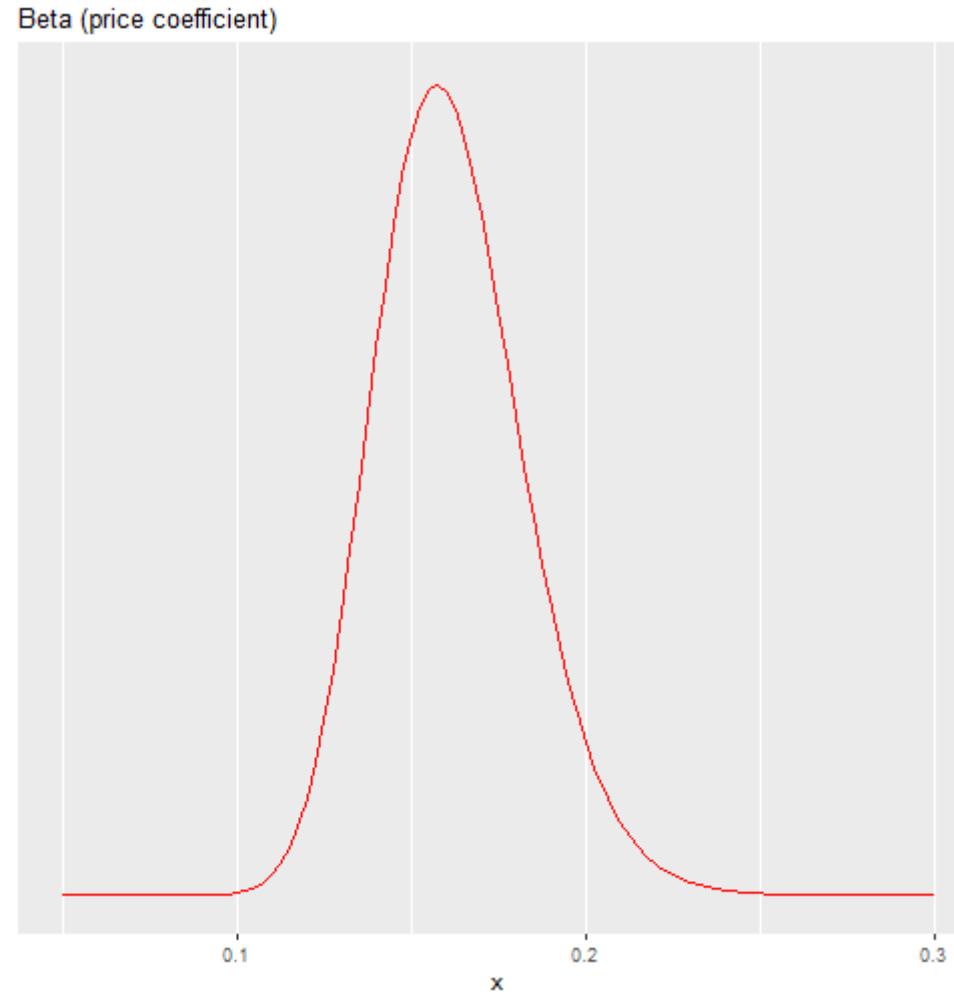
# 【R分析】価格係数の分布

- 価格係数  $\beta_i \sim \log N(\theta_\beta, \sigma_\beta^2)$  についてプロットする。

```
p2 <- ggplot(data = data.frame(x = c(0.05, 0.3) ), aes(x)) +
  stat_function(fun = dlnorm, n = 101,
                args = list(mean = dist_price$mean, sd = dist_price$sigma), colour = "red") +
  ylab("") +
  scale_y_continuous(breaks = NULL) + ggtitle("Beta (price coefficient)")
```

```
plot(p2)
```

# 価格係数の分布



# 需要曲線のプロット

- ランダム係数ロジットモデルの推定結果にもとづいて、需要曲線をプロットする。
- 解釈：推定しているものは消費者の選択確率のモデル。しかしながら、選択確率 = 市場シェアと解釈可能。
- 選択確率(市場シェア)に潜在的な消費者数をかけ合わせると、市場全体の需要が得られる。

