マーケティング・データ分析の学習メモ

Table of contents

第 1 章

はじめに

このノートは、神戸大学の田頭拓己先生が書かれた「マーケティング・データ分析」に関する学習内容をまとめたものです。統計学の基礎を学習し，その知識を使ってマーケティングについて各自が持っている問題意識や，抱えている課題について深く理解し，解決するための糸口を見つける力を養う，という点で非常に優れたテキストです。

1.1 準備

各自の PC(Windows か Mac を想定) の好きな場所に `Marketing_data_analysis` というフォルダを作成し，そこを作業ディレクトリにしていることを想定しています。この文章の意味が分からない人は，R の入門書を確認してください。作業ディレクトリの中に `data` というフォルダを作成し，そこにテキストで使うデータファイルを保存してある，という前提で進めます。

第 I 部

第 1 部

第 1 章 R と Rstudio に慣れる

R と R を便利に使うための統合環境である Rstudio の基本的な使い方を学びます。R 初学者はしっかり読んで準備しておいてください。ただ、この章の内容は，数多く出版されている R の解説書で代替できるため，割愛します。

また，この本ではコードを書く環境として Rstudio を想定していますが，個人的には Microsoft 社が開発している Visual Studio Code (VSCode) や Posit 社が開発している Positron を推奨しているので，この章の内容には触れません。

第 2 章リサーチ・デザインと問いの設計

経営学部生がビジネスに関する問いを立てる際に重要なポイントを解説しています。

必読です。

ただ R のコードは出てこないので、ここでは解説しません。各自で熟読しておきましょう。

第 3 章尺度と質問紙調査設計

マーケティング研究でよく利用されている質問紙を使ったアンケート調査の設計方法について解説しています。いろんなところで利用されているアンケートですが、設計段階でいろんな人の思いが錯綜し、本来知りたかった事を調査できていないことが多々あります。

第 4 章は「データ整理と要約」というテーマで、R を使ってデータの整理や要約を行う方法について解説しています。データ分析の前にデータを適切に整理し、要約することは非常に重要です。この章では、R の基本的なデータ操作や要約統計量の計算方法について学びます。

第 2 章

データ整理と要約

第 4 章は、データを読み込んで、形を整えて、特徴をつかむためのデータを要約する方法を学びます。R は最初から用意されている基本関数だけでも十分強力ですが、外部で開発された様々なパッケージを使うと、さらに便利にデータ操作ができます。

はじめに外部のパッケージをインストールして R の機能を拡張します。R でデータを操作する際に超強力なパッケージ群である `tidyverse` を読み込みます。あとで使う Excel ファイルを読み込むための `readxl` パッケージも一緒に読み込みます。

テキストでは、`install.packages()` 関数でパッケージをインストールして、`library()` 関数で読み込んでいますが、ここでは pacman‘パッケージを使って、必要なパッケージを一括でインストールおよび読み込みをします。

まず始めに、`pacman` パッケージをインストールします。この作業はここで 1 回だけです。以降はこのコードは実行しなくて大丈夫です。

```
# first time only
install.packages("pacman")
```

次に、`pacman` パッケージの `p_load()` 関数を使って，`tidyverse` と `readxl` パッケージをインストールおよび読み込みします。`::` という記法はパッケージ名を明示的に指定して関数を使う方法ですが、ここでは `pacman` パッケージの `p_load()` 関数を指定して使っています。

```
pacman::p_load(tidyverse, readxl, gt, gtExtras)
```

一度 `pacman` パッケージをインストールすれば、あとは `pacman::p_load()` 関数を使うだけで、指定したパッケージがインストールされていなければ自動的にインストールし、読み込みまで行ってくれます。非常に便利です。

作業ディレクトリ内の `data` フォルダに入っている `2022idpos.csv` というデータを読み込みます。何を読み込んだのか確認するために，基本関数 `head()` でデータの先頭 6 行を表示させます。

```
idpos <- readr::read_csv("data/2022idpos.csv", na = ".")
idpos |> head()
```

```
# A tibble: 6 x 4
     id date       spent coupon
```

```
  <dbl> <chr>     <dbl>  <dbl>
1    12 2019/9/25 14326      1
2    32 2019/9/10 10232      1
3    30 2019/9/9   6881      1
4    29 2019/9/4   6365      0
5    46 2019/9/10  7595      1
6    44 2019/9/14  7858      0
```

読み込んだデータに含まれている変数の名前を確認するため，基本関数 names() を使います。

readr パッケージの read_csv() 関数は，CSV 形式のデータを読み込むための関数です。na = "."という引数は，データ内の . を欠損値として扱うことを指定しています。

```
idpos |> names()
```

```
[1] "id"     "date"   "spent"  "coupon"
```

データの詳細を確認するため，tidyverse の dplyr パッケージの glimpse() 関数を使います。

```
idpos |> glimpse()
```

```
Rows: 3,000
Columns: 4
$ id     <dbl> 12, 32, 30, 29, 46, 44, 44, 32, 3, 34, 36, 3, 42, 18, 38, 4, 19~
$ date   <chr> "2019/9/25", "2019/9/10", "2019/9/9", "2019/9/4", "2019/9/10", ~
$ spent  <dbl> 14326, 10232, 6881, 6365, 7595, 7858, 9405, 1821, 8375, 1828, 6~
$ coupon <dbl> 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, ~
```

Excel のようにデータを表示させたいなら，基本関数 View() を使います。

```
View(idpos)
```

これで，自分が読み込んだデータがどんなものかを確認できました。

2.0.1　dplyr でデータの要約

tidyverse の dplyr パッケージを使うと，データの要約が簡単にできます。ここでは，

- summarise() 関数：データの要約統計量を計算する
- mutate() 関数：新しい変数を作成する
- filter() 関数：条件に合う行を抽出する
- select() 関数：特定の変数を抽出する
- arrange() 関数：データを並び替える
- rename() 関数：変数の名前を変更する

の使い方を学びます。

先ほど読み込んだデータには，

- id：顧客 ID

- date：購入日
- spent：購入金額
- coupon：クーポン利用の有無 (1=利用, 0=未利用)

という 4 変数に 3,000 の観測値が含まれています。このデータを dplyr の各関数を使って，以下のような操作をしてみます。

まず，

1. select() 関数で spent と coupon を抽出し，
2. arrange() 関数で spent の降順に並び替え，
3. mutate() 関数で spent の金額を 10 倍した spent10 という新しい変数を作成し，
4. filter() 関数で spent10 が 10000 以上の行を抽出し，
5. summarise() 関数で spent10 の平均を計算し，
6. rename() 関数で spent10 を spent_times10 に名前変更する，

という一連の操作を行います (この処理に意味はないです。)

```
# コード 4-11（見本コード）
idpos |>
  select(spent, coupon) |>
  arrange(desc(spent)) |>
  mutate(spent10 = spent * 10) |>
  filter(spent10 >= 10000) |>
  summarise(mean_spent10 = mean(spent10)) |>
  rename(spent_times10 = mean_spent10)
```

```
# A tibble: 1 x 1
  spent_times10
          <dbl>
1         82391.
```

すると，spent を 10 倍して，1 万円以上の買い物の平均値を計算した結果が spent_times10 として表示されます。

このように加工したデータを csv ファイルとして保存するには、readr パッケージの write_csv() 関数を使います。write_csv() 関数は、第 1 引数に保存したいデータフレームを指定し、第 2 引数に保存先のファイルパスを指定します。

```
readr::write_csv(idpos, path = "data/new_data.csv")
```

2.0.2　企業データの処理

MS Excel のファイルを読み込むこともできます。ここでは，readxl パッケージの read_xlsx() 関数を使って，企業データを読み込みます。

```r
firmdata <- readxl::read_xlsx("data/MktRes_firmdata.xlsx")
firmdata |> glimpse()
```

```
Rows: 1,431
Columns: 31
$ fyear                  <dbl> 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,~
$ legalname              <chr> "ハウステンボス株式会社", "ハウステンボス株
式会社", "ハウステンボス株式会社", "ハ~
$ ind_en                 <chr> "Miscellaneous Services", "Miscellaneous Servic~
$ parent                 <chr> "エイチ・アイ・エス", "エイチ・アイ・エス", "
エイチ・アイ・エス", "エイチ・アイ・~
$ fiscal_month           <chr> "2011/10", "2012/10", "2013/10", "2014/10", "20~
$ current_liability      <dbl> 65509, 76206, 85459, 98384, 122993, 102805, 131~
$ ltloans                <dbl> 0, 4781, 23411, 22780, 14319, 77042, 101603, 11~
$ total_liability        <dbl> 73428, 96734, 125233, 179036, 194254, 237245, 3~
$ current_assets         <dbl> 102810, 111697, 137515, 196789, 212979, 233531,~
$ ppent                  <dbl> 12383, 40554, 45511, 48704, 60761, 62291, 83001~
$ total_assets           <dbl> 139018, 173497, 215913, 281332, 308245, 332385,~
$ net_assets_per_capital <dbl> 65589, 76763, 90680, 102295, 113990, 95139, 111~
$ sales                  <dbl> 380805, 431483, 479478, 523246, 537456, 523705,~
$ sga                    <dbl> 61158, 65654, 69953, 80033, 88284, 90769, 98822~
$ operating_profit       <dbl> 9407, 11316, 11843, 15906, 19970, 14274, 15915,~
$ net_profit             <dbl> 8958, 10881, 11190, 11271, 14025, 1305, 15835, ~
$ pnet_profit            <dbl> 8300, 9331, 8903, 9050, 10890, 267, 13259, 1106~
$ re                     <dbl> 47658, 55966, 63664, 71612, 82150, 80988, 92731~
$ adv                    <dbl> 8565, 9691, 10694, 11665, 12969, 12647, 12371, ~
$ labor_cost             <dbl> 30624, 31743, 31648, 37240, 40402, 41505, 46583~
$ rd                     <dbl> 0, 0, 0, 0, 0, 0, 0, 176, 0, 0, 0, 0, 0, 0, 0, ~
$ other_sg               <dbl> 14962, 17078, 19720, 21648, 24043, 24710, 26067~
$ emp                    <dbl> 6265, 8310, 9026, 9652, 10143, 10845, 13510, 13~
$ temp                   <dbl> 1751, 2470, 2750, 3071, 3469, 3535, 3422, 3179,~
$ tempratio              <dbl> 0.2184381, 0.2291280, 0.2335258, 0.2413739, 0.2~
$ indgrowth              <dbl> -0.0038649153, 0.0780674898, -0.0737512474, 0.0~
$ adint                  <dbl> 0.02249183, 0.02245975, 0.02230342, 0.02229353,~
$ rdint                  <dbl> 0.0000000000, 0.0000000000, 0.0000000000, 0.000~
$ mkexp                  <dbl> 0.16060188, 0.15215895, 0.14589408, 0.15295482,~
$ op                     <dbl> 0.02470293, 0.02622583, 0.02469978, 0.03039870,~
$ roa                    <dbl> 0.0597044987, 0.0537819098, 0.0412342008, 0.032~
```

31 変数 1,431 観測値からならなるデータが読み込まれました。必要な変数が変わること

もあるかと思うので、先に必要な変数名をベクトル vars_list として保存しておきます。

必要な観測値と変数だけを抽出して、新しい変数を作成し、並び替えて、新しいオブジェクト firm2018_check に格納します。

```r
# 必要な変数のリスト
vars_list <- c("legalname", "fyear", "sales", "labor_cost", "emp", "temp")

# コード 4-28
firm2018_check <- firmdata |>
  filter(fyear == 2018) |> # 2018 年のデータを抽出
  select(all_of(vars_list)) |> # 必要な変数を選択
  mutate(wage = labor_cost / (temp + emp), na.rm = TRUE) |> # 新しい変数 wage を作成
  # temp + emp が 0 でないチェックが必要かも
  arrange(desc(wage)) # wage の降順に並び替え
head(firm2018_check, n = 10) # 先頭 10 行を表示
```

```
# A tibble: 10 x 8
   legalname                        fyear   sales labor_cost   emp  temp  wage na.rm
   <chr>                            <dbl>   <dbl>      <dbl> <dbl> <dbl> <dbl> <lgl>
 1 株式会社リクルート               2018 2310756     388583 45856  2449  8.04 TRUE
 2 株式会社 大丸松坂屋百貨店        2018  459840      62692  6695  3581  6.10 TRUE
 3 株式会社 帝国ホテル              2018   58426      17307  1940   998  5.89 TRUE
 4 株式会社 髙島屋                  2018  912848      83779  7761  8849  5.04 TRUE
 5 株式会社コメリ                   2018  346862      43991  4646  4777  4.67 TRUE
 6 株式会社オートバックスセブン     2018  213840      22139  4171   747  4.50 TRUE
 7 株式会社ロイヤルホテル           2018   40884      13115  2049   894  4.46 TRUE
 8 オルビス株式会社                 2018  248574      28555  4181  2330  4.39 TRUE
 9 株式会社ファンケル               2018  122496      15103  1381  2213  4.20 TRUE
10 近畿日本ツーリスト株式会社       2018  411821      38186  6956  2189  4.18 TRUE
```

2.1　4 データの要約

all_of() 関数は、select() 関数内で使用され、変数名のベクトルを指定して、その変数を選択するために使います。これにより、変数名が動的に指定でき、コードの柔軟性が向上します。

大規模データをあつかうとき、データそのものを眺めていても特徴をつかむことは難しいので、そのデータを特徴付ける代表値を計算して、そこからデータの特徴をつかみます。主要な代表値として、平均値 (mean) や中央値 (median)、散らばり具合を示す分散 (variance) や標準偏差 (standard deviation) があります。それぞれ、

- 平均値：mean() 関数
- 中央値：median() 関数

- 分散：var() 関数
- 標準偏差：sd() 関数

を使って計算します。

先ほどのデータ firm2018_check の sales 変数について、代表値を計算してみます。ここでは、dplyr パッケージの summarize() 関数を使って、平均、中央値、分散、標準偏差を一度に計算します。

```
firm2018_check |>
  summarize(
    mean_sales   = mean(sales),
    median_sales = median(sales),
    var_sales    = var(sales),
    sd_sales     = sd(sales)
  )
```

```
# A tibble: 1 x 4
  mean_sales median_sales var_sales sd_sales
       <dbl>        <dbl>     <dbl>    <dbl>
1   1242425.       526675   3.69e12 1921062.
```

2.2　カテゴリ変数の要約

つぎに、データ上は数値や文字列として記録されていても、その数値や文字列がカテゴリーを表す名義尺度 (nominal scale) である場合を考えましょう。

例えば、firmdata の中の産業分類を表す ind_en 変数は、文字列で表現されていますが、その文字列はある観測値が所属するカテゴリーを表しています。fyear 変数は数値ですが、これは年度を表すカテゴリー変数です。

カテゴリ内の観測値の頻度を確認するには基本関数 table() を使いますが、ここでは、dplyr::count() 関数を使って、ind_en 変数の各カテゴリーの頻度を計算し、gt パッケージで表形式で表示します。

```
firm2018 <- firmdata |>
  filter(fyear == 2018) # 2018 年のデータを抽出
firm2018 |>
  dplyr::count(ind_en, name = "企業数") |>
  gt::gt() |>
  gtExtras::gt_theme_538()
```

教科書のように table() 関数を使うと簡単に頻度表が作れますが、オブジェクトの型が table 型になってしまい、後で他の処理に使いにくくなるので、data.frame 型のまま処理できる dplyr::count() 関数を使いました。

広告費を表す adient 変数の観測値が全企業の中央値より大きいか否か、でダミー変数 ad_dummy を作成し、ind_en 変数と ad_dummy 変数のクロス集計表を作成します。

教科書を先取りしますが、if_else() 関数を使ってダミー変数を作成し、count() 関

ind_en	企業数
Air Transportation	8
Amusement Services	4
Bakery Products	1
Communication Services	2
Cosmetics & Toilet Goods	3
Department Stores	7
Foods, NEC	1
Home & Pre-Fabs	2
Hotels	5
Miscellaneous Services	27
Miscellaneous Wholesales	2
Motor Vehicles	4
Musical Instrument	1
Railroad (Major)	27
Railroad (Minor)	2
Real Estate - Sales	1
Retail Stores, NEC	35
Supermarket Chains	14
Trucking	1

数でクロス集計表を作成し、`gt` パッケージで表形式で表示します。`tidyr` パッケージの `pivot_wider()` 関数を使って、クロス集計表が見やすくなるように、縦は産業名、横は広告費が中央値より大きいか否か、で表示するように変形しています。

`table()` 関数を使うと簡単にクロス集計表が作れますが、あえて `data.frame` 型のまま処理できる `dplyr::count()` 関数を使っています。そのため、データが矩形データのまま扱えるので、後で他の処理に使いやすくなるものの、表にすると見にくくなるので、`tidyr::pivot_wider()` 関数で見やすく変形しています。

```
firm2018 <- firm2018 |>
  mutate(
    ad_dummy = if_else(adint > median(adint, na.rm = TRUE), 1L, 0L)
    )
firm2018 %>%
  count(ind_en, ad_dummy) |>
  tidyr::pivot_wider(names_from = ad_dummy, values_from = n,
↪  values_fill = 0) |>
  gt::gt() |>
  gtExtras::gt_theme_538()
```

次に、カテゴリ変数ごとに処理を行いたい場合、例えば、産業分類ごとに売上高の平均値や標準偏差を計算したい場合は、`group_by()` 関数を使って、カテゴリ変数でグループ化してから、`summarize()` 関数で集計します。

ind_en	0	1
Air Transportation	4	4
Amusement Services	4	0
Bakery Products	0	1
Communication Services	1	1
Cosmetics & Toilet Goods	0	3
Department Stores	0	7
Foods, NEC	0	1
Home & Pre-Fabs	0	2
Hotels	5	0
Miscellaneous Services	17	10
Miscellaneous Wholesales	1	1
Motor Vehicles	0	4
Musical Instrument	0	1
Railroad (Major)	27	0
Railroad (Minor)	2	0
Real Estate - Sales	0	1
Retail Stores, NEC	11	24
Supermarket Chains	2	12
Trucking	1	0

```
# コード 4-39
firm2018 |>
  group_by(ind_en) |>
  summarize(
      obs = n(),
      sales_m = mean(sales),
      sales_sd = sd(sales),
      adint_m = mean(adint),
      adint_sd = sd(adint)
      ) |>
  gt::gt() |>
  gt::fmt_number(
    columns = c(sales_m, sales_sd, adint_m, adint_sd),
    decimals = 2
  ) |>
  gtExtras::gt_theme_538()
```

ind_en	OBS	sales_m	sales_sd	adint_m	adint_sd
Air Transportation	8	1,772,786.50	305,239.60	0.00	0.00
Amusement Services	4	298,137.50	263,017.35	0.00	0.00
Bakery Products	1	1,059,442.00	NA	0.01	NA
Communication Services	2	547,087.50	172,735.58	0.02	0.03
Cosmetics & Toilet Goods	3	140,669.33	100,063.48	0.11	0.05
Department Stores	7	843,248.29	348,818.99	0.02	0.01
Foods, NEC	1	504,153.00	NA	0.02	NA
Home & Pre-Fabs	2	4,143,505.00	0.00	0.01	0.00
Hotels	5	62,134.80	58,060.28	0.00	0.00
Miscellaneous Services	27	311,867.22	456,036.57	0.01	0.02
Miscellaneous Wholesales	2	176,520.00	52,778.45	0.02	0.03
Motor Vehicles	4	5,279,121.75	4,233,187.72	0.03	0.00
Musical Instrument	1	434,373.00	NA	0.04	NA
Railroad (Major)	27	1,302,920.52	1,037,834.05	0.00	0.00
Railroad (Minor)	2	260,502.00	0.00	0.00	0.00
Real Estate - Sales	1	1,861,195.00	NA	0.01	NA
Retail Stores, NEC	35	571,019.17	547,246.68	0.02	0.03
Supermarket Chains	14	4,335,164.07	3,511,346.59	0.01	0.01
Trucking	1	1,118,094.00	NA	0.00	NA

第 II 部

第 2 部 仮説検証型データ分析

　第 5 章では基本統計学の復習を、第 6 章では回帰分析、第 7 章では回帰分析の応用としてダミー変数や交差項、変数変換について学習します。第 8 章では消費者選択を分析するための離散選択モデルとして、ロジットモデルとプロビットモデル、多項選択モデルを学びます。

第 3 章

第 5 章基礎統計学復習

経営学部の 1 回生がおおよそ学んでいるであろう統計学の内容を、R コードとともに復習します。テキストの内容に加えて、コードの効率化や関数化についても解説します。

```
# コード 5-1
pacman::p_load(tidyverse, readxl, pwr, knitr)
```

まずサイコロを作ります。1 から 6 の数値からなるベクトル `dice` を作ります。つぎに `sample()` 関数を用いて、`dice` の中から 1 つの数をランダムに取り出します。

`sample()` 関数の引数は以下の通りです。

- `x`: 抽出元のベクトル
- `size`: 抽出する数
- `replace`: 置換を許可するかどうか

```
# コード 5-2
set.seed(442)
# サイコロ
dice <- 1:6
# サイコロを 1 回振る
d <- sample(dice, size = 1)
# 結果の表示
d
```

```
[1] 6
```

