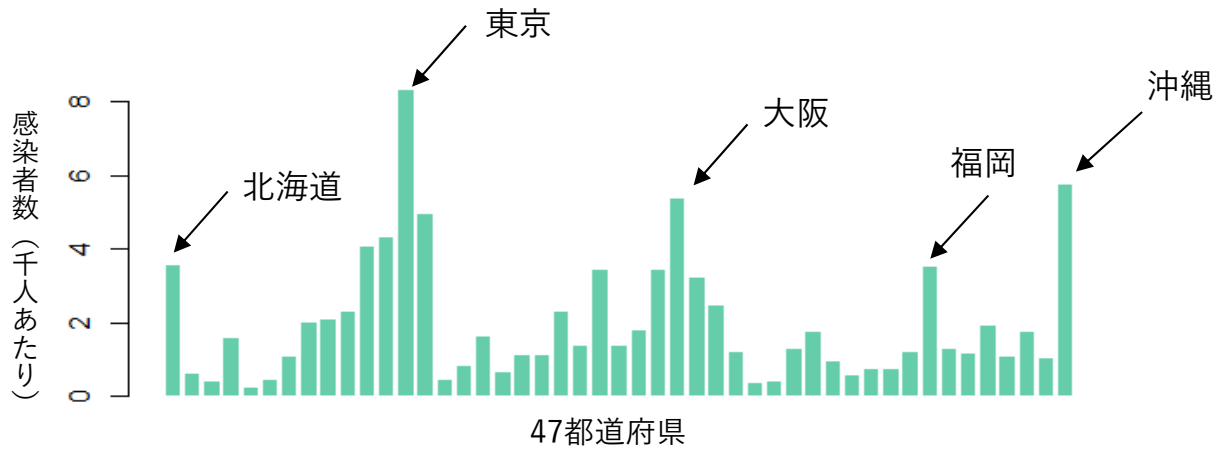
1-4　データ分析
－ 補助教材（Rによる計算例）－

東京大学 数理・情報教育研究センター

2021年2月23日

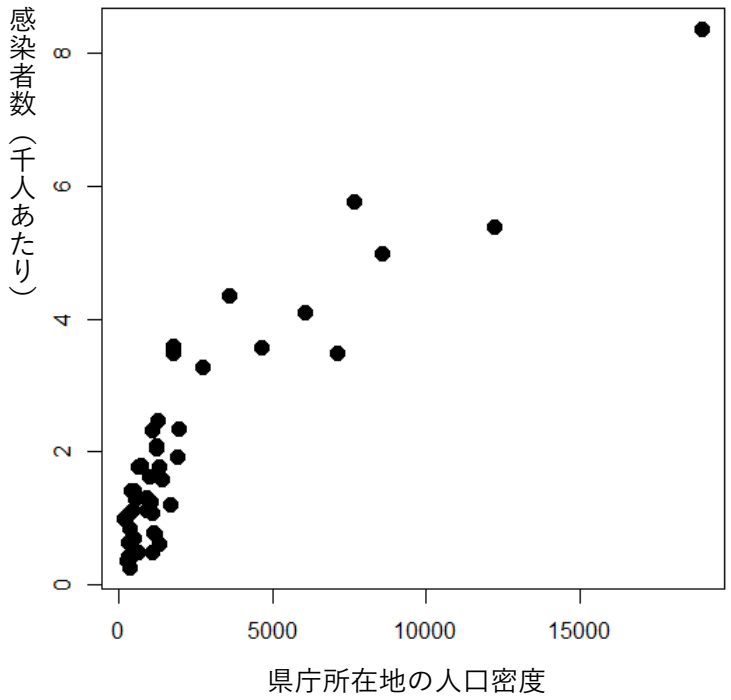
４７都道府県の人口千人あたりの新型コロナウイルスの累積感染者数

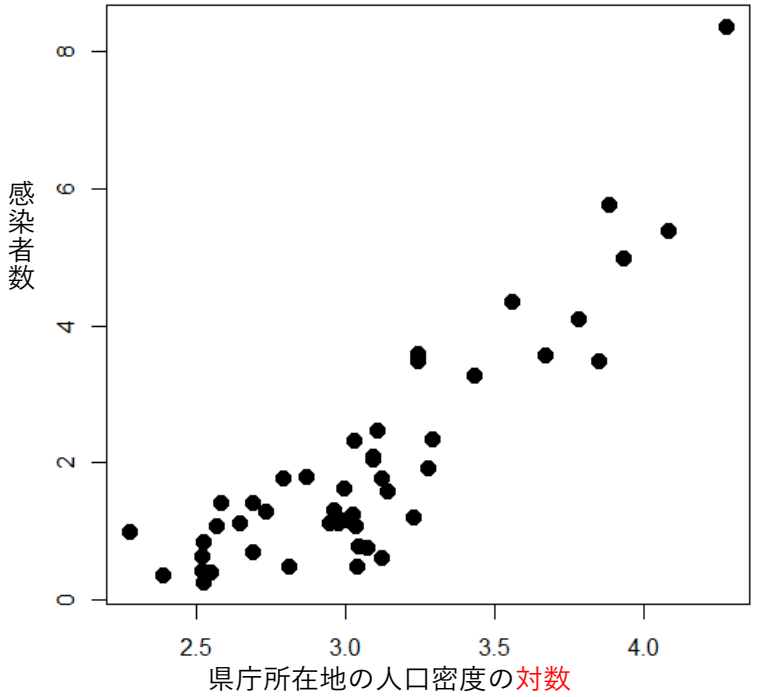
```
z <- read.csv( "Tokyo_Covid-19_regression.csv" )
par( mar=c(2,2,1,1))
plot( z[,2], type="h",lwd=5,col="aquamarine3" )

または
barplot( z[,2],col="aquamarine3",border=F )
```



４７都道府県の人口千人あたりの新型コロナウイルスの累積感染者数

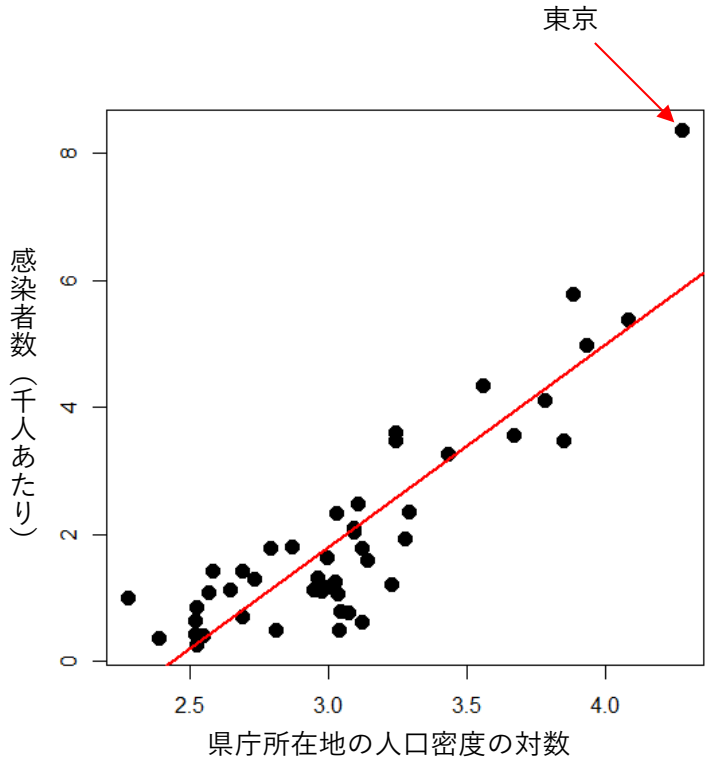
```
x <- read.csv( "Tokyo_Covid-19_regression.csv" )
par( mar=c(2,2,1,1))
plot( z[,c(6,2)], pch=16 )

#  横軸を対数表示
plot( log10(z[,6]),z[,2], pch=16 )
```





単回帰モデル

```
#  直線回帰
x <- log10( z[,6] )
y <- z[,2]
fit <- lm( y~x )
plot( x,y,pch=16 )
abline( fit, col="red" )

fit       # 回帰係数（切片，傾き）
lm(formula = y ~ x)
Coefficients:
(Intercept)            x
     -7.779        3.189
```

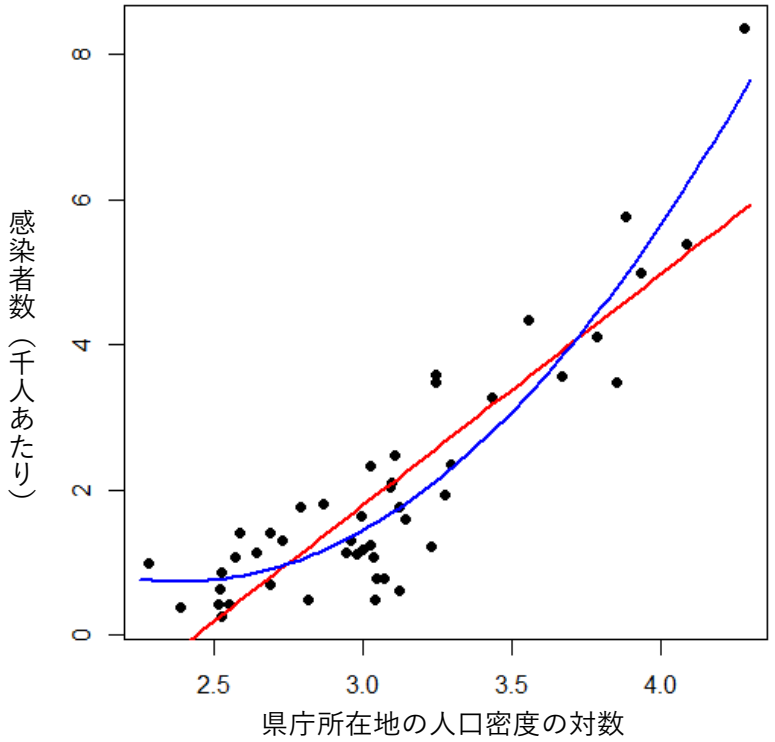


多項式回帰モデル

```
##### 多項式回帰 #####
plot( x,y,pch=16 )
newdata <- seq( 2.3,4.3, length=100 )

# 直線回帰
fit <- lm( y ~ x )
beta <- coef(fit)
pred <- beta[1] + beta[2]*newdata
points( newdata,pred, type="l" , col="red" )

# 2次回帰
fit <- lm( y ~ poly(x,2,raw=TRUE ) )
beta <- coef(fit)
pred <- beta[1] + beta[2]*newdata + beta[3]*newdata^2
points( newdata,pred, type="l" , col= "blue" )
```

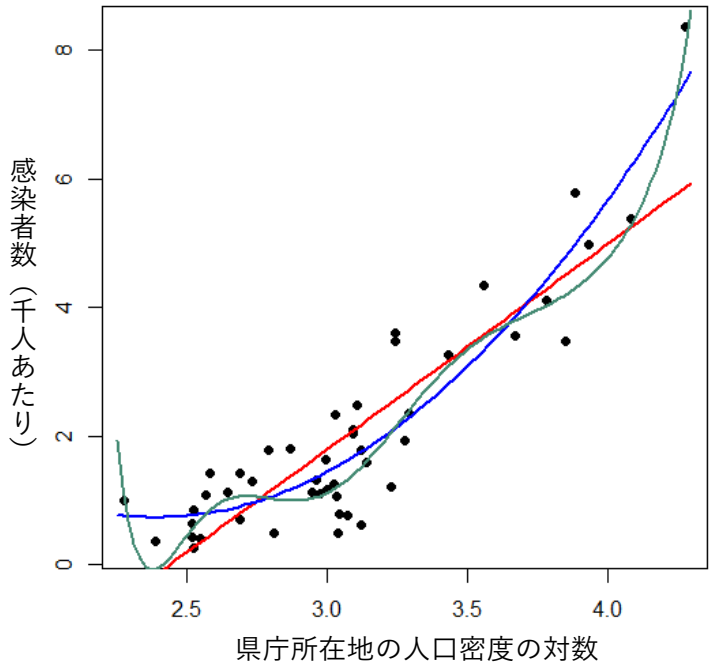
```
fit

Call:
lm(formula = y ~ 1 + x + I(x^2))
Coefficients:
(Intercept)           x      I(x^2)
     11.499       -9.026        1.891
```