# 設定

- 1000人の消費者がいると考える。
- 各消費者の選好パラメタは推定された分布に従っており、ランダムに決まっている。
- 各消費者の購買確率を計算し、それを足し合わせることで、全体の需要を予測する。

# 【R分析】下準備

- 推定した分布パラメタにもとづいて、個々の消費者のパラメタをドローする。

```
# 以下では乱数を用いるので乱数のシードを固定
set.seed(101)

# 消費者数
R = 1000

# 消費者のパラメタを、乱数によって発生させる。
alpha_Kinoko_vec = rnorm(n = R, mean = dist_Kinoko$mean, sd = abs(dist_Kinoko$sigma) )
alpha_Takenoko_vec = rnorm(n = R, mean = dist_Takenoko$mean, sd = abs(dist_Takenoko$sigma) )
beta_vec = rlnorm(n = R, mean = dist_price$mean, sd = abs(dist_price$sigma) )
```

# 乱数について

- 今、1000人の消費者が異なったパラメタをもっており、そのパラメタは推定した分布に従う。
- 1000人のパラメタについて、その分布に従う**乱数**として発生させる。
  - `rnorm` と `rlnorm` が正規乱数・対数正規乱数
- 【重要】乱数を発生させる際には、乱数のシードを固定する。
  - `set.seed(101)` に相当する。
  - これをしないと、分析を回す度に異なった乱数が生成される。(再現性の観点から重要)

# 【R分析】ロジット確率計算のための関数

- パラメタ  $\beta, \alpha_{Kinoko}, \alpha_{Takenoko}$  と価格を与えることで、選択確率を計算する。

```
f_logit_prob <- function(alpha_Kinoko, alpha_Takenoko, beta, price_Kinoko, price_Takenoko){  
  util_Kinoko <- alpha_Kinoko - beta*price_Kinoko  
  util_Takenoko <- alpha_Takenoko - beta*price_Takenoko  
  
  prob_Kinoko <- exp( util_Kinoko ) / ( 1 + exp( util_Kinoko ) + exp( util_Takenoko ) )  
  prob_Takenoko <- exp( util_Takenoko ) / ( 1 + exp( util_Kinoko ) + exp( util_Takenoko ) )  
  prob_Other <- 1 - (prob_Kinoko + prob_Takenoko)  
  
  return( cbind(prob_Kinoko, prob_Takenoko, prob_Other))  
}
```

# 【R分析】需要関数

```
f_demand <- function( alpha_Kinoko_vec, alpha_Takenoko_vec, beta_vec, price_Kinoko, price_Takenoko )  
  R = length(alpha_Kinoko_vec) # Number of consumers  
  
  # 結果を保存するベクトルを事前に準備  
  prob_Kinoko = numeric(R)  
  prob_Takenoko = numeric(R)  
  prob_Other = numeric(R)  
  
  for (r in 1:R){  
    result = f_logit_prob( alpha_Kinoko_vec[r], alpha_Takenoko_vec[r], beta_vec[r], price_Kinoko,  
    prob_Kinoko[r] = result[1]  
    prob_Takenoko[r] = result[2]  
    prob_Other[r] = result[3]  
  }  
  
  # 選択確率をすべての消費者について足し合わせて、各オプションの需要を得る。  
  return( c(sum(prob_Kinoko), sum(prob_Takenoko), sum(prob_Other)))  
}
```

