DEEFを非負関数一般の分解とみなす

ryamada

2020年11月8日

DEEF は確率密度関数を指数型分布族表現分解する手法

関数内積の行列を推定し、その対数を取って固有値分解し座標を取り出す手法がDEEF。

難数内積は確率密度分布に限らず存在するから

$$exp(C(x) + F(x)\Theta - \psi(\Theta))$$

と表せる関数についてなら、適用可能。

ただし、この表現が取れる関数は、非負値関数。

パッケージ deefの例として得られるCx,Fxを用いて、積分して１にならない関数も復元してみる

正規分布様の関数の集合であることがわかる。

正規分布様の関数が釣り鐘型とすると、その反転も「同類」らしいことがわかる。

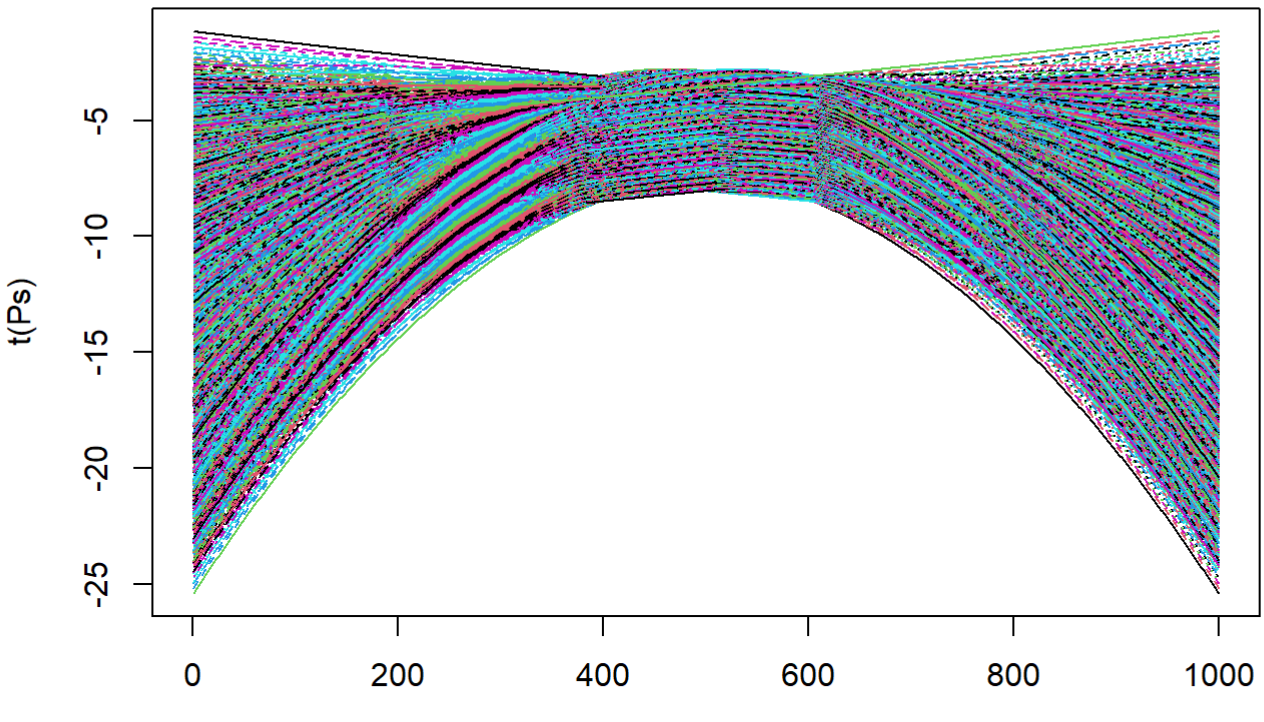
```
library(deef)
P <- Distset2D$P
ip_mat <- Distset2D$ip_mat
result <- DEEF(Distset2D$P,Distset2D$ip_mat)
eigen_value <- result$eigenvalue
Theta <- result$Theta
Fx <- result$Fx
Cx <- result$Cx
```

```
# 1,2,900 are the main components

axes <- c(1,2,900)

# observed theta coordinate range of these three axes
theta_range <- apply(Theta[,axes],2,range)
print(theta_range)
```

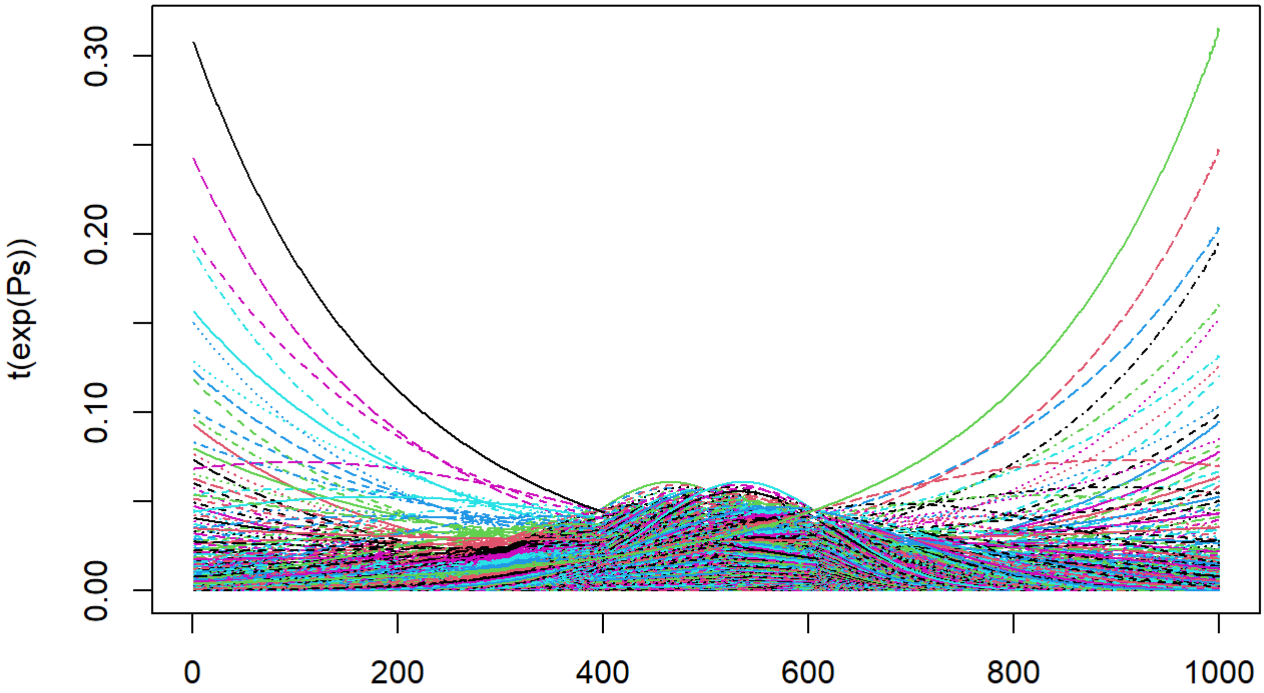
```
##            [,1]       [,2]      [,3]
## [1,] -0.2146877 -0.2399604 -1.170557
## [2,]  0.1675535  0.2399604 -1.013178
```

```
# 分布関数である制約を取り払って、theta1,theta2,theta900を指定して関数を再現する
theta1 <- seq(from = -0.25, to = 0.25, length = 11)
theta2 <- seq(from = -0.25, to = 0.25, length = 11)
theta3 <- seq(from = -1.30, to = 0, length = 21) # 負固有値の軸の座標値は負

thetas <- as.matrix(expand.grid(theta1,theta2,theta3))


F3 <- t(Fx[axes,])

Ps <- matrix(0,length(thetas[,1]),length(Cx))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- F3 %*% c(thetas[i,]) + Cx
}
```

```
matplot(t(Ps),type="l")
```




```
matplot(t(exp(Ps)),type="l")
```



双対空間の測地線(直線)をなす関数とはどんなものか

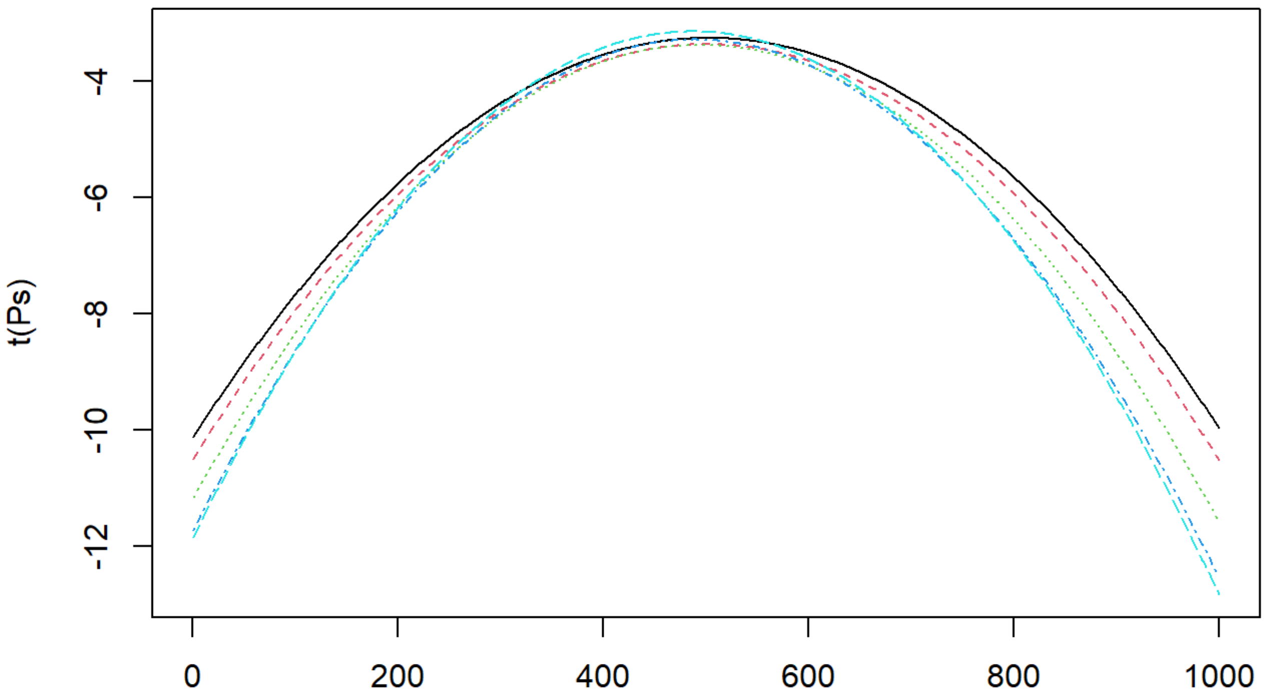
θ_1, θ_2が張る平面に乗せた半球面上の座標 $(\theta_1, \theta_2, \theta_3)$を発生させ、対応する関数を示す。

