◆『Wilson の波浪予報公式』を題材にした R 入門　◆ 北野 利一 ◆ ２００５年１０月 ４日

　波浪とは，風波とも呼ばれ，海上の気象に応じて，日々変化するものである．波浪の予測は，第２次世界大戦中には，最新技術であった．現在，波浪予測は，沿岸構造物および海洋構造物の設計および施工には不可欠であり，波浪情報は多くのマリンレジャーにも活用されており，天気予報と同様に人々の暮らしになじみ深いものになっている．

　発生する波浪は，風の強さと範囲により，その波高などの波浪特性が決まる．Sverdrup と Munk (1947) は，風から海面波へのエネルギーの供給と，これにより生じる波高と波速の関係をはじめて明らかにした．Wilson (1965) は，その関係を以下のように整理した．

```
g*H/U^2      = 0.03 * (1 - 1/(1 + 0.004*(g*F/U^2)^(1/2))^2)           ... (1a)

g*T/(2*pi*U) = 1.37 * (1 - 1/(1 + 0.008*(g*F/U^2)^(1/3))^5)           ... (1b)
```

ここで，U は風速，F は吹送距離，g は重力加速度であり，H および T は，波浪の有意義の波高と周期である（多くの場合は，H1/3 および T1/3 と記す）．ただし，上式は吹送時間が十分に長い場合（すなわち，波浪の発達が飽和状態に達した場合）にのみ適用できる．吹送時間が，最小吹送時間 tmin に達しない時には，与えられた吹送時間 t に応じた最小吹送距離 Fmin を吹送距離 F として用いる．

　最小吹送時間 tmin は，発達しながら進行する波群の伝播距離として，以下のように定義される．

```
tmin = integrate Cg^(-1) from 0 to F                                  ... (2)
```

波群の進行速度 Cg は，波の群速度であり，深海波の群速度 Cg は，波速 C0 を用いて，

```
Cg = C0/2 ;  C0 = g*T/(2*pi)                                          ... (3)
```

と表せることから，無次元化された吹送距離 g*F/U^2 を与えれば，上式により群速度を算出できることがわかる．式(2)において，最小吹送時間 tmin を無次元化すれば，

```
g*tmin/U = integrate a half of left hand side of (1b) from 0 to g*F/U^2    ... (4)
```

となる．また，最小吹送距離 Fmin は，与えられた吹送時間 t が得られるように，次式を満足するように決めた吹送距離である．

```
g*t/U = integrate a half of left hand side of (1b) from 0 to g*Fmin/U^2    ... (5)
```

以上から，最小吹送時間だけでなく，最小吹送距離によっても，波浪の発達が飽和状態に達しているかどうかを判断できることがわかる．

風速 U，吹送距離 F，吹送時間 t の３つの数字が与えられた時，発生する波浪の波高 H と周期 T は，以下の手順に従って算出することができる．

　１）吹送時間 t から最小吹送距離 Fmin を求める（この際に，風速 U は必要となる）．
　２）最小吹送距離 Fmin と，与えられた吹送距離 F の大きさを比較し，小さい方を吹送距離 F' とする．
　３）吹送距離 F' と風速 U を式(1a，b)に与えることにより，波高 H と周期 T を算出する．

なお，ここで注意すべきは，式(1)のように，吹送距離により波高と周期を算出する公式を用いる場合には，最小吹送時間 tmin を求める必要はないことである．ただし，吹送距離が現実的には明確でないことにも注意が必要である．

　さて，以上の事柄を R により，具体的に計算してみる．

【1】Wilson の公式は，式(1a) および (1b) をほぼそのまま入力すればよい．関数（function）として扱う点が異なる．

```
## wilson's formulas
wilson.h <-
function(fetch) 0.30 * (1 - 1/(1 + 0.004 * sqrt(fetch))^2)
#
wilson.p <-
function(fetch) 1.37 * (1 - 1/(1 + 0.008 * fetch^(1/3))^5)
```

【2】Wilson の公式を図示する（合田テキストの図 3.18 に相当）

```
# Fig. 3.18 (in part)
Fetch <- c(rep(c(10, 100, 1000, 10000), rep(6, 4)) * c(1,2,3,4,6,8), 100000)
plot(log10(Fetch), log10(wilson.h(Fetch)), axes=F, ylim=c(-2, 0.25), ylab="", type="l")
axis(1, 1:5, paste("10^", 1:5, sep=""))
axis(2, c(-2:0, log10(1.5)), c(0.01,0.1,1,1.5))
lines(log10(Fetch), log10(wilson.p(Fetch)), lty=2)
abline(h=-2:0, lty=3)
legend(4.2,-1.4, c("H1/3","T1/3"), lty=c(1,2))
```