# 【R分析】きのこの需要関数

- たけのこの価格を200円に固定したときの、きのこの需要関数を求めてみる。

```
price_Takenoko = 200

# きのこの価格を100円から250円まで動かす。
price_vec = seq(from = 100, to = 250, by = 5)
kinoko_vec = numeric(length(price_vec))

# ループで需要を計算
for ( i in 1:length(price_vec)){
  result <- f_demand( alpha_Kinoko_vec, alpha_Takenoko_vec,
                      beta_vec, price_vec[i], price_Takenoko = 200 )
  kinoko_vec[i] <- result[1]
}

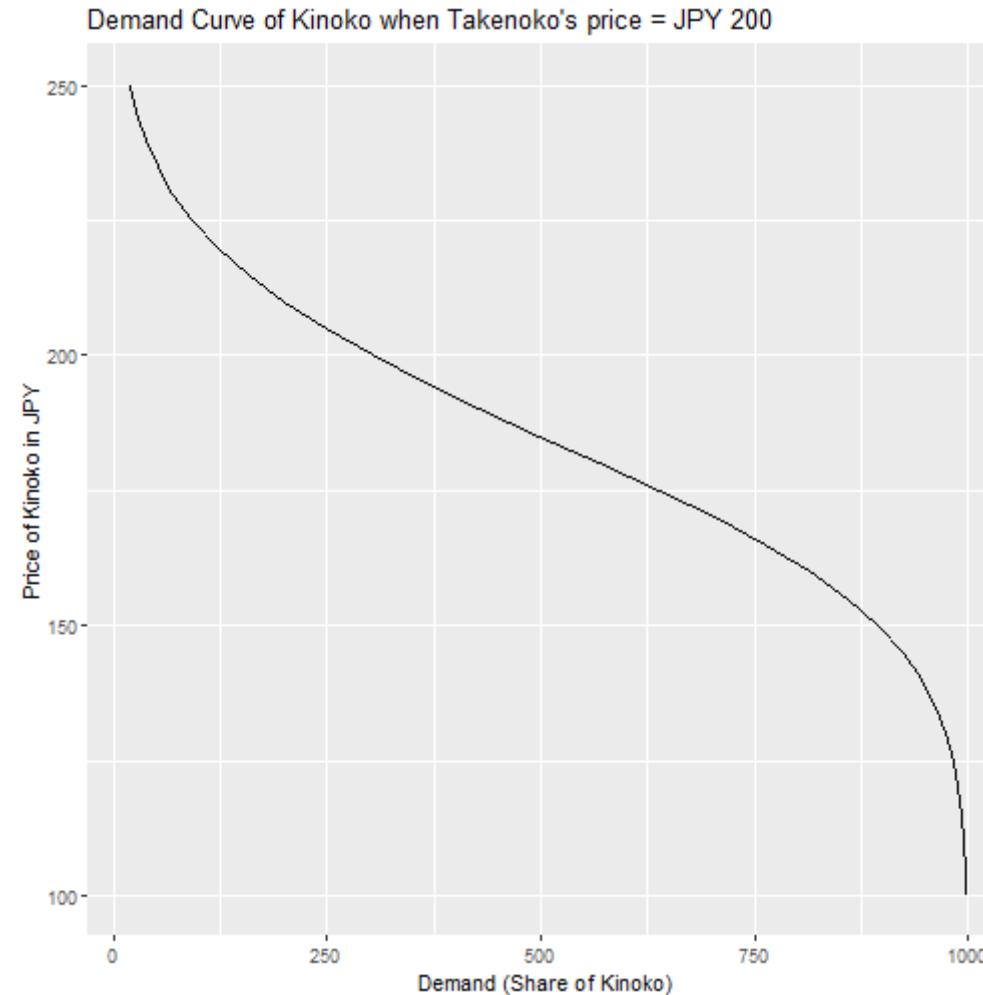
# データフレームに結果を保存
data_demand_kinoko = tibble( price = price_vec,
                             demand = kinoko_vec,
                             revenue = price_vec*kinoko_vec)
```

# 【R分析】プロット

```
fig_demand <- ggplot(data = data_demand_kinoko, aes(x = demand, y = price) ) +  
  geom_line() +  
  xlab("Demand (Share of Kinoko)") +  
  ylab("Price of Kinoko in JPY") +  
  ggtitle("Demand Curve of Kinoko when Takenoko's price = JPY 200")
```

```
plot(fig_demand)
```