球面の前に、ある円弧でやってみる。

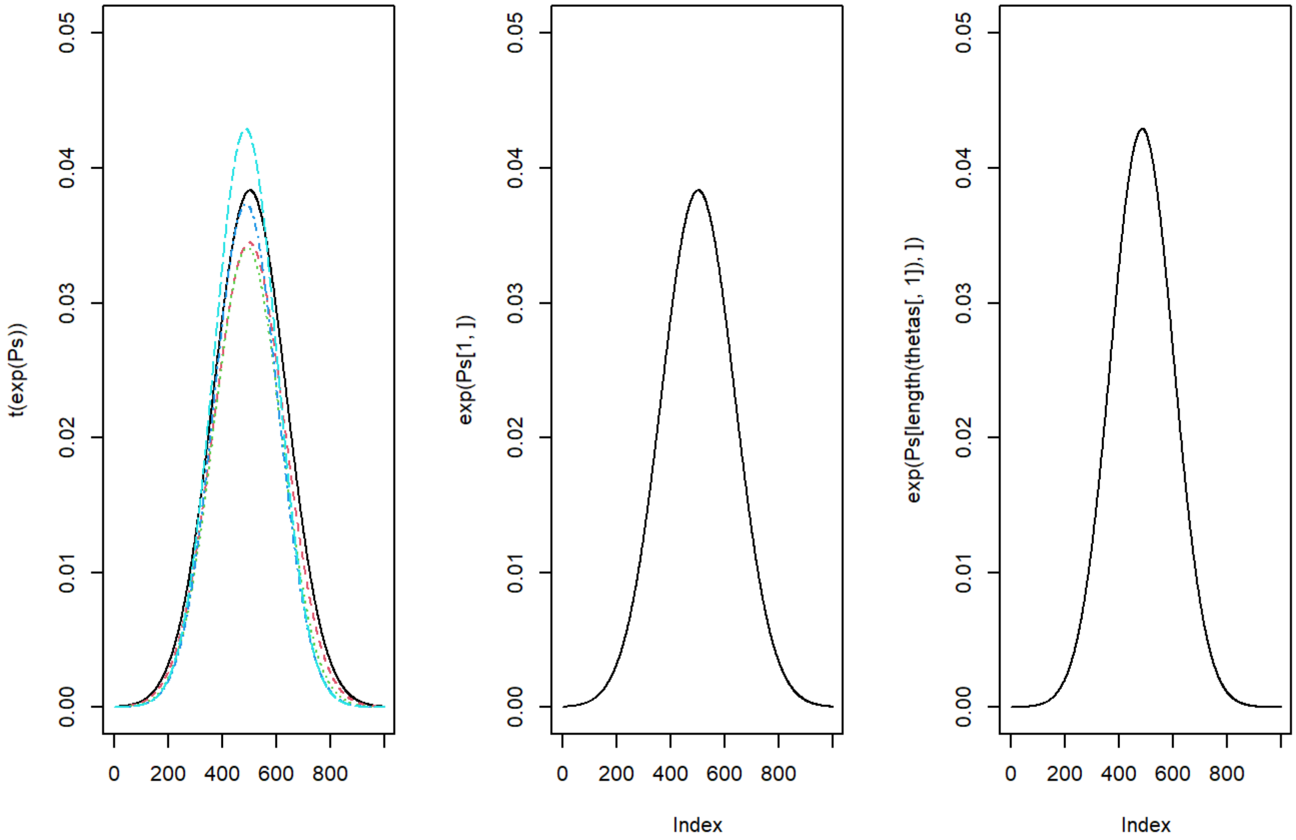
まずは少なめに。

分散が小さめな釣り鐘様関数から始まって、少し太って、また痩せる、という経過を取るらしい。

```
# 双曲空間での、半円弧(測地線・直線)に並ぶ分布たち
# t,phiで半球面座標を指定する
t <- seq(from=pi,to=2*pi,length=5)
phi <- runif(1) * 2 * pi # theta1,theta2平面上は特定の直線のみにしておく
t_phi <- expand.grid(t,phi)

# theta1 theta2 面の円の中心を定める
alpha <- -0.2
beta <- -0.5
# 半径
r <- 0.05

thetas <- cbind(cos(t_phi[,1])*cos(t_phi[,2]) + alpha,cos(t_phi[,1])*sin(t_phi[,2]) + beta,sin
(t_phi[,1])) * r


Ps <- matrix(0,length(thetas[,1]),length(Cx))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- F3 %*% c(thetas[i,]) + Cx
}

matplot(t(Ps),type="l")
```



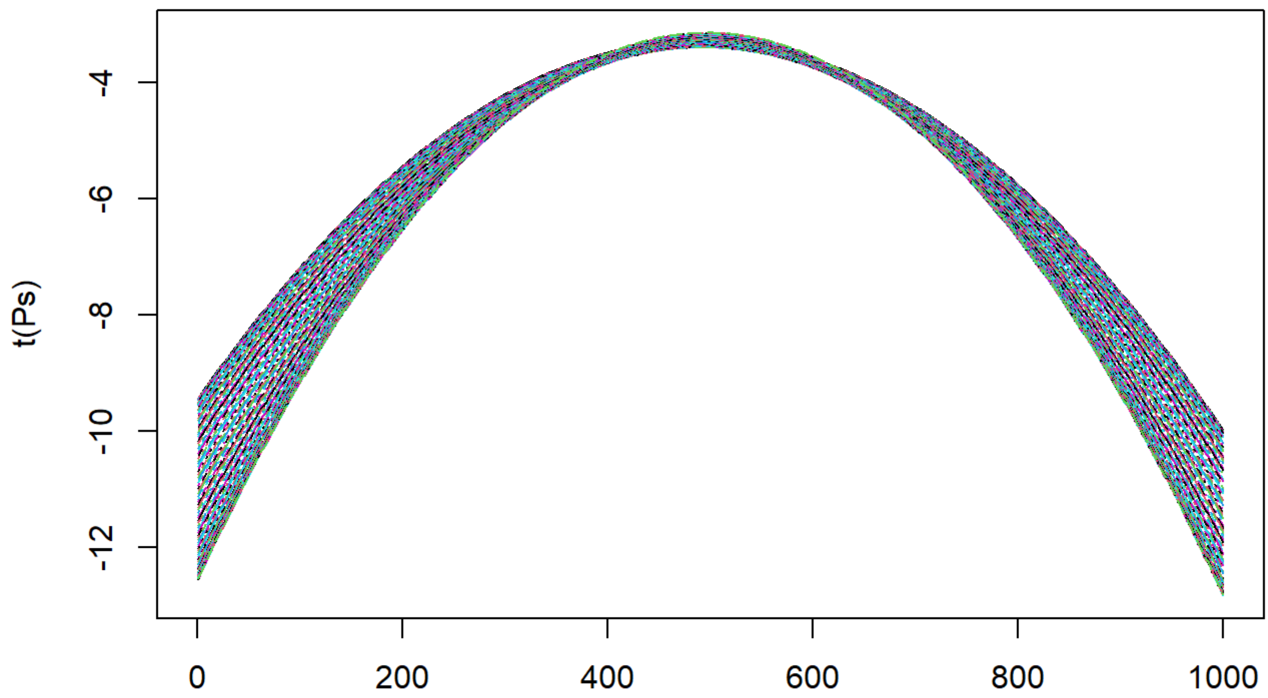

```
par(mfcol=c(1,3))
matplot(t(exp(Ps)),type="l",ylim=c(0,0.05))

plot(exp(Ps[1,]),type="l",ylim=c(0,0.05)) # theta_3 = 0 の片側
plot(exp(Ps[length(thetas[,1]),]),type="l",ylim=c(0,0.05)) # theta_3 = 0 の片側
```