同じコードを 3 回以上書いているなと思ったら、ループ処理や関数化を検討しましょう。`for()` 関数を使ったり、サイコロを投げる回数を引数にした関数を作成したりする方法があります。

```
d1 <- sample(dice, size = 10, replace = TRUE)
d2 <- sample(dice, size = 10, replace = TRUE)
d3 <- sample(dice, size = 10, replace = TRUE)
```

をまとめて書くと、次のようになります。

```
d <- list() # 空のリストを作成
for (i in 1:3) { # ループ処理
  d[[i]] <- sample(dice, size = 10, replace = TRUE)
}
d # 結果の表示
```

```
[[1]]
 [1] 4 4 3 2 6 3 5 2 6 5

[[2]]
 [1] 3 2 6 6 5 4 6 4 3 1

[[3]]
 [1] 6 1 1 5 2 1 4 5 2 2
```

これをさらに発展させて、後で変更する可能性のある変数である試行回数やサイコロを振る回数を別に設定しておいて、コードを読みやすく、また変更しやすくします。

```
trial_count <- 3 # 試行回数
sample_sizes <- 10 # サイコロを振る回数

# 空の数列を作成
d_mean <- numeric(trial_count)
for (i in 1:trial_count) {
  d_mean[i] <- mean(sample(dice, size = sample_sizes, replace = TRUE))
}
d_mean
```

```
[1] 3.2 3.3 4.0
```

試行回数 `trials` とサイコロを振る回数 `sample_sizes` を引数にした関数を作成すると、さらに便利になります。ここでは、自作関数 `dice_mean()` を定義してみましょう。デフォルト値として、試行回数を 3 回、サイコロを振る回数を 10 回に設定しています。

```
dice_mean <- function(trial_count = 3, sample_sizes = 10) {
  d_mean <- numeric(trial_count) # 空のリストを作成
  for (i in 1:trial_count) {
    d_mean[i] <- mean(sample(dice, size = sample_sizes, replace =
    TRUE))
  }
  return(d_mean)
}
```

```
dice_mean() # デフォルト値で実行
```

```
[1] 3.4 2.5 4.4
```

```
dice_mean(trial_count = 5, sample_sizes = 20) # 引数を指定して実行
```

```
[1] 3.15 3.55 3.65 3.50 3.25
```

```
dice_mean(5, 20) # 省略形でも OK
```

```
[1] 3.80 4.30 2.75 3.70 3.05
```

この関数を使って、サイコロを 10 回、100 回、1,000 回振ったときの平均をそれぞれ 3 回ずつ試行してみましょう。

```
set.seed(352)
size = c(10, 100, 1000)
d_mean <- list()

for (i in seq_along(size)) {
  d_mean[[i]] <- dice_mean(trial_count = 3, sample_sizes = size[i])
}

d_mean_mat <- do.call(rbind, d_mean)

# 列名を付与
colnames(d_mean_mat) <- paste0("試行", 1:ncol(d_mean_mat))
# 行名を付与
rownames(d_mean_mat) <- paste0("サイコロを", size, "回振る")
# 表として出力
knitr::kable(
  d_mean_mat,
  caption = "サイコロの標本平均（各サイズ 3 回ずつ）",
  align = "ccc"
)
```

Table3.1: サイコロの標本平均（各サイズ 3 回ずつ）

	試行 1	試行 2	試行 3
サイコロを 10 回振る	3.200	3.000	2.400
サイコロを 100 回振る	3.440	3.340	3.560
サイコロを 1000 回振る	3.433	3.446	3.554

tidyverse の purrr パッケージを使うと、さらに簡潔に書けます。

```
set.seed(352)
trial_count <- 3 # 試行回数
sizes <- c(10, 100, 1000) # サイコロを振る回数
# map_dfr() でデータフレームを直接作成
d_mean_df <- map_dfr(sizes, function(n) {
  # replicate() で 3 回繰り返し試行
  results <- replicate(
    trial_count,
    mean(sample(dice, size = n, replace = TRUE))
    )
  set_names(results, paste0("試行", 1:trial_count))
  })

result <- d_mean_df |>
  mutate(条件 = paste0("サイコロを", sizes, "回振る")) |>
  relocate(条件) #「条件」列を一番左へ移動

# 出力
kable(
  result,
  caption = "サイコロの標本平均（各サイズ 3 回ずつ）",
  align = "cccc"
)
```

Table3.2: サイコロの標本平均（各サイズ 3 回ずつ）

条件	試行 1	試行 2	試行 3
サイコロを 10 回振る	3.200	3.000	2.400
サイコロを 100 回振る	3.440	3.340	3.560
サイコロを 1000 回振る	3.433	3.446	3.554

3.1 区間推定

いま、標準正規分布に従う母集団から抽出した無作為標本 $Z_1, Z_2, ..., Z_n$ を考えます。標準正規分布の確率分布関数 $\phi(z)$ を用いると、次のように表せます。

$$P\left(-z_{\frac{\alpha}{2}} \leq \frac{\bar{Z}-\mu}{\frac{\sigma}{\sqrt{n}}} \leq z_{\frac{\alpha}{2}}\right) = 1-\alpha$$

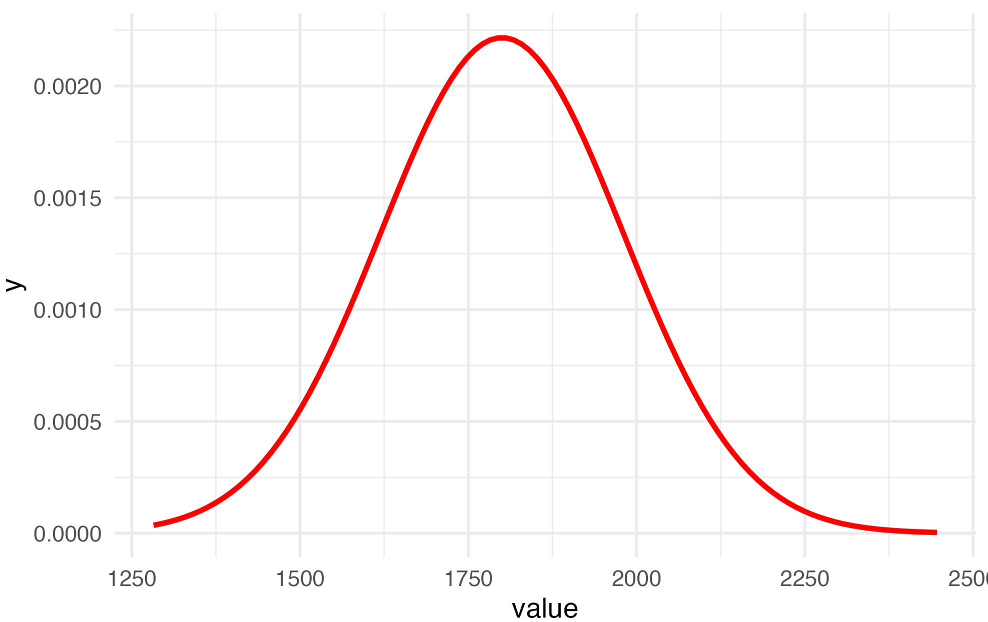
3.1.1　電球の寿命データで区間推定

例として、次のような白熱電球の寿命データが手元にあるものとしましょう。ここで、以下のことが分かっているものとします。

- 平均寿命が 1800 時間である。
- 寿命の標準偏差$ $は 180 時間である。
- 平均寿命は正規分布に従う。

図にすると次のようになります。

```r
averate <- 1800
standard_dev <- 180
rnorm(1000, mean = averate, sd = standard_dev) |>
  as_tibble() |>
  ggplot(aes(x = value)) +
  stat_function(
    fun = dnorm,
    args = list(mean = averate, sd = standard_dev),
    color = "red",
    size = 1
    ) +
  theme_minimal()
```



この分布から抽出したと想定される 16 個の電球の寿命データが以下の通りです。

```r
# コード 5-5
bulb <- c(# 白熱電球の寿命データ
    1939.6, 1680.3, 1982.1, 2215.6,
    2092.5, 1928.9, 2003.8, 1955.5,
    1800.1, 1659.5, 2066.2, 2107.2,
    2085.5, 1878.6, 2007.6, 1816.1
  )
mean(bulb)
```

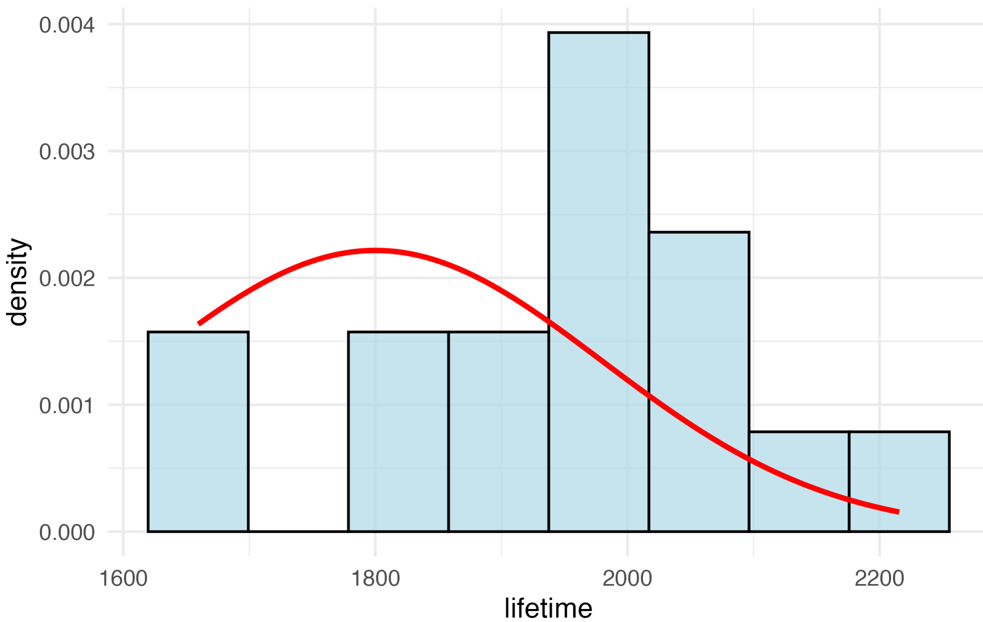
```
[1] 1951.194
```

```r
sd(bulb)
```

```
[1] 154.4567
```

先ほどの分布にこのデータのヒストグラムを重ねると、次のようになります。

```r
# ヒストグラムの作成
bulb_df <- tibble(lifetime = bulb)
bulb_df |>
  ggplot(aes(x = lifetime)) +
  geom_histogram(
    aes(y = ..density..),
    bins = 8,
    fill = "lightblue",
    color = "black",
    alpha = 0.7
    ) +
  stat_function(
    fun = dnorm,
    args = list(mean = averate, sd = standard_dev),
    color = "red",
    size = 1
    ) +
  theme_minimal()
```



そのとき、この手元にある 16 個の電球の寿命データの平均値が、母平均 1800 の周りでどの範囲に入るかを 95 %の信頼水準で推定してみましょう。

```
# t 検定の実施
bulb_ci <- t.test(bulb)
bulb_ci$conf.int
```

```
[1] 1868.890 2033.498
attr(,"conf.level")
[1] 0.95
```

```
# 信頼区間の出力（デフォルトで 95 %信頼水準）
```

```
# コード 5-6
qnorm(0.025, lower.tail = FALSE)
```

```
[1] 1.959964
```

```
# 標準正規分布における上側 2.5 %点の分位点を求める。
```

```
# コード 5-7
n <- length(bulb) # 標本サイズの計算
z <- qnorm(0.025, lower.tail = FALSE) # 上側 2.5 %点の分位点
xbar <- mean(bulb) # 標本平均の計算
sigma <- 180 # 母標準偏差の設定
```

```r
# 信頼区間の計算
upper <- xbar + z * (sigma / sqrt(n))
lower <- xbar - z * (sigma / sqrt(n))

# 結果のまとめと出力
ci.bulb <- matrix(c(lower, upper), nrow = 1)
colnames(ci.bulb) <- c("ci.lower", "ci.upper")
knitr::kable(
  ci.bulb,
  caption = "Bulb data CI (95 %)",
  align = "cc"
  )
```

Table3.3: Bulb data CI (95 %)

ci.lower	ci.upper
1862.995	2039.392

```r
# コード 5-8
mean(bulb)
```

```
[1] 1951.194
```

```r
# コード 5-9
t.test(bulb, alternative = "two.sided", mu = 1800)
```

```

	One Sample t-test

data:  bulb
t = 3.9155, df = 15, p-value = 0.001377
alternative hypothesis: true mean is not equal to 1800
95 percent confidence interval:
 1868.890 2033.498
sample estimates:
mean of x
 1951.194
```

```r
# コード 5-10
n <- length(bulb) # 標本サイズの計算
z <- qnorm(0.025, lower.tail = FALSE) # 上側 2.5 %点の分位点
xbar <- mean(bulb) # 標本平均の計算
sigma <- 180 # 母標準偏差の設定
mu <- 1800 # 帰無仮説の平均値
# Z 値の計算
Z <- (xbar - mu) / (sigma / sqrt(n))
Z
```

```
[1] 3.359861
```

```r
# コード 5-11
# データの読み込みと前処理
firmdata <- readxl::read_xlsx("data/MktRes_firmdata.xlsx")
firm2018 <- firmdata |>
  filter(fyear == 2018) |>
  mutate(
    ad_dummy = ifelse(adint > median(adint), 1, 0)
    )
# t 検定の実施
t.test(sales ~ ad_dummy, data = firm2018)
```

```

	Welch Two Sample t-test

data:  sales by ad_dummy
t = -3.3989, df = 85.686, p-value = 0.001029
alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
95 percent confidence interval:
 -1674283.1  -438496.1
sample estimates:
mean in group 0 mean in group 1 
       725009.7       1781399.3 
```

```r
# コード 5-12
var.test(sales ~ ad_dummy, data = firm2018, ratio = 1)
```

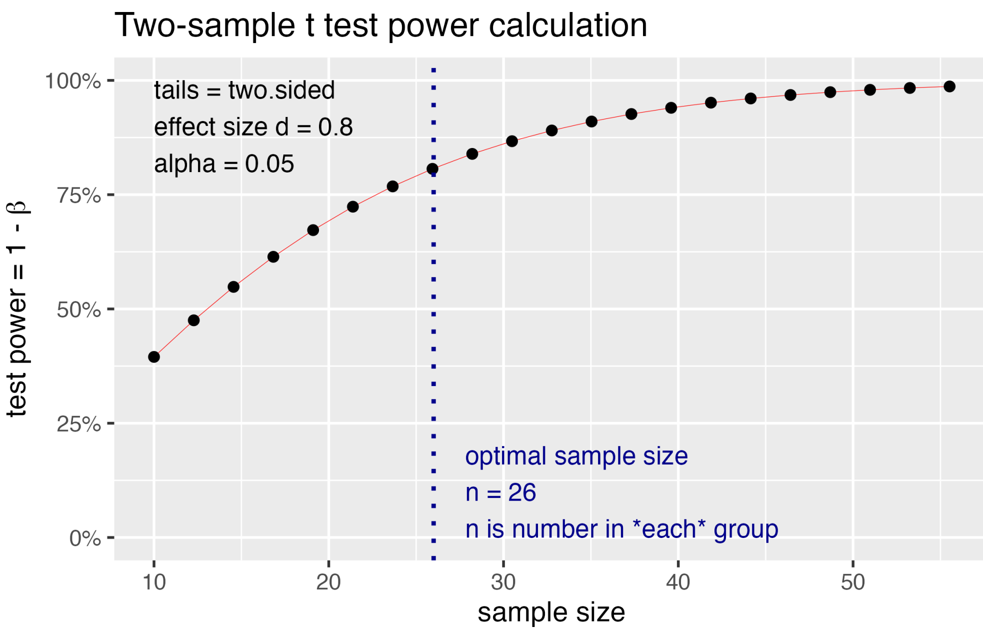
```

	F test to compare two variances
```

```
data:  sales by ad_dummy
F = 0.10863, num df = 74, denom df = 71, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.06821636 0.17258882
sample estimates:
ratio of variances
         0.1086276
```

```
# 効果量 d=0.8、検出力 0.8、有意水準 0.05 の場合の
# 必要標本サイズの推定
est_n <- pwr.t.test(d = 0.8, power = 0.8, sig.level = 0.05)
est_n
```

```

     Two-sample t test power calculation

              n = 25.52458
              d = 0.8
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

```
# コード 5-15
plot(est_n)
```




```
# コード 5-16
pwr.t.test(d = 0.8, n = 26, sig.level = 0.05)
```

```
     Two-sample t test power calculation

              n = 26
              d = 0.8
      sig.level = 0.05
          power = 0.8074866
    alternative = two.sided

NOTE: n is number in *each* group
```

第 4 章

第 6 章 回帰分析

6 章は変数間の関係を考察する手法の 1 つである**回帰分析** (Regression Analysis) について説明します。

いま手元に売上高 (y とする) と従業員数 (x とする) という 2 つのデータの組 (y_i, x_i) があるとします。i は企業の番号を表しており，n 社のデータがあるとします。こんな感じのデータです。

企業番号	売上高	従業員数
1	1000	5
2	2000	10
3	3000	15
...	...	...
n	4500	25

いま、従業員数 x が増えると売上高 y は増えるのか否か、という分析をしたいとしましょう。つまり、従業員数と売上高との間には**関係**があるのかどうかを調べたい、ということです。しかし、関係があるといってもいろんな関係がある可能性があります。そこで、変数間の関係で最も基本的なものとして**線形関係** (linear relation) を考えます。

n はサンプルサイズといいます。サンプル数という言い方は誤りです。サンプル数は 1 です。

y は x と線形関係であると考え分析のモデルを構築します。これを「単回帰モデル」と呼び、次のように表します。

中学校の数学で学ぶ $y = ax + b$ です。

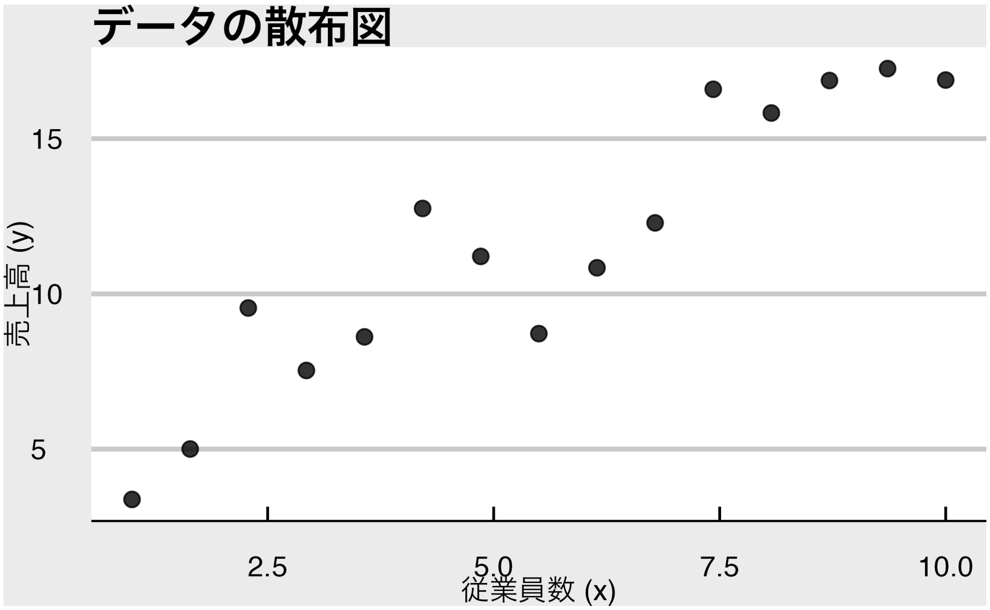
$$y_i = \beta_0 + \beta_1 x_i + u_i$$

ここで各変数は以下のように定義されます。

- y_i: **従属変数** (dependent variable)、応答変数、または目的変数。
- x_i: **独立変数** (independent variable)、または説明変数。
- β_0: **切片** (intercept)。
- β_1: **傾き** (slope) または回帰係数。
- u_i: **誤差項** (error term)。モデルがデータを説明できない確率的な変動部分を表します。

手元にある売上高 y と従業員数 x のデータを使って、横軸を従業員数、縦軸を売上高とした散布図に描くと次のようになります。

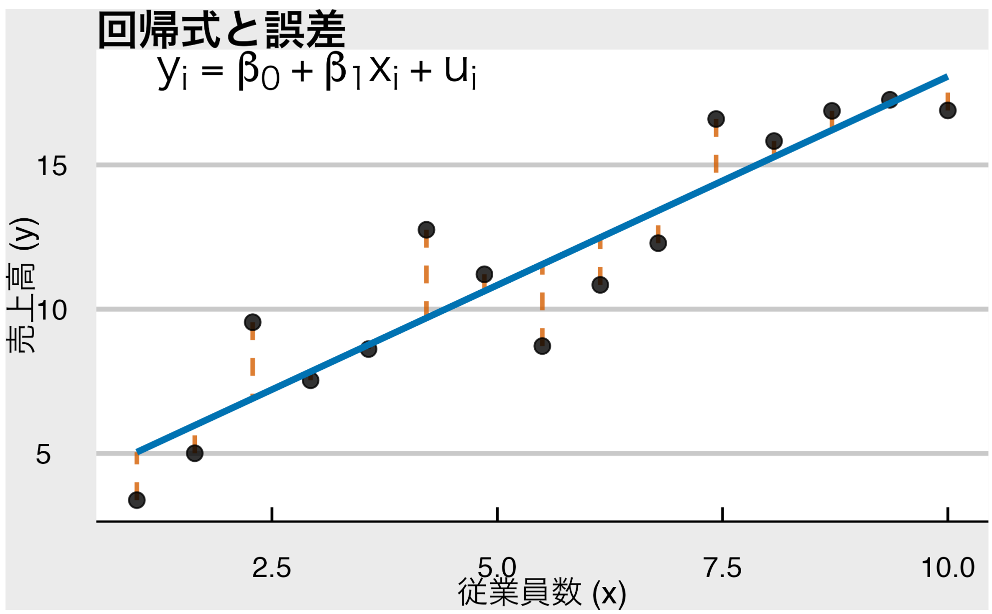
```
pacman::p_load(tidyverse, ggthemes, GGally, gt, gtExtra, knitr,
↪   kableExtra)

set.seed(123) # 乱数のタネ
n <- 15 # サンプルサイズ
x <- seq(1, 10, length.out = n) # 独立変数
y <- 3 + 1.5 * x + rnorm(n, mean = 0, sd = 2) # 従属変数
data <- data.frame(x = x, y = y) # データフレームの作成
my_font <- if (Sys.info()["sysname"] == "Darwin") "HiraKakuProN-W3"
↪   else "sans"