$$y_n = 11.499 - 9.026x_n + 1.891x_n^2$$



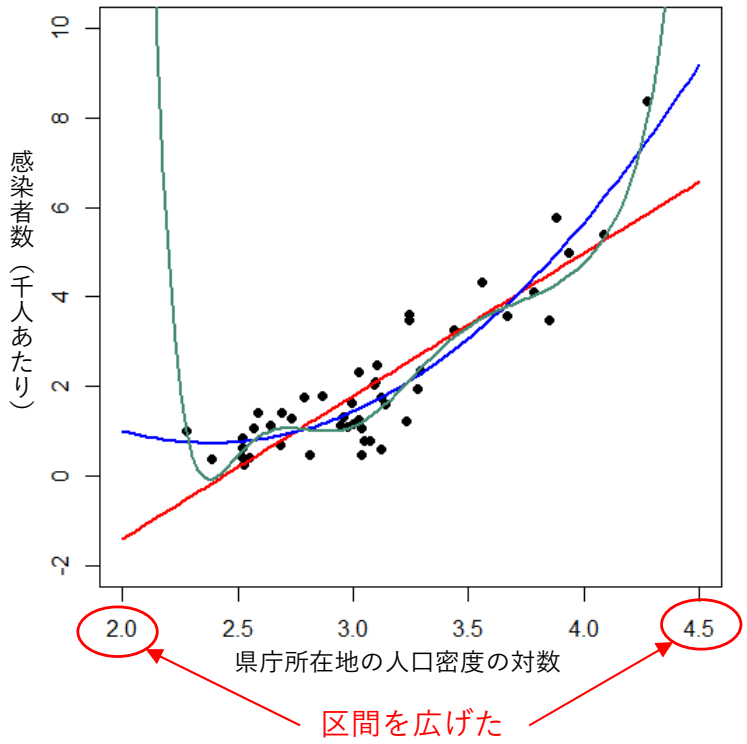
多項式回帰

```
#####  多項式回帰  #####
plot( x,y,pch=16,xlim=c(2.3,4.3) )
newdata <- seq( 2.0,4.5, length=100 )

#  直線回帰
fit <- lm( y ~ x )
beta <- coef(fit)
pred <- beta[1] + beta[2]*newdata
points( newdata,pred, type="l" , col="red",lwd=2 )

#  2次回帰
fit <- lm( y ~ poly(x,2,raw=TRUE ) )
beta <- coef(fit)
pred <- beta[1] + beta[2]*newdata + beta[3]*newdata^2
points( newdata,pred, type="l" , col= "blue",lwd=2 )

#  8次回帰
fit <- lm( y ~ poly(x,8,raw=TRUE ) )
beta <- coef(fit)
pred <- beta[1]
for (i in 1:8){
   pred <- pred + beta[i+1]*newdata^i
}
points( newdata,pred, type="l" , col= "aquamarine3",lwd=2 )
```



柔軟なモデル利用時の注意：　過学習

```
#####  多項式回帰  #####
plot( x,y,pch=16,xlim=c(2.3,4.3) )
newdata <- seq( 2.0,4.5, length=100 )

#  直線回帰
fit <- lm( y ~ x )
beta <- coef(fit)
pred <- beta[1] + beta[2]*newdata
points( newdata,pred, type="l" , col="red",lwd=2 )

#  2次回帰
fit <- lm( y ~ poly(x,2,raw=TRUE ) )
beta <- coef(fit)
pred <- beta[1] + beta[2]*newdata + beta[3]*newdata^2
points( newdata,pred, type="l" , col= "blue",lwd=2 )

#  8次回帰
fit <- lm( y ~ poly(x,8,raw=TRUE ) )
beta <- coef(fit)
pred <- beta[1]
for (i in 1:8){
   pred <- pred + beta[i+1]*newdata^i
}
points( newdata,pred, type="l" , col= "aquamarine3",lwd=2 )
```



平滑化実効再生産数について

y_n：新規感染者数

t_n：7日移動平均

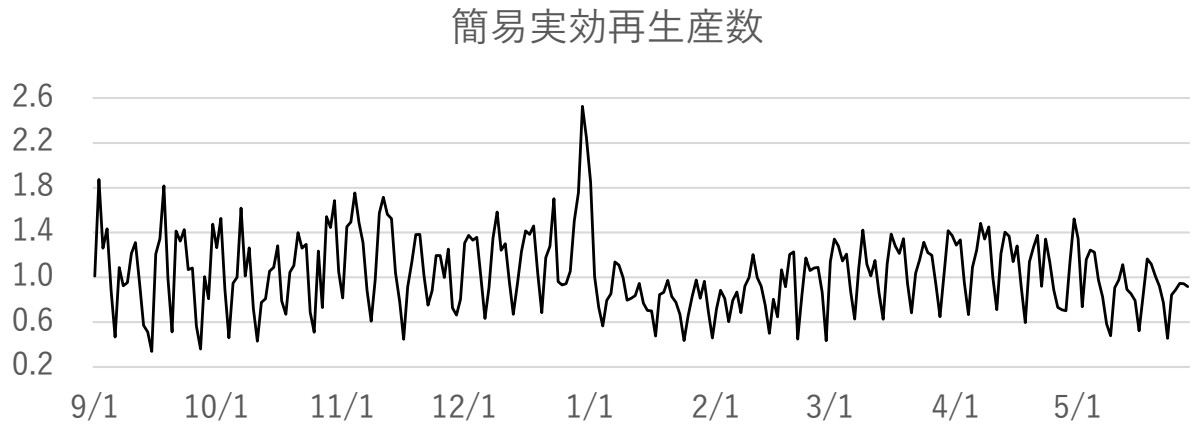
$$t_n = \frac{1}{7}(y_{n-3} + \cdots + y_n + \cdots + y_{n+3})$$

r_n：簡易実効再生産数

$$r_n = \frac{y_{n+8}}{t_{n+3}}$$

s_n：(片側)平滑化簡易実効再生産数

$$s_n = \frac{1}{7}(y_n + y_{n-1} + \cdots + y_{n-6})$$



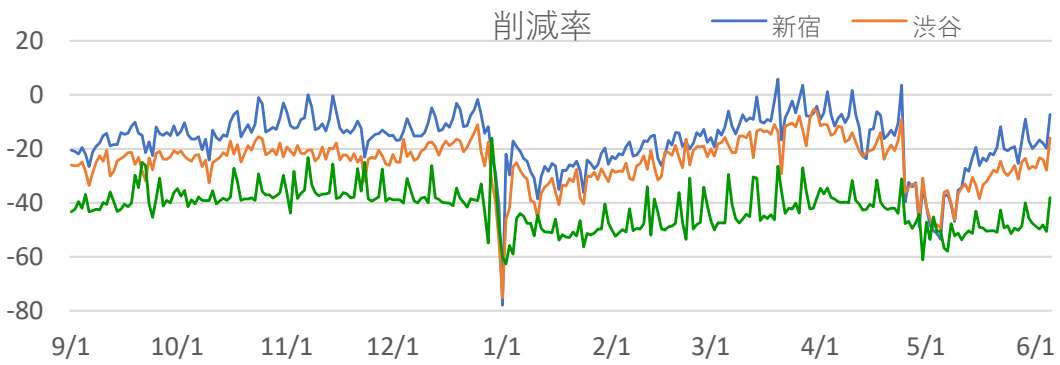
簡易実効再生産数の定義（鳥取大学）
https://www.tottori-u.ac.jp/secure/17752/1125.pdf

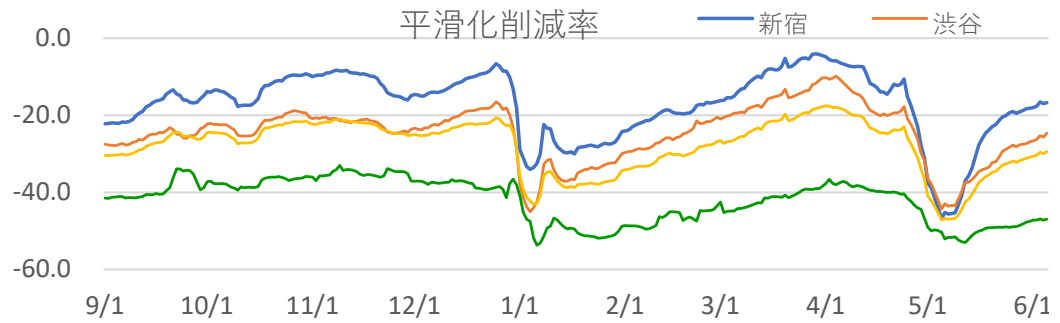
データ出展元「都内の最新感染動向」
https://stopcovid19.metro.tokyo.lg.jp/

都心人出について

$$\text{都心人出}(n)=\frac{1}{3}(\text{新宿駅}(n)+\text{渋谷センター街}(n)+\text{品川駅}(n)$$

ただし，各地点のデータは7日間移動平均で平滑化

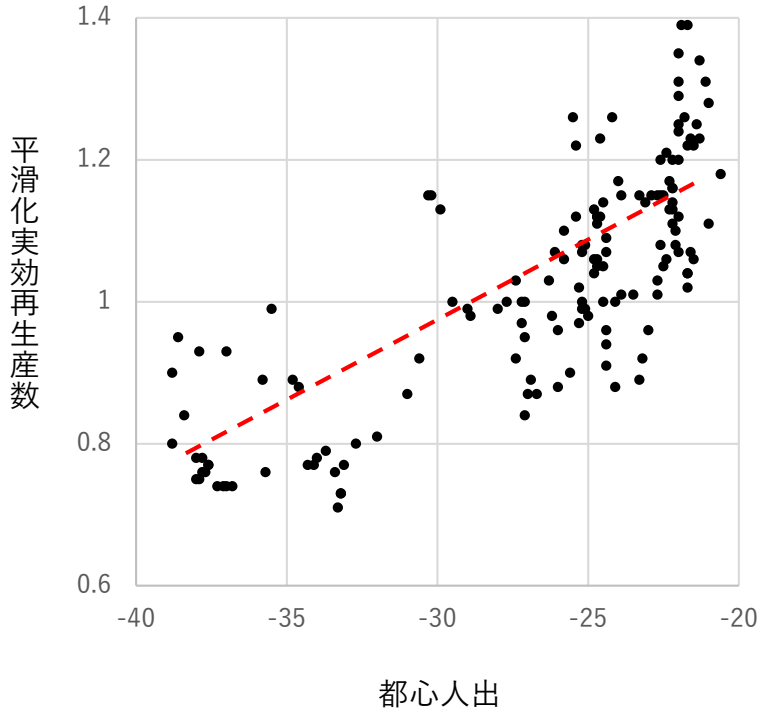




出展元「NTTドコモ　モバイル空間統計」

このサイトで公開されている各地点の削減率一覧データのうち，新宿駅，渋谷センター街，品川駅の3か所感染拡大前比データを利用

例：実効再生産数について

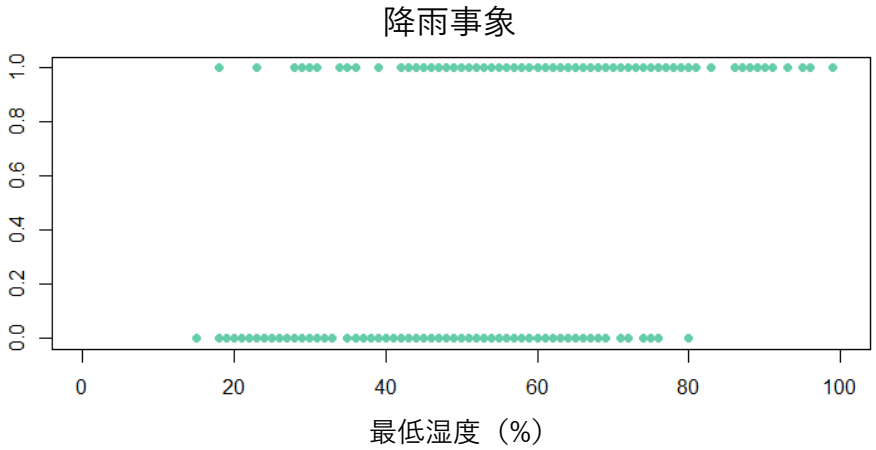


2．ロジスティック回帰分析

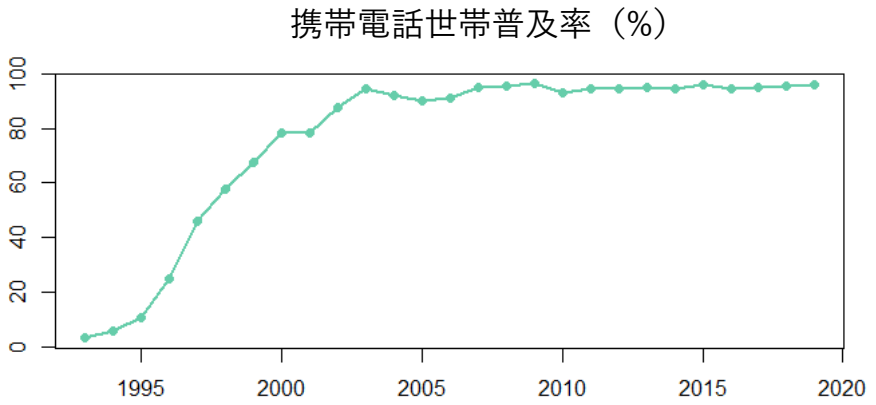
データ出展元
・携帯電話世帯普及率
　総務省情報通信政策局「通信利用動向調査報告書世帯編」
　http://honkawa2.sakura.ne.jp/6350.html