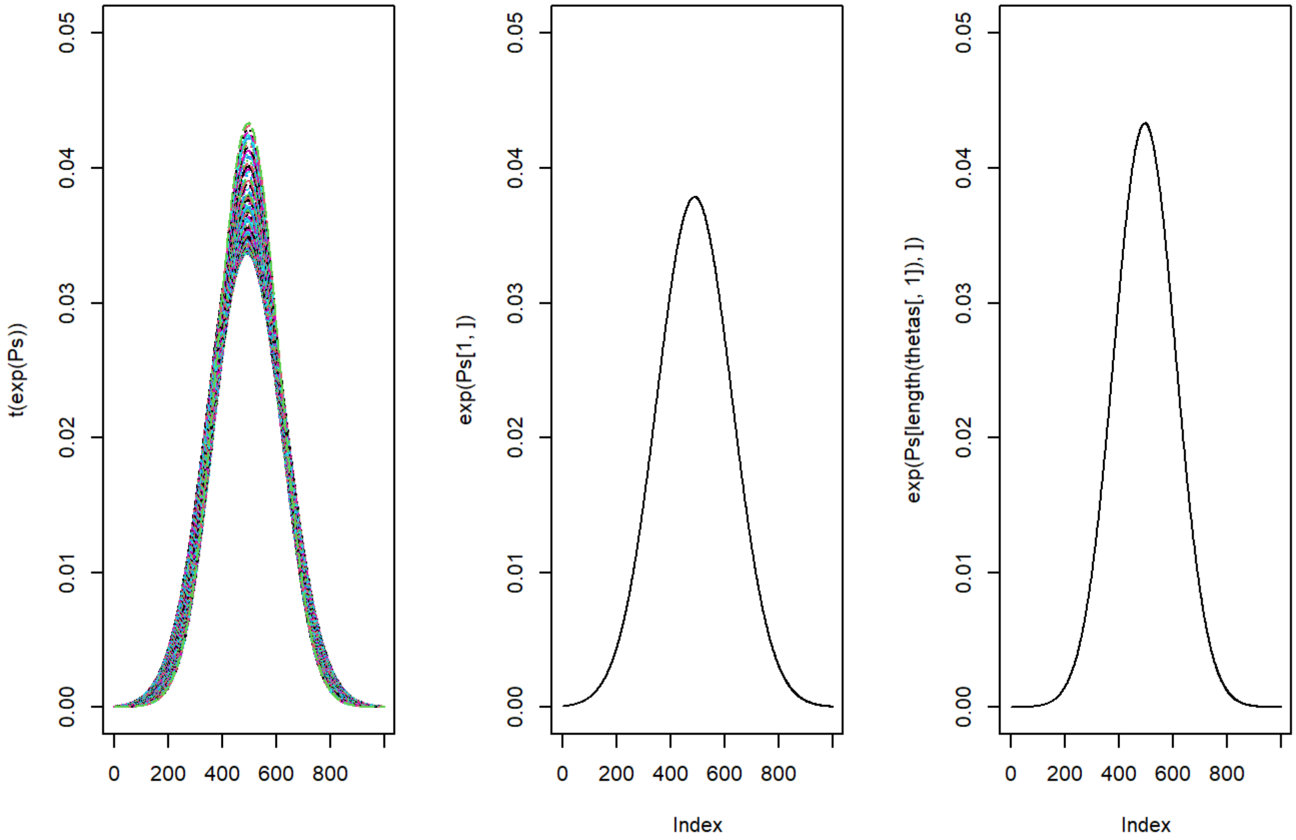

```
par(mfcol=c(1,1))
```

関数の数を増やす。

```
# 双曲空間での、半円弧(測地線・直線)に並ぶ分布たち
# t,phiで半球面座標を指定する
t <- seq(from=pi,to=2*pi,length=105)
phi <- runif(1) * 2 * pi # theta1,theta2平面上は特定の直線のみにしておく
t_phi <- expand.grid(t,phi)

# theta1 theta2 面の円の中心を定める
alpha <- -0.2
beta <- -0.5
# 半径
r <- 0.05

thetas <- cbind(cos(t_phi[,1])*cos(t_phi[,2]) + alpha,cos(t_phi[,1])*sin(t_phi[,2]) + beta,sin
(t_phi[,1])) * r


Ps <- matrix(0,length(thetas[,1]),length(Cx))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- F3 %*% c(thetas[i,]) + Cx
}

matplot(t(Ps),type="l")
```



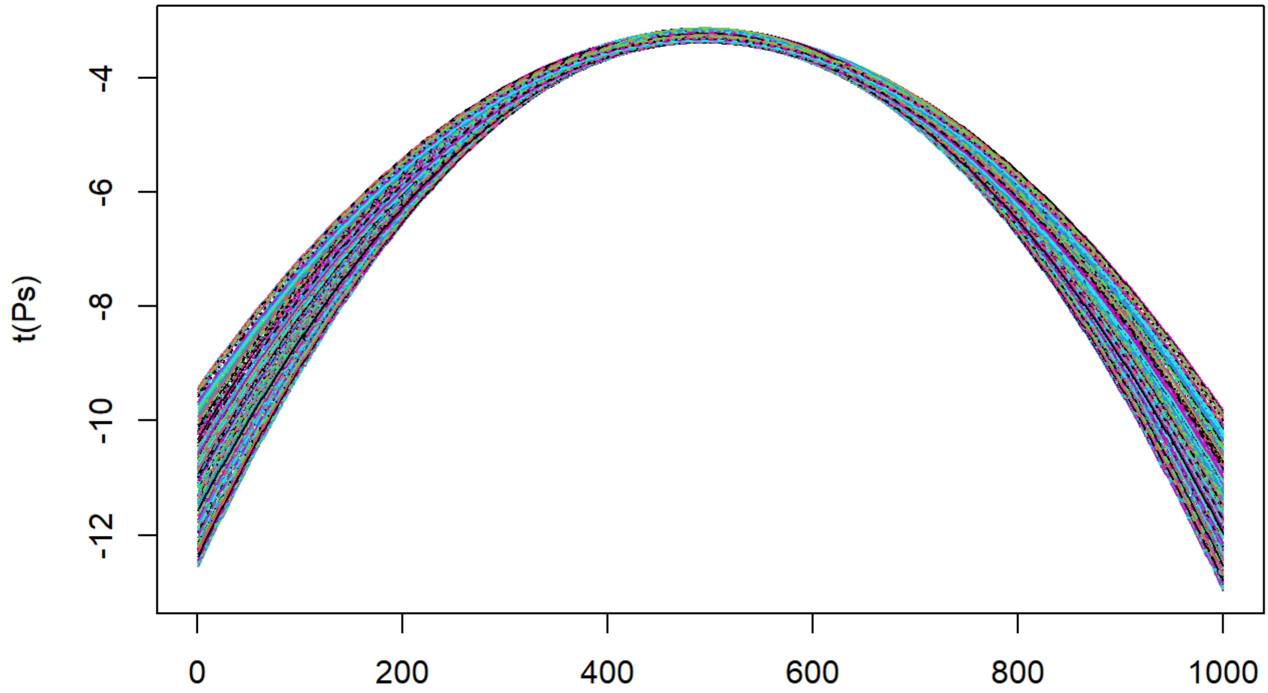

```
par(mfcol=c(1,3))
matplot(t(exp(Ps)),type="l",ylim=c(0,0.05))

plot(exp(Ps[1,]),type="l",ylim=c(0,0.05)) # theta_3 = 0 の片側
plot(exp(Ps[length(thetas[,1]),]),type="l",ylim=c(0,0.05)) # theta_3 = 0 の片側
```



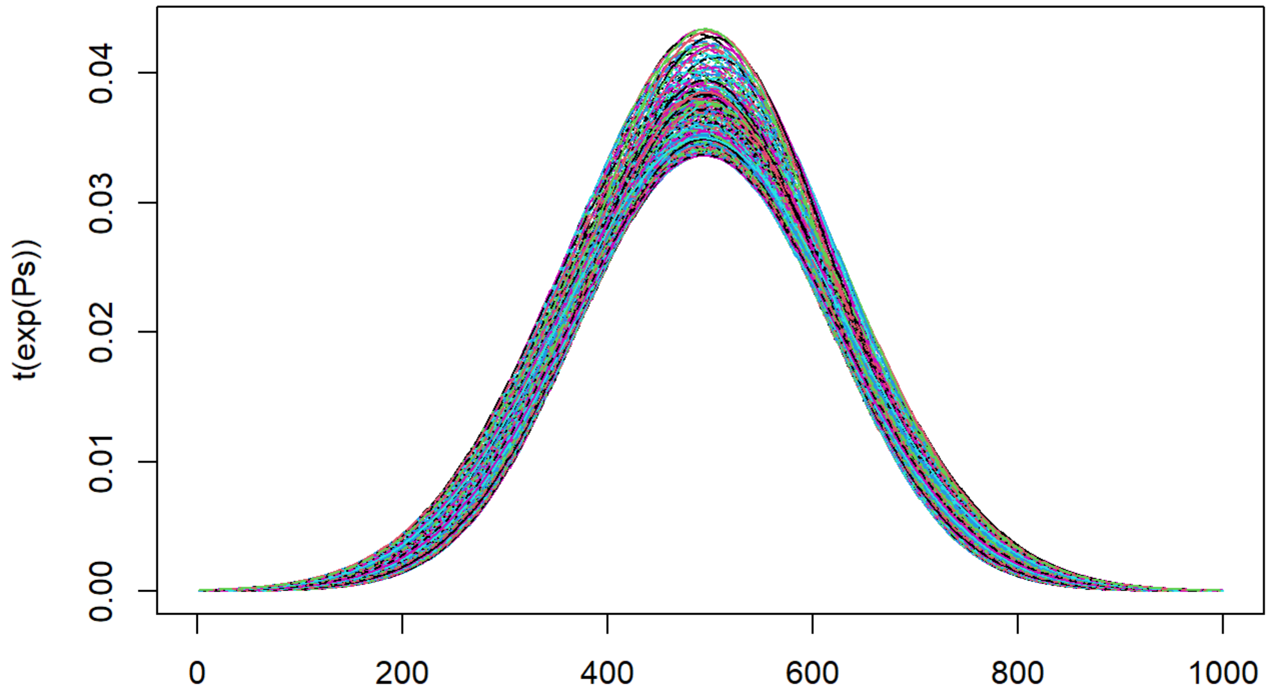

```
par(mfcol=c(1,1))
```

半球面でやってみる。

```
# 双曲空間での、半円弧(測地線・直線)に並ぶ分布たち

t <- seq(from=pi,to=2*pi,length=25)
phi <- seq(from=0,to=2*pi,length=21)
t_phi <- expand.grid(t,phi)

# theta1 theta2 面の円の中心を定める
alpha <- -0.2
beta <- -0.5
# 半径
r <- 0.05

thetas <- cbind(cos(t_phi[,1])*cos(t_phi[,2]) + alpha,cos(t_phi[,1])*sin(t_phi[,2]) + beta,sin
(t_phi[,1])) * r


Ps <- matrix(0,length(thetas[,1]),length(Cx))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- F3 %*% c(thetas[i,]) + Cx
}

matplot(t(Ps),type="l")
```




```
matplot(t(exp(Ps)),type="l")
```