# 作図
g <- ggplot(data, aes(x = x, y = y)) +
    geom_point(size = 2.5, color = "black", alpha = 0.8) +
    theme_economist_white(base_family = my_font) +
        labs(
        title = "データの散布図",
        x = "従業員数 (x)",
        y = "売上高 (y)"
    )

print(g)
```



散布図をみると右肩上がりの関係があるように見えますが、もっと明確に変数間の関係を表すために、そのデータを最も良く説明できる直線をひこうと思います。つまり、切片の β_0 と傾きの β_1 を決めるということです。この直線が「モデル」で、それぞれの黒い点とモデルの直線との縦軸方向への距離を「誤差」または「残差」と呼びます。

```
model <- lm(y ~ x, data = data)
# 予測値を predicted に格納
data$predicted <- predict(model)
# OS に合わせてフォントを自動選択（Mac ならヒラギノ、それ以外ならデフォルト）

g <- ggplot(data, aes(x = x, y = y)) +
    geom_segment(
        aes(xend = x, yend = predicted),
        color = "#D55E00", linetype = "dashed", alpha = 0.8, linewidth
         ↪   = 0.8
    ) +
    geom_point(size = 2.5, color = "black", alpha = 0.8) +
    theme_economist_white(base_family = my_font) +
    labs(
        title = "回帰式と誤差",
        x = "従業員数 (x)",
        y = "売上高 (y)"
    ) +
    geom_smooth(method = "lm", se = FALSE, color = "#0072B2",
     ↪  linewidth = 1.2) +
```

```
    theme(
        plot.title = element_text(face = "bold", size = 16),
        axis.title = element_text(size = 12),
    ) +
    annotate("text",
        x = min(data$x) + 2,
        y = max(data$y) + 1,
        label = "y[i] == beta[0] + beta[1] * x[i] + u[i]",
        parse = TRUE,
        size = 6, family = my_font
    )
print(g)
```



望ましい性質を備えた直線の引き方の 1 つとして、ガウス (Gauss) が構築した最小二乗法 (Ordinaly Least Square Methods: OLS) について説明します。

4.1　分析準備

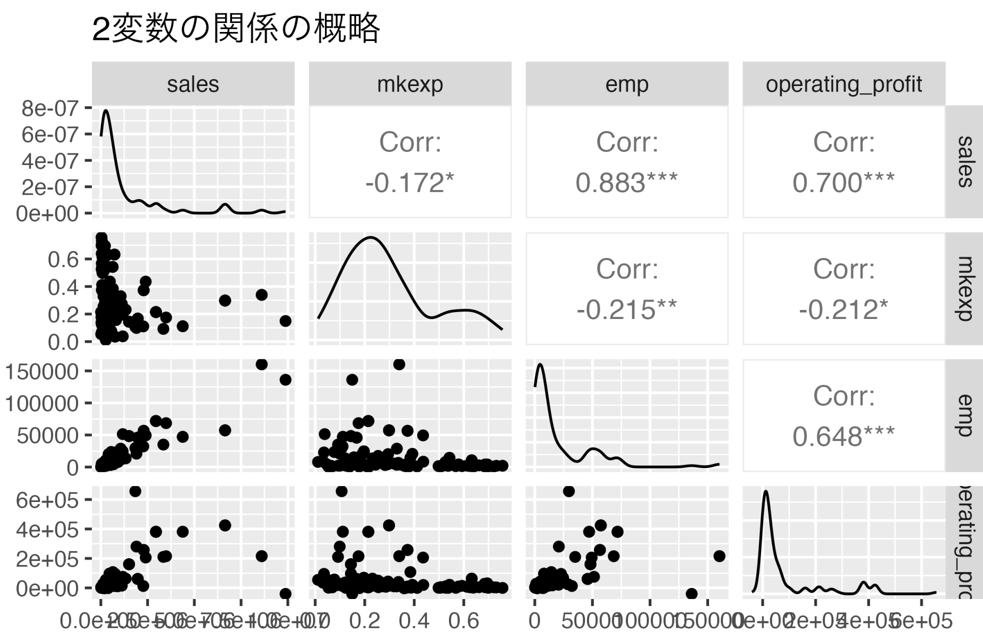
サンプルデータを読み込みます。MS Excel のファイル `.xlsx` なので、`readxl::read_xlsx()` 関数を使います。読み込んだデータから、2019 年のデータを抽出します。

```
firmdata <- readxl::read_xlsx("data/MktRes_firmdata.xlsx")
firmdata19 <- firmdata |>
```

```
    filter(fyear == 2019) # 2019 年を抽出
```

探索的なデータ分析として、変数間の関係を概観するために GGally パッケージの ggpairs() 関数を使います。

```
firmdata19 |>
    select(sales, mkexp, emp, operating_profit) |>
    GGally::ggpairs() +
    labs(title = "2 変数の関係の概略")
```



出力された図を見てみます。

- **対角線（分布）**: 売上高 sales や従業員数 emp、営業利益 operating_profit は値の小さい左側にデータが集まっており、右側に裾を引く歪んだ分布であることがわかります。
- **右上（相関係数）**: Corr: として相関係数が表示されています。売上高 sales と従業員数 emp や営業利益 operating_profit の相関が高く、また営業利益と従業員数にも強い正の相関があります。
- **左下（散布図）**: 2 変数間の散布図が表示されています。異常値の有無などを確認できます。

こういった図でデータを概観し、線形回帰モデルが適切かどうかを判断したり、異常値の有無を確認して分析結果の解釈に役立てたりします。

次に、図ではなく数値としてデータの特徴をつかむため記述統計量を計算します。ここでは基本関数 summary() を使って計算します。

```
ds1 <- firmdata19 |>
    select(sales, mkexp, emp, operating_profit) |> # 変数を選択
    summary() # 記述統計量

knitr::kable(ds1, align = "cccc") |>
    kableExtra::kable_styling(latex_options = "hold_position")
```

sales	mkexp	emp	operating_profit
Min. : 11333	Min. :0.01137	Min. : 163	Min. :-40469
1st Qu.: 183525	1st Qu.:0.16714	1st Qu.: 3454	1st Qu.: 7743
Median : 464450	Median :0.25448	Median : 7826	Median : 23904
Mean :1199403	Mean :0.29868	Mean : 20249	Mean : 81088
3rd Qu.:1164243	3rd Qu.:0.37506	3rd Qu.: 24464	3rd Qu.: 63068
Max. :9878866	Max. :0.75650	Max. :160227	Max. :656163

4.2　単回帰モデルと予測値

膨大なデータを目の前にした我々は、そのデータを最も良く「説明」できるシンプルなモデルを特定したいと考えます。最も基本的な関数 $f : x \to y$ として 1 次関数を考えます。

$$y = \beta_0 + \beta_1 x$$

ここで、β_0 は切片、β_1 は傾きを表します。何らかの方法で β_0 と β_1 を決めたとしよう。その切片を $\hat{\beta}_0$、傾きを $\hat{\beta}_1$ とします。x に手元にあるデータ x_i を代入して計算できる値を**予測値** (predicted value) といい、$\hat{y}_i$ と表します。

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

このとき、実際の観測値 y_i と予測値 $\hat{y}_i$ の差を**残差** (residual) といい、e_i で表します。

$$e_i = y_i - \hat{y}_i$$

回帰分析で最も良く用いられており、望ましい性質をもつ推定値を計算できる方法は最小二乗法といい、この残差 e_i の二乗和 $\sum e_i^2$ を最小にするように $\hat{\beta}_0$ と $\hat{\beta}_1$ を決定します。

R では `lm()` 関数を使って計算できます。ここでは、売上高が従業員数で説明できるかを調べる単回帰モデルを構築します。

$$sales_i = \beta_0 + \beta_1 emp_i + u_i$$

R だと**従属変数 ~ 独立変数**という形で回帰式を指定します。

```
# 回帰モデルを構築
ols_model <- "sales ~ emp" # 回帰式の定義
```

モデルを最小自乗法で推定するには、lm() 関数を使います。lm() 関数の第 1 引数に回帰式を、第 2 引数にデータフレームを指定します。推定結果を reg1 というオブジェクトに格納し、回帰係数を表示するには coef() 関数を使います。

```
reg1 <- lm(formula = ols_model, data = firmdata19)
coef(reg1) # 回帰係数
```

```
(Intercept)         emp
22113.98050    58.14081
```

最小二乗法で推定された係数は以下のようになります。

$$\hat{sales_i} = 22113.98050 + 58.14081 \times emp_i$$

従業員が 10 名のときの売上高の予測値は 22695.3886 となり、従業員が 1 名増えるごとに売上高が約 58.14 増加すると予測されます。

4.3 回帰分析における推定と検定

4.3.1 回帰分析と統計的推測

この節は回帰分析の理屈を学習する上で重要な章なので、時間を掛けて学習することをオススメします。

4.3.2 企業データを用いた回帰分析の実行

上で推定した回帰分析の結果をみていきます。lm() 関数で推定した結果を reg1 というオブジェクトに格納しているので、その中身をみてみましょう。詳細を表示するには summary() 関数を使います。

```
summary(reg1)
```

summary() 関数は引数に指定するオブジェクトの型により異なる結果を返します。ここでは、lm オブジェクトを引数に指定しているので、回帰分析の結果が表示されます。

```

Call:
lm(formula = ols_model, data = firmdata19)

Residuals:
     Min       1Q   Median       3Q      Max
-1834902  -280228   -34549   136598  3292521
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 22113.980  89224.761   0.248    0.805
emp            58.141      2.569  22.628   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 875800 on 144 degrees of freedom
Multiple R-squared:  0.7805,    Adjusted R-squared:  0.779
F-statistic:   512 on 1 and 144 DF,  p-value: < 2.2e-16
```

```
confint(reg1, level = 0.99)
```

```
                   0.5 %       99.5 %
(Intercept) -210798.52845 255026.48944
emp              51.43362     64.84799
```

4.3.3 回帰係数の解釈

この節は難しいので、教科書の解説に補足を加えながら説明します。例として次のような重回帰モデルを考えます。

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + u_i$$

ここで x_{1i} の微少な変化が y_i に与える影響を知りたいとします。この回帰モデルを x_{1i} で偏微分すると、次のようになります。

$$\frac{\partial y_i}{\partial x_{1i}} = \beta_1$$

つまり x_{2i} を一定として x_{1i} が 1 単位変化した場合の y_i の変化量は β_1 である、ということです。このことを**他の変数を一定とした場合の効果** (ceteris paribus effect) といい、様々な要因で決定されているであろう従属変数に対して、特定の独立変数が与える影響を測定する際に重要な考え方となります。

4.3.4 単回帰モデルと重回帰モデルの比較

営業利益 `operating_profit` を従属変数、広告宣伝費 `adv` を独立変数とした単回帰モデルを推定します。

```
reg2 <- lm(operating_profit ~ adv, data = firmdata19)
summary(reg2)
```

```
Call:
lm(formula = operating_profit ~ adv, data = firmdata19)

Residuals:
    Min      1Q  Median      3Q     Max
-450526  -55552  -40672    -791  599084

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.708e+04  1.057e+04   5.401 2.68e-07 ***
adv         1.257e+00  2.159e-01   5.823 3.61e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 117600 on 144 degrees of freedom
Multiple R-squared:  0.1906,    Adjusted R-squared:  0.185
F-statistic: 33.91 on 1 and 144 DF,  p-value: 3.613e-08
```

confint() 関数を使って信頼区間を計算します。

```
confint(reg2)
```

```
                   2.5 %       97.5 %
(Intercept) 36189.311039 77968.894422
adv             0.830362     1.683719
```

次に、営業利益 operating_profit を従属変数、広告宣伝費 adv、温度 temp、従業員数 emp、労働費用 labor_cost、総資産 total_assets、研究開発費 rd を独立変数とした重回帰モデルを推定します。教科書とは異なりますが、先にモデルを定義してから lm() 関数に渡す方法で推定します。

```
multi_model <- "operating_profit ~ adv + temp + emp + labor_cost +
  total_assets + rd"
```

この回帰モデルを lm() 関数で推定し、結果を summary() 関数で表示します。

```
reg3 <- lm(formula = multi_model, data = firmdata19)
summary(reg3)
```

```
Call:
lm(formula = multi_model, data = firmdata19)

Residuals:
```

```
    Min      1Q  Median      3Q     Max 
-360544  -27618  -15279    3467  284094 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.209e+04  8.114e+03   2.723  0.00731 ** 
adv          -1.431e+00  2.954e-01  -4.845 3.34e-06 ***
temp         -1.878e+00  6.313e-01  -2.975  0.00346 ** 
emp          -1.510e+00  7.036e-01  -2.146  0.03358 *  
labor_cost    8.808e-01  1.692e-01   5.207 6.76e-07 ***
total_assets  3.516e-02  5.840e-03   6.020 1.47e-08 ***
rd            1.385e+00  5.268e-01   2.629  0.00953 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 78940 on 139 degrees of freedom
Multiple R-squared:  0.6479,	Adjusted R-squared:  0.6327 
F-statistic: 42.63 on 6 and 139 DF,  p-value: < 2.2e-16
```

confint() 関数を使って信頼区間を計算します。

```
confint(reg3)
```

```
                    2.5 %        97.5 %
(Intercept)  6048.51599480  3.813231e+04
adv            -2.01511513 -8.470054e-01
temp           -3.12602311 -6.298318e-01
emp            -2.90144939 -1.190356e-01
labor_cost      0.54637427  1.215313e+00
total_assets    0.02361376  4.670793e-02
rd              0.34334525  2.426472e+00
```

第 5 章

第 7 章

```
pacman::p_load(tidyverse, readxl, sjPlot, marginaleffects, car,
 ↪  modelsummary)
# コード 7-1
# library(tidyverse)
firmdata <- readxl::read_xlsx("data/MktRes_firmdata.xlsx")
firmdata19 <- firmdata %>%
  filter(fyear == 2019)
#産業名を特定したオブジェクト"Retail"の作成
retail <- c("Retail Stores, NEC", "Supermarket Chains",
            "Department Stores")
# Retail を使ったカテゴリー変数の作成
firmdata19 <- firmdata19 %>%
  mutate(format = ifelse(ind_en %in% retail, "Retail", "Other"))
#カテゴリーの頻度チェック
with(firmdata19, table(format))
```

```
format
 Other Retail 
    90     56 
```

```
# コード 7-2
fit.d1 <- lm(op ~ mkexp + format, data = firmdata19)
summary(fit.d1)
```

```

Call:
lm(formula = op ~ mkexp + format, data = firmdata19)

Residuals:
```

```
      Min        1Q    Median        3Q       Max
-0.095926 -0.031427 -0.008661  0.014345  0.261068

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.101694   0.008833  11.514  < 2e-16 ***
mkexp        -0.066003   0.024959  -2.644 0.009097 **
formatRetail -0.031770   0.009303  -3.415 0.000831 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05368 on 143 degrees of freedom
Multiple R-squared:  0.1379,    Adjusted R-squared:  0.1258
F-statistic: 11.44 on 2 and 143 DF,  p-value: 2.469e-05
```

```
# コード 7-3
firmdata19 <- firmdata19 %>%
  mutate(retail = ifelse(format == "Retail", 1, 0))

# 確認
with(firmdata19, table(retail, format))
```

```
      format
retail Other Retail
     0    90      0
     1     0     56
```

```
# コード 7-4
fit.d2 <- lm(op ~ mkexp + retail, data = firmdata19)
summary(fit.d2)
```

```

Call:
lm(formula = op ~ mkexp + retail, data = firmdata19)

Residuals:
      Min        1Q    Median        3Q       Max
-0.095926 -0.031427 -0.008661  0.014345  0.261068

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

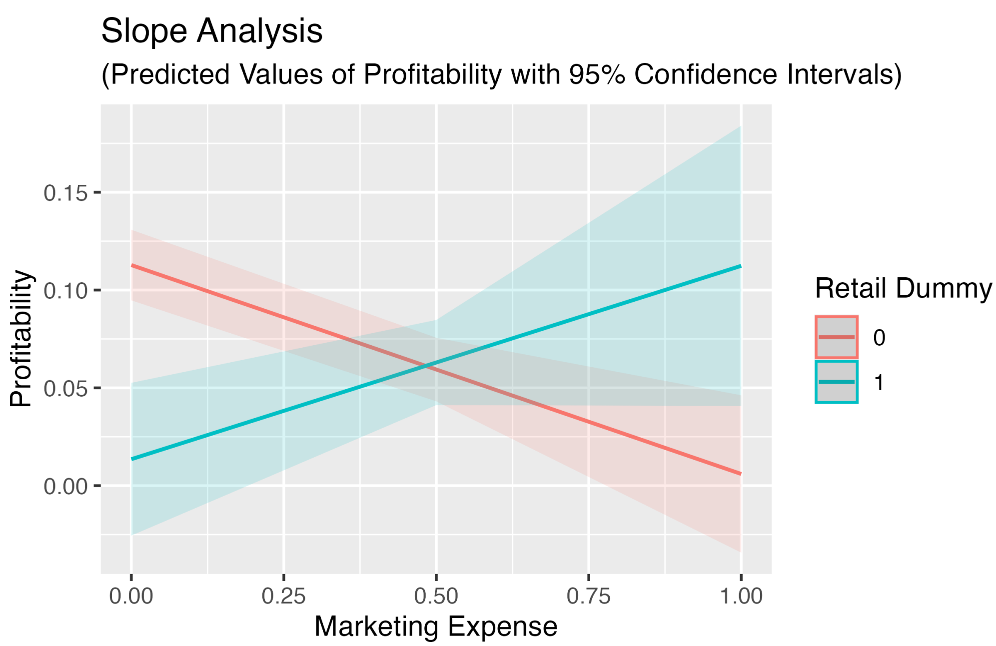
```
(Intercept)  0.101694   0.008833  11.514  < 2e-16 ***
mkexp       -0.066003   0.024959  -2.644 0.009097 **
retail      -0.031770   0.009303  -3.415 0.000831 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05368 on 143 degrees of freedom
Multiple R-squared:  0.1379,	Adjusted R-squared:  0.1258
F-statistic: 11.44 on 2 and 143 DF,  p-value: 2.469e-05
```

```
# コード 7-5
fit.d3 <- lm(op ~ mkexp * retail, data = firmdata19)
summary(fit.d3)
```

```

Call:
lm(formula = op ~ mkexp * retail, data = firmdata19)

Residuals:
      Min        1Q    Median        3Q       Max
-0.100914 -0.027907 -0.006023  0.020847  0.254339

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.112780   0.009123  12.362  < 2e-16 ***
mkexp        -0.106801   0.026898  -3.971 0.000113 ***
retail       -0.099242   0.021752  -4.563 1.08e-05 ***
mkexp:retail  0.205666   0.060393   3.405 0.000859 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0518 on 142 degrees of freedom
Multiple R-squared:  0.203,	Adjusted R-squared:  0.1862
F-statistic: 12.06 on 3 and 142 DF,  p-value: 4.461e-07
```

```
# コード 7-6
# install.packages("sjPlot")

# コード 7-7
# # library(sjPlot)
```

```
pred <- plot_model(fit.d3,
  type = "pred",
  terms = c("mkexp", "retail"),
  ci.lvl = .95) +
  labs(
    title = "Slope Analysis",
    subtitle = "(Predicted Values of Profitability with 95% Confidence
     ↪  Intervals)",
    x = "Marketing Expense",
    y = "Profitability"
  ) + scale_color_discrete(name = "Retail Dummy")
pred
```




```
# コード 7-8
Headphone07 <- readr::read_csv("data/headphone07.csv", na = ".")

#データフレームの確認
glimpse(Headphone07)
```

```
Rows: 221
Columns: 4
$ ID        <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
$ sales     <dbl> 118.8377, 548.6312, 197.3075, 104.2657, 748.8251, 947.8850, ~
```

```
$ rd        <dbl> 404.0893, 252.1270, 444.3374, 407.5876, 841.7605, 336.8744, ~
$ promotion <dbl> 75.63163, 102.74572, 97.98040, 83.46613, 105.69250, 80.17476~
```

```r
# コード 7-9
fit_int <- lm(sales ~ rd * promotion, data = Headphone07)
summary(fit_int)
```

