Rの導入とパラメータ推定

2016/5/9(月)

スタートアップゼミ#4

M1 三木真理子

目次

- Rの導入編
  - RとRstudio概観
  - Rを使ってみる
    （基本演算・関数の定義・様々なデータ型の扱い）

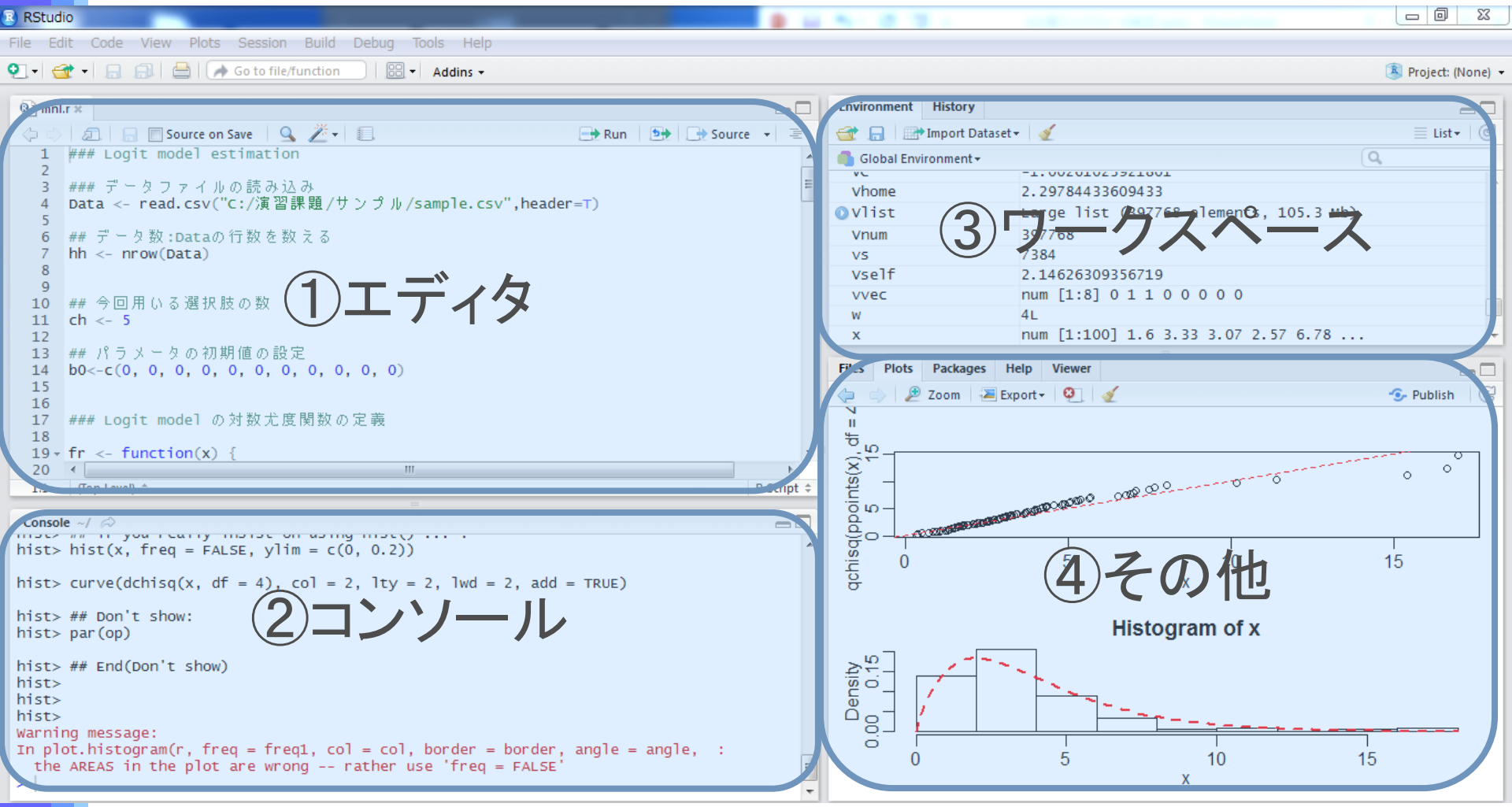
- パラメータ推定編
  - 推定の前の諸注意
  - 推定コードの確認(MNLモデル推定コードを例に)
  - 推定結果の解釈と書き方