割合やカテゴリ変数のデータ

```
install.packages("arules")
library(arules)
library(TSSS)
rain <- read.csv( "Tokyo_rain.csv" )
rain2 = rain[,c(2,3)]
par( mar=c(2,2,1,1))
plot( rain2,pch=16,col="aquamarine3",xlim=c(0,100),ylim=c(-0.02,1.02) )
```

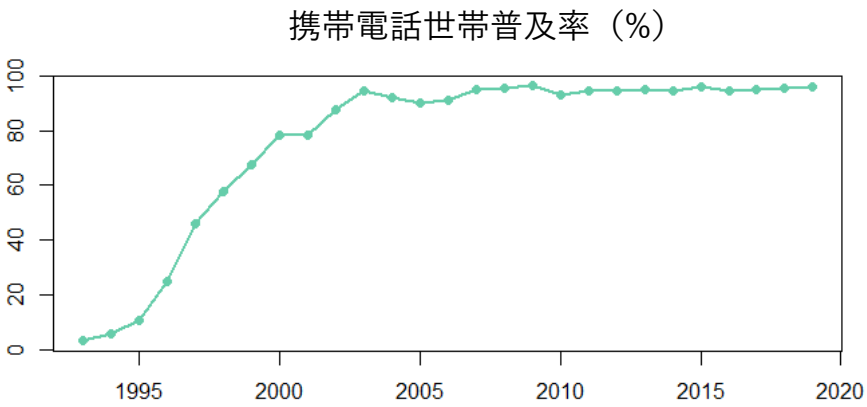



```
phone <- read.csv("mobile_phone.csv" )
phone2 = phone[,c(1,2)]
par( mar=c(2,2,1,1))
plot( phone2,pch=16,col="aquamarine3")
lines( phone2, lwd=2,col="aquamarine3")
```



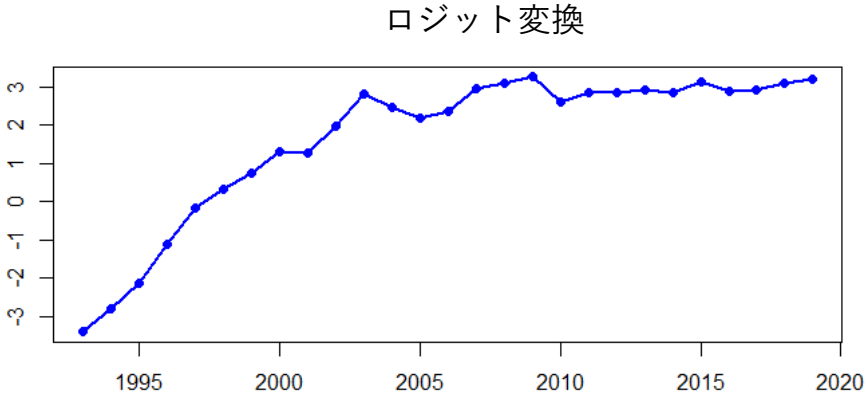
ロジット変換（例）

```
phone <- read.csv("mobile_phone.csv" )
phone2 = phone[,c(1,2)]
par( mar=c(2,2,1,1))
lot( phone2,pch=16,col="aquamarine3")
lines( phone2, lwd=2,col="aquamarine3")
```



$$q_n = a_0 + a_1 x_n$$

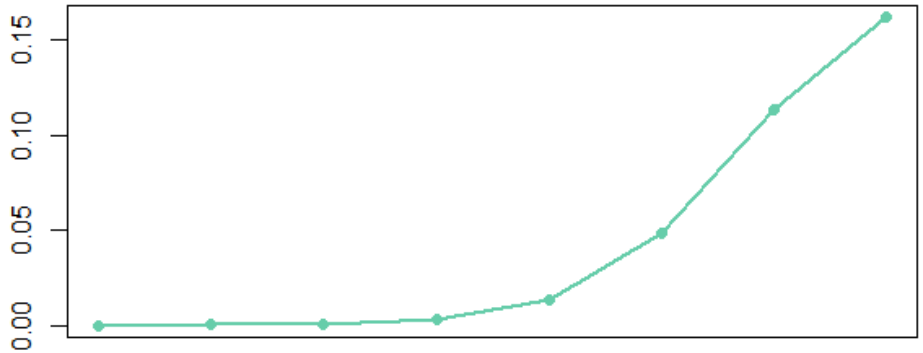
```
phone3 <- phone[,3]
logit_p <- log( phone3/(1-phone3) )
plot( phone[,1],logit_p,pch=16,col="blue")
lines( phone[,1],logit_p, lwd=2,col="blue")
```



ロジット変換（例2）

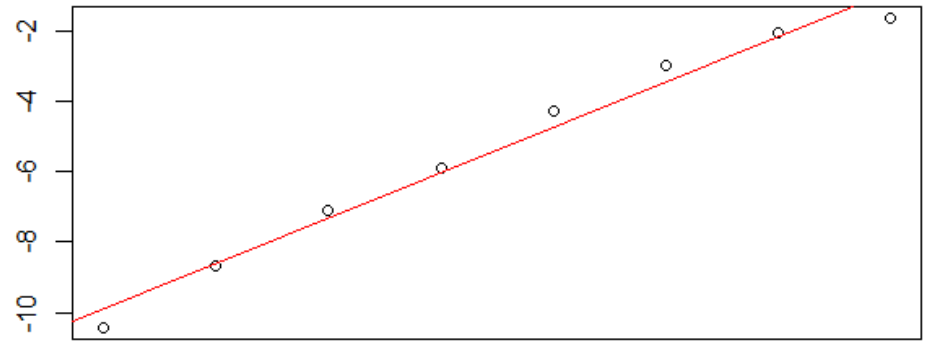
```
lethality <- read.csv( "age_vs_lethality.csv" )
x = lethality[,1]
y = lethality[,2]
par( mar=c(2,2,1,1))
plot(y,pch=16,col="aquamarine3",xaxt="n")
lines(y, lwd=2,col="aquamarine3")
```

年代別致死率



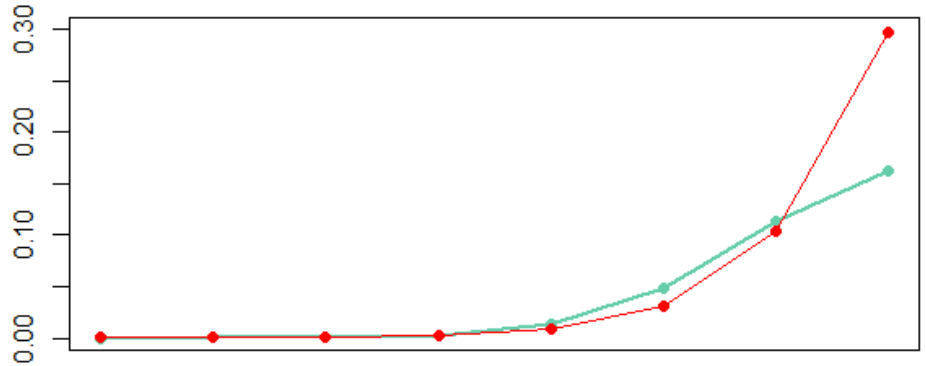

```
logit_y <- log( y/(1-y))
x <- seq( 1,8,by=1 )
plot( logit_y, xaxt="n")
fit <- lm( logit_y~x )
# plot( x,y,pch=16 )
abline( fit, col="red" )
```

ロジット変換と回帰直線



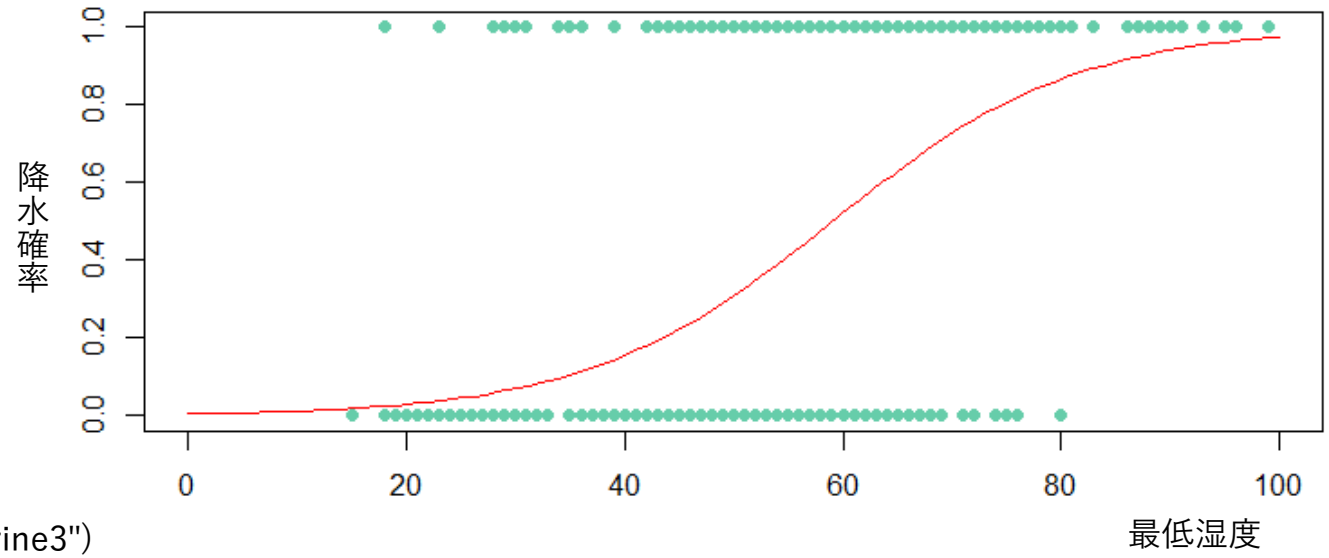

```
par( mar=c(2,2,1,1))
plot(y,pch=16,col="aquamarine3",xaxt="n",ylim=c(0,0.3))
lines(y, lwd=2,col="aquamarine3")
x <- seq( 1,8,by=1 )
f <- fit$coefficients[1] + fit$coefficients[2]*x
z <- exp(f)/(exp(f)+1)
lines( z, col="red")
points( x,z,pch=16,col="red" )
```

年代別致死率と回帰直線



例：降雨確率

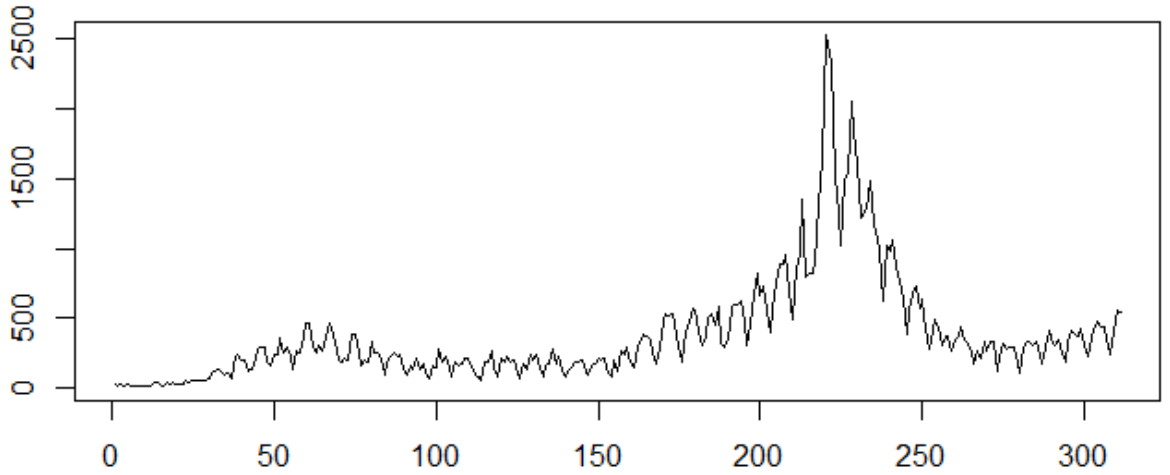
```
rain <- read.csv( "Tokyo_rain.csv" )
rain2 = rain[,c(2,3)]

x <- rain2[,1]
y <- rain2[,2]
z <- rain[,1]
res = glm( formula=y~x, family=binomial)
res[[1]]
res[[1]][1]
res[[1]][2]
#
#  Odds ratio
exp(res[[1]][2])
#
#  Plot logistic regression
#
fit = fitted( res )
#par( new=TRUE )
plot( x,fit,xlim=c(0,100) )

par( new=FALSE )
plot( x,y,xlim=c(0,100),pch=16,col="aquamarine3")
newdata <- seq( 0,100, length=100 )
ex  <- exp( res[[1]][1]+res[[1]][2]*newdata)
pred <- ex/(ex + 1)
lines( newdata,pred,type="l",col="red")
```



