```
#関数定義,for,if,乱数生成
### 関数定義 ################
tmp <- function(x){
    y <- x^2
    return(y)
}
str(tmp)
tmp(2)
tmp <- function(x) {y <- x^2; return(y)} # 波括弧とセミコロンで複数命令を一行に
tmp(3)
tmp <- function(x) return(x^2)  # 一命令なら波括弧は不要
tmp(4)
tmp <- function(x) x^2          # 最後の命令の値が返り値とされる
tmp(3)
tmp <- function(x,y) x*y        # 多変数関数
tmp(2,3)
tmp <- function(x=1,y) x*y      # 既定値付き変数
tmp(,3)                         # =tmp(1,3)
tmp <- function(x=1,y=2) x*y
tmp()                           # =tmp(1,2)
tmp(y=4)                        # =tmp(1,y)
tmp(y=4,x=3)
x <- 3
tmp <- function(x) {x <- 2*x; return(x)}
tmp(3)
x                               # 関数内部の変数は外部には影響しない
tmp <- function(x) {y <<- 2*x; return(y)} # 永続付値 <<-
tmp(2)
y
tmp <- function(x) {x <<- 2*x; return(x)} # 中身のyをxにしたらどうか？
tmp(2)   # 2
x        # 4, すでに変数が存在すれば変数のスコープは関数の外側になる.
### リスト返り値で複数の値を返す
tmp <- function(x,y) list(x=2,sin.x=sin(x),y=y,log.y=log(y))
z <- tmp(2,3)
str(z)
z$sin.x  # z[[2]] でも可
z$log.y  # z[[4]] でも可
##練習問題1を解きましょう
#
### for文
for(x in 1:3) cat("x=",x,"x^2=",x^2,fill=TRUE)
# Rではループは遅くなりがち
x <- NULL; system.time(for(i in seq(10^5)) x <- c(x,sin(i)))
str(x)
# 最初に入れ物を用意しておくと早くなる ※重要
x <- numeric(10^5); system.time(for(i in seq(10^5)) x[i] <- sin(i))
# ベクトル化演算は早い ※重要
system.time(x <- sin(seq(10^5)))
# apply関数族による繰り返しの隠蔽
system.time(x <- sapply(1:10^5, FUN=function(i) sin(i)))          # 速度は向上し
ない
# 括弧で閉じてfor文の中身を複数行化
z <- numeric(3)
for(x in 1:3){
    y <- x^2
    z[x] <- y
}
z
#
### 条件文
tmp <- function(x) if(x < 0) -x else x   # 絶対値
tmp(2)
tmp(-2)
tmp(c(-2,2))                             # ベクトル化されていない
tmp <- function(x) ifelse(x < 0, -x, x)  # ベクトル化if文
x <- seq(-2,2); tmp(seq(-2,2))
tmp <- function(x) if(x < 0) sin else cos
tmp(-2)                                  # sin
tmp(-2)(pi/3)                            # = sin(pi/3)
#{}を使って複数命令を実行
tmp <- function(x){
```

```
    if(x < 0){
        y <- x^2
        z <- x^3
    }else{
        y <- x^3
        z <- x^2
    }
    return(c(y,z))
}
tmp(2)
tmp(-2)
##練習問題2を解きましょう
#
###乱数生成
# r+(乱数名)で乱数の生成
runif(10)  # 一様分布から１０個のサンプルを生成
rnorm(10)  # 標準分布から１０個のサンプルを生成
rnorm(10,2,4)  # 平均2標準偏差4の正規分布から10個のサンプルを生成
# ヒストグラムの表示
hist(rnorm(100)) #標準正規分布100サンプル
hist(rnorm(100),breaks=20) #ビンを細かく
hist(rnorm(1000),breaks=30) #サンプルを増やす
hist(rnorm(10000),breaks=40) #サンプルをさらに増やす. 安定してきた.
# 和, 平均, 分散, 標準偏差
sum(runif(10))    #和
mean(runif(10))   #平均
var(runif(10))    #分散
sd(runif(10))     #標準偏差
#サンプル数を増やしてみる
mean(runif(10^4))  # 理論値 0.5
var(runif(10^4))   # 理論値 1/12=0.08333...
#サンプル数をさらに増やしてみる
mean(runif(10^7))  # 理論値 0.5
var(runif(10^6))   # 理論値 1/12=0.08333...
#収束の様子を図示してみよう
x <- numeric(7); for(i in 1:7) x[i]<-mean(runif(10^i))
plot(x,type='l')
#何度か実行して重ね書き
par(new=F)
for(j in 1:5){
 for(i in 1:7) x[i]<-c(mean(runif(10^i)));
 plot(x,type='l',col=j,ylim=c(0.45,0.55));  #ylimでy軸の範囲を指定
 par(new=T);       #重ね書きモードにする
}
par(new=F)
#ちゃんと収束していることがわかる(大数の法則)
##
#こちらが指定した集合からのサンプリング
sample(seq(1,5),10,rep=TRUE)            # 離散乱数(復元抽出)
sample(seq(1,10),10)                    # 非復元抽出(ランダム置換)
table(sample(seq(1,10),100,rep=TRUE))  # 集計
#乱数種の設定
set.seed(1234);runif(2)                 # 乱数種を与える
set.seed(1234);runif(2)
set.seed(4321);runif(2)
#分布の形
# d+(関数名)で確率密度関数
pd <- dnorm(seq(-4,4,by=0.1))
pd
plot(seq(-4,4,by=0.1),pd,type='l')  #表示
curve(dnorm,-4,4)                   #これでも良い. [-4,4]の範囲
curve(dgamma(x,3,1),0,10)      #ガンマ分布の確率密度関数, 形状母数3, 尺度母数1
curve(dunif(x),-1,2)           #一様分布の確率密度関数
#ヒストグラムと重ね書き
z <- rgamma(500,2,1)           #ガンマ分布から500サンプル生成
plot(sort(z),dgamma(sort(z),2,1),type='l') #密度関数のプロット
hist(z, freq = FALSE,add=T,breaks=20)  #ヒストグラムを上書き
#累積分布関数
x <- seq(-5,5,by=0.1)
plot(x,pnorm(x),type='l')
```