小括

Fx,Cxが関数群を指定し、Θがその具体的な形を定める。

ある特定のタイプの関数を納めた空間が表現されており、それをΘ軸が張っており、そこの内積は双曲幾何的に定義されていることを意味している。

このΘが張った空間全体のうちの部分空間が、確率密度分布になっている。

微分して１になっている部分を取り出せばよい。

Fxを勝手に指定してみよう

Fxが２つの関数であり、片方が実固有値、もう片方が虚固有値とする。

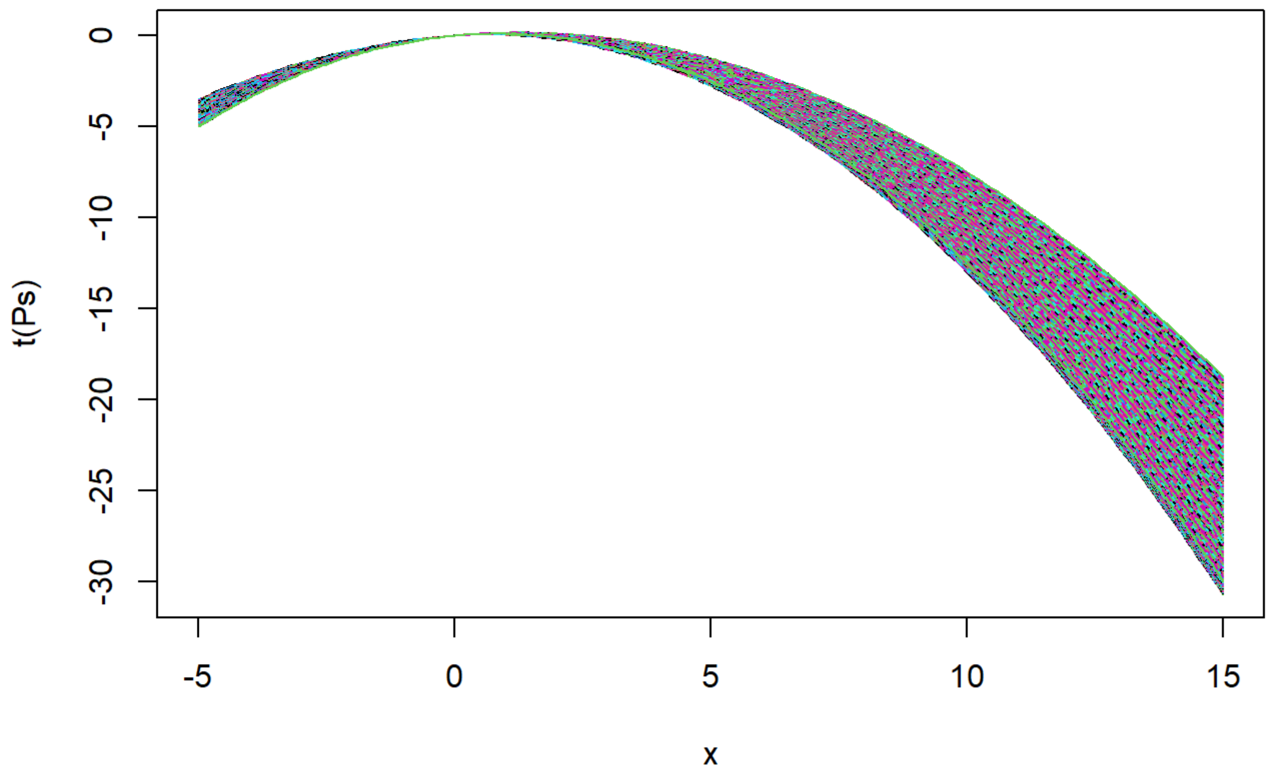
任意にFxを選んでやっても、議論は同じになる。

Fxが自由なことが、ノンパラの意義。

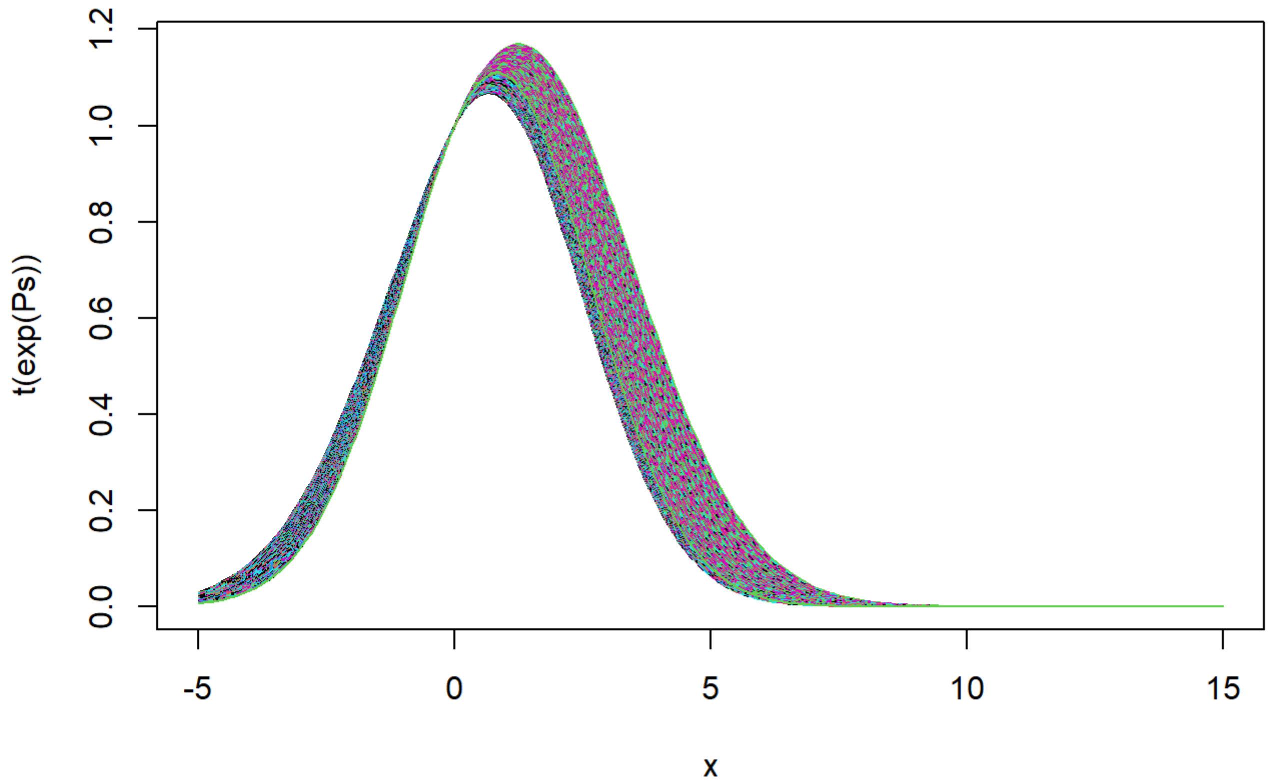
データからcomponent関数がマイニングできるのが嬉しい。

例１

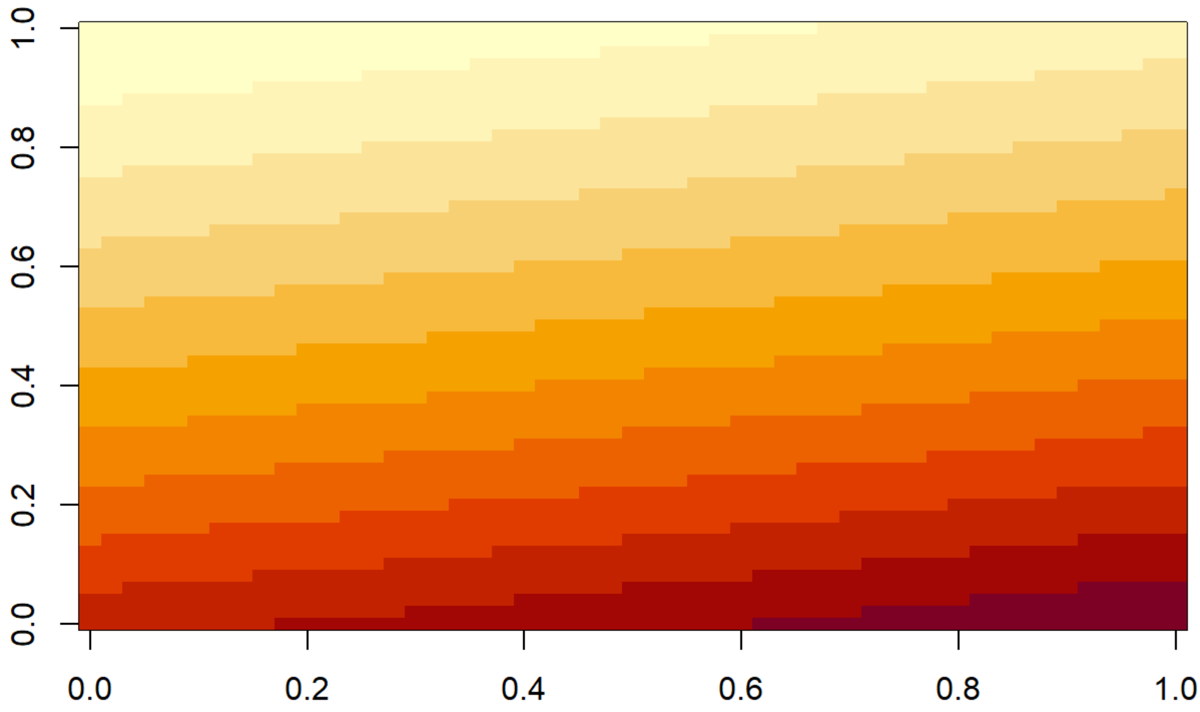
```
F1 <- function(x){
    x # F1 := x ならば、theta2 = 0 のときに指数関数
    #sin(x)
    #ret <- rep(0,length(x))
    #ret[which(abs(x-1) < 1)] <- 1

}
F2 <- function(x){
    -x^2
    #sin(2*x)
}

theta1 <- seq(from=0.20,to=0.25,length=51)
theta2 <- seq(from=0.10,to=0.15,length=51)
thetas <- expand.grid(theta1,theta2)

x <- seq(from=-5,to=15,length=201)

Ps <- matrix(0,length(thetas[,1]),length(x))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- thetas[i,1] * F1(x) + thetas[i,2] * F2(x)
}

matplot(x,t(Ps),type="l")
```