1.1. Rとは

- 統計処理を目的とするプログラミング言語
- 無償＋オープンソースのソフトフェア
  →誰もが便利なパッケージを無償で使える！
- R関連のWebサイト
  - R-Tips
    (http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html)
    Rの基本操作集．参考書的に使える．
  - Rwiki
    (http://www.okadajp.org/RWiki/?RjpWiki)
    Rに関する情報を日本語でまとめてあるwiki.
    誰でも編集できる．
- 描画など，可視化が簡単にできる(詳しくは次回)
- ベクトル・行列演算のプログラムを書きやすい

1.1. R Studio

- R用の統合開発環境(エディタ＋実行＋表示機能)



※配置の変更：Tools→Global Options→Pane Layout

1.1. RとR Studioのインストール(確認)

- Rのインストール
  (windows) http://cran.ism.ac.jp/bin/windows/
  他のOSのインストール方法もRwikiで調べられる

- R studioのインストール
  http://memorandum2015.sakura.ne.jp/index rstudio.html

1.2. Rを使ってみる

1. 基本的な演算処理
2. 関数の定義と使用
3. データ型とその扱い

■ 基本的な演算処理

^	べき乗
:	公差 ±1の等差数列
%/%, %%	整数の割り算の商，整数の割り算の余り
*, /	積，商
+, -	和，差
<, >, <=, >=	大小比較
==, !=	等しい，等しくない
!	否定
&, |, &&, ||	論理積，論理和，かつ，または
<-	代入

コンソール画面

```
> x <- (5 %/% 2) / 3
> y <- 2^3 - 2*3
> (x > 0) && (y > 0)
[1] TRUE
> (x >= 2) || (y >= 2)
[1] TRUE
> 1:4
[1] 1 2 3 4
```

1.2. 関数の定義と使用

- 定義

（関数名）<- function(引数){処理･･･return(戻り値)}

関数名　　引数の指定

```
abs <- function(a,b){
   if(a >= b){
     return(a-b)
   }else{
     return(b-a)
   }
}
```

引数から戻り値を導く処理
※returnは省略できる
（最後に書かれた変数が戻り値になる）

- 使用(呼び出し)

（関数名）(引数)

```
> x <- abs(2,5)
> print(x)
[1] 3
```

引数に値を代入

戻り値が返ってくる

1.2. データの作成と取り出し–ベクトル編–

- Rの基本姿勢：対象をベクトル単位で扱う
  - ベクトルの作成：c()を用いる

```
x <- c(1,2)　#ベクトル(1,2)をxに代入
```

  - 要素へのアクセス：[]で要素を指定

```
x[1]　#ベクトルxの1番目の要素を取り出す
```

x[正整数ベクトル]	指定した場所の要素を同時に取り出す	x[c(1,3)]
x[負整数ベクトル]	指定した場所の要素を取り除く	x[c(-1,-3)]
x[論理値ベクトル]	TRUEの要素を取り出す	x[x > y]
x[条件式]	条件を満たす要素を取り出す	x[x > 30]

  - ベクトルの演算：sum(x), mean(x), sort(x) など

    ※基本演算で紹介した演算子は，要素に対する演算

コンソール画面

```
> c(1,2) * c(3,4)
[1] 3 8
```

1.2. データの作成と取出し–行列編–

- 行列の作成：matrix(ベクトル，行数，列数)

```
> y <- matrix(1:6, 2, 3)
> y
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> y <- matrix(1:6, 2, 3, byrow = T)
> y
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

- 要素へのアクセス：[(行の指定),(列の指定)]で行う

```
y[1,]    #行列yの1行目を取り出す
y[,1]    #行列yの1列目を取り出す
y[1,1]   #行列yの1行1列目の要素を取り出す
```

- 行列演算：逆行列計算solve(y)，行列式計算det(y) etc.

rowSums(y), rowMeans(y)	行の総和，行の平均
colSums(y), colMeans(y)	列の総和，列の平均
x %*% y	行列積 （x * y ならば要素積）
apply(y, 1, 処理関数)	各行に対して処理関数を実行
apply(y, 2, 処理関数)	各列に対して処理関数を実行

1.2. データの作成と取出し–データフレーム編–

- データ解析に使いやすい型:ヘッダー付のcsvファイル
  - 作成：data.frame(“列名”=(ベクトル), ･･･)

```
data <- data.frame(
            “個人ID” = c(101, 102, 103),
              “性別” = c(1, 2, 1),
                ・ ・ ・
        )
```

コンソール画面