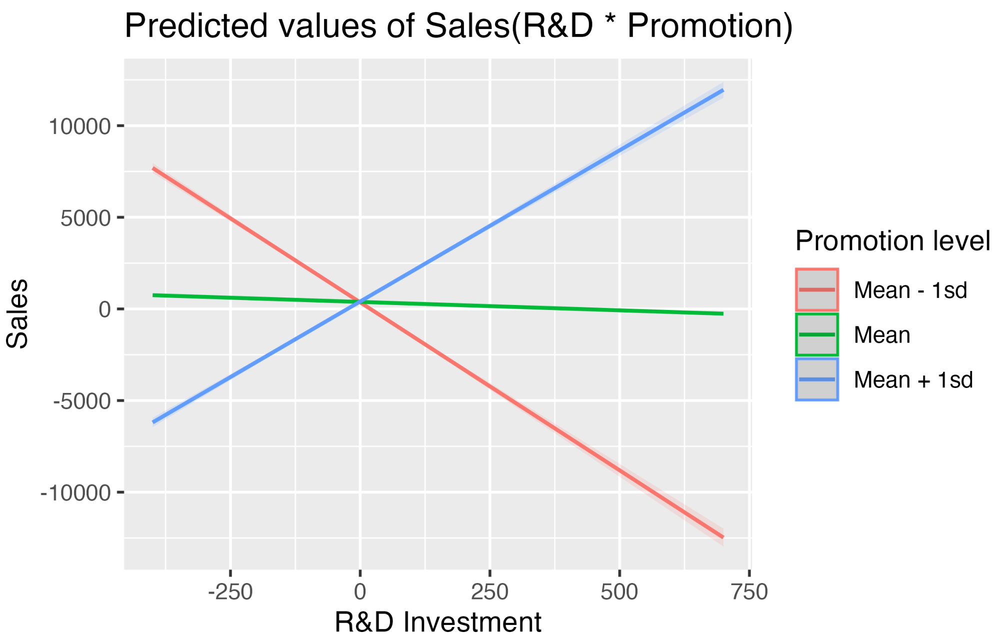
```

Call:
lm(formula = sales ~ rd * promotion, data = Headphone07)

Residuals:
    Min      1Q  Median      3Q     Max 
-46.522 -12.861  -0.638  13.578  70.667 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.033e+04  1.090e+02   186.5   <2e-16 ***
rd           -5.187e+01  2.844e-01  -182.4   <2e-16 ***
promotion    -1.914e+02  1.052e+00  -181.9   <2e-16 ***
rd:promotion  4.979e-01  2.730e-03   182.4   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.55 on 217 degrees of freedom
Multiple R-squared:  0.9935,	Adjusted R-squared:  0.9934 
F-statistic: 1.111e+04 on 3 and 217 DF,  p-value: < 2.2e-16
```

```r
# コード 7-10
Headphone07 <- Headphone07 %>%
  mutate(promotion_c = promotion - mean(promotion, na.rm = TRUE),
         rd_c = rd - mean(rd, na.rm = TRUE))

fit_int_c <- lm(sales ~ rd_c * promotion_c , data = Headphone07)
summary(fit_int_c)
```

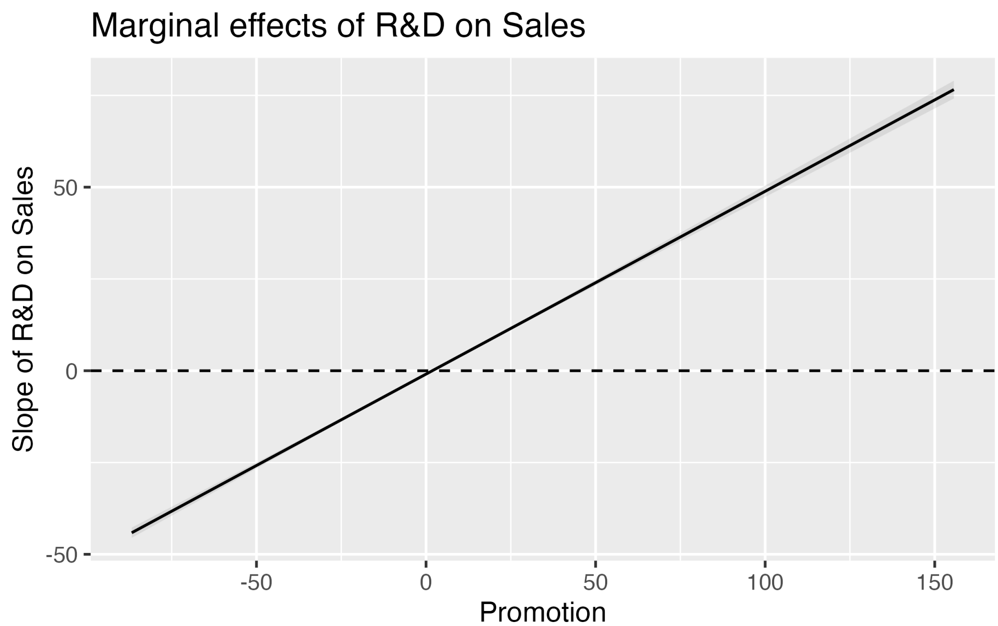
```

Call:
lm(formula = sales ~ rd_c * promotion_c, data = Headphone07)

Residuals:
```

```
    Min      1Q  Median      3Q     Max 
-46.522 -12.861  -0.638  13.578  70.667 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)      382.27812    1.51459  252.40   <2e-16 ***
rd_c              -0.91767    0.01196  -76.75   <2e-16 ***
promotion_c        0.69039    0.03972   17.38   <2e-16 ***
rd_c:promotion_c   0.49792    0.00273  182.37   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.55 on 217 degrees of freedom
Multiple R-squared:  0.9935,    Adjusted R-squared:  0.9934 
F-statistic: 1.111e+04 on 3 and 217 DF,  p-value: < 2.2e-16
```

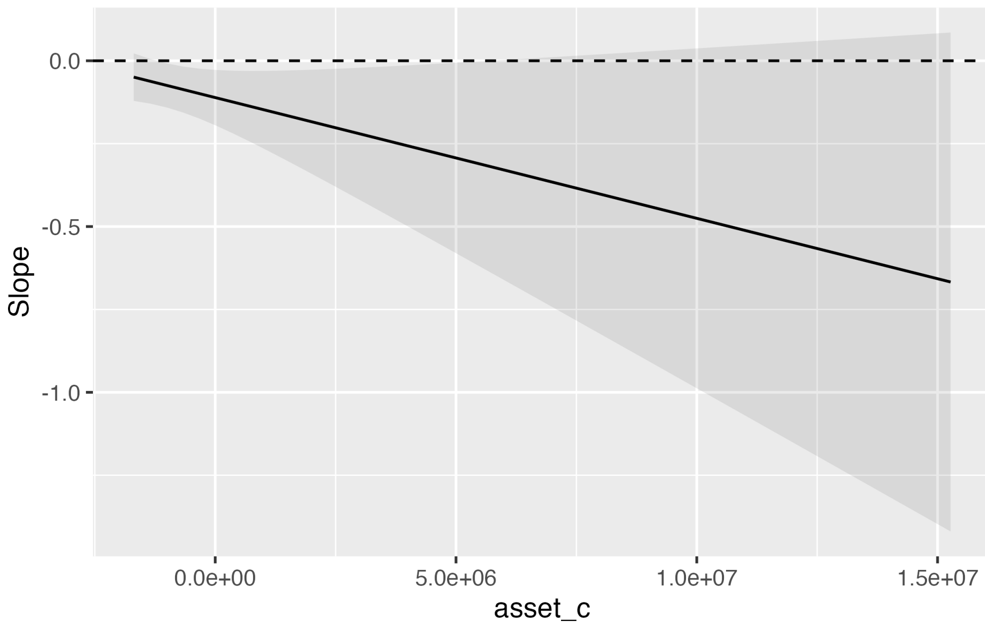
```r
# コード 7-11
leg = c("Mean - 1sd", "Mean", "Mean + 1sd")
int_fig1 <- plot_model(fit_int_c,
 type = "int",
 mdrt.values = "meansd",
 ci.lvl = .9999999999) +
  labs(
    title = "Predicted values of Sales(R&D * Promotion)",
    x = "R&D Investment",
    y = "Sales") +
  scale_color_discrete(
    name = "Promotion level",
    labels = leg)
int_fig1
```




```
# コード 7-12
# install.packages("marginaleffects")

# コード 7-13
# library(marginaleffects)
int_fig2 <- plot_slopes(fit_int_c, variables = "rd_c",
  condition = "promotion_c", conf_level = .99999999) +
  labs(title = "Marginal effects of R&D on Sales",
       x = "Promotion", y = "Slope of R&D on Sales") +
  geom_hline(aes(yintercept = 0), linetype = "dashed")
int_fig2
```




```
# コード 7-14
firmdata19 <- firmdata19 %>%
  mutate(mkexp_c = mkexp - mean(mkexp),
         asset_c = total_assets - mean(total_assets))
fit_int2 <- lm(op ~ mkexp_c * asset_c, data = firmdata19)
int_fig3 <- plot_slopes(fit_int2, variables = "mkexp_c",
  condition = "asset_c", conf_level = 0.99) +
  geom_hline(aes(yintercept = 0), linetype = "dashed")
int_fig3
```




```
# コード 7-15
firmdata19 <- firmdata19 %>%
  mutate(adv = (adv - mean(adv)) / sd(adv),
         rd = (rd - mean(rd)) / sd(rd),
         ad_rd = adv + rd)
fit_linear <- lm(sales ~ adv + rd, data = firmdata19)
summary(fit_linear)
```

```

Call:
lm(formula = sales ~ adv + rd, data = firmdata19)

Residuals:
     Min       1Q   Median       3Q      Max 
-4357995  -463348  -236824   123663  2692251 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1199403      74285  16.146   <2e-16 ***
adv          1632326      74996  21.766   <2e-16 ***
rd             29915      74996   0.399    0.691    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 897600 on 143 degrees of freedom
Multiple R-squared:  0.7711,    Adjusted R-squared:  0.7679
F-statistic: 240.8 on 2 and 143 DF,  p-value: < 2.2e-16
```

```r
# コード 7-16
fit_comp <- lm(sales ~ adv + ad_rd, data = firmdata19)
summary(fit_comp)
```

```

Call:
lm(formula = sales ~ adv + ad_rd, data = firmdata19)

Residuals:
     Min       1Q   Median       3Q      Max
-4357995  -463348  -236824   123663  2692251

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1199403      74285  16.146   <2e-16 ***
adv          1602411     111739  14.341   <2e-16 ***
ad_rd          29915      74996   0.399    0.691
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 897600 on 143 degrees of freedom
Multiple R-squared:  0.7711,    Adjusted R-squared:  0.7679
F-statistic: 240.8 on 2 and 143 DF,  p-value: < 2.2e-16
```

```r
# コード 7-17
fit_prod <- lm(log(sales) ~ log(labor_cost) + log(ppent),
  data = firmdata19)

summary(fit_prod)
```

```

Call:
lm(formula = log(sales) ~ log(labor_cost) + log(ppent), data = firmdata19)

Residuals:
     Min       1Q   Median       3Q      Max
-1.40821 -0.36129 -0.00933  0.38591  1.66673
```

```
Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)      3.17129    0.31733   9.994   <2e-16 ***
log(labor_cost)  0.53739    0.03679  14.605   <2e-16 ***
log(ppent)       0.34588    0.02797  12.366   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5169 on 143 degrees of freedom
Multiple R-squared:  0.8747,    Adjusted R-squared:  0.8729 
F-statistic: 499.1 on 2 and 143 DF,  p-value: < 2.2e-16
```

```r
# コード 7-18
price <- readr::read_csv("data/price_data.csv")

fit_q <- lm(log(q) ~ log(p), data = price)
summary(fit_q)
```

```
Call:
lm(formula = log(q) ~ log(p), data = price)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.11220 -0.13668  0.02804  0.16166  0.51039 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  9.47781    0.08135   116.5   <2e-16 ***
log(p)      -0.18008    0.01354   -13.3   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2309 on 998 degrees of freedom
Multiple R-squared:  0.1506,    Adjusted R-squared:  0.1497 
F-statistic: 176.9 on 1 and 998 DF,  p-value: < 2.2e-16
```

```r
# コード 7-19
# install.packages("car")
```

```
# コード 7-20
# library(car)
linearHypothesis(fit_q, c("log(p)= -1"))
```

```
Linear hypothesis test:
log(p) = - 1

Model 1: restricted model
Model 2: log(q) ~ log(p)

  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1    999 248.811
2    998  53.225  1    195.59 3667.3 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# コード 7-21
# install.packages("modelsummary")

# コード 7-22
# library(modelsummary)
msummary(fit_int, statistic = 'conf.int')
```

```
# コード 7-23
msummary(fit_int, gof_omit = "Log.Lik.|AIC|BIC|RMSE")
```

```
# コード 7-24
var_nam <- c("rd" = "R&D", "promotion" = "Promotion",
             "rd:promotion" = "R&D * Promotion",
             "rd_c" = "R&D_c", "promotion_c" = "Promotion_c",
             "rd_c:promotion_c" = "R&D_c * Promotion_c",
             "(Intercept)" = "定数項")
Int <- list()
Int[["Without centering"]] <- fit_int
Int[["With centering"]] <- fit_int_c

msummary(Int,
```

	(1)
(Intercept)	20 326.588
	[20 111.797, 20 541.378]
rd	−51.872
	[−52.433, −51.312]
promotion	−191.431
	[−193.505, −189.358]
rd × promotion	0.498
	[0.493, 0.503]
Num.Obs.	221
R2	0.994
R2 Adj.	0.993
AIC	1969.2
BIC	1986.2
Log.Lik.	−979.617
F	11 110.648
RMSE	20.36

	(1)
(Intercept)	20 326.588
	(108.978)
rd	−51.872
	(0.284)
promotion	−191.431
	(1.052)
rd × promotion	0.498
	(0.003)
Num.Obs.	221
R2	0.994
R2 Adj.	0.993
F	11 110.648

Table 5.1: Comparing Interaction Models

	Without centering	With centering
R&D	−51.872***	
	[−52.433, −51.312]	
Promotion	−191.431***	
	[−193.505, −189.358]	
R&D * Promotion	0.498***	
	[0.493, 0.503]	
R&D_c		−0.918***
		[−0.941, −0.894]
Promotion_c		0.690***
		[0.612, 0.769]
R&D_c * Promotion_c		0.498***
		[0.493, 0.503]
定数項	20 326.588***	382.278***
	[20 111.797, 20 541.378]	[379.293, 385.263]
Num.Obs.	221	221
R2	0.994	0.994
R2 Adj.	0.993	0.993
F	11 110.648	11 110.648

+ p <0.1, * p <0.05, ** p <0.01, *** p <0.001
Values in [ ] show 95% confidence intervals

```
  coef_map = var_nam,
  title = "Comparing Interaction Models",
  notes = "Values in [ ] show 95% confidence intervals",
  stars = TRUE,
  statistic = 'conf.int', conf_level = .95,
  gof_omit = "Log.Lik.|AIC|BIC|RMSE")
```

```
# コード 7-25
msummary(Int,
  title = "Comparing Interaction Models",
```

Table 5.2: Comparing Interaction Models

	Without centering	With centering
(Intercept)	20 326.588***	382.278***
	[20 111.797, 20 541.378]	[379.293, 385.263]
rd	−51.872***	
	[−52.433, −51.312]	
promotion	−191.431***	
	[−193.505, −189.358]	
rd × promotion	0.498***	
	[0.493, 0.503]	
rd_c		−0.918***
		[−0.941, −0.894]
promotion_c		0.690***
		[0.612, 0.769]
rd_c × promotion_c		0.498***
		[0.493, 0.503]
Num.Obs.	221	221
R2	0.994	0.994
R2 Adj.	0.993	0.993
F	11 110.648	11 110.648

+ p <0.1, * p <0.05, ** p <0.01, *** p <0.001
Values in [ ] show 95% confidence intervals

```
notes = "Values in [ ] show 95% confidence intervals",
stars = TRUE,
statistic =  'conf.int', conf_level = .95,
gof_omit = "Log.Lik.|AIC|BIC|RMSE",
output = "latex")
```

第 6 章

第 8 章消費者の選択と離散選択モデル

6.1 顕示選好の理論

Kreps (1988) “Notes on the Theory of Choice” に基づいて，単一個人意思決定モデルの基礎的な考え方を説明します。

「選択」(choice) という現象は，ある集合 A がある個人にとっての実際に選択可能な選択肢の集合であったときに，その中から個人によって 1 つの選択肢 $a \in A$ が選ばれる，という現象をいいます。ここに，「選ぶ」というのは，第三者に観察できる行為として選ぶ，ということに限って用いられます。また，問題となっている集合 A の各要素 $a \in A$ を「選択肢」(alternative) と呼びます。

選択現象を問題にしている選択肢 a が確率的構造を持っているとき，その選択肢をとくに**クジ** (lottery) と呼びます。

説明するべき対象を選択肢とします。

$$X \supseteq A \mapsto a$$

ここで A は選択肢の集合で，a はその選択肢を選んだ結果となります。X は整合性を問題にできる選択肢集合のいちばん最大のもの，となります。

X 上の，この人の選択関数

$$C : X \subseteq A \mapsto C(A) \subseteq X$$

が安定的に存在すると仮定します。ここで $C(A)$ は，A の中からこの人が選んだ選択肢集合を表します。

X を考察する選択行動で，選択の対象となりうる潜在的な選択肢を集めた集合を表す。選択集合とは，ある $A \in \mathcal{P}(X)$ が実際に選択可能な選択肢の集合であったときに，その中から 1 つの選択肢が選ばれる現象をいいます。「選ばれる」といういうとき，現に選ばれた 1 つの選択肢以外にも「それでも同様に良かった」というものがあるとき，それらの「そのどれでも良かった」選択肢を全て集めた集合 C を，A からの選択結果とみなします。

選択関数を定義すると,
$C : \mathcal{P}(X) \to \mathcal{P}(X)$ が,

$$\forall A \in \mathcal{P}(X);\ C(A) \subseteq A$$

であるとき，X 上の選択関数 (choice function) である，といいます。ここで，$\mathcal{P}(X)$ は，X の部分集合 1 つ 1 つを全部集めてできる冪集合 (power set) のうち，X の中から二者択一の結果だけを集めたデータを表します。

6.1.1 選好モデル

二項関係 (binary relation) を定義します。X の（たまたま同じであることを許す）2 つの要素について語られた関係が，X 上の二項関係 (binary relation) であるとは，$X \times X$ のすべての要素 (x, y) について，x は y に対してその関係にあるか，またはその関係にないか，を必ず一方をいうことができる場合を言います。

つまり，選択肢集合 A に選択肢 x と y があり，その人の選択結果 $C(\{x, y\}) = ?$ を見ます。その人が選択するものは,

$$C(\{x,y\}) = \phi \quad x \text{ も } y \text{ も選べたのに選ばなかった。}$$
$$C(\{x,y\}) = \{x\} \quad x\text{を選び}, y\text{を選ばなかった}$$
$$C(\{x,y\}) = \{y\} \quad y\text{を選び}, x\text{を選ばなかった}$$
$$C(\{x,y\}) = \{x,y\} \quad x\text{も}y\text{も選んだ。}$$

6.1.2 選好

X 上の選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ が与えられたとき，X の (それがたまたま同じであることを許す)2 つの要素についての関係を，各 $(x, y) \in X \times X$ に対して，$y \notin C(\{x, y\})$ なら，x は y に対してその関係にあり，$y \in C(\{x, y\})$ なら x は y に対してその関係にない，というやり方で定められるとき，その関係 (それは二項関係である) を，(選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ によって生成された) X 上の**強選好** (strict preference) といいます。

この二項関係を記号 $\succ$ で表わし，x が y に対して強選好の関係にあるとき，x を y よりも厳密に選好する，あるいは x は y よりも厳密に選好されるといって，$x \succ y$ と書き，x が y に対して強選好の関係にないとき，x は y よりも厳密に選好されるわけではない，といって $x \not\succ y$ と書きます。