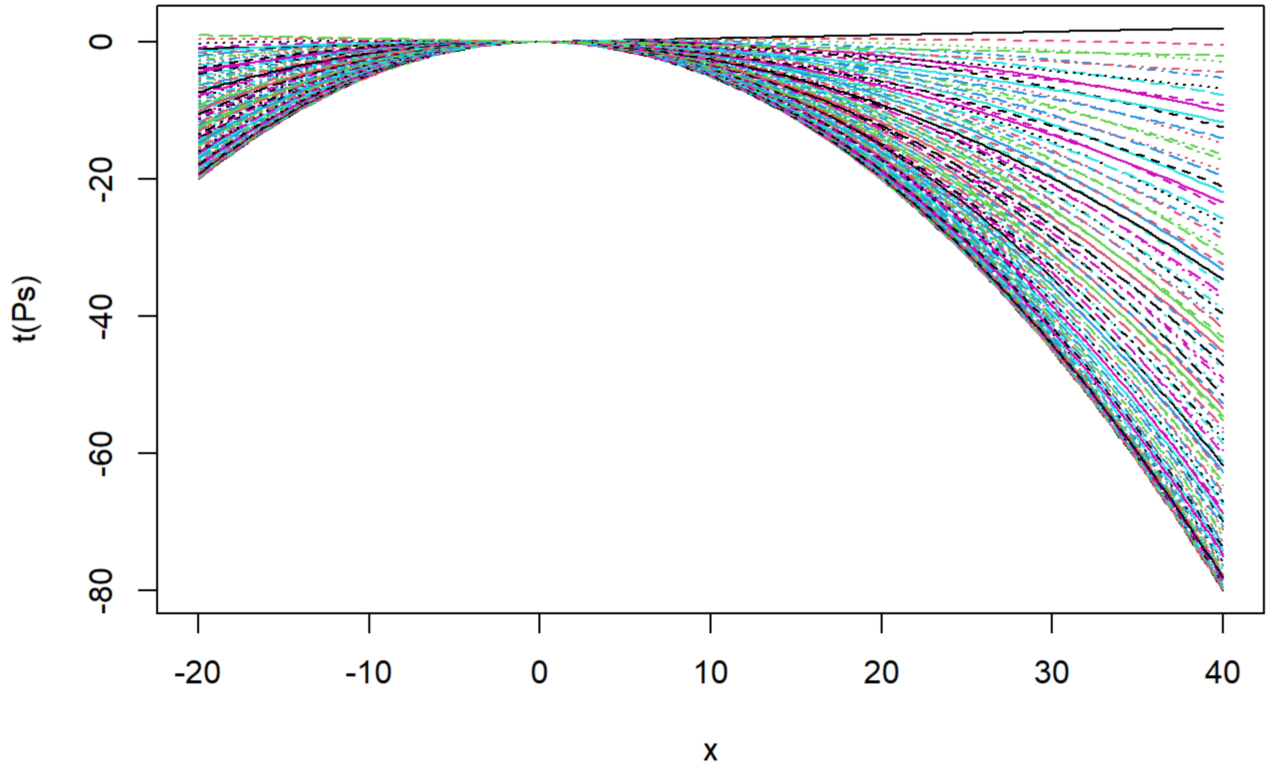

```
matplot(x,t(exp(Ps)),type="l")
```



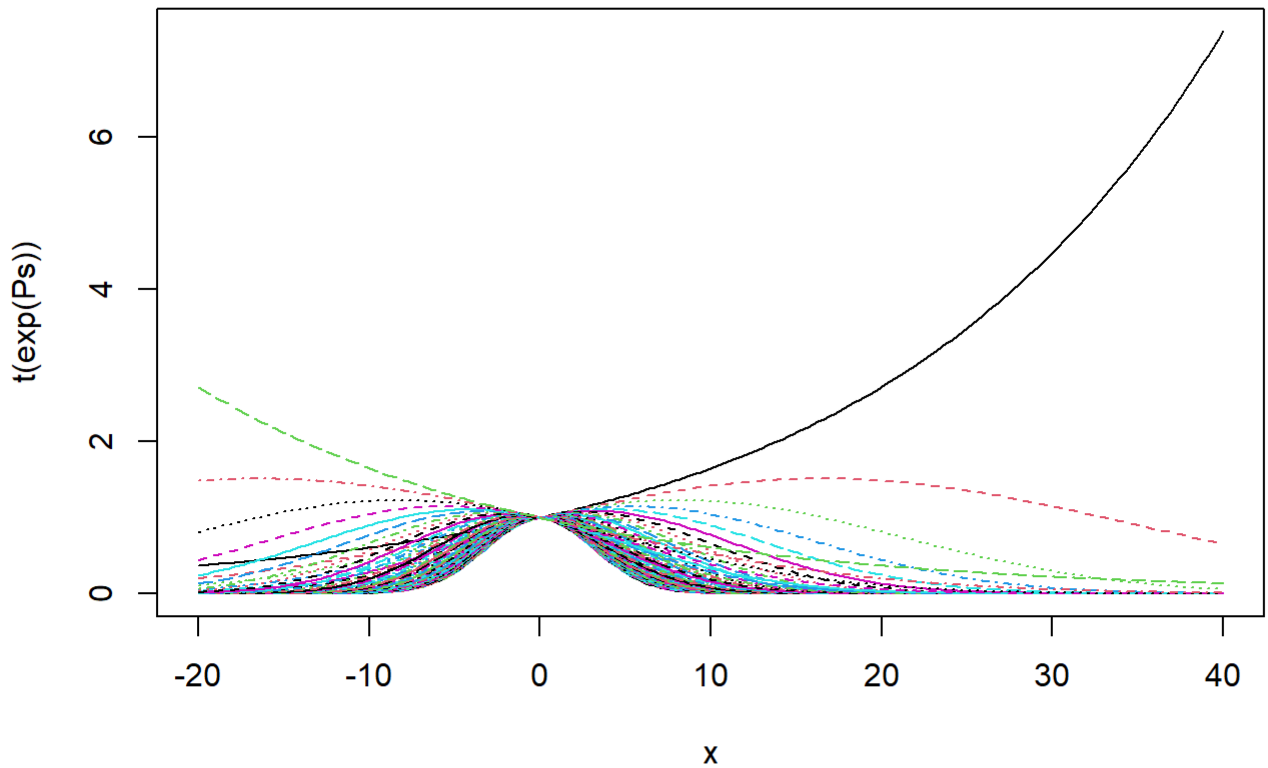
```
Int <- apply(exp(Ps),1,sum) * (x[2]-x[1])

#Int.st <- Int/max(Int)

#plot(thetas,col=gray(1-Int.st),pch=20,xlab="theta1",ylab="theta2")

image(matrix(log(Int),length(theta1),length(theta2)))
```




```
#############

# 双曲空間での、半円弧(測地線・直線)に並ぶ分布たち

t <- seq(from=0,to=pi,length=105)

thetas <- cbind(cos(t),sin(t)) * 0.05

x <- seq(from=-20,to=40,length=201)

Ps <- matrix(0,length(thetas[,1]),length(x))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- thetas[i,1] * F1(x) + thetas[i,2] * F2(x)
}

matplot(x,t(Ps),type="l")
```

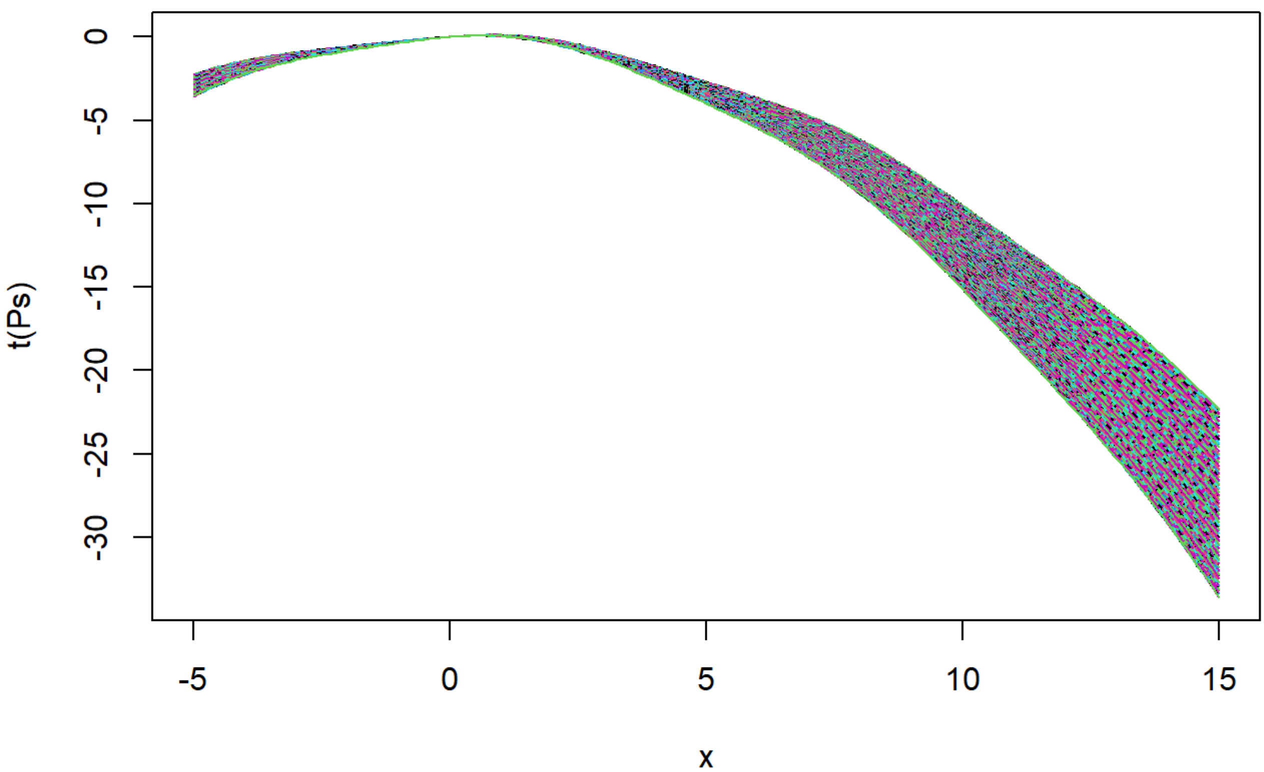



```
matplot(x,t(exp(Ps)),type="l")
```

