```
############################################################
#
# The HSS Advising Survey Data (with the fake
# "yes, advising is OK" variable)
#

data <- data.frame(scan(what=list("",0,0,0,0,0,0),sep="&"))
Economics&    28 & 40 & 0.132 & 126 & 0.128 & 0.97
English&      23 & 39 & 0.128 & 115 & 0.117 & 0.91
History&      10 & 21 & 0.069 &  48 & 0.049 & 0.70
ModLang&       3 &  8 & 0.026 &  16 & 0.016 & 0.62
Philosophy&    1 &  4 & 0.013 &   7 & 0.007 & 0.54
Psychology&   11 & 37 & 0.122 & 104 & 0.105 & 0.87
SDS&          22 & 54 & 0.178 & 161 & 0.163 & 0.92
Statistics&    3 &  6 & 0.020 &   8 & 0.008 & 0.41
Interdisc/IS& 46 & 76 & 0.250 & 233 & 0.236 & 0.95
Undeclared&   13 & 19 & 0.062 & 168 & 0.170 & 2.73

names(data) <- c("Dept","Yesses","S.Tot","S.Prop","P.Tot","P.Prop","Weights")

data

############################################################
#
# Convert yes/no data that has been summarized within strata
# into a tall array, each row of which corresponds to a single
# survey respondent.
#

get.ys <- function(yesses,totals,weights) {
  #
  # Convert a set of poststratum weights, counts of yesses
  # and total counts to individual observations with
  # corresponding weights

  nos <- totals-yesses

  individs <- NULL
  for (h in 1:length(weights)) {
    individs <- rbind(individs,
                      cbind(rep(1,yesses[h]),rep(weights[h],yesses[h])))
    individs <- rbind(individs,
                      cbind(rep(0,nos[h]),rep(weights[h],nos[h])))
  }
  return(individs)
}



tall.data <- get.ys(data$Yesses,data$S.Tot,data$Weights)
  # this is only needed for summarized yes/no data
  # like in the hss table.  If you already have individual
  # quantitative variables y with associated weights w
  # for each person in the sample, you don't need to do this.



dim(tall.data)
tall.data[1:10,]



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



```
#
# an implementation of Taylor approximation
# (linearization) to estimate the variance
# of a post-stratification weighted average
#

ts.variance <- function(y,w) {
  # given individual observations y and associated weights w,
  # compute
  #
  # * weighted ybar
  # * taylor-series variance

  if (missing(w)) {
    w <- y[,2]
    y <- y[,1]
  }

  n  <- length(y)
  wy <- w*y

  ybar.weighted <- sum(wy)/sum(w)

  wy.bar   <- sum(wy)/n
  w.bar    <- sum(w)/n
  s2.wy    <- sum((wy - wy.bar)^2)/(n-1)
  s2.w     <- sum((w - w.bar)^2)/(n-1)
  cov.wy.w <- sum((wy-wy.bar)*(w-w.bar))/(n-1)

  var.sum.wy       <- n*s2.wy
  var.sum.w        <- n*s2.w
  cov.sum.wy.sum.w <- n*cov.wy.w
  sum.w            <- sum(w)

  tsvar <-  (var.sum.wy
             - 2*ybar.weighted*cov.sum.wy.sum.w
             + (ybar.weighted)^2*var.sum.w) / (sum.w)^2

  return(list(ybar.weighted=ybar.weighted,var.ts=tsvar))

}



ts.variance(tall.data)



############################################################
#
# an implementation of the jackknife procedure
# to estimate the variance of a post-
# stratification weighted average
#

jk.variance <- function(y,sti,P.Tot) {
  #
  # given
  # * individual observations y
  # * stratum identifiers sti
  # * stratum population totals P.Tot
  #
  # compute
```

```
  #
  # * weighted ybar
  # * jackknifed ybar
  # * jackknifed variance
  #
  # Nb., the "sti" can be any unique set of stratum indicators
  # (1 = stratum 1, 2 = stratum 2, etc.), and should be of
  # the same length as y.  The "P.Tot" should be a vector of
  # population totals, one for each stratum.