# 推定された需要関数

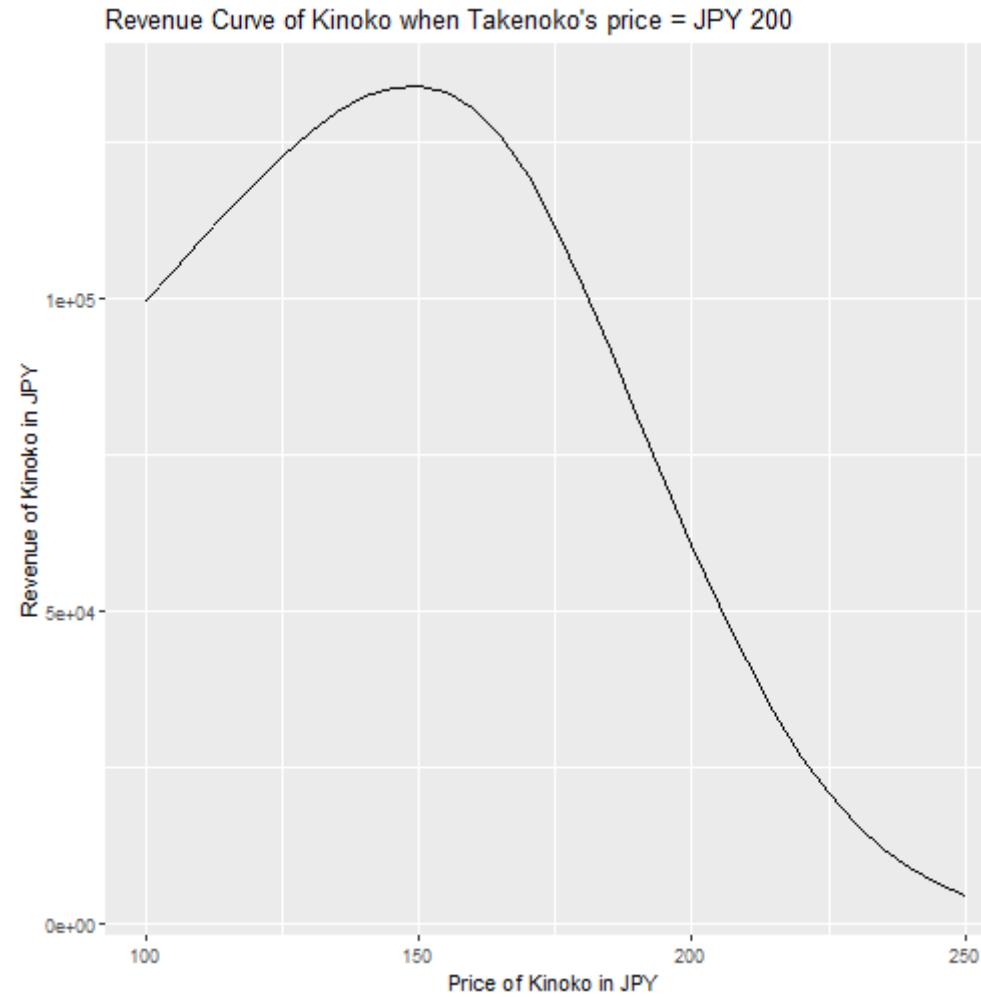


# 【R分析】 収入関数のプロット

```
fig_rev <- ggplot(data = data_demand_kinoko, aes(x = price, y = revenue) ) +  
  geom_line() +  
  xlab("Price of Kinoko in JPY") +  
  ylab("Revenue of Kinoko in JPY") +  
  ggtitle("Revenue Curve of Kinoko when Takenoko's price = JPY 200")
```

```
plot(fig_rev)
```

# 推定された収入関数



# ひとまずのまとめ1：データの種類と限界点

- 今回用いたサーベイデータを表明選好データ(Stated preference)と呼ぶ。
  - 仮想のアンケートを用いて、人々の選好に関する情報を収集している。
- 一方、現実の購買行動データを、顯示選好データ(Revealed preference)と呼ぶ。
  - POSデータ、購買履歴データ、など。
- 表示選好データにおいては、常に「その行動が真の行動を反映しているのか？」を考えなければならない。
- 同時に、サーベイ一般の問題として、サーベイ対象者が興味ある母集団全体を反映しているのかについても要検討。