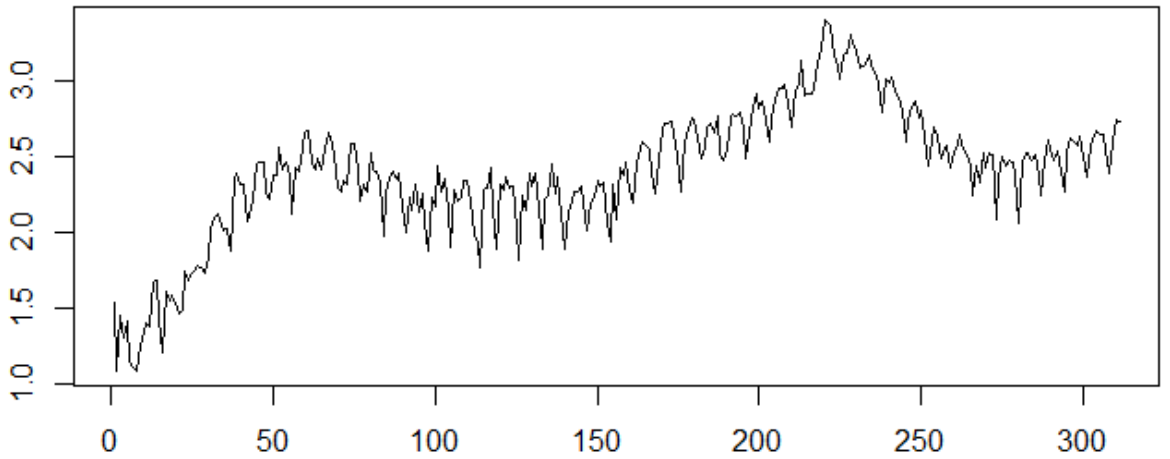
3．時系列分析

COVID-19 データ

```
covid_19 <- as.ts( read.csv("Covid-19_tokyo_20210408.csv") )
par(mar=c(2,2,1,1))
plot( covid_19 )
```

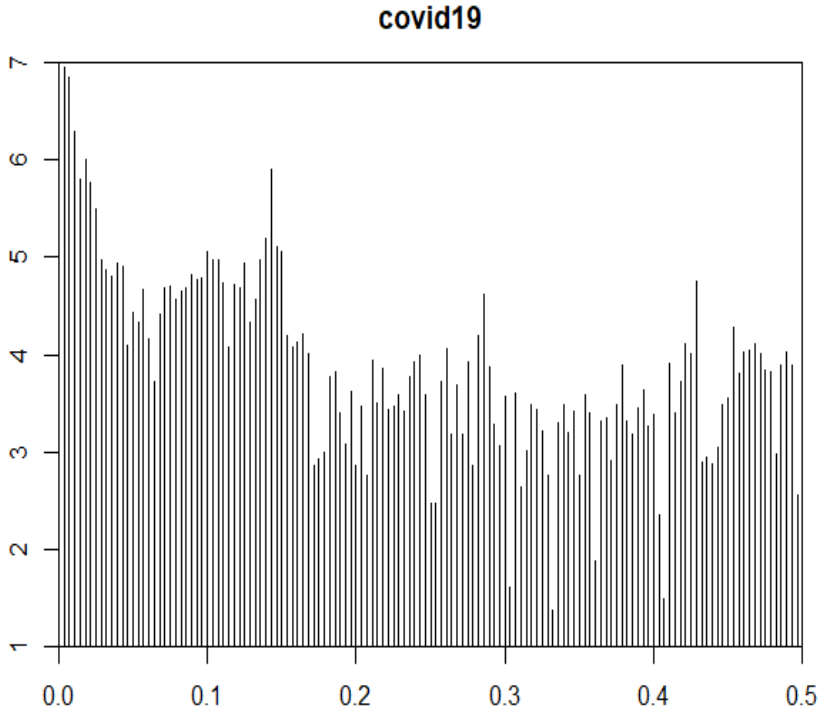



```
plot(log10( covid_19 ) )
```



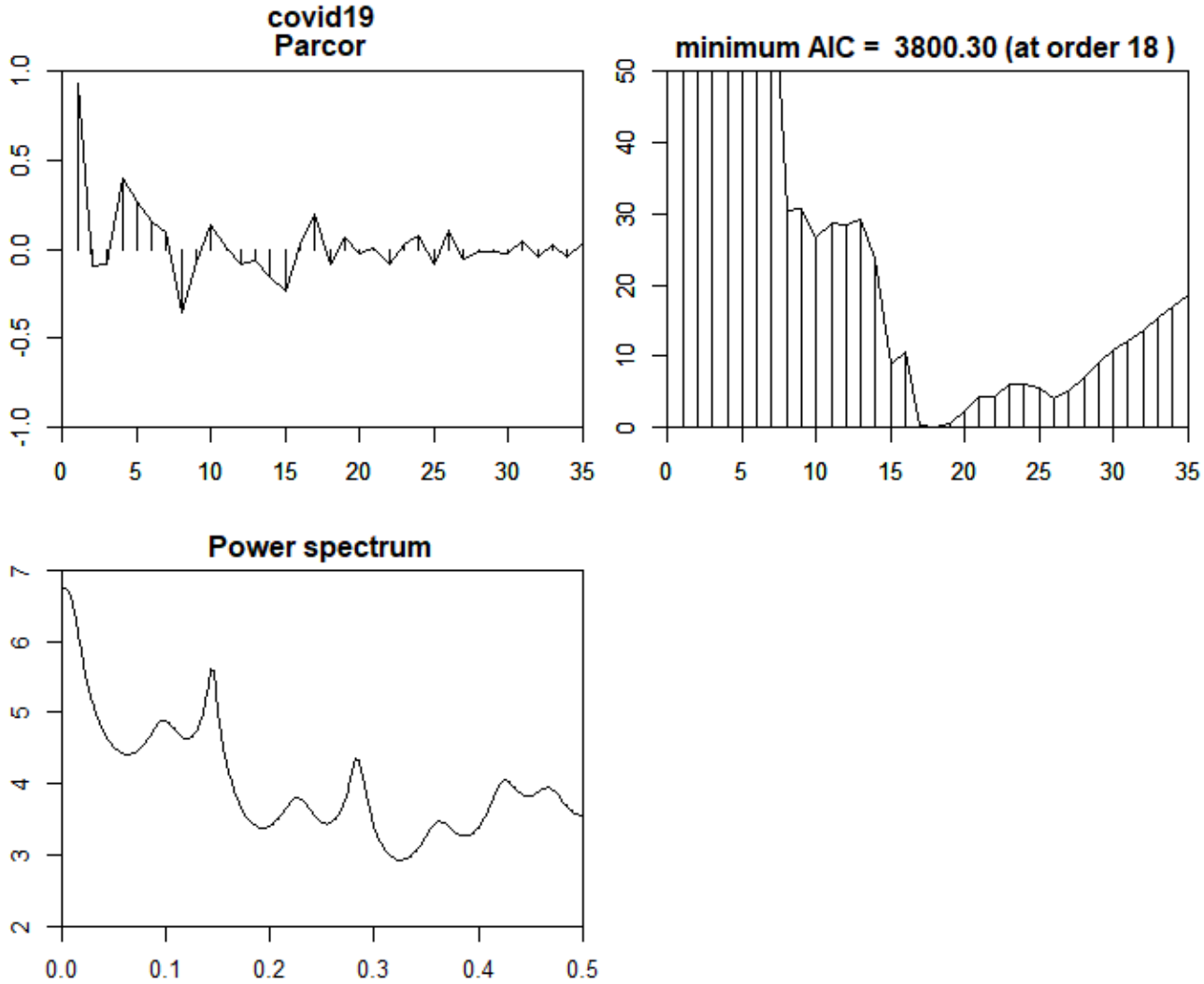
ピリオドグラムの計算：　Covid-19データ

```
library(TSSS)
par(mar=c(2,2,3,1))
period( covid19, window=0 )
```



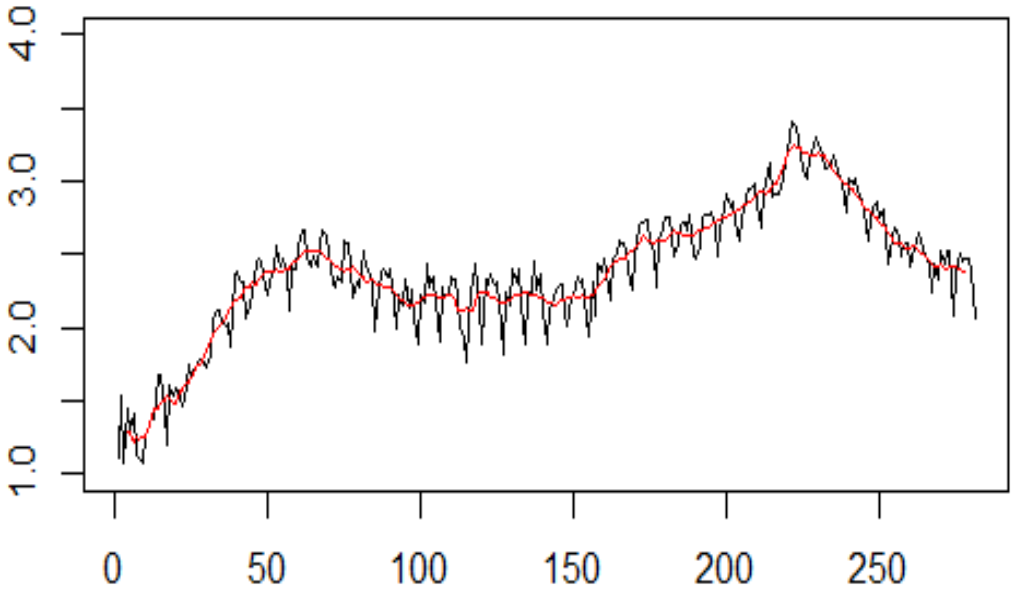
ARスペクトルの計算：　Covid-19データ

```
library(TSSS)
par(mar=c(2,2,3,1))
arfit( covid19, window=0 )
```