```
> data
  個人ID 性別 代表交通手段
1    101    1        自転車
2    102    2        自転車
3    103    1          鉄道
```

  - 要素へのアクセス：[(行指定),(列指定)]または$列名

```
> data[1,c(1,2)]
  個人ID 性別
1    101    1
```

```
> data$性別
[1] 1 2 1
```

  - データフレーム用の演算
    - by(data, data$列名, 処理関数)
      指定した列の変数の値でデータフレームを分割して,
      それぞれについて処理関数を実行.

1.2. データの作成と取出し–リスト編–

- いろんな型や大きさの要素をまとめるのに使う
  - 作成：list(要素1，要素2，…)

```
> list <- list(c(1,2),matrix(1:6,2,3),3)
> list
[[1]]
[1] 1 2

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

[[3]]
[1] 3
```

  - 要素へのアクセス：[]または[[]]

list[k]	リストの第k成分を取り出す(戻り値はリスト)
list[[n]]	リストの第k成分の要素を取り出す(戻り値はリスト中の要素)

```
> list[2]
[[1]]
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> list[[2]]
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

1.2. データの抽出・結合

- 主な関数

subset(x, 条件式)	条件にあう行のみ抽出する
subset(x, 条件式, ベクトル)	ベクトルで指定した列について，条件にあう行のみを抽出する
rbind(x, y)	行ベクトル単位でxとyを結合（行が増える）
cbind(x, y)	列ベクトル単位でxとyを結合（列が増える）

- これまでの知識でできること
  - PTデータをデータフレーム型にして，移動目的ごとに移動距離の平均を求める
  - 実際の選択結果に応じてデータを分割して，それぞれを処理してからまた結合させる

目次

- Rの導入編
  - RとRstudio概観
  - Rを使ってみる
    (基本演算・関数の定義・様々なデータ型の扱い)

- パラメータ推定編
  - 推定の前の諸注意
  - 推定コードの確認(MNLモデル推定コードを例に)
  - 推定結果の解釈と書き方

パラメータ推定とは

- パラメータ推定とは

  ある母集団からランダムにサンプリングされたデータを用いて，母集団の特性値である未知パラメータを求める

- 推定の前に必要なこと
  - 政策変数(何を評価したいのか)を決める
  - 仮説(何を確かめたいのか)を立てる
    →基礎分析でデータと向き合うことが必要！
    失敗例）通勤トリップの多いデータで回遊行動モデルを推定
    やみくもにパラメータを設定したらt値が低い

[基礎分析のやり方例]
Excelでクロス集計，Javaで集計プログラムを組む，
Rでヒストグラムを書いてみるetc.

効用関数設定上の注意

$$U_{in} = V_{in} + \varepsilon_{in}$$
$$V_{in} = \theta_1 x_{1n} + \theta_2 x_{2n} + \cdots \theta_K x_{Kn}$$

θ :未知パラメータベクトル
x_n：個人nの説明変数ベクトル

- 説明変数同士が強く相関していないか(多重共線性)
  例)移動距離dと移動時間tをともに説明変数にする
- 定数項が選択肢と同数になっていないか(不定)
  例)(電車・バス・自転車)の3肢選択でそれぞれに定数項
- 識別できないパラメータが入っていないか(識別性)
  例)複数のダミー変数を同一の選択肢組に与える

推定コード例を確認

- PPデータで交通手段選択モデル(MNL)を推定

$$
\begin{aligned}
V_{train} &= t_1 X_1 + c_1 X_2 + t_2 X_3 + \beta_1 \\
V_{bus} &= t_1 X_1 + c_1 X_2 + t_2 X_3 + w_1 S_1 \\
V_{car} &= t_1 X_1 + \beta_2 \\
V_{bike} &= t_1 X_1 + w_1 S_1 + \beta_3 \\
V_{walk} &= t_1 X_1 + \beta_4
\end{aligned}
$$

X_i =説明変数(選択肢iの特性)

X_1 =所要時間[分/10]

X_2 =費用[円/100]

X_3 =乗車外時間 [分/10]

S_i =説明変数(選択者の社会経済属性)

S_1 =女性ダミー

t_i, c_i, w_i, β_iはパラメータ（特にβ_iは定数項）

推定コード概観

- MNLの推定コード構成

	A	B	C	D	E	F	G	H
1	トリップID	性別コード	年齢	職業コード	目的コード	曜日コード	出発地コー	出発
2	17147	1	33	1	400	1	12071	
3	19473	1	33	1	400	1	12070	
4	22842	1	40	1	400	6	11280	
5	15767	2	32	1	400	3	10321	
6	20405	2	32	1	400	4	10321	
7	16888	2	30	1	400	7	10839	
8	17931	2	30	1	400	3	10323	
9	18232	2	30	1	400	4	10323	
10	19828	2	30	1	400	2	11492	

列名
要素1
要素2
要素3
.
.
.
.
.