ある X 上の強選好 $\succ$ は，それが定義されたもとの選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ を明示する必要があるときは，$\succ_c$ と書く。

x, y の二者択一の結果，$C(\{x, y\})$ を,

$$x \not\succ y \Longleftrightarrow y \in C(\{x,y\}) \leftarrow x \text{ は } y \text{ より厳密に選好されるわけではない。}$$
$$x \succ y \Longleftrightarrow y \notin C(\{x,y\}) \leftarrow x\text{は}y\text{より厳密に選好される。}$$

のように書いたものを，この人の (強) 選好という。

つぎに、X 上の強選好 $\succ$ が与えられたとき、各 $A \in \mathcal{P}(X)$ に対し、

$$C(A, \succ) = \{x \in A \mid \nexists y \in A \quad \text{s.t.} \quad y \succ x\}$$

を対応づける（集合体集合の）関数 $C(\cdot, \succ): \mathcal{P}(X) \to \mathcal{P}(X)$ を、X 上の選択の**選好モデル** (preference model) といいます。

まとめると、

$$C: \mathcal{P}(X) \to C(A) \in \mathcal{P}(X)$$

のうち、「二者択一の結果を、X の各ペア (x, y) ごとに、$x \succ y$ か $x \not\succ y$ かを書いて、書いたものを X 上の（この人の）強選好」といいます。選好モデルとは、

$$C: \mathcal{P}(X) \ni A \to C(A) \in \mathcal{P}(X)$$

を

$$\begin{aligned} &C(\cdot, \succ): \mathcal{P}(X) \ni A \to C(A, \succ) \in \mathcal{P}(X) \\ &\text{such that} \quad C(A, \succ) = \{x \in A \mid \nexists y \in A \quad \text{s.t.} \quad y \succ x\} \end{aligned}$$

となり、この選好 $\succ$ をもつ人が、実際に A に直面したときに選ぶと予測するということを意味します。これは $C(\cdot, \succ)$ のデータから予測されます。

$C(\cdot, \succ): \mathcal{P}(X) \ni A \to C(A, \succ) \in \mathcal{P}(X)$ は、$C: \mathcal{P}(X) \ni A \to C(A) \in \mathcal{P}(X)$ のうち、$|X|^2$ 個のデータで、$C: \mathcal{P}(X) \ni A \to C(A) \in \mathcal{P}(X)$ 全体の $2^{|X|}$ 個の値を予測しているのです。

合理性モデルでは、「もし・・・なら、$\forall A \in \mathcal{P}; C(A, \succ) = C(A)$ となります。

ふりかえると、

- 説明する対象は、選択関数 $C: \mathcal{P}(X) \in A \mapsto C(A) \in \mathcal{P}(X)$ である。
- データを取ってくるパラメータとしての選好 $\succ$ (つまり、X 上のすべての二者択一の結果) を考える。
- 選択を説明するモデルは選好モデル $C(A, \succ) = \{x \in A \mid \nexists y \in A \quad \text{s.t.} \quad y \succ x\}$
- 選択理論とは、「$C: \mathcal{P}(X) \to \mathcal{P}(X)$ が・・・の条件を満たすとき、C のうち ××から作った$C(A, ××) = C(A)」が成り立つような「･･･」と「××」を考えます。

ただ、選好 $\succ$ としてどんなものでも認めてしまうと、

$$\exists \succ, \exists A \quad \text{s.t.} \quad C(A, \succ) = \emptyset$$

となる場合があります。たとえば、$A = X = \{x, y, z\}$ で、$C(\{x, y\}) = \{x\}$、$C(\{y, z\}) = \{y\}$、$C(\{x, z\}) = \{z\}$ かつ、$C(A) \neq \emptyset$. 場合、$C(\{x, y, z\}) = \emptyset$ となります。

つまり、

- $z \succ x$ より $x \notin C(\{x, y, z\}, \succ)$
- $x \succ y$ より $y \notin C(\{x, y, z\}, \succ)$
- $y \succ z$ より $z \notin C(\{x, y, z\}, \succ)$

すなわち、$C(\{x, y, z\}, \succ) = \emptyset$ となります。

て、次のように選択関数 $C_{\succ}: \mathcal{P}(X) \to \mathcal{P}(X)$ を定めます。

では、どのような二者択一行動を示す人なら、その人の選好モデルは完結性があるのかを考えます。次の 2 つの条件が満たされるとき、選好 $\succ$ は**完結性** (completeness) を持つといいます。

1. **反対称性** (asymmetry)：$\forall x, y \in X;$,「$x \succ y$ かつ $y \succ x$」ということはない。つまり二者択一行動が $C(\{x, y\}) = \{\emptyset\}$ ということがない。
2. **負推移性** (negative transitivity)： $\forall x, y \in X;$,「もし $x \nsucc y$ かつ $y \nsucc z$」ならば「$x \nsucc z$」である。つまり二者択一行動としては、もし $y \in C(\{x, y\})$ かつ $z \in C(\{y, z\})$ ならば、$z \in C(\{x, z\})$ である。

これより、X 上の強選好 $\succ$ は、次のように定義できます。

! 反対称性・負推移性

$$\nexists (x, y) \in X \times X; \quad \text{s.t} \quad x \succ y, \quad y \succ x$$

であるとき、**反対称的** (assymmetric) である、あるいは反対称性を満たすといい、

$$\forall x, y, z \in X; \quad \text{if} \quad x \nsucc y, \quad y \nsucc z, \quad \text{then} \quad x \nsucc z$$

であるとき、**負推移的** (negatively transitive) である、あるいは負推移性を満たすといいます

! 選好関係

X 上の強選好 $\succ$ は、反対称的かつ負推移的であるとき、X 上の**選好関係** (preference relation) であるといいます。

選好 $\succ$ が反対称的かつ負推移的であるとき、**推移的** (acyclic) であるといい、$x_1 \succ x_2$ かつ $x_2 \succ x_3$ かつ $\cdots$ かつ $x_{n-1} \succ x_n$ なら $x_1 \neq x_n$ となります。

6.1.3 X が有限集合の場合

X 上の強選好 $\succ$ が

$$\forall x \in X; \quad x \nsucc x$$

であるとき、**非反射的** (irreflexive) である、あるいは非反射性を満たすといいます。

i 推移性

X 上の強選好 $\succ$ は、

$$\forall x, y, z \in X; \quad \text{if} \quad x \succ y, \ y \succ x, \quad \text{then} \quad x \succ z$$

であるとき、**推移的** (transitive) である、あるいは推移性を満たすといいます。
証明は省略（背理法により示せます）。

非周期性

次に、X 上の強選好 $\succ$ は、

$$
\begin{aligned}
&\forall x_1, x_2, \cdots, x_n \in X; \\
&\quad \text{if} \quad x_1 \succ x_2,\ x_2 \succ x_3,\ \ldots,\ x_{n-1} \succ x_n, \quad \text{then} \quad x_1 \neq x_n
\end{aligned}
$$

であるとき、**非周期的** (acyclic) である、あるいは非周期性を満たすといいます。
証明は省略（帰納法により示せます）。

Important

X 上の強選好 $\succ$ は、X 上の選好関係ならば、非反射的かつ推移的かつ非周期的である。

X 上の選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ は、

$$\forall A \in \mathcal{P}(X); \quad \text{if} \quad A \neq \emptyset \quad \text{then} \quad C(A) \neq \emptyset$$

であるとき、**非空** (nonempty) であるといいます。
これらの定理と定義から、次の定理が示せます。

Important

X を有限集合 ($|X| < \infty$) とします。X 上の強選好 $\succ$ は、X 上の (反対称的と負推移性を満たす) 選好関係ならば、$A \neq \emptyset$ のとき、$C(\cdot, \succ) : \mathcal{P}(X) \to \mathcal{P}(X)$ は非空となります。

が選好関係であるとき、X が有限集合であれば、任意の $A \in \mathcal{P}(X)$ に対し、$C(A, \succ) \neq \emptyset$ となります。

ここまでの説明により、ある個人の選択行動を説明するために必要な条件は、ある個人が (有限の) 選択肢の中から選び取るとき、その人の選好が反対称的かつ負推移的であれば、選択肢集合が非空なら、その人がもつ選択関数は非空である、ということがわかりました。

6.2　顕示選考理論

選択肢集合 X が有限であるとします。すると、X 上の強選好 $\succ$ が X 上の選好関係であるならば、$C(\cdot, \succ) : \mathcal{P}(X) \to \mathcal{P}(X)$ は非空まつ**ハウタッカーの公理** (Houthakker's axiom) を満たします。

ハウタッカーの公理

X 上の選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ は、

$$\begin{aligned} &\forall x, y \in X, \quad \forall A, B \in \mathcal{P}(X); \\ &\quad \text{if} \quad x, y \in A \cap B, \quad x \in C(A), \quad y \in C(B) \\ &\quad \text{then} \quad x \in C(B), \quad y \in C(A) \end{aligned}$$

であるとき、**ハウタッカーの公理**を満たす、といいます。

具体例で説明します。たとえば、あなたの買い物の選択肢として

- $A = \{$みかん, リンゴ, バナナ$\}$、
- $B = \{$バナナ, リンゴ, ぶどう$\}$

があるとします。

このとき、あなたは

- A の中からリンゴを選び（$C(A) = \{$リンゴ$\}$）、
- B の中からバナナを選ぶ（$C(B) = \{$バナナ$\}$）

とします。

リンゴとバナナはいずれも $A \cap B$ に含まれています。ハウタッカーの公理は、この状況のもとでは、選択肢集合が B に変わってもリンゴが選ばれ、選択肢集合が A に変わってもバナナが選ばれなければならない、ということを要請します。

つまり、共通して含まれる選択肢については、集合ごとに選択の優劣が入れ替わるような行動は許されない、という選択の一貫性を表しています。

センの α 規則性

X 上の選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ は、

$$\begin{aligned} &\forall x \in X, \quad \forall A, B \in \mathcal{P}(X); \\ &\quad \text{if} \quad x \in B \subset A, \quad x \in C(A), \\ &\quad \text{then} \quad x \in C(B) \end{aligned}$$

であるとき、センの α **規則性** (Sen's property α) を満たす、といいます。

センの α 規則性の具体例

あなたの選択肢の全集合を X とし、$A = \{$みかん, リンゴ, バナナ$\}$、$B = \{$リンゴ, バナナ$\}$ とします。このとき $B \subset A$ が成り立っています。

いま、あなたが集合 A からリンゴを選んでいる、すなわち $C(A) = \{$リンゴ$\}$ であるとします。リンゴは B にも含まれています。

センの α 規則性は、この状況において、選択肢集合が A からその部分集合である B に縮小されても、リンゴは引き続き選ばれなければならない、すなわち

$C(B) = \{$リンゴ$\}$ であることを要請します。
言い換えると、ある選択肢がより大きな集合の中で選ばれていたのであれば、その選択肢を排除しない範囲で選択肢集合を減らしても、その選択肢が選ばれなくなることは許されない、という性質を表しています。

センの β 規則性

X 上の選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ は、

$$\begin{aligned} &\forall x, y \in X, \quad \forall A, B \in \mathcal{P}(X); \\ &\quad \text{if} \quad x, y \in C(A), \quad A \subset B, \quad y \in C(B), \\ &\quad \text{then} \quad x \in C(B) \end{aligned}$$

であるとき、センの β **規則性** (Sen's property β) を満たす、といいます。

センの β 規則性の具体例

選択肢の全集合を X とし、$A = \{$リンゴ, バナナ$\}$、$B = \{$みかん, リンゴ, バナナ$\}$ とします。このとき $A \subset B$ が成り立っています。
いま、集合 A からはリンゴとバナナの両方を選んでいる、すなわち $C(A) = \{$リンゴ, バナナ$\}$ とします。
次に、選択肢が増えた集合 B からはバナナを選んでいる、すなわち $C(B) = \{$バナナ$\}$ であるとします。
センの β 規則性は、この状況のもとでは、集合 A で同時に選ばれていたリンゴとバナナのうち、集合 B に移ってもバナナが選ばれているのであれば、リンゴもまた集合 B から選ばれなければならない、ということを要請します。
したがって、この規則性を満たす選択関数では、$C(B)$ は $\{$バナナ$\}$ ではなく、少なくともリンゴを含み、たとえば $C(B) = \{$リンゴ, バナナ$\}$ でなければならない、ということになります。
この例は、集合を拡大したときに、もともと同時に選ばれていた選択肢の一部だけが残るような振る舞いを排除する性質を具体的に示しています。

これらの公理と規則性から次の定理が示せます。

定理

X 上の選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ が非空とします。このとき、$C : \mathcal{P}(X) \to \mathcal{P}(X)$ がハウタッカーの公理を満たすための必要十分条件は、それがセンの α 規則性と β 規則性を満たすことです。

具体例で説明してみます。

$X = \{$みかん, リンゴ, バナナ, ぶどう$\}$ とします。あなたの好みが

$$\text{リンゴ} \sim \text{バナナ} \succ \text{みかん} \succ \text{ぶどう}$$

だとします。つまりリンゴとバナナは同程度に最も好ましく、次にみかん、最後にぶどう、という順番です。

この好みに基づく選択関数 C を、「与えられた集合の中の "最も好ましいもの" をすべて選ぶ」と定めます。つまり任意の $A \subseteq X$ に対して

- A にリンゴまたはバナナが含まれていれば、$C(A) = A \cap \{$リンゴ, バナナ$\}$
- そうでなければ（リンゴもバナナも無いなら)、残りの中で最上位のもの（ここではみかん、なければぶどう）を選ぶ

というルールです。たとえば

- $C(\{$みかん, リンゴ, バナナ$\}) = \{$リンゴ, バナナ$\}$,
- $C(\{$バナナ, ぶどう$\}) = \{$バナナ$\}$,
- $C(\{$みかん, ぶどう$\}) = \{$みかん$\}$

のようになります。

この選択関数 C はセンの α 規則性を満たします。理由は、ある集合 A で選ばれた x は A の中で「最上位」なので、x を含む部分集合 B ⊂ A に選択肢を減らしても、x より上位のものが新たに現れることはなく、x は依然として最上位（最大元）だからです。

また　規則性も満たします。実際、ある集合 A で x,y ∈ C(A) ということは、x と y がともに A の中で最上位（この例ではリンゴとバナナのように同率首位）ということです。集合を広げた B ⊃ A でも、もし y が依然として選ばれているなら（y ∈ C(B))、y より上位の選択肢は B に存在しないので、同率首位である x も排除されず x ∈ C(B) となります。

ここから、ハウタッカーの公理を満たしていることを、同じ果物で具体的に確かめます。たとえば

A={みかん, リンゴ, バナナ},　B={リンゴ, バナナ, ぶどう}

とおくと、

C(A)={リンゴ, バナナ},　C(B)={リンゴ, バナナ}

です。いま x=リンゴ, y=バナナ とすると、確かに x,y ∈ A ∩ B かつ x ∈ C(A)、y ∈ C(B) が成り立ちます。そして結論として要求される x ∈ C(B)、y ∈ C(A) も実際に成り立っています（どちらの集合でもリンゴとバナナが選ばれているため)。

また、X 上の選択関数 $C : \mathcal{P}(X) \to \mathcal{P}(X)$ が非空 ($A \neq \emptyset$ なら $C(A) \neq \emptyset$) かつハウタッカーの公理を満たすならば、$\succ_c$ は X 上の選好関係 $\succ$ であり (反対称性と負推移性を満たす) であり、しかも $A \neq \emptyset$ ならば、

$$\forall A \in \mathcal{P}(X); \quad C(A, \succ_c) = C(A)$$

が成り立ちます。

```
pacman::p_load(tidyverse, readxl, modelsummary, mfx, DescTools,
 ↪  mlogit)
```

```
# コード 8-1
df1 <- data.frame(
    Y = c(0, 0, 0, 0, 0, 1, 1, 1, 1),
    X = c(3.4, 5.22, 7.06, 2.81, 4.11, 10.34, 13.67, 15.99, 9.09)
  )
lpm1 <- lm(Y ~ X, data = df1)
pred_lpm1 <- predict(lpm1)
pred_lpm1
```

```
           1            2            3            4            5            6
-0.006342894  0.173357681  0.355032986 -0.064597476  0.063760077  0.678888967
           7            8            9
 1.007681776  1.236750640  0.555468243
```

```
# コード 8-2
choice_df <- readxl::read_xlsx("data/choice_data.xlsx")
head(choice_df)
```

```
# A tibble: 6 x 6
     y1    y2    p1    p2    a1    a2
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     1     0 10.7  16.8      0     1
2     0     1 11.6   9.15     1     0
3     1     0  8.97  9.31     1     0
4     1     0  4.62  8.03     0     0
5     1     0  7.81 19.2      1     1
6     1     0  7.49  8.84     0     0
```

```
# コード 8-3
probit1 <- glm(y1 ~ p1 + a1,
               family = binomial(link = probit),
               data = choice_df)
summary(probit1)
```

```

Call:
glm(formula = y1 ~ p1 + a1, family = binomial(link = probit),
    data = choice_df)
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.04117    0.22350   9.133  < 2e-16 ***
p1          -0.24720    0.02213 -11.171  < 2e-16 ***
a1           0.62763    0.08656   7.251 4.13e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1384.5  on 999  degrees of freedom
Residual deviance: 1205.5  on 997  degrees of freedom
AIC: 1211.5

Number of Fisher Scoring iterations: 3
```

```
# コード 8-4
ols1 <- lm(y1 ~ p1 + a1,
           data = choice_df)
logit1 <- glm(y1 ~ p1 + a1,
              family = binomial(link = logit),
              data = choice_df)
models <- list()
models[["Linear probability model"]] <- ols1
models[["Probit model"]] <- probit1
models[["Logit model"]] <- logit1
modelsummary(models,
             title = "モデル比較",
             notes = "Values in [ ] show robust standard errors",
             stars = TRUE,
             statistic = "std.error",
             vcov = "robust",
             gof_map = "nobs")
```

```
# 適合度指標において，サンプルサイズ（"nobs"）のみ表示するという指示
```

```
# コード 8-5
# Average Marginal Effects（限界効果の平均）
```

Table 6.1: モデル比較

	Linear probability model	Probit model	Logit model
(Intercept)	1.196***	2.041***	3.332***
	(0.065)	(0.212)	(0.361)
p1	−0.085***	−0.247***	−0.404***
	(0.006)	(0.021)	(0.036)
a1	0.221***	0.628***	1.035***
	(0.029)	(0.087)	(0.145)
Num.Obs.	1000	1000	1000

+ p <0.1, * p <0.05, ** p <0.01, *** p <0.001

Values in [ ] show robust standard errors

```
probit1_ame <- probitmfx(y1 ~ p1 + a1,
                         data = choice_df, atmean = FALSE)
probit1_ame
```

```
Call:
probitmfx(formula = y1 ~ p1 + a1, data = choice_df, atmean = FALSE)

Marginal Effects:
        dF/dx  Std. Err.        z     P>|z|
p1 -0.0847786  0.0060787 -13.9469 < 2.2e-16 ***
a1  0.2188866  0.0289816   7.5526 4.266e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

dF/dx is for discrete change for the following variables:

[1] "a1"
```

```
# コード 8-7
# Marginal Effects at Mean（平均値における限界効果）
probit1_mem <- probitmfx(y1 ~ p1 + a1,
                         data = choice_df, atmean = TRUE)
probit1_mem
```

```
Call:
```

```
probitmfx(formula = y1 ~ p1 + a1, data = choice_df, atmean = TRUE)

Marginal Effects:
        dF/dx  Std. Err.        z     P>|z|
p1 -0.0984235  0.0088074 -11.1750 < 2.2e-16 ***
a1  0.2446609  0.0324366   7.5427 4.602e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

dF/dx is for discrete change for the following variables:

[1] "a1"
```

```
# コード 8-8
marginal <- c(
  "p1 marginal" = "p1",
  "a1 marginal" = "a1"
)

marg_model <- list(
  "(1) Probit_AME" <- probit1_ame,
  "(2) Probit_MEM" <- probit1_mem
)

marg_sum <- modelsummary(
  marg_model,
  statistic = "std.error",
  coef_map = marginal,
  stars = TRUE,
  shape = term:component ~ model,
  add_rows = data.frame("Marginal effect type",
                        "Average Marginal Effect",
                        "Marginal Effect at Mean"),
  title = "限界効果サマリー",
  gof_map = "nobs"
)
marg_sum
```

Table 6.2: 限界効果サマリー

	(1)	(2)
p1	−0.085***	−0.098***
	(0.006)	(0.009)
a1	0.219***	0.245***
	(0.029)	(0.032)
Num.Obs.	1000	1000
Marginal effect type	Average Marginal Effect	Marginal Effect at Mean