  H <- length(unique(sti))
  if (length(P.Tot)!=H) {
    stop("Need 'P.Tot' to contain the population sizes of the strata.")
  }

  P.Prop <- P.Tot/sum(P.Tot)

  n <- length(y)
  if (length(sti)!=n) {
    stop("Need 'sti' to contain stratum indicators for each y.")
  }

  weights <- function(sti,P.Prop) {
    S.Tot <- table(sti)
    S.Prop <- S.Tot/length(sti)
    S.Wgts <- P.Prop[order(unique(sti))]/S.Prop
    w <- rep(0,length(sti))
    for (h in 1:length(S.Wgts)) {
      stn <- names(S.Wgts)[h]
      w[sti==stn] <- S.Wgts[stn]
    }
    return(w)
  }

  w <- weights(sti,P.Prop)
  ybar.weighted <- sum(y*w)/sum(w)

  ybar.r <- NULL
  for (r in 1:n) {
    y.r <- y[-r]
    sti.r <- sti[-r]
    w.r <- weights(sti.r,P.Prop)
    ybar.r <- c(ybar.r,sum(w.r*y.r)/sum(w.r))
  }

  ybar.reps <- mean(ybar.r)
  s2.reps <- var(ybar.r)

  jkvar <- (n-1)/n * (n-1) * s2.reps

  return(list(ybar.weighted=ybar.weighted,ybar.reps=ybar.reps,
              var.jk=jkvar))

}



jk.variance(tall.data[,1],tall.data[,2],data$P.Tot)
```



```
#####################################################################
#
# Here's an implementation of the Taylor Series method by
# hand, rather than by function, for your curiosity...
#

ybar.weighted <- sum(data$Weights*data$Yesses)/sum(data$Weights*data$S.Tot)

w.bar <- sum(data$Weights*data$S.Tot)/sum(data$S.Tot)

wy.bar <- sum(data$Weights*data$Yesses)/sum(data$S.Tot)

s2.wy <- (sum(data$Yesses*(data$Weights - wy.bar)^2)
          + sum((data$S.Tot-data$Yesses)*(0 - wy.bar)^2))/(sum(data$S.Tot)-1)

s2.w <- sum(data$S.Tot*(data$Weights - w.bar)^2)/(sum(data$S.Tot)-1)

cov.wy.w <- (sum(data$Yesses*(data$Weights - wy.bar)*(data$Weights-w.bar))
             + sum((data$S.Tot-data$Yesses)*(0 - wy.bar)*(data$Weights-w.bar))
             ) / (sum(data$S.Tot)-1)

var.sum.wy <-   sum(data$S.Tot)*s2.wy
var.sum.w <-    sum(data$S.Tot)*s2.w
cov.sum.wy.w <- sum(data$S.Tot)*cov.wy.w
sum.w <-        sum(data$S.Tot)*w.bar

var.ts <- (var.sum.wy
           - 2*ybar.weighted*cov.sum.wy.w
           + (ybar.weighted)^2*var.sum.w) / (sum.w)^2

#############################
#
# results:
#

ybar.weighted
w.bar
wy.bar
s2.wy
s2.w
cov.wy.w
var.sum.wy
var.sum.w
sum.w

var.ts

sum(data$Yesses)/sum(data$S.Tot)
.53*(1-.53)/304

1-304/986

#
# For the taylor series variance we get
#
#    var.ts = 0.0009733187
#
# for the unweighted estimate  p = 0.526 the naive
# "unweighted" variance would be
#
#    p*(1-p)/304 = 0.0008194079
#
# We usually expect the weighted variance to be larger
```

```
# than the unweighted one.
#
# The fpc (which we would multiply either variance estimate by)
# is
#
#  1 - 304/986 = 0.6916836
#


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