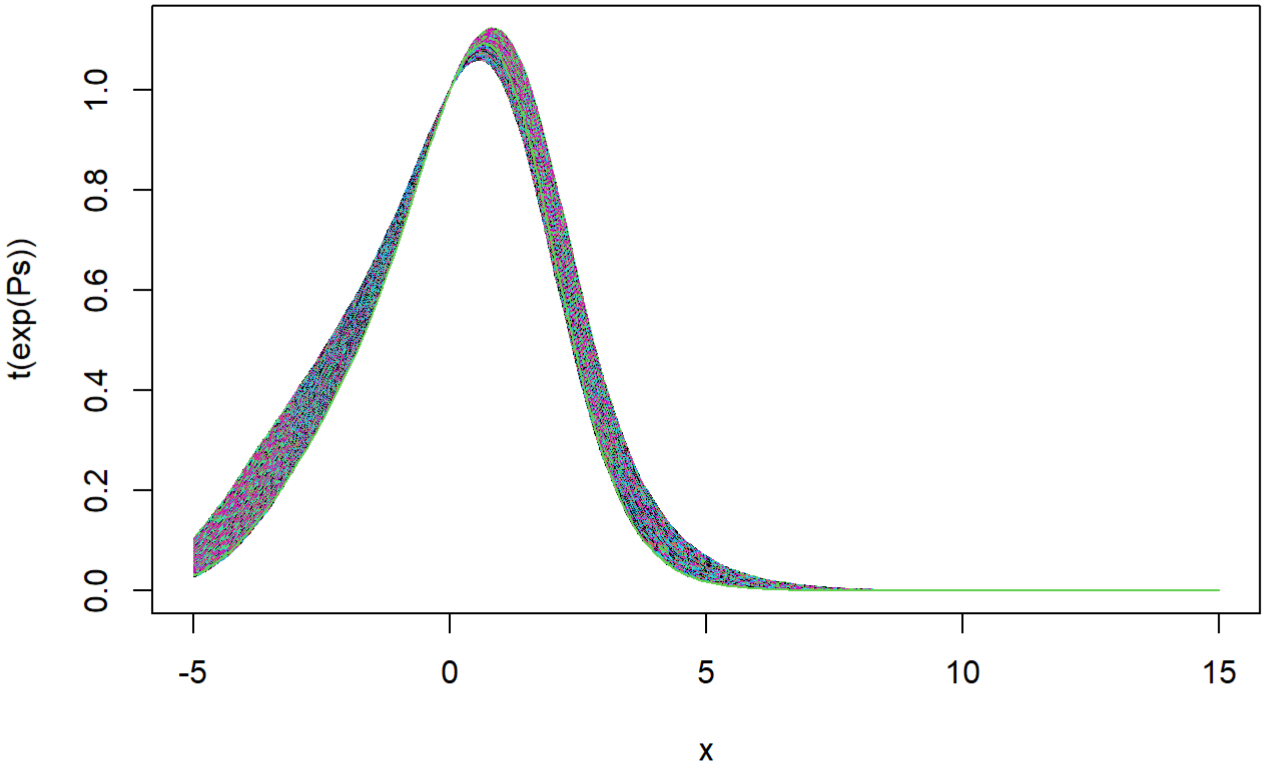


例2

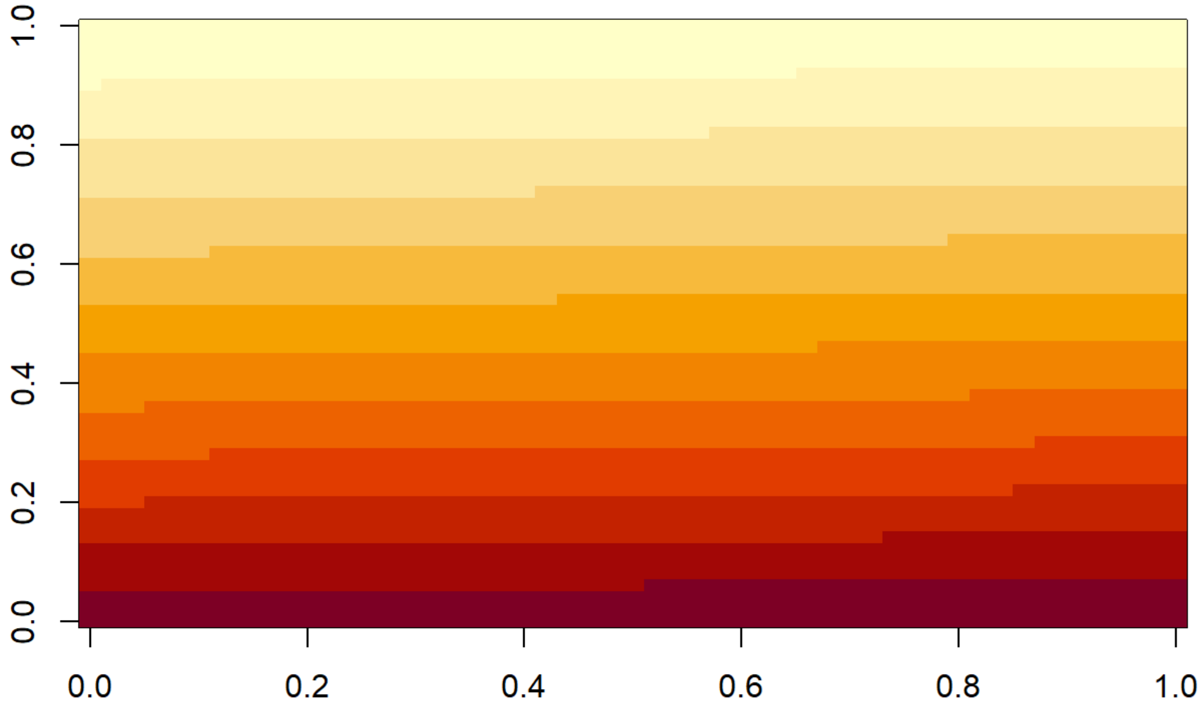
```
F1 <- function(x){
    #x # F1 := x ならば、theta2 = 0 のときに指数関数
    sin(x)
    #ret <- rep(0,length(x))
    #ret[which(abs(x-1) < 1)] <- 1

}
F2 <- function(x){
    -x^2
    #sin(2*x)
}

theta1 <- seq(from=0.20,to=0.25,length=51)
theta2 <- seq(from=0.10,to=0.15,length=51)
thetas <- expand.grid(theta1,theta2)

x <- seq(from=-5,to=15,length=201)

Ps <- matrix(0,length(thetas[,1]),length(x))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- thetas[i,1] * F1(x) + thetas[i,2] * F2(x)
}

matplot(x,t(Ps),type="l")
```