- データの読み込み
  ：csvファイルをデータフレーム型で読み込む

```
1  ###データファイルの読み込み
2  Data <- read.csv("C:/input/ensyu.csv",header=T)
3
4  ##データ数を数える
5  hh <- nrow(Data)
6
7  ##パラメータの初期値の設定
8  b0 <- numeric(8)
9
```

ファイルの場所を指定　　列名つき=T，列名なし=F

対数尤度関数の定義

- 関数の定義：(関数名)＝function(引数){処理}
- MNLの対数尤度の式

$$\ln L(\theta) = \sum_{n=1}^{N} \sum_{i \in I_n} \delta_{in} \ln(P_{in})$$

$$P_{in} = \frac{\exp(V_{in})}{\sum_{j \in I_n} \exp(V_{jn})}$$

δ_{in}:個人nが選択肢iを実際に選んだ場合1,
そうでなければ0を取る変数
I_n:個人nの選択肢集合
V_{in}:個人nにとっての選択肢iの効用の確定項

これを1つ1つ定めていく.

対数尤度関数の定義

1. パラメータを宣言
2. 効用確定項を計算
3. 選択確率を計算
4. 選択結果を判別
5. 対数尤度を計算

データフレームの利点を生かし，
簡単に各サンプルについて計算

■ パラメータ宣言

```
12 fr <- function(x){
13   ###パラメータの宣言
14   ##定数項
15   b1 <- x[1]
16   b2 <- x[2]
17   b3 <- x[3]
18   b4 <- x[4]
19   ##目的地までの所要時間
20   t1 <- x[5]
21   ##費用
22   c1 <- x[6]
23   ##乗車外時間
24   t2 <- x[7]
25   ##女性ダミー
26   w1 <- x[8]
```

関数名 <- function(引数){処理・・・

引数のx：自動的にベクトルと見なされる

パラメータの数がnのとき，
x[1]からx[n]までのn個の数に応じて
尤度関数を計算

以降，b1～b4, t1～t2, c1, w1 を使って
確定項の計算をしていく

効用確定項の計算

$$\exp(V_{in}) = \exp(\theta_1 x_{1n} + \theta_2 x_{2n} + \cdots)$$

```
###交通手段は以下の5つ
##鉄道(train)
train <- Data$代替手段生成可否train*exp(t1*Data$所要時間train/100
                                 +c1*Data$費用train/100
                                 +t2*(Data$アクセス時間train+Data$イグレス時間train)/100
                                 +b1*matrix(1,nrow=hh,ncol=1))
##バス(bus)
bus <- Data$代替手段生成可否bus*exp(t1*Data$所要時間bus/100
                             +c1*Data$費用bus/100
                             +t2*(Data$アクセス時間bus+Data$イグレス時間bus)/100
                             +w1*(Data$性別コード-1))
##自動車(car)
car <- Data$代替手段生成可否car*exp(t1*Data$所要時間car/100
                             +b2*matrix(1,nrow=hh,ncol=1))
##自転車(bike)
bike <- Data$代替手段生成可否bike*exp(t1*Data$所要時間bike/100
                             +w1*(Data$性別コード-1)
                             +b3*matrix(1,nrow=hh,ncol=1))
##徒歩(walk)
walk 成可否walk*exp(t 時間walk/100
                             +b4*matrix( col=1))
```

データフレーム型の扱い
(データ名)$列名

行列型の指定
(hh行1列，全要素1)