+ p <0.1, * p <0.05, ** p <0.01, *** p <0.001

```
# コード 8-9
DescTools::PseudoR2(probit1)
```

```
 McFadden
0.1293345
```

```
# コード 8-10

# 価格差変数作成
choice_df <- choice_df %>%
  mutate(p_ratio = p1 - p2)

probit2 <- glm(y1 ~ p_ratio + a1 + a2,
               family = binomial(link = probit),
               data = choice_df)
summary(probit2)
```

```

Call:
glm(formula = y1 ~ p_ratio + a1 + a2, family = binomial(link = probit),
    data = choice_df)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.54634    0.07756  -7.044 1.87e-12 ***
p_ratio     -0.22180    0.01524 -14.559  < 2e-16 ***
a1           0.66314    0.09162   7.238 4.55e-13 ***
a2          -0.32291    0.09884  -3.267  0.00109 **
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1384.5  on 999  degrees of freedom
Residual deviance: 1066.9  on 996  degrees of freedom
AIC: 1074.9

Number of Fisher Scoring iterations: 4
```

```r
# コード 8-11
DescTools::PseudoR2(probit2)
```

```
McFadden
0.229384
```

```r
# コード 8-12
probit2_ame <- probitmfx(y1 ~ p_ratio + a1 + a2,
                         data = choice_df, atmean = FALSE)
probit2_ame
```

```
Call:
probitmfx(formula = y1 ~ p_ratio + a1 + a2, data = choice_df,
    atmean = FALSE)

Marginal Effects:
             dF/dx  Std. Err.        z     P>|z|
p_ratio -0.0669199  0.0029405 -22.7581 < 2.2e-16 ***
a1       0.2036848  0.0270475   7.5306 5.049e-14 ***
a2      -0.0969891  0.0291848  -3.3233 0.0008897 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

dF/dx is for discrete change for the following variables:

[1] "a1" "a2"
```

```r
# コード 8-13
data(Cracker, package = "mlogit")
head(Cracker, n = 20)
```

```
   id disp.sunshine disp.keebler disp.nabisco disp.private feat.sunshine
1   1             0            0            0            0             0
2   1             0            0            0            0             0
3   1             1            0            0            0             0
4   1             0            0            0            0             0
5   1             0            0            0            0             0
6   1             0            0            0            0             0
7   1             0            0            1            0             0
8   1             0            0            1            0             0
9   1             0            0            1            0             0
10  1             1            0            1            0             0
11  1             0            0            1            0             0
12  1             0            0            0            0             0
13  1             1            0            0            0             0
14  1             0            1            1            0             0
15  1             0            0            0            0             0
16  1             0            0            1            0             0
17  2             0            0            0            0             0
18  2             1            0            1            0             1
19  2             1            0            0            0             1
20  2             1            0            0            0             0
   feat.keebler feat.nabisco feat.private price.sunshine price.keebler
1             0            0            0             98            88
2             0            0            0             99           109
3             0            0            0             49           109
4             0            0            0            103           109
5             0            0            0            109           109
6             0            0            0             89           109
7             0            0            0            109           109
8             0            0            0            109           119
9             0            0            0            109           121
10            0            0            0             79           121
11            0            0            0            109           113
12            0            0            0            109           121
13            0            0            0             89           121
14            0            0            0            109           109
15            0            0            0            109           109
16            0            0            0            129           104
17            0            0            0             79            99
```

```
18           0           0           0          69          105
19           0           0           0          79          125
20           0           0           0          79          125
   price.nabisco price.private   choice
1            120            71  nabisco
2             99            71  nabisco
3            109            78 sunshine
4             89            78  nabisco
5            119            64  nabisco
6            119            84  nabisco
7            129            78 sunshine
8            129            78  nabisco
9            109            78  nabisco
10           109            78  nabisco
11           109            96  nabisco
12            99            86  nabisco
13            99            86  nabisco
14           129            96  nabisco
15           129            79  nabisco
16           129            96  nabisco
17            69            69  nabisco
18            89            65 sunshine
19           106            69 sunshine
20           106            69 sunshine
```

```
# コード 8-14
cracker <- mlogit.data(Cracker, choice = "choice", shape = "wide",
                       varying = c(2:13))
head(cracker, n = 20)
```

```
~~~~~~~
 first 20 observations out of 13168
~~~~~~~
   id choice      alt disp feat price chid    idx
1   1  FALSE  keebler    0    0    88    1 1:bler
2   1   TRUE  nabisco    0    0   120    1 1:isco
3   1  FALSE  private    0    0    71    1 1:vate
4   1  FALSE sunshine    0    0    98    1 1:hine
5   1  FALSE  keebler    0    0   109    2 2:bler
6   1   TRUE  nabisco    0    0    99    2 2:isco
```

```
7   1  FALSE  private    0    0    71    2 2:vate
8   1  FALSE sunshine    0    0    99    2 2:hine
9   1  FALSE  keebler    0    0   109    3 3:bler
10  1  FALSE  nabisco    0    0   109    3 3:isco
11  1  FALSE  private    0    0    78    3 3:vate
12  1   TRUE sunshine    1    0    49    3 3:hine
13  1  FALSE  keebler    0    0   109    4 4:bler
14  1   TRUE  nabisco    0    0    89    4 4:isco
15  1  FALSE  private    0    0    78    4 4:vate
16  1  FALSE sunshine    0    0   103    4 4:hine
17  1  FALSE  keebler    0    0   109    5 5:bler
18  1   TRUE  nabisco    0    0   119    5 5:isco
19  1  FALSE  private    0    0    64    5 5:vate
20  1  FALSE sunshine    0    0   109    5 5:hine

~~~ indexes ~~~~
   chid      alt
1      1  keebler
2      1  nabisco
3      1  private
4      1 sunshine
5      2  keebler
6      2  nabisco
7      2  private
8      2 sunshine
9      3  keebler
10     3  nabisco
11     3  private
12     3 sunshine
13     4  keebler
14     4  nabisco
15     4  private
16     4 sunshine
17     5  keebler
18     5  nabisco
19     5  private
20     5 sunshine
indexes:  1, 2
```

```r
# コード 8-15
ml_cracker <- mlogit(choice ~ price | 1 | disp + feat, probit = FALSE,
                     data = cracker)

summary(ml_cracker)
```

```
Call:
mlogit(formula = choice ~ price | 1 | disp + feat, data = cracker,
    probit = FALSE, method = "nr")

Frequencies of alternatives:choice
 keebler  nabisco  private sunshine
0.068651 0.544350 0.314399 0.072600

nr method
5 iterations, 0h:0m:0s
g'(-H)^-1g = 0.000258
successive function values within tolerance limits

Coefficients :
                      Estimate Std. Error  z-value  Pr(>|z|)
(Intercept):nabisco   2.0011226  0.0831336  24.0712 < 2.2e-16 ***
(Intercept):private   0.3193793  0.1238052   2.5797 0.0098888 **
(Intercept):sunshine -0.5435987  0.1139407  -4.7709 1.834e-06 ***
price                -0.0300966  0.0021082 -14.2761 < 2.2e-16 ***
disp:keebler          0.2999316  0.2070369   1.4487 0.1474251
disp:nabisco          0.1011111  0.0773633   1.3070 0.1912249
disp:private         -0.2244555  0.1495236  -1.5011 0.1333199
disp:sunshine         0.4818670  0.1672400   2.8813 0.0039605 **
feat:keebler          0.6678542  0.2581732   2.5868 0.0096859 **
feat:nabisco          0.6047773  0.1404077   4.3073 1.653e-05 ***
feat:private          0.1726981  0.2004446   0.8616 0.3889213
feat:sunshine         0.8304895  0.2340938   3.5477 0.0003886 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-Likelihood: -3333.8
McFadden R^2:  0.052798
```

```
Likelihood ratio test : chisq = 371.66 (p.value = < 2.22e-16)
```

```
# コード 8-16
# パラメータの推定値（beta）の抽出
beta <- probit2$coefficients

# パラメータ beta と説明変数の線形結合を作成（beta_0 は定数項なので 1 をか
↪ ける）
est_probit2 <- cbind(1, choice_df$p_ratio, choice_df$a1,
                     choice_df$a2) %*% beta

# pnorm によって標準正規分布の分布関数による計算を実行する。
# 製品 1(c1) を選ぶ確率の予測値
pred_probit_c1 <- pnorm(est_probit2)
mean(pred_probit_c1)
```

```
[1] 0.4789384
```

```
# コード 8-17
# ロジットモデルの場合
logit2 <- glm(y1 ~ p_ratio + a1 + a2,
              family = binomial(link = logit),
              data = choice_df)
beta_logit <- logit2$coefficients
est_logit2 <- cbind(1, choice_df$p_ratio, choice_df$a1,
                    choice_df$a2) %*% beta_logit
pred_logit_c1 <- (exp(est_logit2)) / (1 + exp(est_logit2))
mean(pred_logit_c1)
```

```
[1] 0.479
```

```
# コード 8-18
choice_df_v <- choice_df %>%
  mutate(p1_v = p1 - 1,
         p_ratio_v = p1_v - p2)
est_probit_v <- cbind(1, choice_df_v$p_ratio_v, choice_df_v$a1,
                      choice_df_v$a2) %*% beta

# pnorm によって標準正規分布の分布関数による計算を実行する。
# 製品 1(c1) を選ぶ確率の予測値
pred_probit_v <- pnorm(est_probit_v)
```

```
mean(pred_probit_v)
```

```
[1] 0.5459071
```

第 III 部

第 3 部 探索型データ分析

第 9 章では観察重視アプローチ、第 10 章ではセグメントとクラスター分析を、第 11 章では探索的因子分析、第 12 章では価格感応度測定 (PSM) について学びます。第 9 章は観察重視アプローチと理論重視アプローチを学習しますが、R コードが出てこないので、ここでは割愛します。

第 7 章

第 10 章セグメントとクラスター分析

クラスター分析 (cluster analysis) は、観測値をいくつかのグループに分ける手法です。ここでは、階層的クラスター分析と非階層的クラスター分析 (K-means 法) を学習します。

7.1 階層的クラスター分析

データの中から類似している観測値を段階的にクラスターとしてまとめて、最終的に 1 つのクラスターになるまで繰り返す方法です。類似度は距離で測定され、距離が近いほど類似しているとみなされます。距離概念として、

- ユークリッド距離 (Euclidean distance)
- マンハッタン距離 (Manhattan distance)
- マハラノビス距離 (Mahalanobis distance)

などがあります。デンドログラム (dendrogram) という樹形図を使って、クラスターの結合過程を視覚的に表現します。

2023 年実施の消費者調査アンケートデータを使って、階層的クラスター分析を実行してみます。必要なパッケージを読み込みます。

- `tidyverse`: データ操作と可視化
- `readxl`: Excel データの読み込み
- `cluster`: クラスター分析
- `factoextra`: クラスター分析の可視化
- `ggrepel`: グラフ上のラベル配置
- `useful`: 便利な関数群

そしてデータを読み込みます。`read_xlsx()` 関数の引数として,

- `path`: データファイルのパスとファイル名
- `sheet`: 読み込むシート名
- `na`: 欠損値を表す文字列

```
pacman::p_load(tidyverse, readxl, cluster, factoextra, ggrepel,
 ↪  useful)
# コード 10-1
df_cons <- read_xlsx(
  path = "data/回答データ【消費者調査 2023 年度下期調査】.xlsx",
  sheet = "回答データ【共通調査 2023 年度下期】",
  na = " "
  )
```

読み込んだデータを str() で確認します。

```
df_cons |>str()
```

```
tibble [5,364 x 992] (S3: tbl_df/tbl/data.frame)
 $ 2023 年下期 no: num [1:5364] 1 2 3 4 7 8 9 10 11 12 ...
 $ 県番号        : num [1:5364] 13 13 27 14 13 13 14 11 25 14 ...
 $ 地域分類      : num [1:5364] 3 3 6 3 3 3 3 3 6 3 ...
 $ 性別          : num [1:5364] 2 1 1 1 1 1 2 2 1 2 ...
 $ 年齢          : num [1:5364] 38 42 25 26 30 33 52 53 36 26 ...
 $ 年齢階層      : num [1:5364] 3 4 2 2 3 3 5 5 3 2 ...
 $ 性年代        : num [1:5364] 11 4 2 2 3 3 13 13 3 10 ...
 $ 結婚有無      : num [1:5364] 2 2 1 1 1 1 1 2 1 2 ...
 $ q1            : num [1:5364] 2 1 1 1 1 1 2 2 1 2 ...
 $ q2t           : num [1:5364] 38 42 25 26 30 33 52 53 36 26 ...
 $ q3            : num [1:5364] 13 13 27 14 13 13 14 11 25 14 ...
 $ q4            : num [1:5364] 1 1 2 2 2 2 2 1 2 1 ...
 $ q5            : num [1:5364] 8 2 7 3 3 3 3 8 4 4 ...
 $ q6            : num [1:5364] 2 2 3 4 1 1 1 2 1 2 ...
 $ q7_1          : num [1:5364] 1 8 2 10 3 5 6 1 5 5 ...
 $ q7_2          : num [1:5364] 4 9 3 10 NA NA NA 4 NA 8 ...
 $ q8            : num [1:5364] 4 4 2 5 2 2 4 1 4 4 ...
 $ q9            : num [1:5364] 1 4 1 2 2 4 1 5 6 5 ...
 $ q10           : num [1:5364] 4 7 2 7 1 1 3 5 NA 5 ...
 $ q11_1         : num [1:5364] 5 4 1 3 4 2 4 2 3 2 ...
 $ q11_2         : num [1:5364] 4 5 2 3 4 2 3 2 3 2 ...
 $ q11_3         : num [1:5364] 4 5 2 3 3 2 4 2 3 2 ...
 $ q11_4         : num [1:5364] 4 4 3 3 4 2 4 2 3 2 ...
 $ q11_5         : num [1:5364] 4 4 3 4 4 2 4 2 3 2 ...
 $ q11_6         : num [1:5364] 4 4 3 3 3 2 4 2 3 2 ...
 $ q11_7         : num [1:5364] 4 5 2 3 4 1 4 2 3 2 ...
```

```
 $ q11_8       : num [1:5364] 4 4 3 3 4 1 3 2 3 2 ...
 $ q11_9       : num [1:5364] 4 4 3 2 4 2 5 2 3 3 ...
 $ q11_10      : num [1:5364] 4 5 4 3 3 2 3 2 3 3 ...
 $ q11_11      : num [1:5364] 4 5 3 3 4 2 4 2 3 2 ...
 $ q11_12      : num [1:5364] 4 4 3 3 3 1 4 2 3 2 ...
 $ q11_13      : num [1:5364] 4 4 2 3 3 1 3 3 3 2 ...
 $ q11_14      : num [1:5364] 4 4 2 3 4 1 4 4 3 2 ...
 $ q11_15      : num [1:5364] 4 4 3 4 4 2 3 4 3 2 ...
 $ q11_16      : num [1:5364] 4 4 3 3 4 2 5 3 3 2 ...
 $ q11_17      : num [1:5364] 4 5 3 3 5 2 4 3 3 2 ...
 $ q11_18      : num [1:5364] 4 4 2 3 4 2 4 3 3 3 ...
 $ q11_19      : num [1:5364] 4 5 2 3 5 2 5 3 3 4 ...
 $ q11_20      : num [1:5364] 4 4 3 3 4 2 4 3 3 2 ...
 $ q11_21      : num [1:5364] 4 4 3 3 5 2 3 3 3 4 ...
 $ q11_22      : num [1:5364] 4 4 3 3 4 2 3 3 3 3 ...
 $ q11_23      : num [1:5364] 4 4 4 3 4 1 5 3 3 2 ...
 $ q11_24      : num [1:5364] 4 5 3 3 5 1 4 3 3 2 ...
 $ q11_25      : num [1:5364] 4 4 3 4 4 1 4 3 3 2 ...
 $ q12_1       : num [1:5364] 2 2 3 4 1 3 3 2 3 2 ...
 $ q12_2       : num [1:5364] 2 1 4 3 2 3 2 5 3 4 ...
 $ q12_3       : num [1:5364] 2 2 3 3 1 4 2 5 3 4 ...
 $ q12_4       : num [1:5364] 2 2 3 3 1 4 2 5 3 4 ...
 $ q12_5       : num [1:5364] 2 1 4 3 2 5 3 5 3 4 ...
 $ q13_1       : num [1:5364] 2 2 3 4 2 4 2 2 2 2 ...
 $ q13_2       : num [1:5364] 2 2 3 3 2 5 2 2 2 3 ...
 $ q13_3       : num [1:5364] 2 1 3 3 2 5 2 3 2 2 ...
 $ q13_4       : num [1:5364] 4 5 3 3 5 1 3 3 1 2 ...
 $ q14_1       : num [1:5364] 4 3 3 3 3 2 2 4 3 2 ...
 $ q14_2       : num [1:5364] 4 4 3 3 3 2 2 4 3 3 ...
 $ q14_3       : num [1:5364] 4 4 3 3 3 1 2 4 3 4 ...
 $ q14_4       : num [1:5364] 4 4 3 3 2 1 4 4 3 3 ...
 $ q14_5       : num [1:5364] 4 4 3 3 3 2 5 4 3 3 ...
 $ q14_6       : num [1:5364] 4 5 2 2 3 2 4 4 4 4 ...
 $ q14_7       : num [1:5364] 4 4 3 3 4 2 2 3 3 3 ...
 $ q14_8       : num [1:5364] 4 5 3 4 3 2 2 3 3 3 ...
 $ q14_9       : num [1:5364] 4 4 3 3 5 2 3 3 3 4 ...
 $ q14_10      : num [1:5364] 4 5 2 3 5 1 4 3 3 4 ...
 $ q14_11      : num [1:5364] 4 5 3 3 4 1 2 3 4 2 ...
 $ q14_12      : num [1:5364] 4 4 3 3 3 1 4 3 3 3 ...
```

```
 $ q14_13      : num [1:5364] 4 4 3 3 4 2 4 3 3 4 ...
 $ q14_14      : num [1:5364] 4 4 3 3 3 2 2 3 3 3 ...
 $ q14_15      : num [1:5364] 4 5 3 3 3 2 4 3 4 3 ...
 $ q14_16      : num [1:5364] 4 5 2 3 4 1 2 3 3 3 ...
 $ q14_17      : num [1:5364] 4 5 2 3 4 1 4 4 4 3 ...
 $ q15_1       : num [1:5364] 1 2 3 3 2 4 3 3 3 2 ...
 $ q15_2       : num [1:5364] 2 2 3 3 3 5 2 3 3 4 ...
 $ q15_3       : num [1:5364] 2 2 3 3 2 5 2 3 3 3 ...
 $ q15_4       : num [1:5364] 2 1 3 2 3 5 2 3 3 2 ...
 $ q15_5       : num [1:5364] 2 2 3 3 3 4 3 3 3 2 ...
 $ q15_6       : num [1:5364] 2 2 3 2 2 4 1 3 2 3 ...
 $ q15_7       : num [1:5364] 2 2 5 2 3 4 2 3 3 3 ...
 $ q16_1       : num [1:5364] 2 1 3 4 3 3 3 2 4 2 ...
 $ q16_2       : num [1:5364] 2 2 3 3 3 4 4 3 4 2 ...
 $ q16_3       : num [1:5364] 2 2 3 3 3 4 2 3 3 3 ...
 $ q16_4       : num [1:5364] 2 2 3 3 3 4 3 3 3 3 ...
 $ q17_1.01    : num [1:5364] 0 1 0 0 1 1 0 0 0 1 ...
 $ q17_1.02    : num [1:5364] 0 1 0 0 0 0 0 0 0 0 ...
 $ q17_1.03    : num [1:5364] 0 0 0 0 0 0 0 0 0 1 ...
 $ q17_1.04    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.05    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.06    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.07    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.08    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.09    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.10    : num [1:5364] 0 0 0 0 1 0 0 0 0 0 ...
 $ q17_1.11    : num [1:5364] 0 0 0 0 0 0 0 1 0 0 ...
 $ q17_1.12    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.13    : num [1:5364] 0 0 0 0 1 0 0 1 0 0 ...
 $ q17_1.14    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.15    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.16    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
 $ q17_1.17    : num [1:5364] 0 1 0 0 1 0 0 0 0 0 ...
 $ q17_1.18    : num [1:5364] 0 0 0 0 0 0 0 0 0 0 ...
  [list output truncated]
```

992 変数，5364 観測値からなるデータを読み込みました。東京都 (13) と兵庫県 (28) の回答者を対象に，ブランド志向性 (`q12_4`) と価格志向性 (`q13_3`) の 2 変数を使ってクラスター分析を実行します。

教科書では，カテゴリー変数の数値を文字列に変換するために，`case_when()` 関数を

使っていますが，この処理だと変数の型が文字列となり，カテゴリー変数として R が認識しなくなります。そこでここでは，`factor()` 関数を使って，変数の型を因子型に変換し，ラベルを付与する方法を採用します。

```
# 東京と兵庫の県番号リスト作成
ken_number   <- c(13, 28)
# 因子型のラベル作成
pref_labels  <- c("Tokyo", "Hyogo")
gender_label <- c("Male", "Female")
marital_label <- c("Married", "Not Married")