## ひとまずのまとめ2：さらなる分析

- 今回のアンケートでは消費者属性についても質問している。属性によってきのこ・たけのこへの選好が異なるのでは？
- 後半の実習パートで見ていこう。

# フルスクラッチでモデル推定

# フルスクラッチで推定コードをプログラミング

- 自分自身で推定・シミュレーションのコードをプログラミングする状況が多々ある。
  - むしろパッケージで実行可能な構造推定手法は限定的
- 以下では、多項ロジットモデルの推定のコードについて、ゼロから書いていこう。
- 手順：
  - Step 1: 多項ロジットモデルの尤度関数を定義する。
  - Step 2: 最尤法による推定：定義した尤度関数について数値最適化を行う。-> 点推定値
  - Step 3: 標準誤差の計算

# 多項ロジットモデルの尤度関数(再掲)

- 尤度関数は

$$L(\theta) = \prod_{i=1}^N \prod_{k=1}^5 P_k(j = y_{i,k} | \theta)$$

- ここで

$$P_k(j|\theta) = \frac{\exp(\alpha_j - \beta \cdot p_{j,k})}{1 + \exp(\alpha_{Kinoko} - \beta \cdot p_{Kinoko,k}) + \exp(\alpha_{Takenoko} - \beta \cdot p_{Takenoko,k})}$$

- パラメタは  $\theta = (\beta, \alpha_{Kinoko}, \alpha_{Takenoko})$

# 対数尤度(再掲)

- 対数尤度関数を用いる

$$\log L(\theta) = \sum_{i=1}^N \sum_{k=1}^5 \log P_k(j = y_{i,k} | \theta)$$

- ポイント：この対数尤度を最大化するような値は解析的には得られない。数値計算の出番！！

# 【R分析】Step 1: 対数尤度関数の定義

- 関数f\_logit\_probは上で定義済み。

```
f_likelihood_logit <- function(param, data){  
  alpha_Kinoko <- param[1]  
  alpha_Takenoko <- param[2]  
  beta    <- param[3]  
  
  P1 <- f_logit_prob(alpha_Kinoko, alpha_Takenoko, beta, 200, 200)  
  P2 <- f_logit_prob(alpha_Kinoko, alpha_Takenoko, beta, 180, 200)  
  P3 <- f_logit_prob(alpha_Kinoko, alpha_Takenoko, beta, 200, 170)  
  P4 <- f_logit_prob(alpha_Kinoko, alpha_Takenoko, beta, 220, 200)  
  P5 <- f_logit_prob(alpha_Kinoko, alpha_Takenoko, beta, 190, 210)
```

# 【R分析】 続き

```
pred <- rbind(P1, P2, P3, P4, P5)
pred <- as_tibble(pred)
pred$occasion <- c("Q1", "Q2", "Q3", "Q4", "Q5")

data %>%
  left_join(pred, by = "occasion") %>%
  mutate( choice_prob = case_when( choice == 0 ~ prob_0ther,
                                    choice == 1 ~ prob_Kinoko,
                                    choice == 2 ~ prob_Takenoko) ) %>%
  mutate(log_choice_prob = log(choice_prob)) %>%
  select(log_choice_prob) -> result

likelihood <- sum(result)

return(likelihood)

}
```

## 【R分析】Step 2: 対数尤度関数の最大化

- 定義した対数尤度関数を最大化するようなパラメタを求めたい。
- どうやるか？ -> 数値最適化
- まずは数値最適化について簡単に説明する。

# 数値最適化

- 定義した関数について、その関数を最大にするような変数を求める作業。
- 簡単な例： $f(x, a) = (x - a)^2$ .  $a$  はパラメタ。

```
# 関数の定義  
fx <- function(x,a){ (x-a)^2 }
```

- 当然ながら、 $x = a$  で最小化される。
- 数値最適化を行って確かめてみよう。

# optimize関数による数値最適化

```
# 関数を最小化する。  
xmin <- optimize( f = fx,  
                    interval = c(-10,10),  
                    a = 2.4)  
print(xmin)
```

```
## $minimum  
## [1] 2.4  
##  
## $objective  
## [1] 0
```

- `f`: 最小化したい、自分で定義した関数。なお、`optimize`は1変数のみ。(多変数は後述)
- `interval`: 変数を探索する範囲。
- `a`: 自分で定義した関数に追加の引数が必要な場合は指定する。