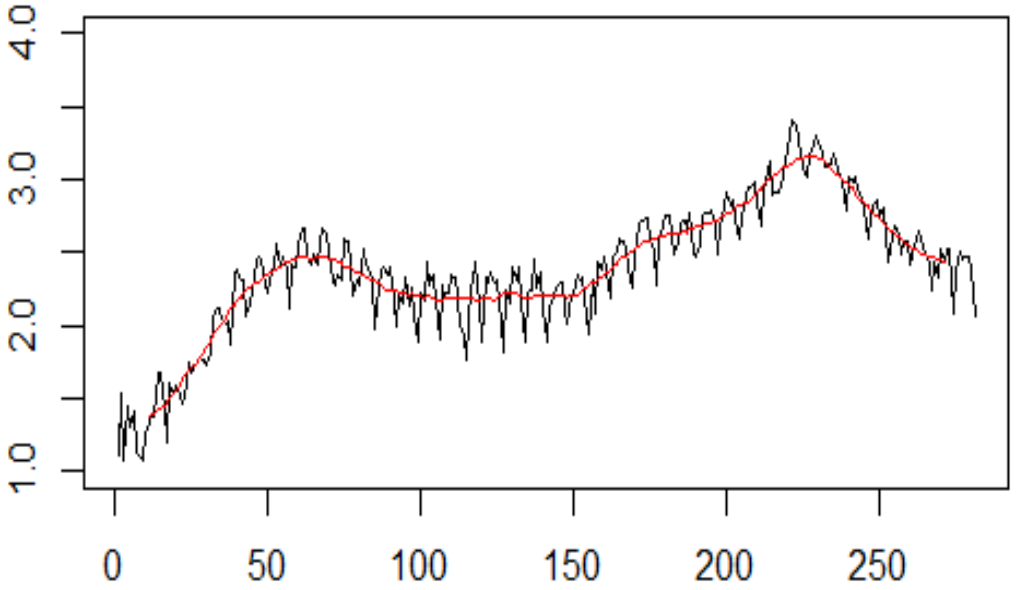


移動平均（1週間と3週間）

```
y <- log10( covid19 )
plot( y, ylim=c(1,4) )
ndata<-length( y )
t <- y
t[1:ndata] <-NA
kfilter <- 3            # 項数=2*kfilter+1
n0 <-kfilter+1
n1 <-ndata-kfilter
for(i in n0:n1){
i0 <-i-kfilter
i1 <-i+kfilter
t[i] <-mean( y[i0:i1] )
}
lines( t, col=2, lwd=2, ylim=c(1,4) )
```



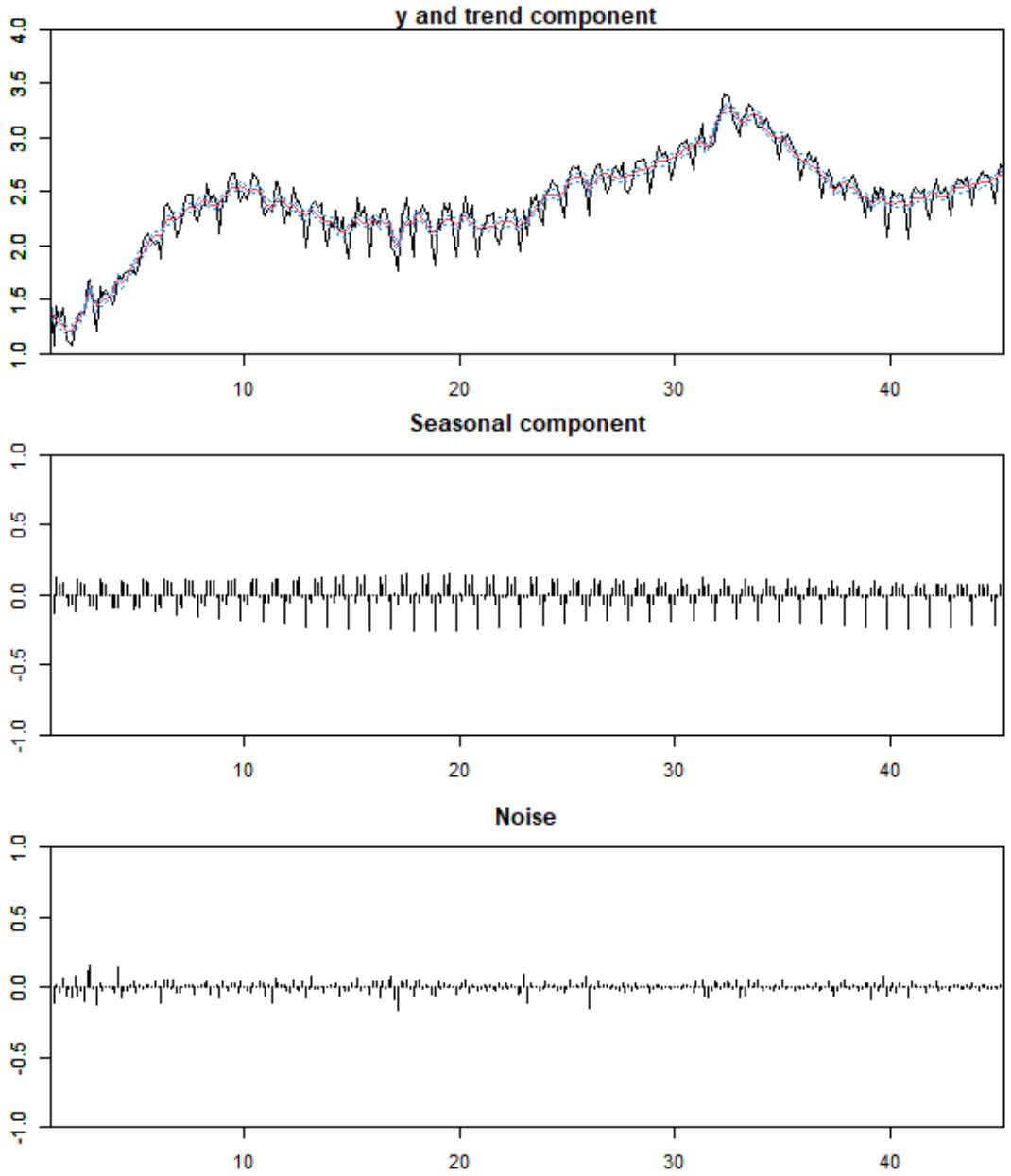
kfilter <- 10
他は同じ



COVID-19 データ：　季節調整

```
y <- ts( log10(covid_19),frequency=7 )
par(mar=c(2,2,1,1))
plot( y )

par(mar=c(2,2,3,1))
season( y , tau2.ini=c(0.5,0.0001) )
```

tau2	9.22756e-01 5.78841e-02
sigma2	3.34980e-03
log-likelihood	228.505
aic	-439.010



period の指定と ts属性の frequency が紛らわしいのですが
ヘルプにもありますように ts属性がNULLの場合 periodの指定が有効になります.

・ts属性を使わず　period = 7 で指定したい場合

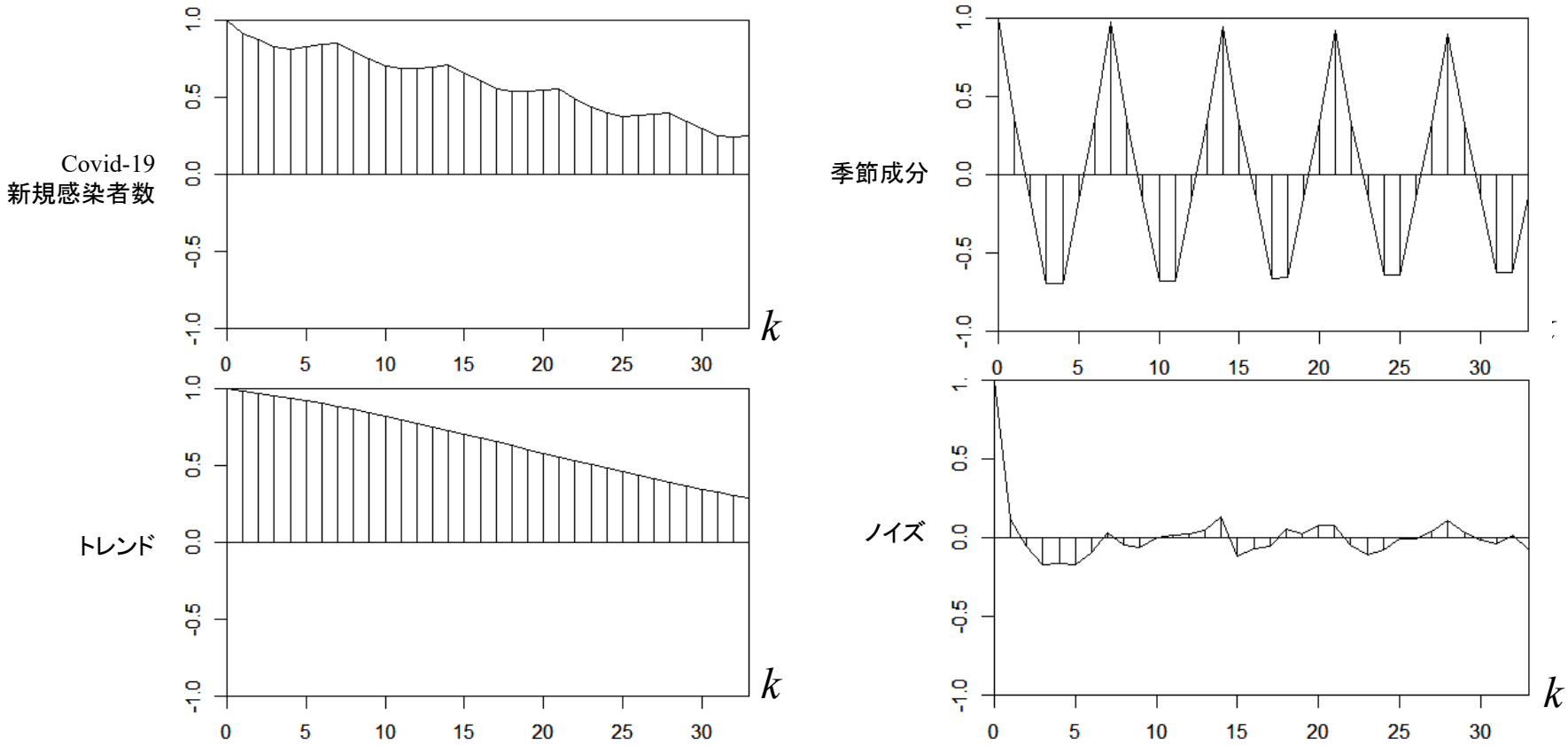
```
data <- read.csv("Tokyo_new.csv")
# ここで covid は list になっているので
covid <- log10(data[[1]]) または covid <- unlist(log10(data))
season( covid,trend.order=2,seasonal.order=1,period=7)
```

・ts属性を使う場合

```
data <- read.csv("Tokyo_new.csv")
# weeklyデータを指定するには frequency = 365.25/7 となります.
covid <- ts(log10(data), frequency=365.25/7)
season( covid,trend.order=2,seasonal.order=1,period=7)
```

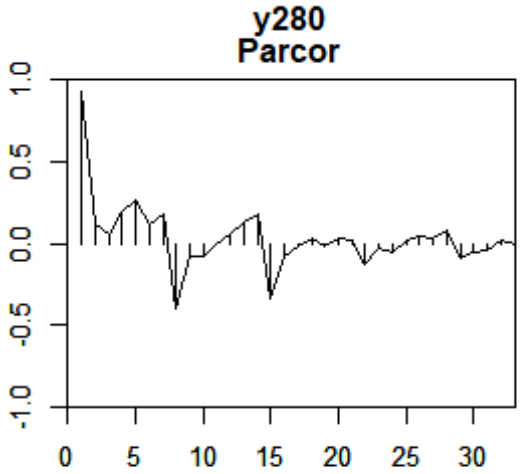
Covid-19 データの自己相関関数

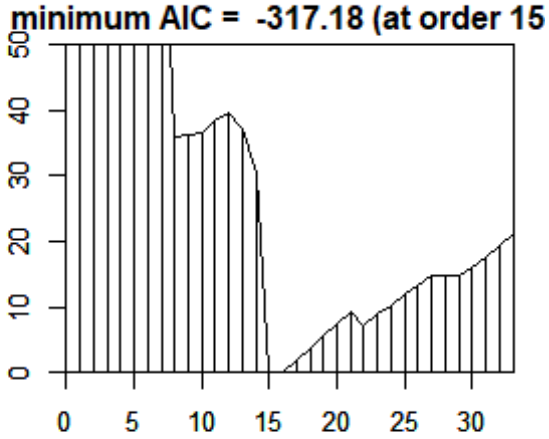
```
library(TSSS)
y <- log10( covid19 )
unicor( y )
z <- season( y , tau2.ini=c(0.5,0.0001) )

unicor( z$trend )
unicor( z$seasonal )
noise <- y - z$trend - z$seasonal
unicor( noise )
```

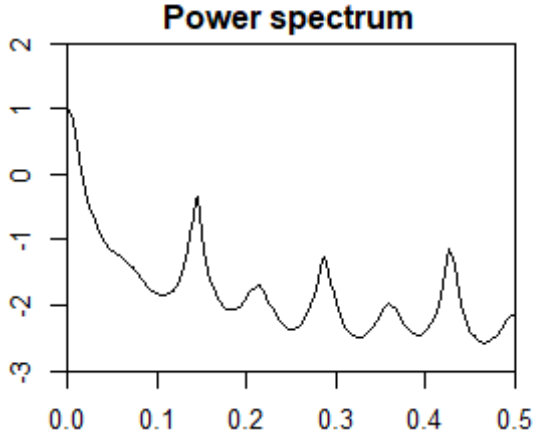


ARモデルによるCovid-19データの予測(1) ARモデルの推定

```
y <- log10( covid19 )
y280 <- y[1:280]
par( mar=c(2,2,3,1) )
arfit( y280 )
```