パラメータの桁数を揃える工夫

	代替手段生成可否car
個人1	1
個人2	1
.	1
.	1
.	1
個人n	1
.	1
.	1
.	1
	1

$\cdot\, exp \Big(t_1 \cdot$

所要時間car
46.2
45.6
35.7
55
77.7
5
77.2
8.7
55.4
73.3
50.1

$+b_2 \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \end{pmatrix} \Big)$

選択確率の計算

$$P_{in} = \frac{\exp(V_{in})}{\sum_{j \in I_n} \exp(V_{jn})}$$

```
53  ##選択確率の計算
54  ##分母の計算
55  deno <- (train + bus + car + bike + walk)
56  ##それぞれの計算
57  Ptrain <- train/deno
58  Pbus <- bus/deno
59  Pcar <- car/deno
60  Pbike <- bike/deno
61  Pwalk <- walk/deno
62
63  ##選択確率が0になってしまったら1を返す
64  Ptrain <- (Ptrain != 0) * Ptrain + (Ptrain ==0)
65  Pbus <- (Pbus != 0) * Pbus + (Pbus ==0)
66  Pcar <- (Pcar != 0) * Pcar + (Pcar ==0)
67  Pbike <- (Pbike != 0) * Pbike + (Pbike ==0)
68  Pwalk <- (Pwalk != 0) * Pwalk + (Pwalk ==0)
```

：hh行1列の行列の要素和

hh行1列の行列の要素の割り算
（Pcarのn行目は個人nの車の選択確率Pcar,n）

hh行1列の行列の要素和

$$Pwalk = \begin{cases} 1 \cdot Pwalk + 0, & Pwalk \neq 0 \\ 0 \cdot Pwalk + 1, & Pwalk = 0 \end{cases}$$

※$\ln L(\theta) = \sum_{n=1}^{N} \sum_{i \in I_n} \delta_{in} \ln(P_{in})$より，$P_{in} \neq 0$が必要

$P_{in} = 1$ ならば $\ln(P_{in}) = 0$ なので影響がない

選択結果の判別・対数尤度関数の計算

対数尤度関数 $\ln L(\theta) = \sum_{n=1}^{N} \sum_{i \in I_n} \delta_{in} \ln(P_{in})$

- 選択結果の判別

```
70    ##選択結果
71    Ctrain <- Data$代表交通手段  == "鉄道"
72    Cbus <- Data$代表交通手段   == "バス"
73    Ccar <- Data$代表交通手段   == "自動車"
74    Cbike <- Data$代表交通手段   == "自転車"
75    Cwalk <- Data$代表交通手段   == "徒歩"
```

hh行1列の行列
要素はδ_{in}

- 対数尤度関数の計算

```
77    ##対数尤度の計算
78    LL <- colSums(Ctrain*log(Ptrain)+Cbus*log(Pbus)
79                  +Ccar*log(Pcar)+Cbike*log(Pbike)+Cwalk*log(Pwalk))
80    }
```

列の総和
を計算

使われているのはhh行1列の行列のみ
各行について要素同士を計算

関数frの処理の定義終了を示す．※return(LL)を省略
fr <- function(x){処理・・・・}

対数尤度の最大化

- 最小化を行うoptim関数を使用

・optim(par, fn, method, control = list(), hessian)

par　：決定変数の初期値
fn　：最小化の目的関数
method：計算方法
　　Nelder-Mead, BFGS, CG, L-BFGS-B, SANNのいずれか
control：最大化の場合はcontrol=list(fnscale=-1) と指定
hessian：ヘッセ行列を計算する場合にTRUEを指定

```
82 #####対数尤度関数frの最大化#####
83 ##パラメータ値の最適化
84 res <- optim(b0,fr,method="BFGS",hessian=TRUE, control = list(fnscale=-1))
```

対数尤度関数fr(x)を初期値x=b0から最適化し，計算結果をresに格納.

・optim関数の戻り値：リスト型変数

[要素]最適解(par)，最適解が与える関数値(value),
　　計算回数(counts)，収束判定(convergence),
　　何らかの文字列(message)，ヘッセ行列(hessian)

Optim関数の戻り値

- コンソール画面にprint(res)を入力すると

```
> print(res)
$par
[1]   3.16752201   0.86766970   0.04603973   2.03698685  -7.62220412  -0.04440689 -10.49253379   1.81128428

$value
[1] -344.0754

$counts
function gradient
      38       18

$convergence
[1] 0

$message
NULL

$hessian
            [,1]        [,2]        [,3]       [,4]         [,5]         [,6]        [,7]         [,8]
[1,]  -50.193037  33.0614999   8.1036208   7.168461  4.055994026 -176.657941 -10.2867275   6.518126526
[2,]   33.061500 -61.3103766  12.2136938  13.812340 -0.239495066  149.623725   7.6756546   7.765767293
[3,]    8.103621  12.2136938 -35.2561869  11.346464 -0.435490449   25.456981   1.7493400 -17.739189005
[4,]    7.168461  13.8123404  11.3464643 -34.877569 -3.819365759   18.950901   1.3532355   7.651114885
[5,]    4.055994  -0.2394951  -0.4354904  -3.819366 -1.658178192   14.880729   0.9932775  -0.002066592
[6,] -176.657941 149.6237251  25.4569808  18.950901 14.880729040 -1191.010203 -41.0530410   5.118464969
[7,]  -10.286728   7.6756546   1.7493400   1.353235  0.993277496  -41.053041  -2.8191889   0.801989216
[8,]    6.518127   7.7657673 -17.7391890   7.651115 -0.002066592    5.118465   0.8019892 -21.935007453
```