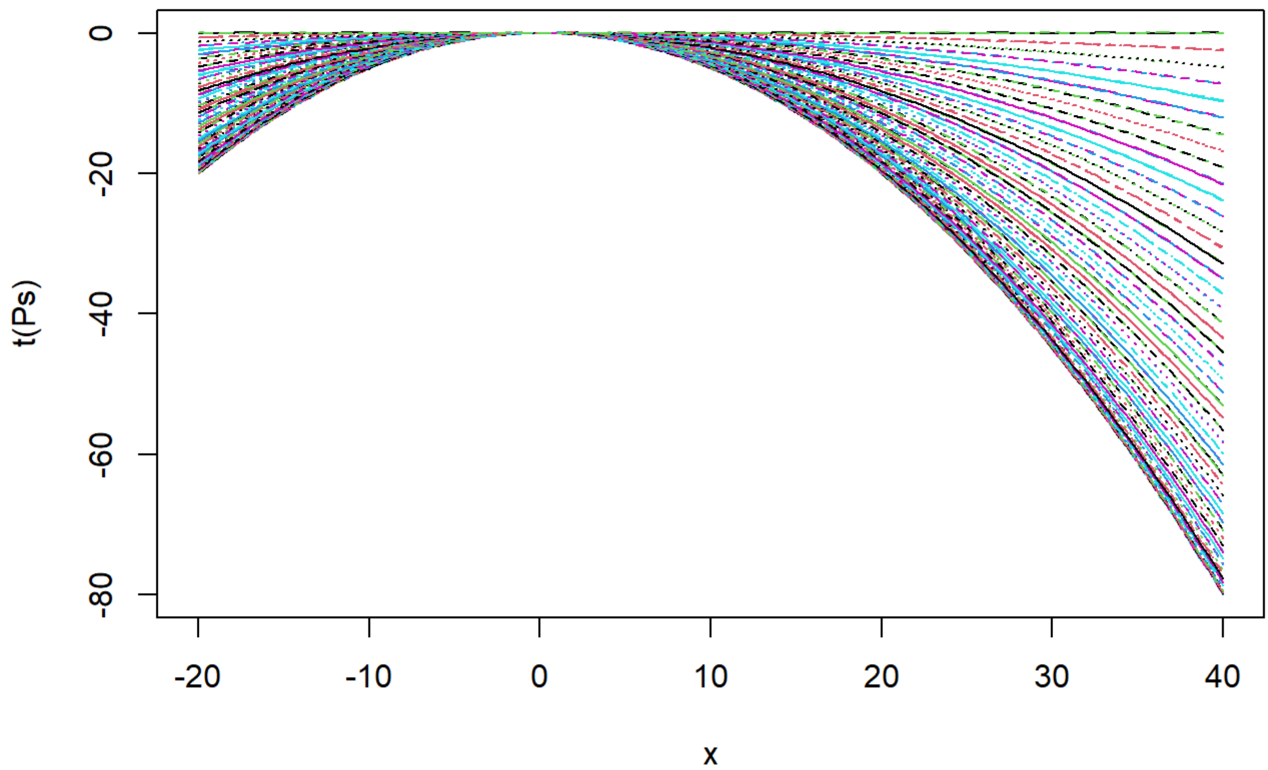

```
matplot(x,t(exp(Ps)),type="l")
```



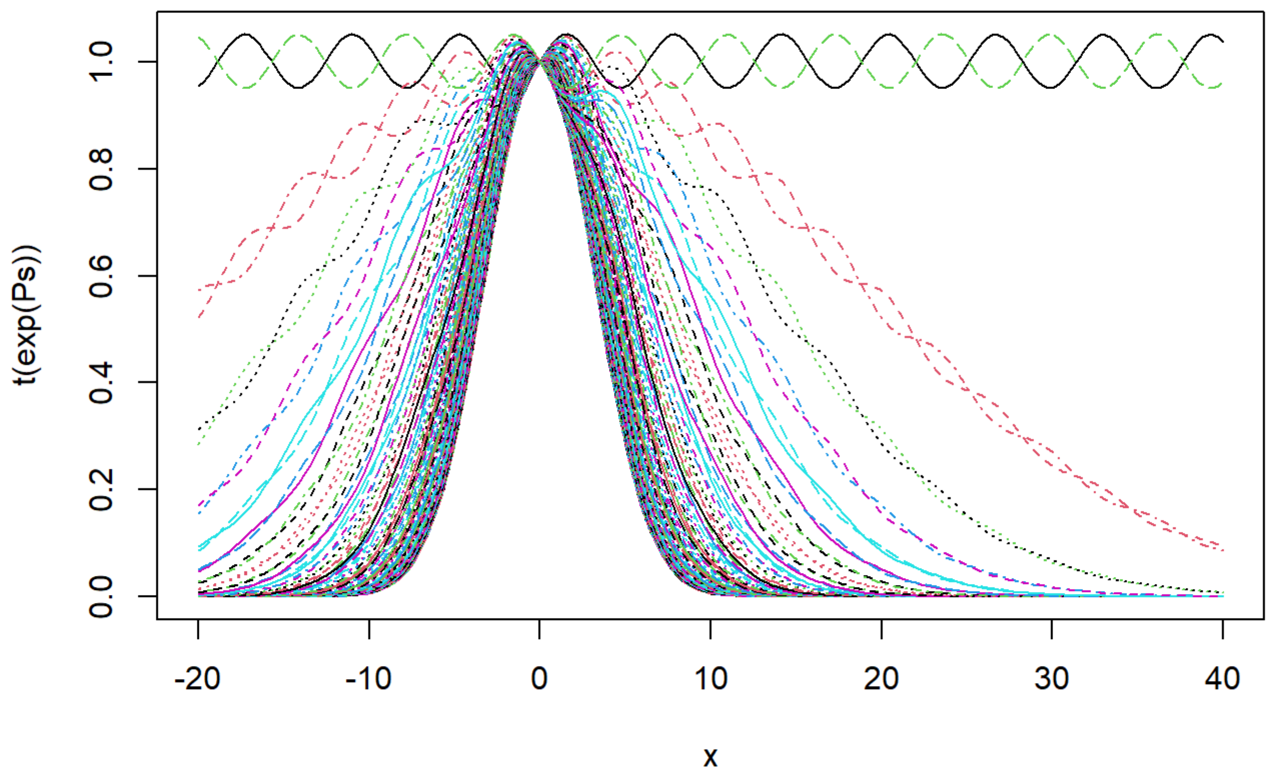

```
Int <- apply(exp(Ps),1,sum) * (x[2]-x[1])

#Int.st <- Int/max(Int)

#plot(thetas,col=gray(1-Int.st),pch=20,xlab="theta1",ylab="theta2")

image(matrix(log(Int),length(theta1),length(theta2)))
```




```
#############

# 双曲空間での、半円弧(測地線・直線)に並ぶ分布たち

t <- seq(from=0,to=pi,length=105)

thetas <- cbind(cos(t),sin(t)) * 0.05

x <- seq(from=-20,to=40,length=201)

Ps <- matrix(0,length(thetas[,1]),length(x))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- thetas[i,1] * F1(x) + thetas[i,2] * F2(x)
}

matplot(x,t(Ps),type="l")
```

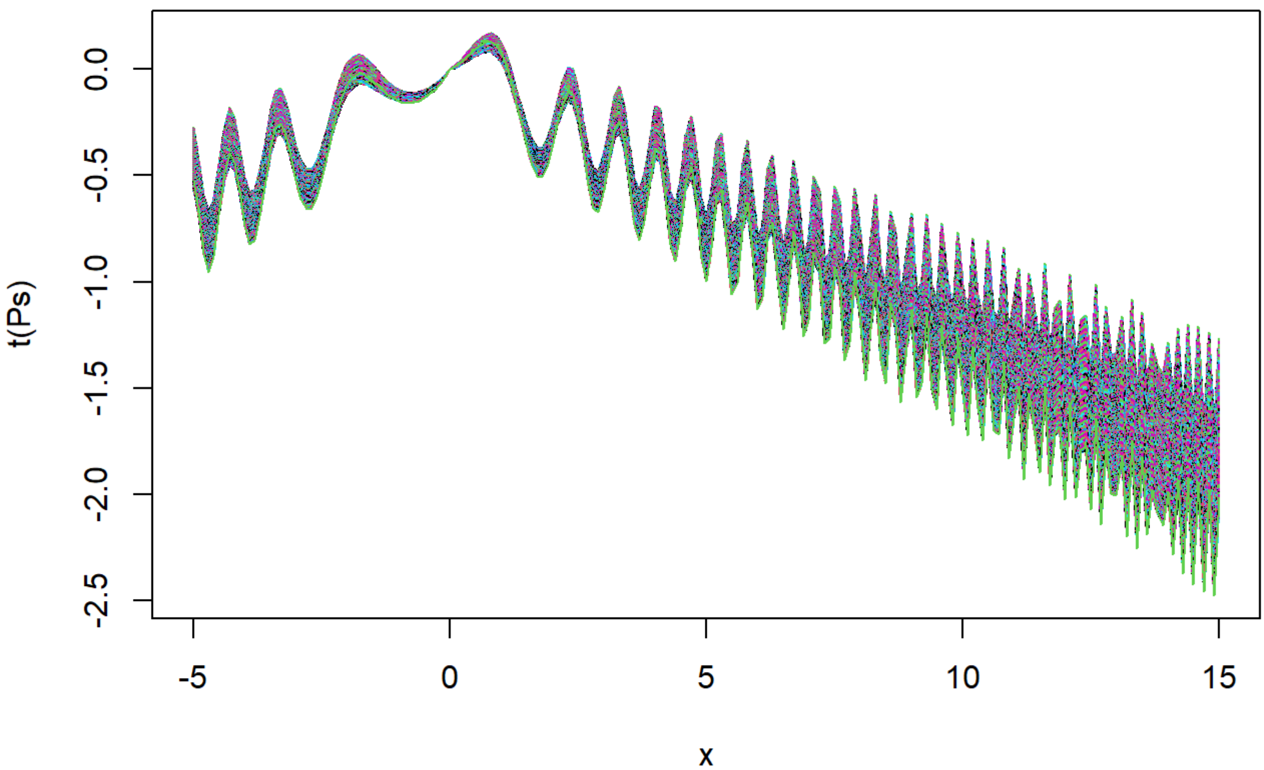



```
matplot(x,t(exp(Ps)),type="l")
```