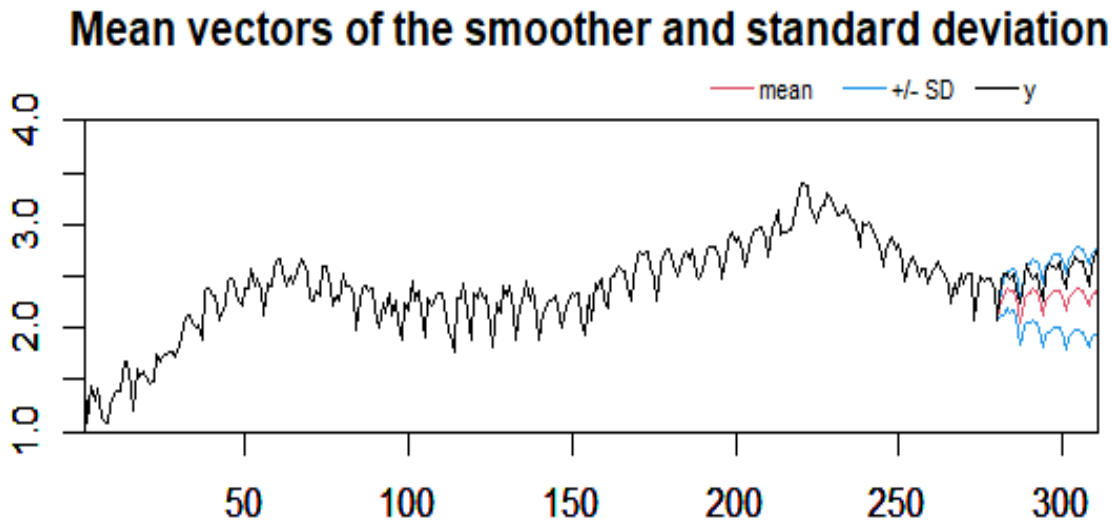


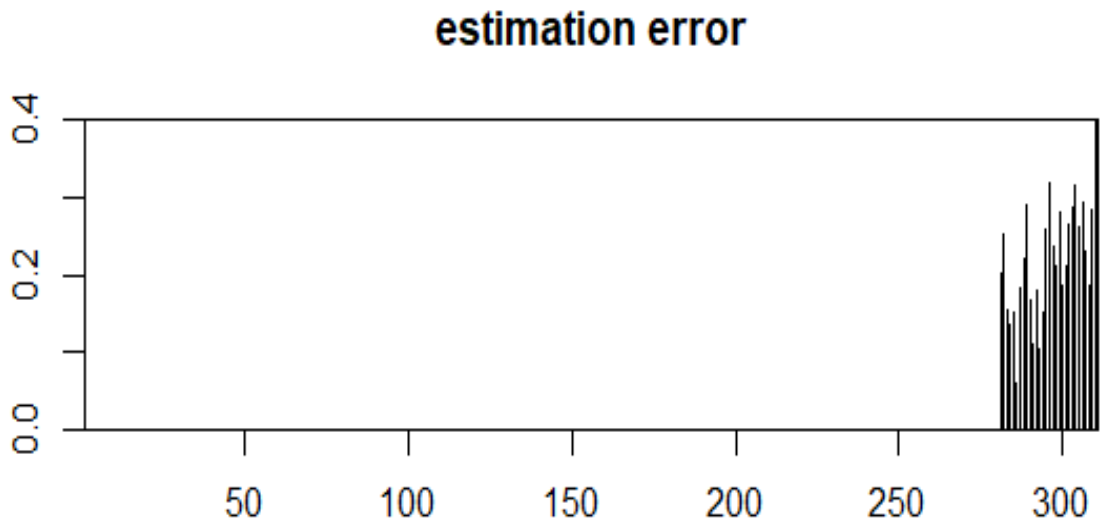




ARモデルによるCovid-19データの予測(2)

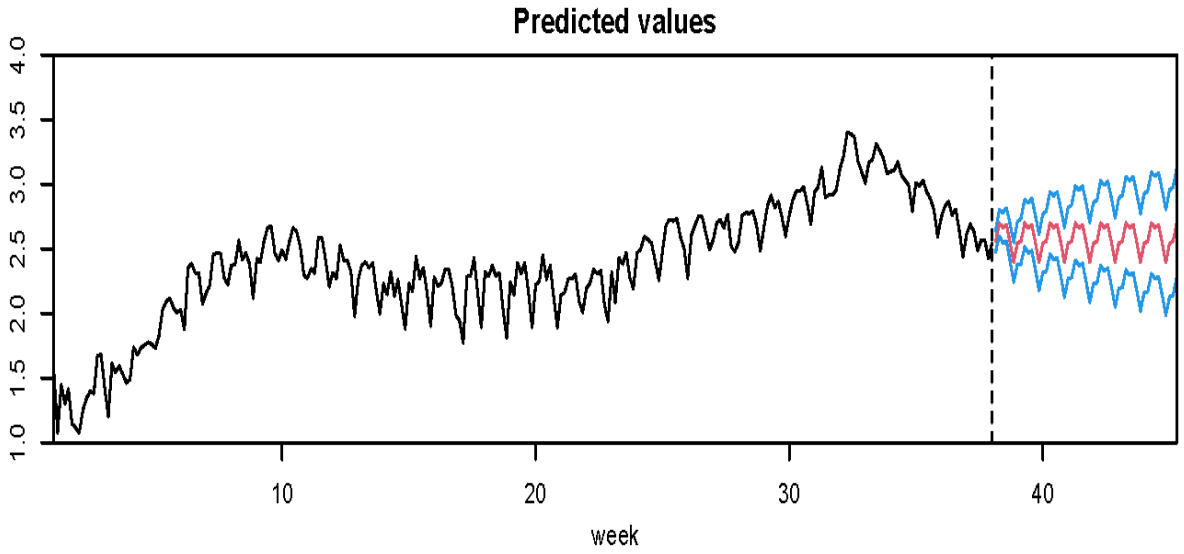
```
n <- length( y )

z1 <- arfit( y280, plot = FALSE)

tau2 <- z1$sigma2
# m = maice.order, k=1
m1 <- z1$maice.order
arcoef <- z1$arcoef[[m1]]
f <- matrix(0.0e0, m1, m1)
f[1, ] <- arcoef
if (m1 != 1)
  for (i in 2:m1) f[i, i-1] <- 1
g <- c(1, rep(0.0e0, m1-1))
h <- c(1, rep(0.0e0, m1-1))
q <- tau2[m1+1]
r <- 0.0e0
x0 <- rep(0.0e0, m1)
v0 <- NULL

s1 <- tsmooth( y, f, g, h, q, r, x0, , filter.end = 280, predict.end =n )
plot(s1, y)
```





季節調整モデルによる長期予測

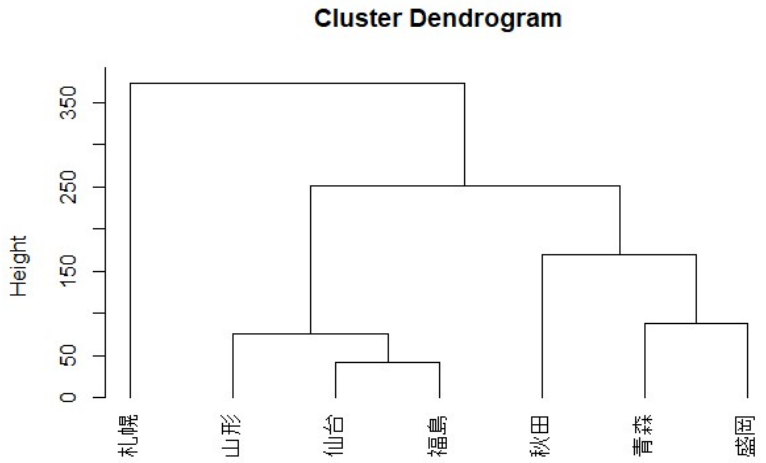
```
y <- ts( log10(covid_19),frequency=7 )
season( y, filter=c(1,280) )
```



４．クラスター分析

移動距離データ

```
dist <- read.csv("distance3.csv", row.name = 1 )
rc <- hclust( as.dist(dist) )
plot(rc)
plot(rc,hang=-1)
```




```
rc$merge
```

```
     [,1] [,2]
[1,]   -4   -7
[2,]   -6    1
[3,]   -2   -3
[4,]   -5    3
[5,]    2    4
[6,]   -1    5
```

```
rc$height
```

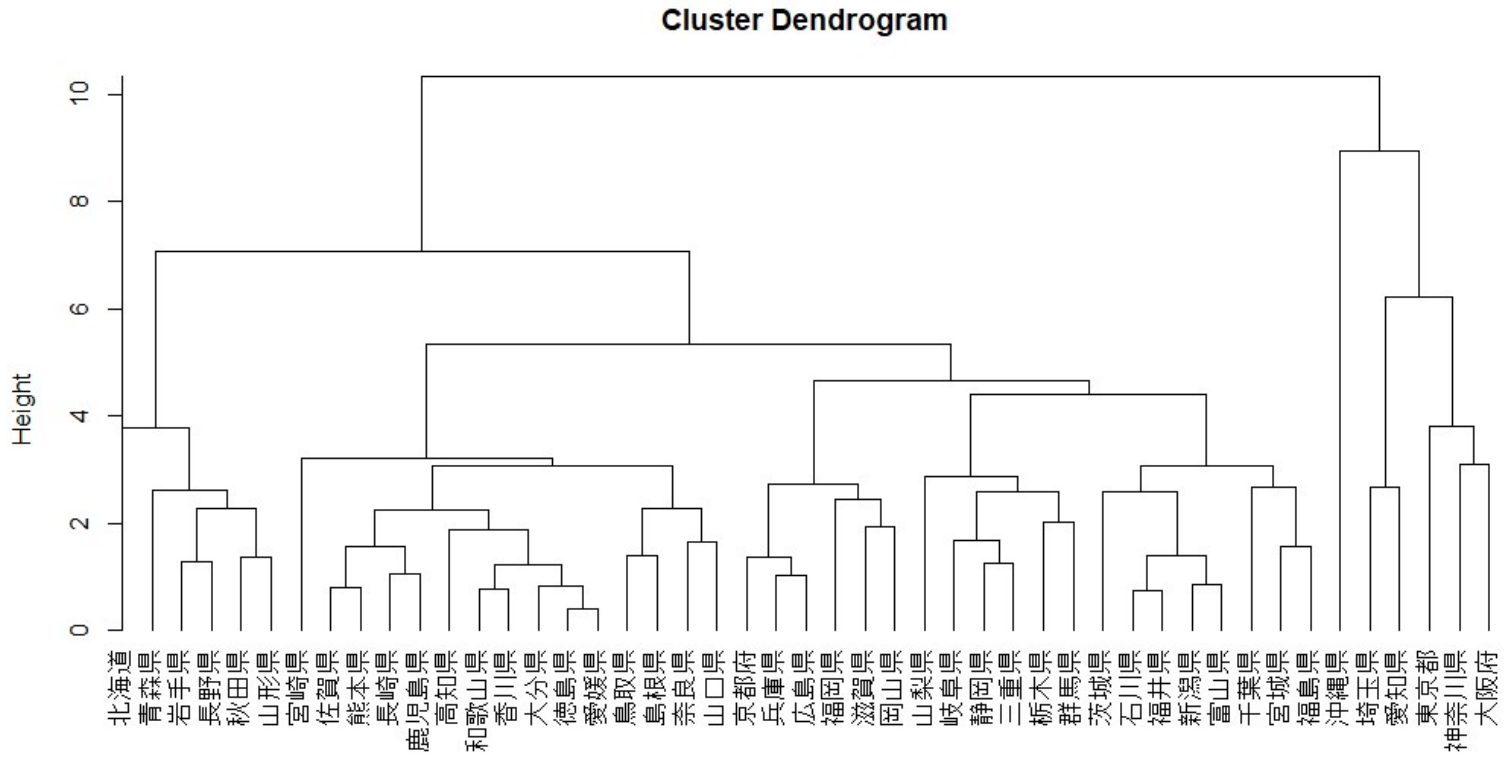
```
[1]  42.0  76.5  88.0 170.0 251.0 373.0
```