# 回答者と項目を抽出
df_cons <- df_cons |>
  select(県番号, 性別, 年齢, 結婚有無, q12_4, q13_3) |>
  filter(
    県番号 %in% ken_number, # 東京と兵庫の回答者
    q12_4 != 999,           # 欠損値は 999
    q13_3 != 999
  ) |>
  mutate(
    q12_4 = 6 - q12_4, # 尺度を反転
    q13_3 = 6 - q13_3, # 尺度を反転
    Pref       = factor(県番号,  levels  = ken_number, labels =
     ↪  pref_labels),
    Gender     = factor(性別,    levels  = c(1, 2), labels =
     ↪  gender_label),
    MaritalSt. = factor(結婚有無, levels = c(1, 2), labels =
     ↪  marital_label)
  )
```

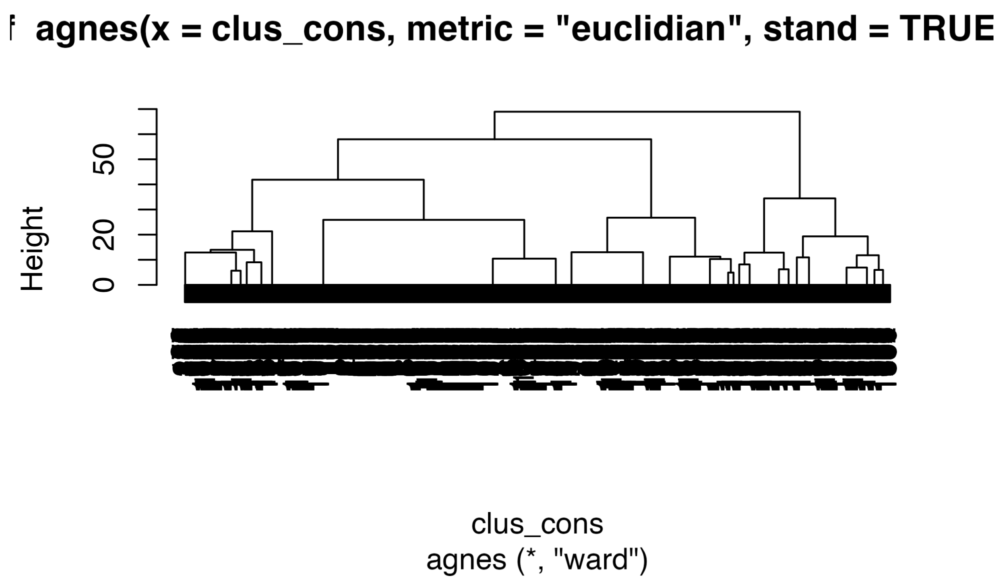
クラスター分析を実行する関数の引数は数値データのみである必要があるので，`select()` で必要な変数だけを抽出し，オブジェクト `clus_cons` に格納します。

`agnes()` 関数を使って階層的クラスター分析を実行します。引数として，

- `clus_cons`: クラスター分析対象データ
- `metric`: 距離尺度
- `method`: クラスター結合方法
- `stand`: 変数の標準化

を指定します。実行結果を `Hier1` に格納し，`pltree()` 関数でデンドログラムを作成します。

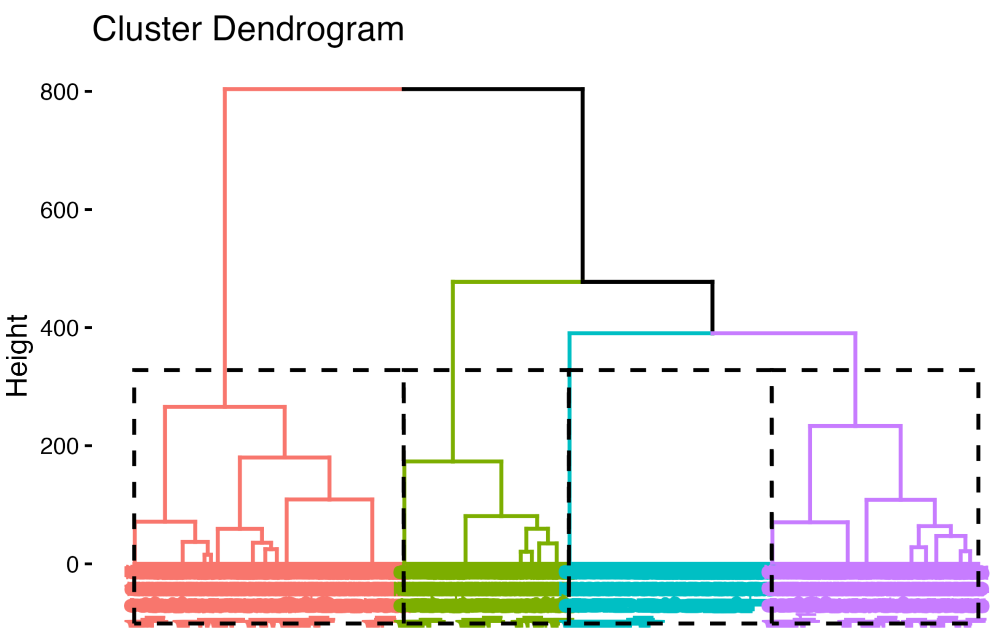
```r
clus_cons <- df_cons |> select(q12_4, q13_3)


Hier1 <- agnes(
  clus_cons, # クラスター分析対象データ
  metric = "euclidian", # 距離尺度
  method = "ward", # クラスター結合方法
  stand = TRUE # 変数の標準化
  )

pltree(Hier1) # デンドログラムの作成
```



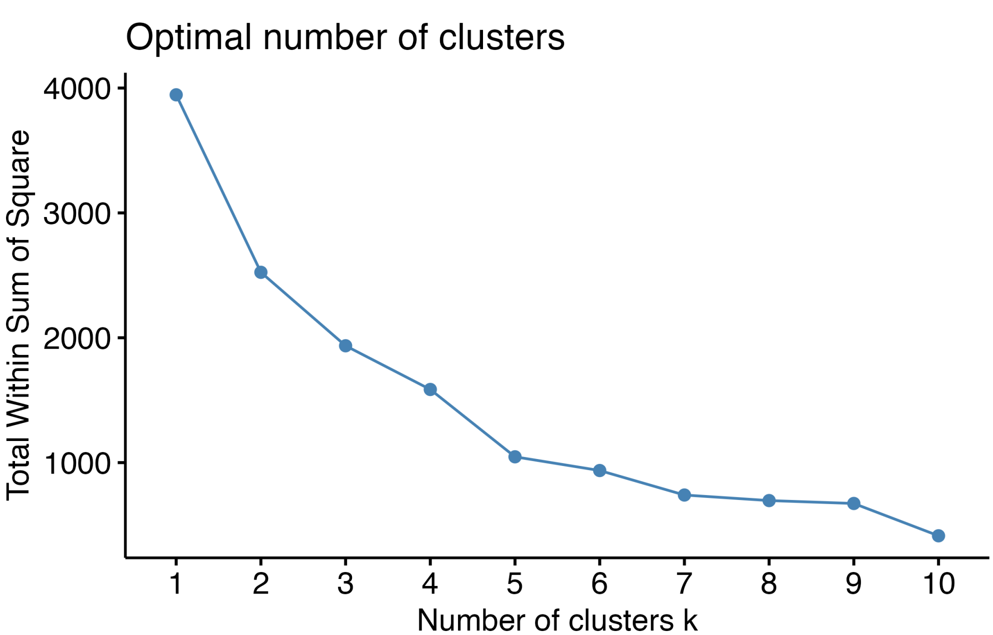
次に，hclust() 関数を使って階層的クラスター分析を実行します。階層別クラスターの分析手順は以下の通りです。

```r
alt_Hier <- clus_cons |>
  dist("euclidian") |> # ユークリッド距離の計算
  hclust("ward.D")   # ウォード法による階層的クラスター分析
```

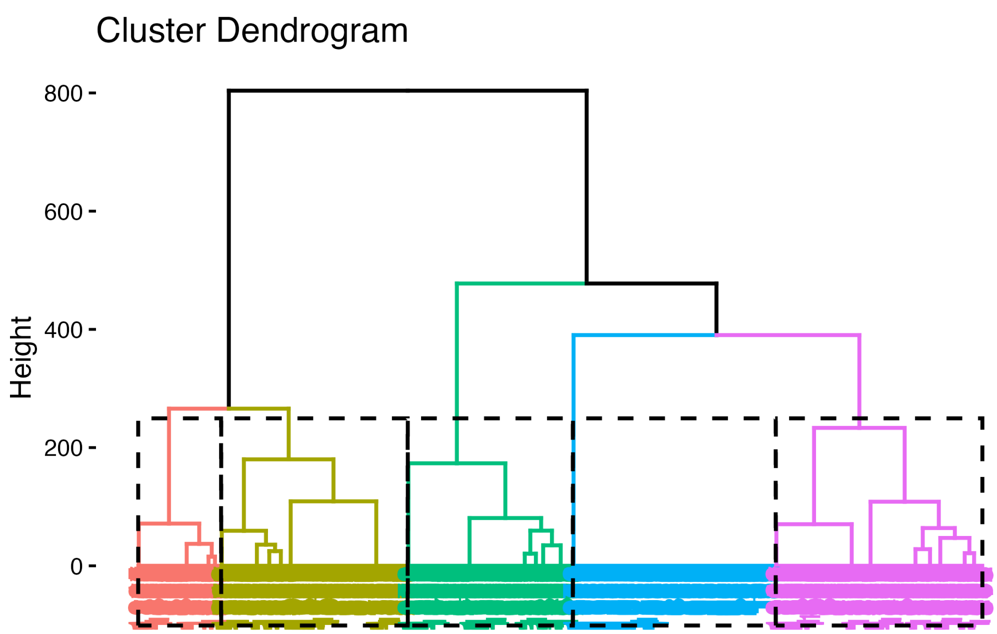
```r
alt_Hier |>
  fviz_dend(
    k = 4,   # クラスター数
    rect = TRUE, # クラスター枠の表示
    rect_border = TRUE # クラスター枠の色表示
  )
```




```
# コード 10-5
fviz_nbclust(
  clus_cons,  # クラスター分析対象データ
  kmeans,  # k-means 法
  method = "wss" # クラスター内変動の総和
  )
```



```
alt_Hier |>
  fviz_dend(
    k = 5, # クラスター数
    rect = TRUE, # クラスター枠の表示
    rect_border = TRUE # クラスター枠の色表示
  )
```



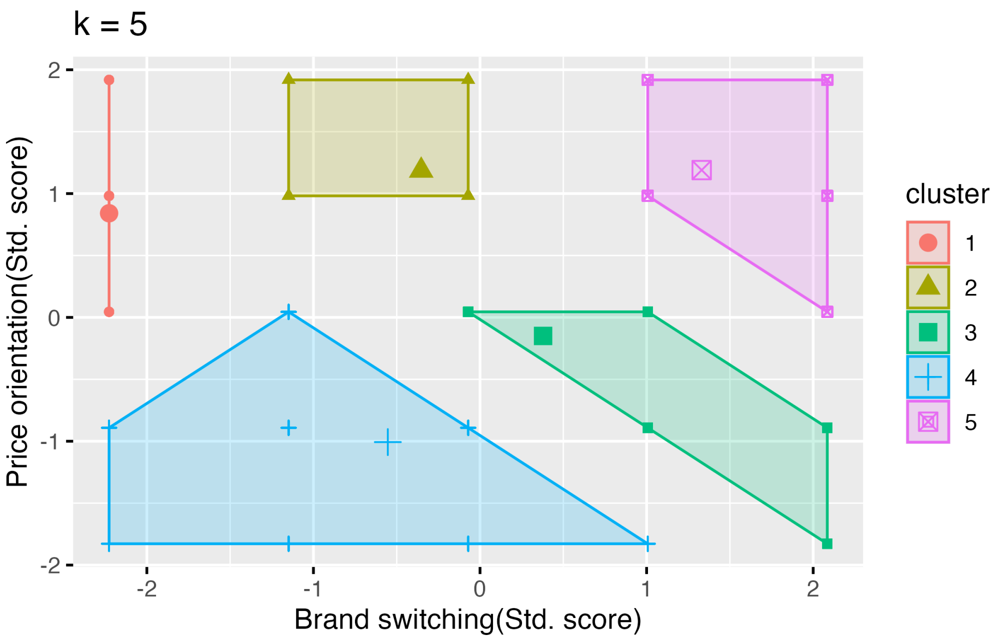

```
# 乱数の種を設定
set.seed(343)

# k-means 法による非階層的クラスター分析
K_shopping <- kmeans(clus_cons, 5)
K_shopping
```

```
K-means clustering with 5 clusters of sizes 80, 298, 760, 592, 243

Cluster means:
     q12_4    q13_3
1 1.000000 3.850000
2 2.738255 4.218121
3 3.417105 2.792105
4 2.552365 1.876689
5 4.300412 4.222222
```

```
Clustering vector:
   [1] 5 5 5 4 3 3 3 4 2 3 3 1 3 3 3 3 4 3 3 4 2 4 5 3 2 4 3 4 5 4 3 3 3 4 3 4 3
  [38] 4 3 3 4 5 4 3 3 4 3 3 4 2 4 2 2 4 4 2 3 3 3 3 4 3 4 3 2 1 2 1 3 4 3 4 2 4
  [75] 2 3 5 3 4 4 4 3 4 3 2 4 3 2 3 3 3 3 4 2 3 3 3 4 4 3 3 3 4 3 3 3 4 4 3 1 3
 [112] 3 3 3 4 4 3 4 5 4 4 5 3 3 4 3 4 4 4 3 3 3 4 1 2 4 3 2 3 1 4 3 2 4 3 3 4 4
 [149] 3 2 5 2 4 2 3 3 3 4 3 3 4 3 4 4 3 4 4 3 3 4 4 3 2 4 5 3 2 4 3 4 3 3 3 3 3
 [186] 4 3 3 3 3 3 2 4 3 3 2 4 4 4 3 2 3 3 4 4 1 3 3 2 4 3 5 3 3 3 4 4 5 3 2 1 4
 [223] 5 3 5 3 4 4 5 3 4 5 3 4 3 3 4 4 5 3 3 3 5 4 3 4 2 3 2 4 3 4 3 3 4 5 4 2 3
 [260] 3 3 3 5 2 5 4 3 2 3 5 1 4 2 3 4 3 4 3 4 3 2 1 3 3 3 2 3 4 4 2 3 4 3 4 4 3
 [297] 3 3 2 4 3 5 4 2 2 4 4 4 4 5 3 1 3 4 4 4 3 3 5 4 5 4 4 3 2 2 4 4 5 4 3 4 5
 [334] 3 3 4 4 3 3 3 2 1 4 3 4 3 4 3 5 5 2 3 1 2 1 3 3 3 4 2 3 4 4 2 3 3 2 2 3 3
 [371] 4 3 3 3 3 3 3 5 3 3 4 4 3 4 3 2 4 3 4 5 3 2 4 2 3 3 4 3 3 3 3 4 4 4 1 4 3
 [408] 4 3 2 3 3 3 3 3 4 3 1 2 4 2 5 2 3 2 3 2 4 2 3 2 3 4 3 3 3 3 3 2 3 4 3 3 3
 [445] 3 3 3 5 3 3 3 1 3 3 4 4 3 2 3 5 3 4 1 5 4 4 3 2 3 3 4 3 5 4 3 4 4 3 3 4 5
 [482] 4 3 4 2 3 5 4 4 4 3 4 3 4 3 3 4 5 5 3 3 3 5 3 3 1 5 3 4 3 4 5 3 2 2 5 5 3
 [519] 2 3 3 4 2 4 3 3 3 2 5 2 4 5 5 3 3 5 3 4 2 3 3 3 3 3 3 4 4 1 4 3 3 4 2 5 4
 [556] 3 4 3 3 2 5 4 5 3 3 4 5 4 3 2 2 4 4 3 2 4 2 3 2 3 3 2 3 3 3 4 5 3 4 3 3 4
 [593] 2 4 3 3 4 3 4 3 4 3 3 4 3 2 2 3 3 3 2 4 3 4 2 5 5 3 5 3 3 5 3 4 3 1 3 2 3
 [630] 3 5 1 4 3 5 3 2 3 2 4 5 2 3 1 4 4 3 2 4 3 4 3 4 3 2 4 3 4 4 3 4 3 5 4 3 3
 [667] 3 3 3 5 2 3 2 3 4 4 3 5 3 3 4 1 3 4 1 3 5 3 5 4 5 3 3 4 2 3 3 5 2 3 3 3 4
 [704] 3 3 4 3 3 3 3 3 5 4 1 2 3 2 4 2 2 3 5 3 3 3 5 3 4 3 5 3 2 3 4 4 5 4 3 1 3
 [741] 4 3 5 4 3 1 3 3 4 4 3 5 4 3 4 4 4 2 3 4 3 5 3 3 3 4 4 3 1 3 5 2 4 1 2 3 2
 [778] 3 2 3 4 3 5 5 4 3 4 4 3 2 3 3 3 1 3 3 3 3 2 4 4 3 3 2 3 4 4 3 3 4 3 5 4 3
 [815] 4 3 4 1 4 1 4 3 4 5 2 4 2 3 4 3 2 5 4 3 4 4 3 4 2 5 3 2 2 4 3 4 4 4 4 3 5
 [852] 3 2 4 5 4 4 5 2 2 3 1 3 4 5 5 3 2 3 3 5 2 4 3 3 2 4 2 3 1 3 2 2 2 3 2 5 1
 [889] 4 4 5 5 5 4 2 5 2 3 4 1 4 3 5 5 3 2 4 4 4 3 2 2 4 3 3 4 3 3 3 3 3 2 3 2 3
 [926] 5 5 4 3 4 3 4 4 3 1 2 4 5 4 3 5 1 2 4 5 4 4 5 3 3 4 3 4 4 3 4 2 4 2 4 5 4
 [963] 3 2 4 5 4 2 4 2 3 4 3 4 3 2 4 2 3 2 3 4 2 4 2 3 3 3 2 2 4 1 3 4 1 4 3 4 4
[1000] 3 5 4 2 5 4 5 2 3 3 3 4 3 4 3 3 4 3 3 4 4 2 3 3 5 3 2 5 3 3 3 3 4 3 4 4 5
[1037] 2 4 3 4 2 3 2 2 4 3 3 3 4 4 2 5 3 3 3 4 3 3 5 4 4 4 2 3 4 5 1 4 3 1 2 5 4
[1074] 4 3 4 5 4 4 3 4 3 3 4 4 4 3 3 3 4 2 4 2 4 3 2 4 4 3 3 2 3 5 5 4 3 3 5 3 4
[1111] 2 2 3 4 5 3 4 5 4 4 3 3 3 3 3 3 4 4 3 3 4 4 3 3 4 3 2 4 5 5 5 5 2 4 3 4 3
[1148] 4 4 2 5 4 3 2 2 3 2 4 3 3 4 4 5 3 2 2 3 4 4 2 4 5 3 5 1 4 3 2 5 3 4 4 3 3
[1185] 4 1 5 2 4 5 3 3 4 4 3 4 3 5 5 2 3 2 3 4 4 3 2 4 4 3 2 4 3 3 3 2 5 4 1 3 5
[1222] 4 4 3 3 3 3 4 4 2 3 4 2 5 5 4 5 4 5 2 5 2 5 4 3 3 4 3 5 3 4 4 3 4 4 4 3 3
[1259] 2 4 2 3 4 4 4 3 4 2 4 2 4 3 4 3 2 2 4 5 5 4 3 3 3 4 4 2 3 3 4 4 3 3 5 1 4
[1296] 3 4 3 2 3 4 3 3 4 5 3 3 4 3 4 3 2 4 4 3 3 3 5 3 4 3 5 4 4 3 4 3 5 1 4 3 1
[1333] 2 4 4 5 3 4 4 3 3 3 4 3 3 5 4 3 1 3 3 5 3 3 3 4 2 3 5 3 2 4 3 4 1 3 4 1 5
```