# 数値最適化の一般論

- Rのデフォルトは`optimize`と`optim`
  - `optimize`は一変数の最適化
  - `optim`は多変数の最適化
- その他にも様々なパッケージ：`optimx`
- 数値最適化のややこしい点：様々なオプションを設定しなければならない。
  - 最適化の方法：微分を使うか否か
  - 初期値の選び方
  - 最適化停止の基準(tolerance): 目的関数や微分の値がどの程度動かなくなったら停止するか？

# 最適化の方法

- 微分を使わない方法：Nelder-Mead、シンプレックス法
- 微分を使う方法：BFGS (quasi-Newton法)
- また、微分を使う方法の場合も、いくつかのVariation
  - 自分で微分を計算して関数として与える
  - 数値微分を内部で計算してもらう。
- 関数が滑らか（微分可能）で、微分計算ができるならば、微分を使う方法の方が早い。
- 関数が滑らかかわからない場合は、微分を使わない方法を使う。ただし、計算時間がかかる。

# 初期値の選び方

- 初期値：どの値から関数の評価を始めるか。
- 非常に重要。
  - 極端な値を選ぶと、関数の内部で変なことが発生し、最適化が進まない。(例：exp関数)
  - 初期値によっては、**大域的最適点**ではなく、**局所最適点**に止まるかも。
- 実践においては
  - 初期値をうまく選ぶ(後述)
  - いろいろな初期値を試して、結果が大きく変わらないことを確かめる&目的関数が小さくなる値を選ぶ。

# 最適化停止の基準(tolerance)

- 目的関数や微分の値がどの程度動かなくなったら停止するか？
- 基準をゆるくすると、計算時間は早いが、得られる値は不正確かもしれない。
- 関数における引数やアウトプットのスケールとも関連する。

# 数値最適化の参考資料

- 数値最適化は非常に広い
  - 今回扱っているのは、制約なしの非線形最適化
  - その他：線形計画、制約つき最適化、非線形方程式、などなど
- 離散選択モデルにおける最適化については、TrainのChapter 8がよくまとめている。
- 経済学一般におけるガイドとしては、Miranda and Fackler "Applied Computational Economics and Finance" ただしMatlabベース。
- Rにおける数値最適化のガイドとして Optimization with R -Tips and Tricks

# 【R分析】Step 2: 対数尤度関数の最大化

- `optimx`パッケージを用いて、対数尤度関数の最大化を行う。
  - 後ほど標準誤差計算に必要なヘシアンの計算のため。

```
# 初期パラメタ
ini <- c(5,5, -0.01)

# 最適化
result <- optimx( par = ini,
                    fn = f_likelihood_logit,
                    method="Nelder-Mead",
                    data = data_for_estimation,
                    control = list(fnscale=-1),
                    hessian = TRUE )

# 推定値の取得
est_vec <- as.numeric(result[1,1:3])
```

# 【R分析】 optimxの引数

- `par`: 初期値
- `fn`: 最適化したい関数
- `method`: 方法。ここではNelder – Mead (シンプレックス法)
- `data`: 定義した`f_likelihood_logit`で必要な引数
- `control`: 最大化なので、`fnscale = -1`としている。
- `hessian`: 数値計算で得られるヘシアン。後ほど。

# どうやって初期値を選んだのか？

- 身も蓋もない説明としては：
  - `mlogit`すでに推定値がわかっている -> パラメタのスケールの当たりがついている。
  - 多項ロジットモデルの対数尤度関数は、**大域的に凹関数** -> 初期値はそこまで重要でない。
- 初期値の選び方に関する実践(個人的経験に基づく)
  - パラメタを適当にいれて、対数尤度関数を計算し、変な値にならないことを確認する。
  - グリッドサーチをする。
  - 多次元関数における一部変数を固定して、一次元でプロットしてみる。

# 実践 1：極端な値を入れない。

- 関数の中に `exp` が入っているときや、分母に変数が入る場合は気をつける。`Inf` が生じて、内部の計算がおかしくなる。

```
exp(2000)
```

```
## [1] Inf
```

```
1/0
```

```
## [1] Inf
```