x=par で fr(x) は最大値 value を取る

最適化計算の繰り返し回数・一階偏微分の計算回数

収束判定：0ならば収束．他の値なら収束していない．

エラーメッセージ

ヘッセ行列

結果の表示

- optim関数の戻り値を使ってt値を求める

```
86  ##パラメータ推定値、ヘッセ行列
87  b <- res$par
88  hhh <- res$hessian
89  ##t値の計算
90  tval <- b/sqrt(-diag(solve(hhh)))
91  ##初期尤度
92  L0 <- fr(b0)
93  ##最終尤度
94  LL <- res$value
```

- 結果をコンソール画面に表示させる

```
97  #####結果の出力#####
98  print(res)
99  ##初期尤度
100 print(L0)
101 ##最終尤度
102 print(LL)
103 ##ρ^2値
104 print((L0-LL)/L0)
105 ##修正済みρ^2値
106 print((L0-(LL-length(b)))/L0)
107 ##パラメータ推定値
108 print(b)
109 ##t値
110 print(tval)
```

推定結果の見方

- パラメータの正負はあっているか
- 尤度比は出ているか
- パラメータの説明力は高いか(t値はどうか)

サンプル数が十分に多い場合,
5%有意：t値の絶対値が1.96以上
1%有意：t値の絶対値が2.26以上

→複数のモデルを推定して仮説を検討する

推定結果の書き方

$$
\begin{aligned}
V_{train} &= t_1X_1 + c_1X_2 + t_2X_3 + \beta_1 \\
V_{bus} &= t_1X_1 + c_1X_2 + t_2X_3 + w_1S_1 \\
V_{car} &= t_1X_1 + \beta_2 \\
V_{bike} &= t_1X_1 + w_1S_1 + \beta_3 \\
V_{walk} &= t_1X_1 + \beta_4
\end{aligned}
$$

X_i =説明変数(選択肢iの特性)

X_1 =所要時間[分/10]

X_2 =費用[円/100]

X_3 =乗車外時間 [分/10]

S_i =説明変数(選択者の社会経済属性)

S_1 =女性ダミー

t_i, c_i, w_i, β_iはパラメータ

(特にβ_iは定数項)

説明変数	パラメータ	t値
所要時間[分/10]	-0.762	-7.41 **
費用[円/100]	-0.044	-0.98
乗車外時間[分/10]	-1.049	-8.07 **
女性ダミー	1.811	5.53 **
定数項（鉄道）	3.168	7.46 **
定数項（自動車）	0.868	2.08 *
定数項（自転車）	0.046	0.13
定数項（徒歩）	2.037	4.81 **
サンプル数	400	
初期尤度	-564.18	
最終尤度	-344.08	
尤度比	0.390	
修正済み尤度比	0.376	

*5%有意　**1%有意

- 単位を書く
- 桁数を揃える
- 有意なパラメータに印をつける
- 表には縦線を引かない

回らないときの基本的な確認事項

- データセットは正しくできているか
  - 空欄はない？
  - 誤字はない？(数字と文字が混在してない？)
- パラメータの設定はあっているか(個数・宣言)
- ファイルの指定場所は正しいか
- 効用関数の式は正しいか
  - 括弧の閉じ忘れよくあります
  - 説明変数を入れ忘れたり抜き忘れたりとか
  - 列名に誤字があったりとか

まとめ(頭に留めておいてほしいもの)

- Rには悔しいくらいにたくさんの便利な関数とパッケージがある
  →調べればたいていのことはできそう
- 中で何をやっているのかに関心を払おう
- データセットを作るのは大変です.
- 推定に関する試行錯誤は実際にやってみないとわからないと思います. やると面白いのでぜひやってみてください.