	北海道	青森	岩手	宮城	秋田	山形	福島
北海道	0						
青森	313.5	0					
岩手	290.5	88	0				
宮城	253.5	146.5	61.5	0			
秋田	373	170	97	153.5	0		
山形	319	247.5	161	76.5	251	0	
福島	283	193.5	115,5	42	205.5	72.5	0

（移動距離：県庁所在地のJR駅を朝8時に出発した時の到達時間）

例：都道府県の多変量データ

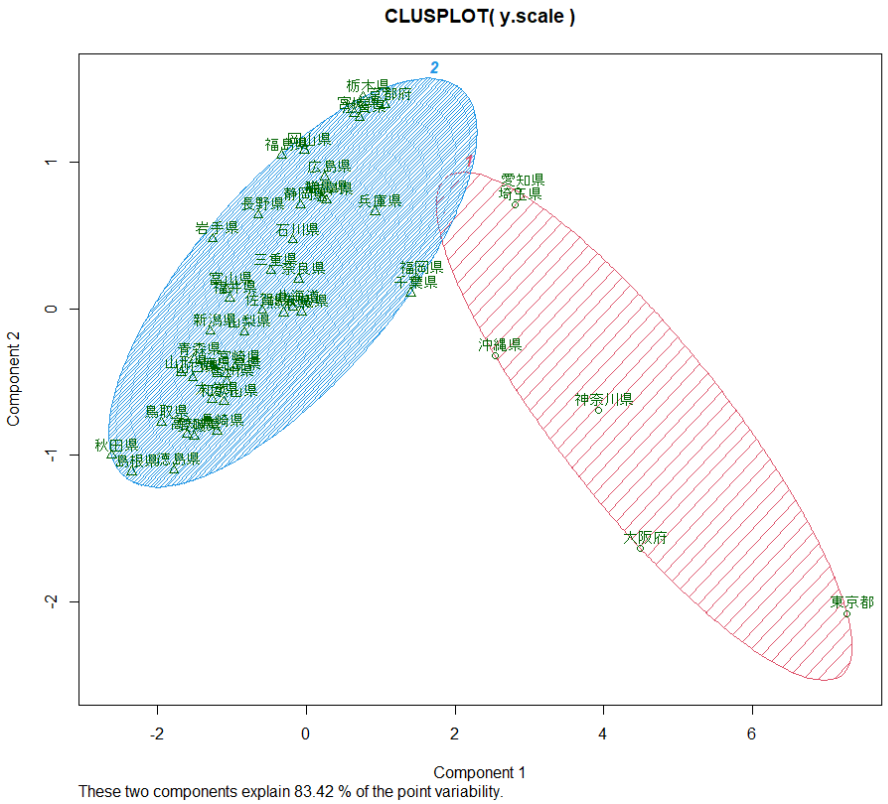
```
ypref <- read.csv( "Tokyo_Covid-19_regression.csv", row.name = 1  )
ypref <- read.csv( "Prefecture_regression.csv", row.name = 1 )
pref <- ypref[,3:11]
pref.scale = scale( pref )
pref.d <- dist( pref.scale )

rc_pref <- hclust( pref.d )
plot(rc_pref)
plot(rc_pref, hang=-1)
```



非階層的クラスタリング: k-means法

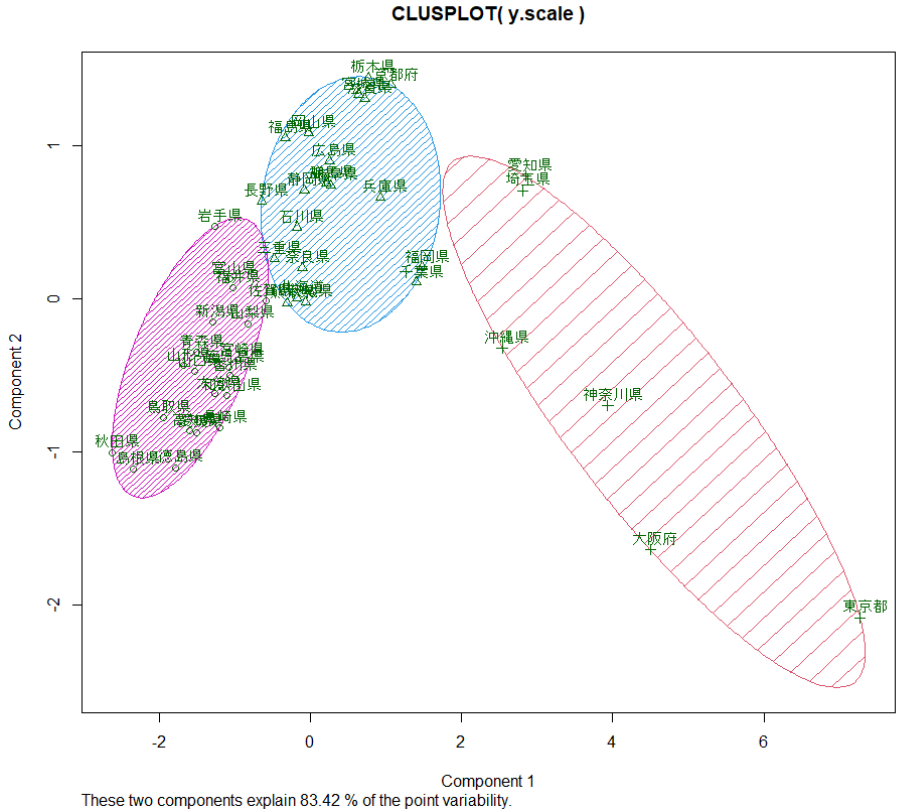
k =2の場合

```
library(cluster)
pref_data <- read.csv( "Prefecture_regression.csv", row.name = 1 )
x <- pref_data[,c(1,3,4,5,6)]
y <- x
y[,1] <- log(x[,1])
y[,3] <- log(x[,3])
y.scale = scale( y )
y_km <- kmeans( y.scale,2)
clusplot( y.scale,y_km$cluster, color=TRUE, shade=TRUE, labels=2,
lines=0 )
```



k =3の場合

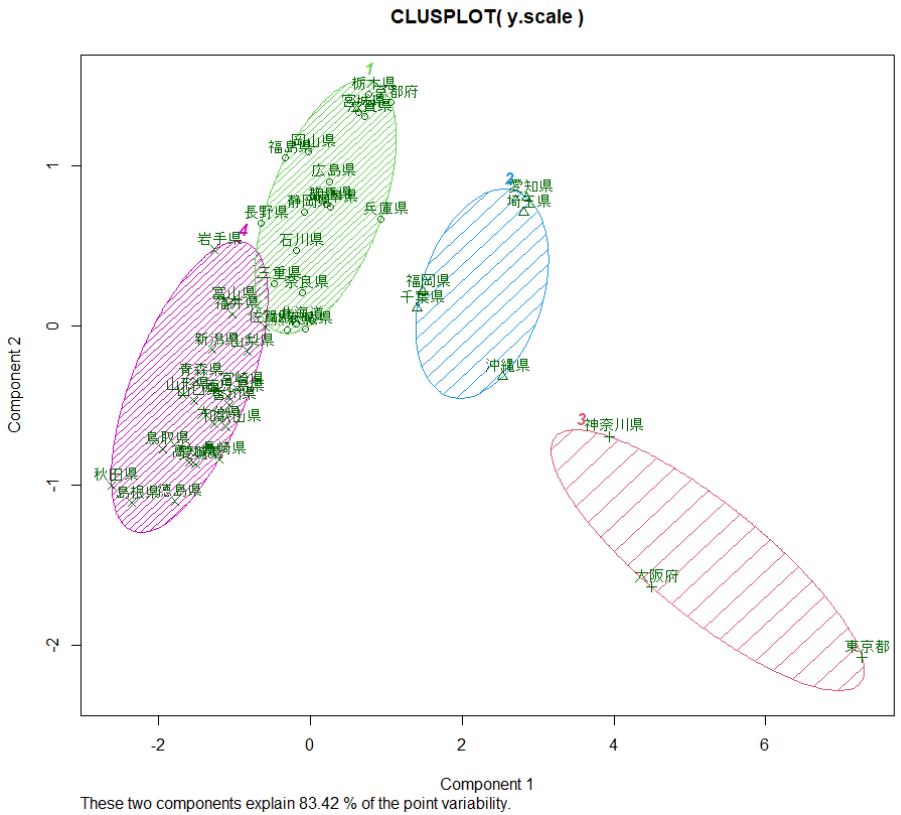
```
y_km <- kmeans( y.scale,3 )
clusplot( y.scale,y_km$cluster, color=TRUE, shade=TRUE, labels=3,
lines=0 )
```



非階層的クラスタリング: k-means法

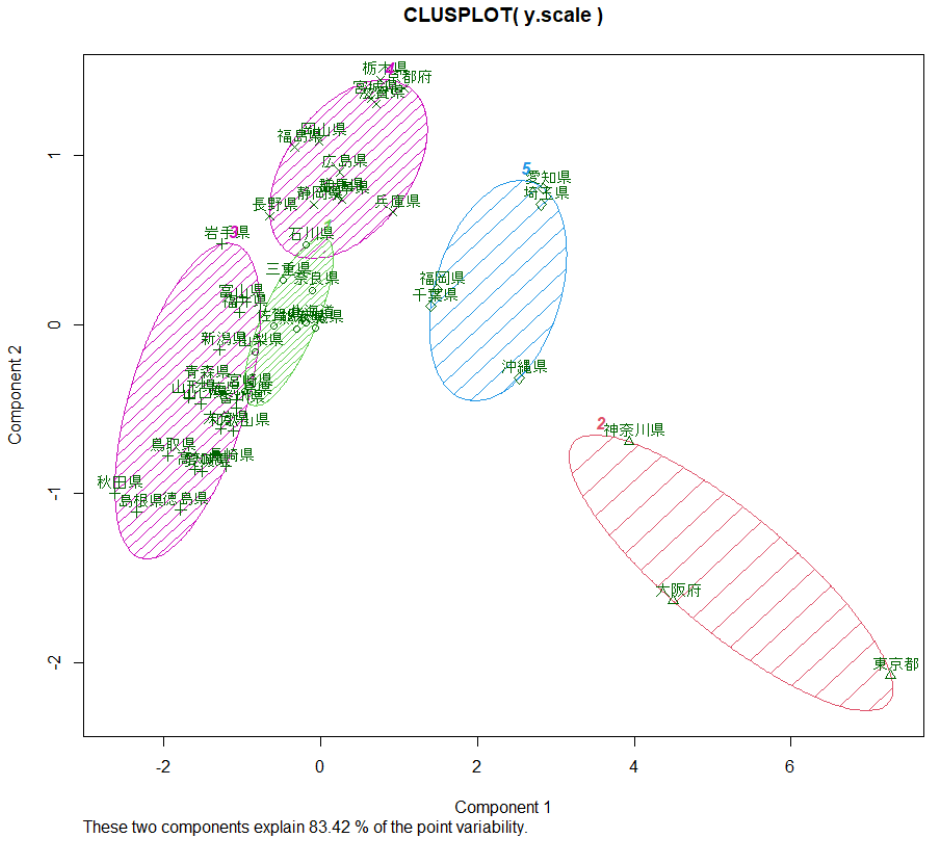
k =4の場合

```
y_km <- kmeans( y.scale,4 )
clusplot( y.scale,y_km$cluster, color=TRUE, shade=TRUE,
labels=2, lines=0 )
```



k =5 の場合

```
y_km <- kmeans( y.scale,5 )
clusplot( y.scale,y_km$cluster, color=TRUE, shade=TRUE, labels=2,
lines=0 )
```