【3】公式を次元量に直すために，次式を用意する．

```
## formulas for dimensional quantities (1) note
wilson.height.m <-
function(F.km, U.ms) wilson.h(9.8*1000*F.km/U.ms^2
                                          )*U.ms^2/9.8
#
wilson.period.s <-
function(F.km, U.ms) wilson.p(9.8*1000*F.km/U.ms^2
                                          )*U.ms/9.8*2*pi
```

【4】再び Wilson の公式を図示する（合田論文の図 1 および 2 に相当）

```
## Fig. 1 (Goda, 2003)
Fetch <- c(rep(c(1, 10, 100, 1000), rep(6, 4)) * c(1,2,3,4,6,8), 10000)
plot(log10(Fetch), log10(wilson.height.m(Fetch, 20)), ylim=c(-1,1),
     xlab="Fetch (km)", ylab="Wave Height (m)", axes=F, type="l")
axis(1,  0:4, paste("10^",  0:4, sep=""))
axis(2, -1:1, c(0.1, 1, 10))
lines(log10(Fetch), log10(wilson.height.m(Fetch, 30)), lty=2)
lines(log10(Fetch), log10(wilson.height.m(Fetch, 10)), lty=3)
lines(log10(Fetch), log10(wilson.height.m(Fetch,  5)), lty=4)
lines(log10(Fetch), log10(wilson.height.m(Fetch, 15)), lty=5)
legend(3,-.3, c("U=30","20","15","10","5"), lty=c(2,1,5,3,4))
#
## Fig. 2 (similar but different appearance)
Fetch <- c(rep(c(1, 10, 100, 1000), rep(6, 4)) * c(1,2,3,4,6,8), 10000)
plot(log10(Fetch), wilson.period.s(Fetch, 20), ylim=c(0,16),
     xlab="Fetch (km)", ylab="Wave Height (m)", axes=F, type="l")
axis(1,  0:4, paste("10^",  0:4, sep=""))
axis(2, 2*(0:8))
lines(log10(Fetch), wilson.period.s(Fetch, 30), lty=2)
lines(log10(Fetch), wilson.period.s(Fetch, 10), lty=3)
lines(log10(Fetch), wilson.period.s(Fetch,  5), lty=4)
lines(log10(Fetch), wilson.period.s(Fetch, 15), lty=5)
legend(0,16, c("U=30","20","15","10","5"), lty=c(2,1,5,3,4))
```

【4】“有名な”風波の予知曲線も描いておく（合田テキストの図 3.19 に相当）

```
#  options(width=150)
F <- c(rep(c(1, 10, 100, 1000), rep(6, 4)) * c(1,2,3,4,6,8), 10000)
U <- c(5,6,7,8,9,10,12,14,16,18,20,22,24,26,28,30,35,40,50,60)
outer(F,U, wilson.height.m) -> H
rownames(H) <- paste(F, "km")
colnames(H) <- paste(U, "m/s")
round(H, 2)
#
outer(F,U, wilson.period.s) -> T
rownames(T) <- paste(F, "km")
colnames(T) <- paste(U, "m/s")
round(T, 2)
#
# Fig. 3.19
contour(log(F)/log(10), U, H,
        levels=c(.25,.5,.75,1,1.5,2,2.5,3,3.5,4:10,12,14,16,18,20,30,35,40,50),
        xlab="Fetch (10^# km)")
axis(2, 5*(1 + 2*0:5), rep("",6)) # axis(4, U)
axis(3, log10(F), F)
#
contour(log(F)/log(10), U, T,
        levels=c(.25,.5,.75,1,1.5,2,2.5,3,3.5,4:10,12,14,16,18,20,30,35,40,50),
        xlab="Fetch (10^# km)", add=TRUE, lty=3)
#
# Fig. 3.19 (bis)
persp(log10(F), U, log2(H), theta=-30)
persp(log10(F), U,       T, theta=-30)
```