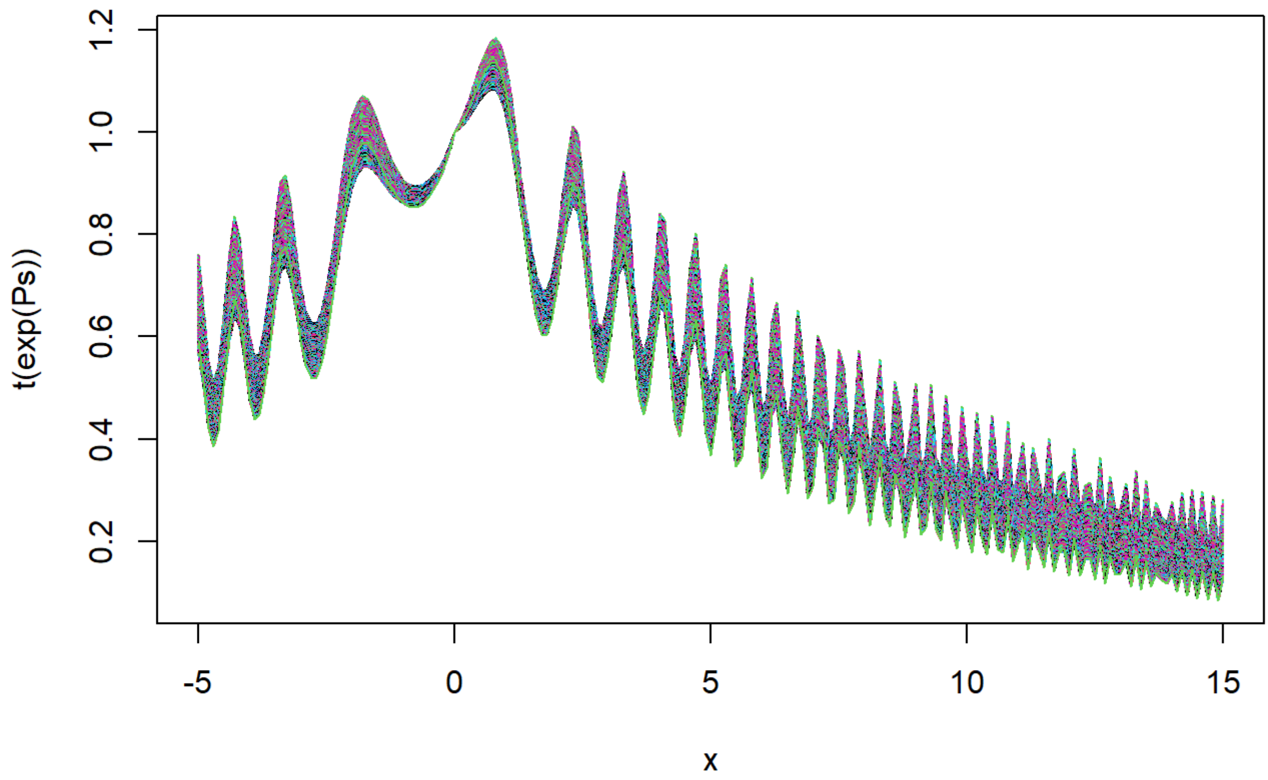


例３

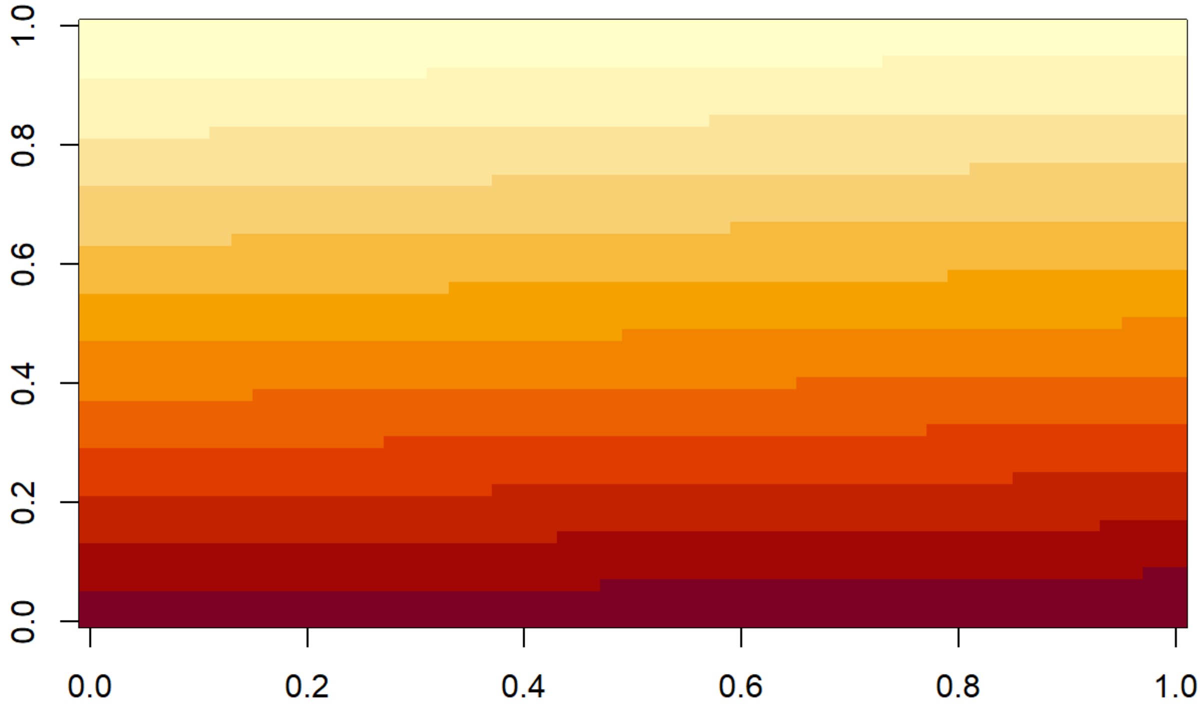
```
F1 <- function(x){
    #x # F1 := x ならば、theta2 = 0 のときに指数関数
    #sin(x)
  sin(x^2+x)
    #ret <- rep(0,length(x))
    #ret[which(abs(x-1) < 1)] <- 1

}
F2 <- function(x){
    #-x^2
    #sin(2*x)
  -abs(x)
}

theta1 <- seq(from=0.20,to=0.25,length=51)
theta2 <- seq(from=0.10,to=0.15,length=51)
thetas <- expand.grid(theta1,theta2)

x <- seq(from=-5,to=15,length=201)

Ps <- matrix(0,length(thetas[,1]),length(x))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- thetas[i,1] * F1(x) + thetas[i,2] * F2(x)
}

matplot(x,t(Ps),type="l")
```



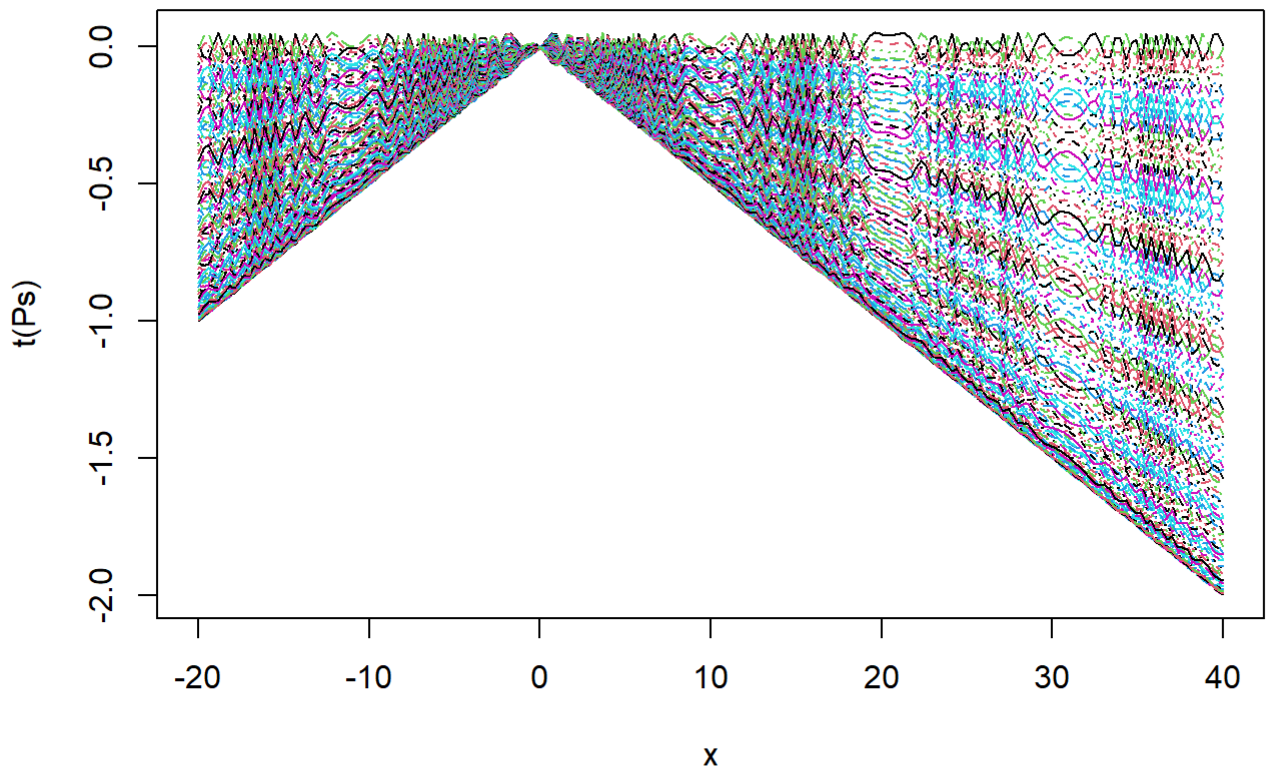
```
matplot(x,t(exp(Ps)),type="l")
```




```
Int <- apply(exp(Ps),1,sum) * (x[2]-x[1])

#Int.st <- Int/max(Int)

#plot(thetas,col=gray(1-Int.st),pch=20,xlab="theta1",ylab="theta2")

image(matrix(log(Int),length(theta1),length(theta2)))
```




```
#############

# 双曲空間での、半円弧(測地線・直線)に並ぶ分布たち

t <- seq(from=0,to=pi,length=105)

thetas <- cbind(cos(t),sin(t)) * 0.05

x <- seq(from=-20,to=40,length=201)

Ps <- matrix(0,length(thetas[,1]),length(x))

for(i in 1:length(Ps[,1])){
    Ps[i,] <- thetas[i,1] * F1(x) + thetas[i,2] * F2(x)
}

matplot(x,t(Ps),type="l")
```




```
matplot(x,t(exp(Ps)),type="l")
```