```
[1370] 3 4 3 3 3 4 3 3 5 4 5 4 3 4 2 4 5 2 4 3 3 5 5 3 2 4 5 4 4 3 3 2 3 2 2 3 3
[1407] 4 4 5 3 4 2 3 2 3 1 2 1 2 3 2 3 3 3 4 4 1 4 2 5 3 5 2 4 4 3 4 4 4 5 3 2 3
[1444] 1 5 3 4 3 3 3 2 4 3 4 4 3 4 3 2 4 3 4 2 4 3 3 5 4 3 5 4 2 5 4 4 2 3 2 3 3
[1481] 3 2 5 5 5 4 4 1 3 4 3 4 3 2 5 3 4 3 3 5 1 4 4 4 3 3 3 3 4 3 4 3 4 2 3 3 3
[1518] 3 5 4 5 5 3 2 4 3 4 4 3 3 4 3 4 3 3 2 4 4 4 5 2 4 4 1 3 3 3 5 3 3 5 3 3 3
[1555] 3 5 4 3 2 2 4 3 3 2 3 3 3 3 3 4 4 5 5 4 2 4 3 2 3 4 4 3 1 4 2 1 5 4 2 4 3
[1592] 3 2 3 3 4 5 3 2 2 4 1 3 2 5 3 3 3 2 4 5 5 5 4 3 5 5 4 2 4 3 5 3 1 4 5 3 2
[1629] 4 5 5 3 2 3 3 4 1 3 5 2 3 3 2 4 3 2 3 2 3 5 3 2 2 5 3 2 4 4 4 3 4 2 4 2 3
[1666] 2 4 2 5 5 1 3 4 2 3 5 3 3 5 5 3 4 4 2 2 4 2 4 3 4 4 4 3 3 2 2 2 4 5 2 3 3
[1703] 5 1 3 4 5 2 3 5 5 3 2 2 4 3 3 5 3 4 4 3 3 5 3 4 4 4 3 3 4 4 5 4 4 3 3 3 3
[1740] 4 3 3 4 2 2 5 3 4 4 2 3 4 4 2 4 3 3 2 3 2 4 2 4 4 4 4 5 2 2 3 4 3 2 4 2 4
[1777] 4 4 4 4 3 2 4 4 3 4 4 5 4 5 2 2 2 2 3 5 4 5 2 2 1 4 4 2 4 1 3 5 2 4 5 3 1
[1814] 3 5 4 2 3 3 3 3 2 4 3 5 3 5 3 4 1 5 5 2 3 4 2 5 4 3 5 4 5 3 4 3 4 5 3 4 4
[1851] 4 3 3 5 1 2 5 3 3 1 2 4 3 4 3 3 4 4 1 5 3 3 3 4 1 2 3 5 5 2 4 3 2 4 4 2 3
[1888] 2 4 5 4 3 2 3 5 3 1 3 2 3 1 2 3 2 4 3 3 3 3 4 4 3 4 4 3 4 2 4 2 3 3 5 3 4
[1925] 5 3 4 3 4 4 5 4 1 4 3 3 3 2 2 5 1 4 4 3 3 3 4 4 4 4 2 4 3 4 1 4 5 2 4 3 5
[1962] 1 3 3 3 4 5 2 3 3 4 3 3

Within cluster sum of squares by cluster:
[1]  54.2000 108.4060 401.9303 622.3750 127.0700
 (between_SS / total_SS =  66.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```
# コード 10-8

fviz_cluster(
  K_shopping,  # k-means 法の結果
  data = clus_cons, # クラスター分析対象データ
  geom = "point" # プロットの形状
  ) +
  labs( # タイトルと軸ラベル
    title = "k = 5",
    x = "Brand switching(Std. score)",
    y = "Price orientation(Std. score)"
  )
```




```
# コード 10-9
df_cons$cluster_id <- factor(K_shopping$cluster)

knitr::kable(head(df_cons), caption = "結合データ")
```

Table7.1: 結合データ

県番号	性別	年齢	結婚有無	q12_4	q13_3	Pref	Gender	MaritalSt.	cluster_id
13	2	38	2	4	4	Tokyo	Female	Not Married	5
13	1	42	2	4	5	Tokyo	Male	Not Married	5
13	1	30	1	5	4	Tokyo	Male	Married	5
13	1	33	1	2	1	Tokyo	Male	Married	4
28	1	25	1	3	3	Hyogo	Male	Married	3
28	1	32	1	3	3	Hyogo	Male	Married	3

```
# コード 10-10

clus_summary <- df_cons |>
  group_by(cluster_id) |>
```

```
  summarize(
    N = n(),
    Loyalty_m = mean(q12_4),
    Price_m   = mean(q13_3),
    Age_m     = mean(年齢),
    Male_r    = sum(Gender == "Male") / n(),
    Tokyo_r   = (sum(Pref == "Tokyo") / n()) / (1465 / 1973),
    Married_r = sum(MaritalSt. == "Married") / n()
  )
# 表示
knitr::kable(clus_summary, caption = "クラスターサマリー")
```

Table7.2: クラスターサマリー

cluster_id	N	Loyalty_m	Price_m	Age_m	Male_r	Tokyo_r	Married_r
1	80	1.000000	3.850000	42.61250	0.5125000	0.8753925	0.5250000
2	298	2.738255	4.218121	38.56040	0.5134228	0.9400188	0.6107383
3	760	3.417105	2.792105	41.88816	0.5065789	1.0100683	0.5618421
4	592	2.552365	1.876689	43.79730	0.4645270	1.0191680	0.4712838
5	243	4.300412	4.222222	34.91770	0.5102881	1.0363938	0.6213992

第 8 章

第 11 章

```
pacman::p_load(psych, GPArotation, tidyverse, readxl, gt, gtExtras,
 ↪  dendextend, cluster, factoextra, useful, ggrepel)

# コード 11-1
# install.packages("psych")
# install.packages("GPArotation")

# コード 11-2
factor_exdata <- readxl::read_excel("data/factor_ex.xlsx", na = ".")

factor_exdata$V5 <- 8 - factor_exdata$V5
rownames(factor_exdata) <- factor_exdata$ID

factor_exdata2 <- factor_exdata %>%
  select(-ID)
knitr::kable(summary(factor_exdata2))
```

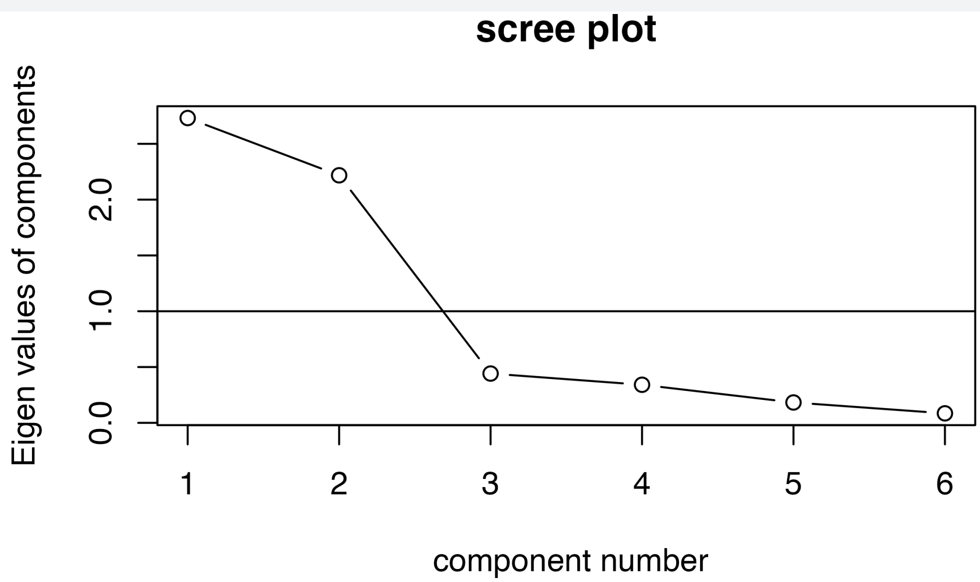
	V1	V2	V3	V4	V5	V6
	Min. :1.000	Min. :2.0	Min. :1.0	Min. :2.0	Min. :1.0	Min. :2.000
	1st Qu.:2.000	1st Qu.:3.0	1st Qu.:2.0	1st Qu.:3.0	1st Qu.:3.0	1st Qu.:3.000
	Median :4.000	Median :4.0	Median :4.0	Median :4.0	Median :4.5	Median :4.000
	Mean :3.933	Mean :3.9	Mean :4.1	Mean :4.1	Mean :4.5	Mean :4.167
	3rd Qu.:6.000	3rd Qu.:5.0	3rd Qu.:6.0	3rd Qu.:5.0	3rd Qu.:6.0	3rd Qu.:4.750
	Max. :7.000	Max. :7.0	Max. :7.0	Max. :7.0	Max. :7.0	Max. :7.000

```
# コード 11-3
knitr::kable(cor(factor_exdata2), caption = "相関行列")
```

Table8.2: 相関行列

	V1	V2	V3	V4	V5	V6
V1	1.0000000	-0.0532178	0.8730902	-0.0861622	0.8576366	0.0041681
V2	-0.0532178	1.0000000	-0.1550200	0.5722121	-0.0197456	0.6404649
V3	0.8730902	-0.1550200	1.0000000	-0.2477879	0.7778480	-0.0180688
V4	-0.0861622	0.5722121	-0.2477879	1.0000000	0.0065819	0.6404649
V5	0.8576366	-0.0197456	0.7778480	0.0065819	1.0000000	0.1364029
V6	0.0041681	0.6404649	-0.0180688	0.6404649	0.1364029	1.0000000

```
# コード 11-4
cor.exdata <- cor(factor_exdata2)

VSS.scree(cor.exdata)
```



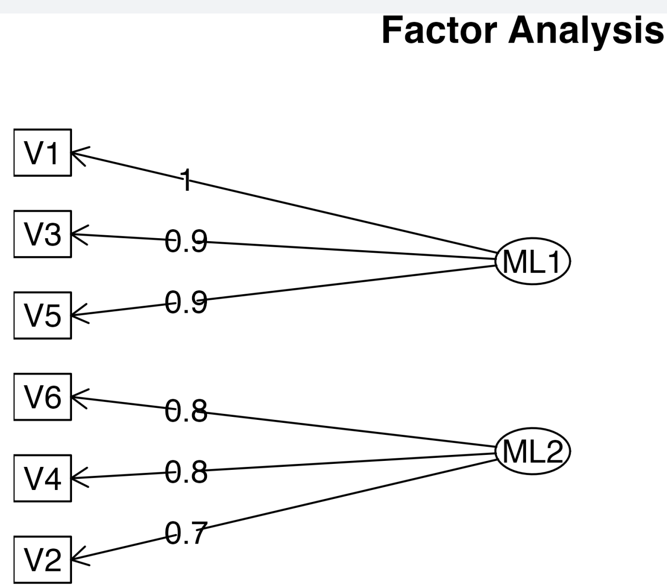

```
# コード 11-5
fa <- fa(r = factor_exdata2, nfactors = 2,
  rotate = "promax", fm = "ml")
fa
```

```
Factor Analysis using method =  ml
Call: fa(r = factor_exdata2, nfactors = 2, rotate = "promax", fm = "ml")
```

```
Standardized loadings (pattern matrix) based upon correlation matrix
     ML1   ML2   h2    u2 com
V1  0.97  0.00 0.94 0.063   1
V2 -0.02  0.75 0.56 0.437   1
V3  0.89 -0.12 0.83 0.174   1
V4 -0.06  0.78 0.62 0.378   1
V5  0.89  0.11 0.79 0.205   1
V6  0.08  0.83 0.69 0.309   1

                       ML1  ML2
SS loadings           2.54 1.89
Proportion Var        0.42 0.32
Cumulative Var        0.42 0.74
Proportion Explained  0.57 0.43
Cumulative Proportion 0.57 1.00

 With factor correlations of
      ML1   ML2
ML1  1.00 -0.06
ML2 -0.06  1.00

Mean item complexity =  1
Test of the hypothesis that 2 factors are sufficient.

df null model =  15  with the objective function =  4.25 with Chi Square =  111.31
df of  the model are 4  and the objective function was  0.21

The root mean square of the residuals (RMSR) is  0.03
The df corrected root mean square of the residuals is  0.05

The harmonic n.obs is  30 with the empirical chi square  0.65  with prob <  0.96
The total n.obs was  30  with Likelihood Chi Square =  5.21  with prob <  0.27

Tucker Lewis Index of factoring reliability =  0.95
RMSEA index =  0.095  and the 90 % confidence intervals are  0 0.314
BIC =  -8.39
Fit based upon off diagonal values = 1
Measures of factor score adequacy
                                                   ML1  ML2
```

```
Correlation of (regression) scores with factors   0.98 0.92
Multiple R square of scores with factors          0.96 0.84
Minimum correlation of possible factor scores     0.92 0.68
```

```
# コード 11-6
fa.diagram(fa)
```

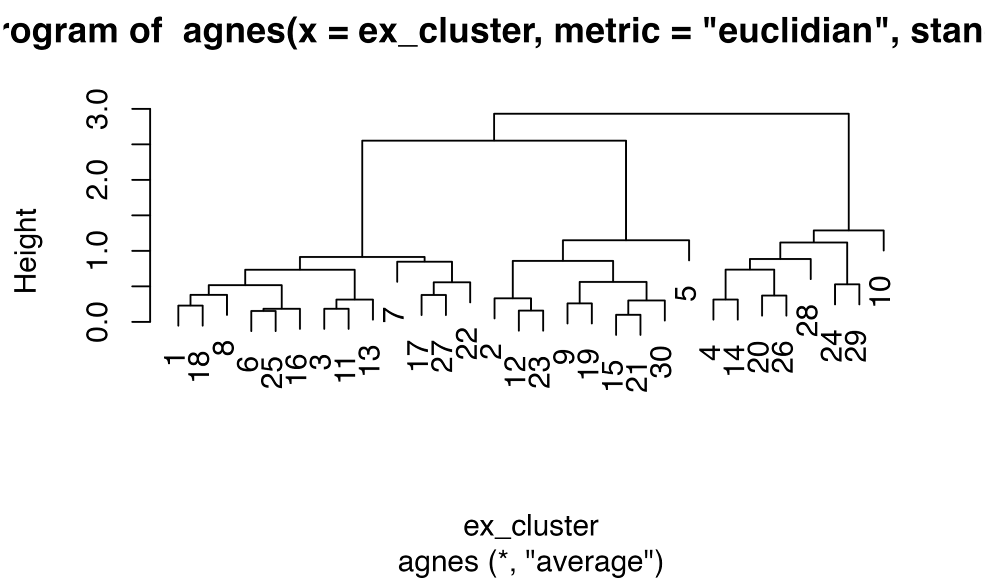



```
# コード 11-7
fs <- data.frame(fa$scores)
factor_exdata2$rowname <- rownames(factor_exdata2)
fs$rowname <- rownames(fs)
factor_exdata2 <- left_join(factor_exdata2, fs, by = "rowname")
knitr::kable(head(factor_exdata2), caption = "結合後データ")
```

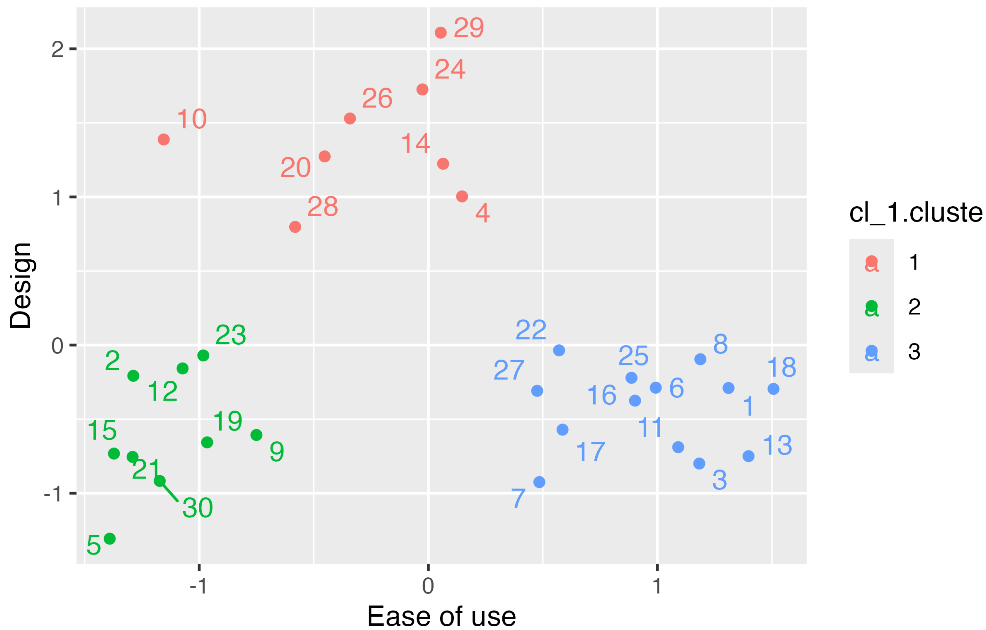
Table8.3: 結合後データ

V1	V2	V3	V4	V5	V6	rowname	ML1	ML2
7	3	6	4	6	4	1	1.3106155	-0.2896470
1	3	2	4	3	4	2	-1.2878038	-0.2073067
6	2	7	4	7	3	3	1.1828544	-0.7996332
4	5	4	6	6	5	4	0.1474225	1.0035837
1	2	2	3	2	2	5	-1.3907544	-1.3067260
6	3	6	4	6	4	6	0.9928111	-0.2875982

```
# コード 11-8

ex_cluster <- factor_exdata2 %>%
  select(ML1, ML2)
Hier1 <- agnes(ex_cluster, metric = "euclidian", stand = TRUE)
pltree(Hier1)
```




```
# コード 11-9
cl_1 <- kmeans(ex_cluster, 3)
clus_fa <- data.frame(cl_1$cluster)
clus_fa$rowname <- rownames(clus_fa)
factor_exdata2 <- left_join(factor_exdata2, clus_fa, by = "rowname")
factor_exdata2$cl_1.cluster <- factor(factor_exdata2$cl_1.cluster)

# Visualizing the clusters with 2 factors
p1 <- ggplot(data = factor_exdata2,
             mapping = aes(x = ML1, y = ML2, color = cl_1.cluster))
p1 + geom_point() +
  geom_text_repel(mapping = aes(label = rownames(factor_exdata2))) +
  labs(x = "Ease of use", y = "Design")
```




```
# コード 11-10
factor_exdata3 <- factor_exdata %>%
  select(-ID)
pca_res <- pca(factor_exdata3, nfactors = 2, rotate = "none")
pca_res
```

```
Principal Components Analysis
Call: principal(r = r, nfactors = nfactors, residuals = residuals,
    rotate = rotate, n.obs = n.obs, covar = covar, scores = scores,
    missing = missing, impute = impute, oblique.scores = oblique.scores,
    method = method, use = use, cor = cor, correct = 0.5, weight = NULL)
Standardized loadings (pattern matrix) based upon correlation matrix
     PC1  PC2   h2    u2 com
V1  0.93 0.25 0.93 0.074 1.1
V2 -0.30 0.80 0.72 0.277 1.3
V3  0.94 0.13 0.89 0.106 1.0
V4 -0.34 0.79 0.74 0.261 1.4
V5  0.87 0.35 0.88 0.122 1.3
V6 -0.18 0.87 0.79 0.210 1.1

                       PC1  PC2
SS loadings           2.73 2.22
Proportion Var        0.46 0.37
Cumulative Var        0.46 0.82
```

```
Proportion Explained  0.55 0.45
Cumulative Proportion 0.55 1.00

Mean item complexity =  1.2
Test of the hypothesis that 2 components are sufficient.

The root mean square of the residuals (RMSR) is  0.07
 with the empirical chi square  3.94  with prob <  0.41

Fit based upon off diagonal values = 0.98
```

```
# コード 11-11
# データの相関行列から固有値と固有ベクトルを抽出
cor_ex <- cor(factor_exdata3)
eigen_list <- eigen(cor_ex)

# 固有ベクトルについての情報をオブジェクトとして定義
pc1_eig <- eigen_list$vectors[, 1]
pc2_eig <- eigen_list$vectors[, 2]

# 固有ベクトルに，固有値の平方根を乗じる
PC1 <- pc1_eig * sqrt(eigen_list$values[1])
PC2 <- pc2_eig * sqrt(eigen_list$values[2])
knitr::kable(data.frame(cbind(PC1, PC2)), caption = "計算結果")
```

Table8.4: 計算結果

PC1	PC2
0.9283425	-0.2532285
-0.3005297	-0.7952496
0.9361812	-0.1308894
-0.3415817	-0.7889663
0.8687553	-0.3507939
-0.1766389	-0.8711581

第 9 章

第 12 章

```
pacman::p_load(pricesensitivitymeter, tidyverse)
#コード 12-1
# install.packages("pricesensitivitymeter")
# library(pricesensitivitymeter)
# library(tidyverse)

#コード 12-2
psm_ex <- read.csv("data/psm_ex.csv", na = ".")
head(psm_ex)
```

```
   tch   ch   ex  tex
1  963 1016 1242 1318
2  866 1082 1272 1303
3  795  824 1290 1297
4  812  933 1211 1335
5 1026  818 1228 1289
6  912  987 1195 1395
```

```
#コード 12-3
output_psm <- psm_analysis(
  toocheap = "tch",
  cheap = "ch",
  expensive = "ex",
  tooexpensive = "tex",
  data = psm_ex
)

summary(output_psm)
```

```
Van Westendorp Price Sensitivity Meter Analysis

Accepted Price Range: 922.5 - 1253
Indifference Price Point: 1101
Optimal Price Point: 1058

---
157 cases with individual price preferences were analyzed (unweighted data).
Total data set consists of 250 cases. Analysis was limited to cases with transiti
(Removed: n = 93 / 37% of data)
```

```
#コード 12-4
psm_plot(output_psm) +
  labs(
    x = "Price",
    y = "Share of Respondents (0-1)",
    title = "Price Sensitivity Meter Plot",
    caption = "Shaded area: range of acceptable prices\nData: Randomly
     ↪ generated") +
  theme_minimal()
```