【5】最小吹送時間を算出する式は，次のように表せばよい．

```
## minimum duration:
wilson.tminimum <-
function(fetch) {
          f <- function(fet) 2/wilson.p(fet)
integrate(f, 0, fetch)$value}
#
# ex.
wilson.tminimum(1)
wilson.tminimum(2)
wilson.tminimum(5)
```

【6】最小吹送時間を吹送距離で無次元化するという考え方もある．

```
wilson.tmin.proto <-
function(fetch, dimensionless=c("g*tmin/U","tmin*U/F")) {
                dimensionless <- match.arg(dimensionless)
      if(length(fetch) != 1) stop("fetch is not vectorized") # !!
          f <- function(fet) 2/wilson.p(fet)
                                 res <- integrate(f, 0, fetch)$value
if (dimensionless == "tmin*U/F") res <- res/fetch
res}
```

なお，上で定義した関数 wilson.tmin は，引数にベクトル量をとることはできない（wilson.h, wilson.p, wilson.height.m, wilson.period.s は，ベクトル量を受け付けることと対比し，エラーメッセージを出すように工夫している点を見よ！）

【7】最小吹送時間を図示する.

```
Fetch <- c(rep(c(10, 100, 1000, 10000), rep(6, 4)) * c(1,2,3,4,6,8), 100000)
tmin <- numeric(0)
for (fe in Fetch) tmin <- c(tmin, wilson.tmin.proto(fe, "tmin*U/F"))
plot(log10(Fetch), tmin, type="l", ylim=c(1,30), ylab="tmin*U/F")
```

上のコードから得られる図は，合田テキストの図 3.18 に相当するものである．吹送時間も対数をとり，以下のコードを用いれば，合田論文の図 3 に相当するものを示すことができる．

```
plot(log10(Fetch), log10(tmin), ylim=c(0,2), axes=FALSE)
axis(1, 1:5, paste("10^", 1:5, sep=""))
axis(2, 0:2, c(1,10,100))
abline(log10(43), -0.27, col="red") # by Goda(2002)
```

【8】最小吹送時間と最小吹送距離の関係は，以下のように近似できる．

```
goda.Fmin <-
function(duration) (duration/43)^(1/0.73)
#
goda.tmin <-
function(fetch) 43*fetch^0.73
#
goda.Fmin.km <-
function(U.ms, t.h)  round(1.0 * t.h^1.37 * U.ms^0.63)
#
goda.tmin.h <-
function(F.km, U.ms) round(1.0 * F.km^0.73 * U.ms^-.46, 1)
```

【9】近似関係により，次のような関数を作ることができる．

```
wilson.g <-
function(F.km, U.ms, t.h) {
        F.given <- F.km
        F.min.km <- goda.Fmin.km(U.ms, t.h)
        F.km <- F.km * (F.min.km/F.min.km)
  if(sum(F.min.km < F.km -> flag) > 0) F.km[flag] <- F.min.km[flag]
#
data.frame(round(wilson.height.m(F.km, U.ms), 2),
          round(wilson.period.s(F.km, U.ms), 2), U.ms,
          paste("(t_min:",
#                         char.white
                                (as.character(goda.tmin.h(F.given, U.ms))), ifelse(flag, ">)", "<)"),
#                         char.white
                                (as.character(        t.h))),
          paste("(F_min:",
#                         char.white
                                (as.character(      round(F.min.km))),      ifelse(flag, "<)", ">)"),
#                         char.white
                                (as.character(            F.given)))) -> res
names(res) <- c("H1/3 (m)", "T1/3 (s)", "U (m/s)", "duration (h)", "fetch (km)")
      res}
```

ここで，例題を解いてみよう．

```
wilson.g(F=120, U=20, t=10) # ex.1
wilson.g(F=350, U=25, t=15) # ex.2
```

こんなこともできる.

```
wilson.g(F=350, U=25, t=15:20)
wilson.g(F=120, U=10:20, t=10)
```

ただし，上の２つめの出力は美しくない（少し不細工である）.
改善のために，以下のような関数を定義した上で，wilson.g におけるコメント # を外して再度実行せよ.

```
char.white <- function(chars) { nch <- nchar(chars)
               max.nchar <- max(nch)
               min.nchar <- min(nch)
if (max.nchar > min.nchar) {white <- list()
     flag <- max.nchar - nch
                                     white[[1]] <- " "
         if (max.nchar > min.nchar +1) {
for (j in 2:(max.nchar - min.nchar)) white[[j]] <- paste(white[[j-1]], " ", sep="")}
for (j in 1:(max.nchar - min.nchar)) {
  if (any(flag == j)) chars[flag == j] <- paste(white[[j]], chars[flag == j], sep="")}}
                      chars}
```