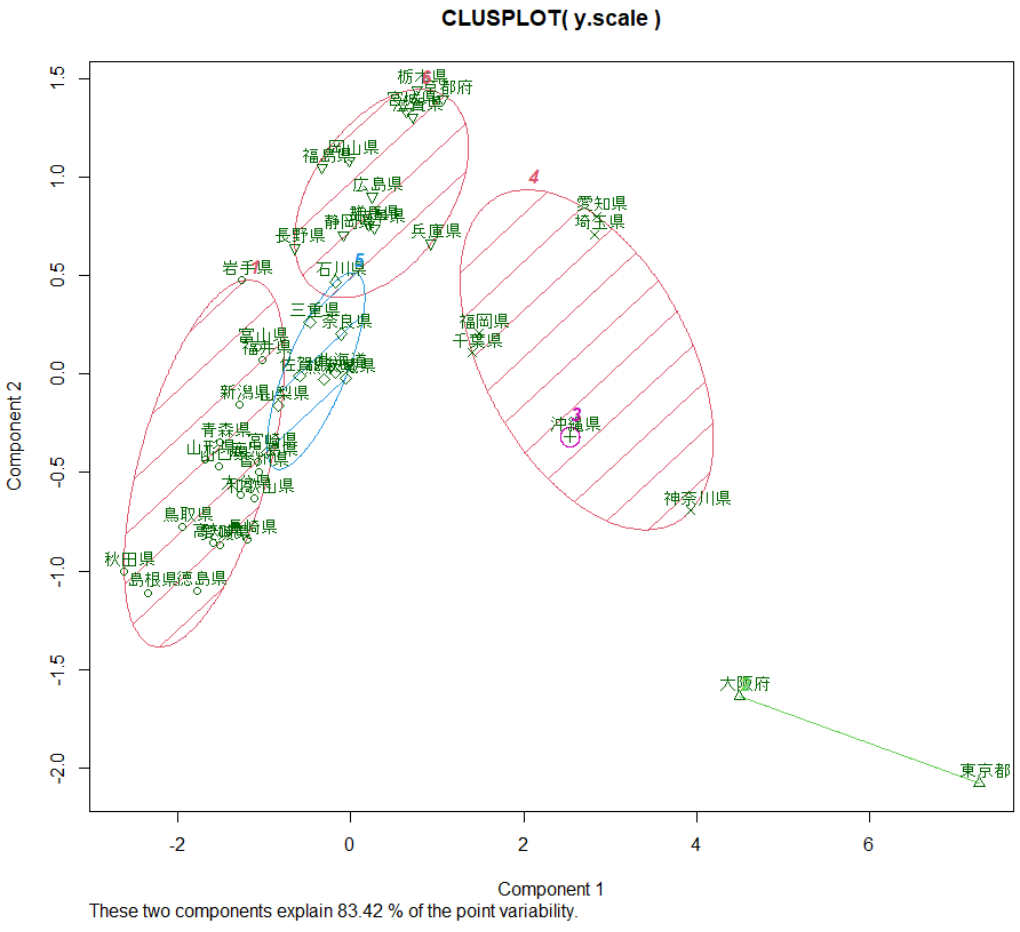
非階層的クラスタリング: k-means法

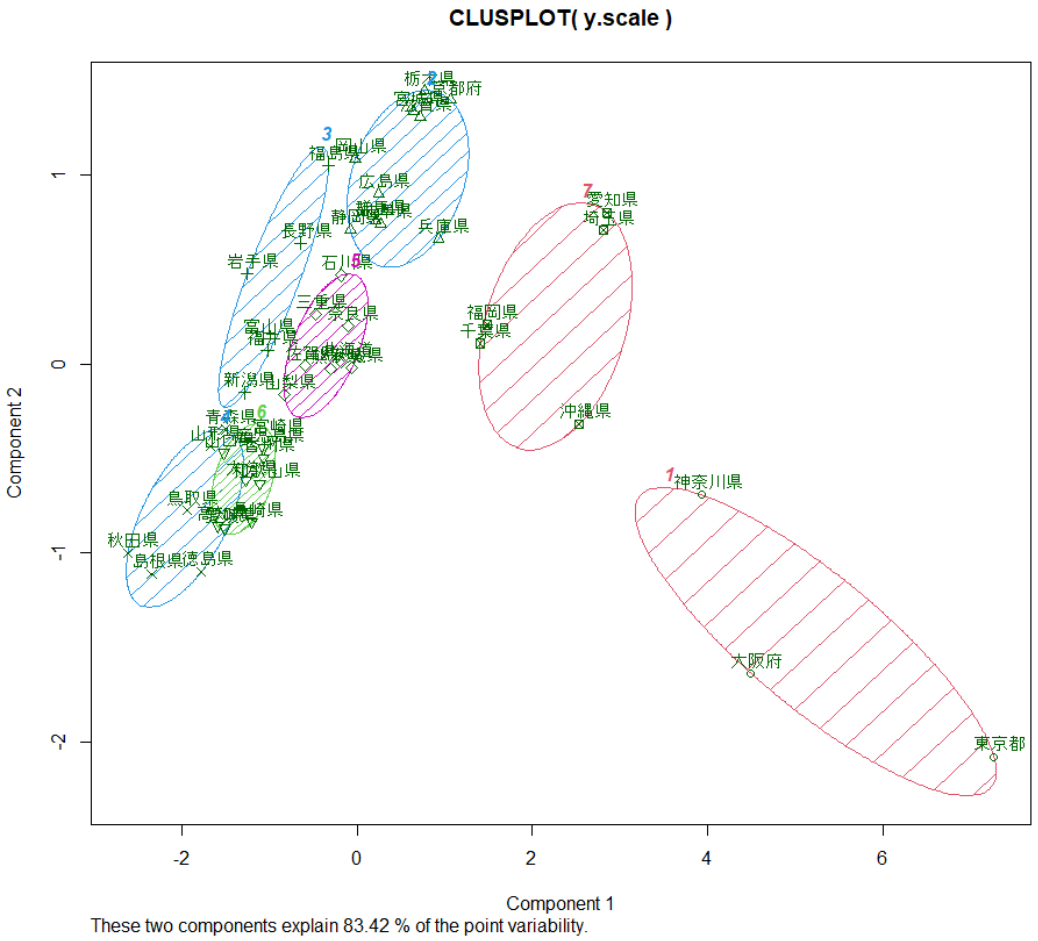
k =6の場合

```
y_km <- kmeans( y.scale,6 )
clusplot( y.scale,y_km$cluster, color=TRUE, shade=TRUE,
labels=2, lines=0 )
```

k =7 の場合

```
y_km <- kmeans( y.scale,7 )
clusplot( y.scale,y_km$cluster, color=TRUE, shade=TRUE, labels=2,
lines=0 )
```





5．パターン発見

データ出展元
　Groceries：　Rパッケージarules 標準データ

Aprioriアルゴリズム(1)

```
install.packages("arules")
library(arules)
# arulesに付属しているサンプルデータセット
data(Groceries)

# Aprioriアルゴリズムによるアソシエーション分析の実行
# support（支持度）が0.5%以上で、cofidence（確信度）が6%以上のルールを
抽出する
A_rules <- apriori(Groceries,parameter = list(support=0.005,confidence=0.6))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.6    0.1    1 none FALSE            TRUE       5   0.005      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 49

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [120 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [22 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Apriorアルゴリズム（１）

```
# 相関ルールを見る
inspect(head(A_rules))
```

```
    lhs                                rhs                  support     confidence coverage    lift     count
[1] {root vegetables,onions}        => {other vegetables}   0.005693950 0.6021505  0.009456024 3.112008 56
[2] {tropical fruit,curd}           => {whole milk}         0.006507372 0.6336634  0.010269446 2.479936 64
[3] {domestic eggs,margarine}       => {whole milk}         0.005185562 0.6219512  0.008337570 2.434099 51
[4] {butter,domestic eggs}          => {whole milk}         0.005998983 0.6210526  0.009659380 2.430582 59
[5] {butter,whipped/sour cream}     => {whole milk}         0.006710727 0.6600000  0.010167768 2.583008 66
[6] {butter,bottled water}          => {whole milk}         0.005388917 0.6022727  0.008947636 2.357084 53
```

単にinspectで結果を表示するとsupport, confidence, lift のいずれかの順に表示されているわけではないので要注意．次頁のように sort で並べなおす．

Aprioriアルゴリズム（１）

```
# 相関ルールを見る
A_rules <- sort( A_rules,d=T, by="support" )　#supportでソート
inspect(head(A_rules))
```

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{butter,yogurt}	=>	{whole milk}	0.009354347	0.6388889	0.01464159	2.500387	92
[2]	{root vegetables,butter}	=>	{whole milk}	0.008235892	0.6377953	0.01291307	2.496107	81
[3]	{root vegetables,other vegetables,yogurt}	=>	{whole milk}	0.007829181	0.6062992	0.01291307	2.372842	77
[4]	{tropical fruit,other vegetables,yogurt}	=>	{whole milk}	0.007625826	0.6198347	0.01230300	2.425816	75
[5]	{tropical fruit,domestic eggs}	=>	{whole milk}	0.006914082	0.6071429	0.01138790	2.376144	68
[6]	{butter,whipped/sour cream}	=>	{whole milk}	0.006710727	0.6600000	0.01016777	2.583008	66

```
A_rules <- sort( A_rules,d=T, by="lift" )　　# lift でソート
inspect(head(A_rules))
```

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{citrus fruit,root vegetables,whole milk}	=>	{other vegetables}	0.005795628	0.6333333	0.009150991	3.273165	57
[2]	{pip fruit,root vegetables,whole milk}	=>	{other vegetables}	0.005490595	0.6136364	0.008947636	3.171368	54
[3]	{pip fruit,whipped/sour cream}	=>	{other vegetables}	0.005592272	0.6043956	0.009252669	3.123610	55
[4]	{root vegetables,onions}	=>	{other vegetables}	0.005693950	0.6021505	0.009456024	3.112008	56
[5]	{tropical fruit,root vegetables,yogurt}	=>	{whole milk}	0.005693950	0.7000000	0.008134215	2.739554	56
[6]	{pip fruit,root vegetables,other vegetables}	=>	{whole milk}	0.005490595	0.6750000	0.008134215	2.641713	54

Aprioriアルゴリズム（２）

```
rules <- apriori(Groceries,parameter =list(support=0.001,confidence=0.5))
# 相関ルールを見る
inspect(head(rules))
```

結果はsupport, confidence, lift のいずれかの順に表示されているわけではないので要注意．次頁のようにsort で並べなおす．

lhs		rhs	support	confidence	coverage	lift	count
[1]	{honey}	=> {whole milk}	0.001118454	0.7333333	0.001525165	2.870009	11
[2]	{tidbits}	=> {rolls/buns}	0.001220132	0.5217391	0.002338587	2.836542	12
[3]	{cocoa drinks}	=> {whole milk}	0.001321810	0.5909091	0.002236909	2.312611	13
[4]	{pudding powder}	=> {whole milk}	0.001321810	0.5652174	0.002338587	2.212062	13
[5]	{cooking chocolate}	=> {whole milk}	0.001321810	0.5200000	0.002541942	2.035097	13
[6]	{cereals}	=> {whole milk}	0.003660397	0.6428571	0.005693950	2.515917	36

Aprioriアルゴリズム（２）

```
rules <- apriori(Groceries,parameter =list(support=0.001,confidence=0.5))
rules <- sort(rules, d=T, by="support" )    #supportの降順
# 相関ルールを見る
inspect(head(rules))
```

```
 lhs                                                rhs                  support     confidence coverage    lift     count
[1] {other vegetables,yogurt}                    => {whole milk}         0.02226741  0.5128806  0.04341637 2.007235 219
[2] {tropical fruit,yogurt}                      => {whole milk}         0.01514997  0.5173611  0.02928317 2.024770 149
[3] {other vegetables,whipped/sour cream}
                                                 => {whole milk}         0.01464159  0.5070423  0.02887646 1.984385 144
[4] {root vegetables,yogurt}                     => {whole milk}         0.01453991  0.5629921  0.02582613 2.203354 143
[5] {pip fruit,other vegetables}                 => {whole milk}         0.01352313  0.5175097  0.02613116 2.025351 133
[6] {root vegetables,yogurt}                     => {other vegetables}   0.01291307  0.5000000  0.02582613 2.584078 127
```

```
rules <- sort(rules, d=T, by="lift" )      # lift の降順
inspect(head(rules))
```

```
 lhs                                                rhs                  support     confidence coverage    lift     count
[1] {Instant food products,soda}                 => {hamburger meat}     0.001220132 0.6315789  0.001931876 18.99565 12
[2] {soda,popcorn}                               => {salty snack}        0.001220132 0.6315789  0.001931876 16.69779 12
[3] {flour,baking powder}                        => {sugar}              0.001016777 0.5555556  0.001830198 16.40807 10
[4] {ham,processed cheese}                       => {white bread}        0.001931876 0.6333333  0.003050330 15.04549 19
[5] {whole milk,Instant food products}           => {hamburger meat}     0.001525165 0.5000000  0.003050330 15.03823 15
[6] {other vegetables,curd,yogurt,whipped/sour cream}
                                                 => {cream cheese }      0.001016777 0.5882353  0.001728521 14.83409 10
```