- ロジットモデルにおいて初期値のパラメタを極端にすると、尤度が計算できない。
  - 例えば、 $\beta_{Kinoko} = 10000$  を初期値にしたとき、 $\exp()$ 関数は発散し、確率は 0 / 1 に張り付く。
  - 尤度関数の中に  $\log(P)$  があるため、尤度が  $-\infty$  になる。
- 選んだ初期値において、最適化したい関数の中身がちゃんと計算されているか（例：確率、尤度）を確認する！！

## 実践 2：グリッドサーチ

- 最適化の変数が 3 つであれば、それぞれ 10 通りの取りうる値を選び、合計 1000 通りの組み合わせについて、目的関数を評価する。
- その結果、最も目的関数が大きく（小さく）なる値を初期値として選んで、数値最適化する。
- 注意点：変数が多くなると大変。いわゆる **次元の呪い**

# 実践 3：一次元（二次元）プロット

- 一次元・二次元ならば、図としてプロット可能。
- 一部のパラメタを（適切に）止めた上で、他のパラメタについて動かしてみる。
- ある程度の当たりをつけた上で、数値最適化する。

# 【参考】 数値最適化・シミュレーションの個人的コツ

- 目的関数の評価時間についてプロファイラーで計算する。
- モンテカルロ実験
  - 真のパラメタを知っている元でデータを生成。
  - そのデータを使って推定を行い、元のパラメタを復元できるか？
  - デバッグの手段としても有効。
- 並列計算の活用
  - いろいろな初期値をたくさん試す場合に有効

## 【参考】個人的コツ続き

- モデルの挙動を把握する(比較静学)
  - 経済学的思考をバグ取りに活用する。数値計算の結果が経済的な直感と合うのか？？
- パラメタの識別を考える。
- パラメタにReasonableな制約を入れる。
  - パラメタが変な値をとったときに、モデルの挙動がおかしくなるのを防ぐ。
- 最適化する変数のスケールをなるべく合わせる。
  - 最適化のToleranceとも関係。

## 【参考】標準誤差の計算

- 最尤推定量の標準誤差の計算。
- 漸近正規性

$$\sqrt{n} \left( \hat{\theta} - \theta_0 \right) \xrightarrow{d} N(0, V)$$

- ここで、

$$A = -E \left[ \frac{\partial^2 \log f(y_i; \theta)}{\partial \theta \partial \theta'} \Bigg|_{\theta=\theta_0} \right]$$

$$V = A^{-1}$$

- この結果にもとづいて、漸近分散の計算を以下のように行うことができる。
- まず、漸近分散の推定として、

$$\hat{A} = -\frac{1}{n} \sum_{i=1}^N \frac{\partial^2}{\partial \theta \partial \theta} \log f(y_i, \theta) = -\frac{\partial^2}{\partial \theta \partial \theta} \frac{1}{N} l(\theta)$$

- これは目的関数(をサンプルサイズで割った値)のヘシアンに相当する。なお、目的関数（対数尤度）のヘシアンは、数値計算などにおいて付随的に計算されることが多い。
- よって漸近分散の推定値  $ase(\hat{\theta})$  は、

$$ase(\hat{\theta}) = \sqrt{diag\left(\frac{1}{n} \hat{A}^{-1}\right)}$$

として与えられる。ここで、 $diag(X)$ は行列Xの対角要素のみを抜き出したベクトルである。

# 【R分析】Step 3: 標準誤差の計算

- 以上を踏まえて、目的関数のヘシアンを数値的に計算すれば良い。
- `optimx`には、`gHgen`というヘシアンを数値計算する関数が存在する。

```
Hessian <- gHgen(par = as.numeric(result[2,1:3]),  
                  fn = f_likelihood_logit,  
                  data = data_for_estimation)  
se_vec <- round(sqrt(diag(solve(-Hessian$Hn))), 3)
```

# 推定結果のまとめ

```
print( rbind(est_vec, se_vec) )  
  
## [,1] [,2] [,3]  
## est_vec 11.57884 11.94554 0.05637882  
## se_vec NA NA NA
```