【10】式(5)をキチンと満足する最小吹送距離を求めよう.
そのためには，Newton 法を用いるべきであり，初期値が必要となる．合田の近似的な関係式により得られる値を初期値として用いる.

```
wilson.Fminimum <-
function(duration) {        res <- goda.Fmin(duration)
while (abs((wilson.tminimum(res) - duration)*wilson.p(res)/2 -> eps) > 10^-5)
                     res <- res - eps
                     res}
```

なお，wilson.tminimum の引数には，ベクトル量を受け付けないので，wilson.Fminimum にも，ベクトル量は代入できないことに注意する.

```
wilson.Fminimum(wilson.tminimum(1))
wilson.Fminimum(wilson.tminimum(2))
wilson.Fminimum(wilson.tminimum(5))
```

【11】Wilson 公式を正確に満足する wilson.g に相当するものは，以下のように作成できる.

```
wilson.proto <-
function(F.km, U.ms, t.h) {
# if(all(!unlist(lapply(list(F.km, U.ms, t.h), length)) == 1)) stop("not vectorized") # wilson.Fmin
                                 flag <- FALSE
                 F.given <- F.km
                 F.min.km <- wilson.Fminimum(9.8*3600*t.h/U.ms)*U.ms^2/9.8/1000
     if (F.km  > F.min.km) {     flag <- TRUE
         F.km <- F.min.km  }
#
data.frame(round(wilson.height.m(F.km, U.ms), 2),
           round(wilson.period.s(F.km, U.ms), 2), U.ms,
           paste("(t_min:",
           round(wilson.tminimum(9.8*(1000*F.given)/U.ms^2
                                                  )*U.ms/9.8/3600, 1), ifelse(flag, ">)", "<)"),
                       t.h),
           paste("(F_min:",
           round(  F.min.km),                                        ifelse(flag, "<)", ">)"),
                  F.given)) -> res
```

```
names(res) <- c("H1/3 (m)", "T1/3 (s)", "U (m/s)", "duration (h)", "fetch (km)")
row.names(res) <- " "
res}
```

この時，例題を以下のように解くことができる.

```
wilson.proto(F=120, U=20, t=10) # ex.1
wilson.proto(F=350, U=25, t=15) # ex.2
```

ただし，wilson.proto に含まれる関数 wilson.Fminimum および wilson.tminimum は，いずれも，引数にベクトル量を受けつけることができない. このままでは，wilson.g(F=350, U=25, t=15:20) というような計算はできないことに注意である.

付属のスクリプトファイルを読み込むことにより，ベクトル量を引数にとる計算は可能となる. 以下で用いる wilson は，wilson.proto を改良したものである.

```
source("MacOSX3:Users:tk:rm171:kitano:wilson.R") # change your path

wilson(F=350, U=25, t=15:20)
wilson(F=120, U=10:20, t=10)
```

以上の議論は，R を用いた計算方法を示すことに主眼があり，Wilson 公式を厳密に解くことに主眼を特に置くものではない. つまり，Wilson 公式は経験式であり，Wilson公式そのものが，観測結果を近似的に示すに過ぎないものであるため，
合田(2002) が提示する Wilson 公式の近似式に問題は少ない.

◆ 参考文献

合田良実(1998): 海岸・港湾（二訂版），彰国社，321p. （特に，p.67-74 を見よ）
合田良実(2002): Wilson 推算式による波浪の簡易計算について，ECOH/YG 技術論文，No.1, 3p. http://www.ecoh.co.jp/donnakaisya/wilsonsprediction.pdf
Goda, Y.(2003): Reevisiting Wilson's formulas for simplified wind-wave prediction, J. WPCOE, ASCE, Vol.129, pp.93-95
Sverdrup and Munk (1947): Wind sea and swell. theory of relation for forecasting, U. S. Navy Hydrogr. Office, Washington, No.601, 44pp.
Wilson, B. W. (1965): Numerical prediction of ocean waves in the North Atlantic for December, 1959., Deutsche Hydrographische Z., Vol.18, pp.114-130.