Example survey data inputs for `incidence_calculator` tool.

Jeff Eaton

Fri Dec 1 19:57:57 2017

Load packages and functions

```
library(survey)
invlogit <- function(x) exp(x)/(1+exp(x))
```

Simulate survey data

Create a survey with 10 strata and 40 clusters sampled in each stratum with equal probability. Cluster population sizes are distributed NegBin(μ=1000, size=10).

```
set.seed(6841634)
cluster <- data.frame(stratum = rep(LETTERS[1:10], each=40),
                      cluster = 1:400,
                      totpop = rnbinom(400, size=10, mu=1000))
cluster$prev <- invlogit(rnorm(40, -2.5, 0.5))
cluster$incid <- cluster$prev / 10
```

Sample 30 respondents per cluster and calculate weights based on sampling probability.

```
data <- cluster[rep(cluster$cluster, each=30),]
data$weight <- data$totpop / 30
```

Normalised sample weights (e.g. as published by DHS).

```
data$weight_norm <- nrow(data) *data$weight / sum(data$weight)
```

Simulate HIV recency testing outcomes and create a factor variable summarizing the outputs of recent testing as recent, HIV positive but not recent, and HIV negative.

```
frr <- 0.01
mdri <- 130/365
T <- 1.0
data$hivstatus <- rbinom(nrow(data), 1, data$prev)

p_recent <- (1-data$prev)/data$prev * data$incid * (mdri - frr*T)
data$recentstatus <- data$hivstatus * rbinom(nrow(data), 1, p_recent)

data$recent <- factor(data$hivstatus + data$recentstatus, 2:0,
                      c("recent", "not recent", "negative"))
```

Analyse survey data for prevalence and proportion recent

```
des <- svydesign(~cluster, strata=~stratum, data=data, weights=~weight_norm)
```

We can directly analyse the prevalence, proporiotn recent, and proportion recent among HIV positive using the survey package. This furnishes

```
svymean(~hivstatus, des, deff="replace")
```

```
##                mean        SE   DEff
## hivstatus 0.0913737 0.0034341 1.7044
```

```
svymean(~recentstatus, des, deff="replace")
```

```
##                    mean         SE   DEff
## recentstatus 0.00286562 0.00050115 1.0547
```

```
svymean(~recentstatus, subset(des, hivstatus==1), deff="replace")
```

```
##                   mean        SE  DEff
## recentstatus 0.0313615 0.0054554 1.062
```

Canonical outputs from standard survey analysis may consist of estimates of population totals or proportions {recent; not recent; HIV negative}. The survey package furnishes estimates for these totals or proportions and the covariance of these estimates accounting for the complex survey design.

```
totals <- svytotal(~recent, des)
totals
```

```
##                      total       SE
## recentrecent        34.387   6.0265
## recentnot recent  1062.097  45.0074
## recentnegative   10903.516 172.5990
```

```
cov2cor(vcov(totals))
```

```
##                  recentrecent recentnot recent recentnegative
## recentrecent       1.00000000       0.01742908     0.03430433
## recentnot recent   0.01742908       1.00000000     0.19748555
## recentnegative     0.03430433       0.19748555     1.00000000
```

```
props <- svymean(~recent, des)
props
```

```
##                       mean     SE
## recentrecent     0.0028656 0.0005
## recentnot recent 0.0885081 0.0034
## recentnegative   0.9086263 0.0034
```

```
cov2cor(vcov(props))
```

```
##                  recentrecent recentnot recent recentnegative
## recentrecent       1.00000000      -0.01384235     -0.1322106
## recentnot recent  -0.01384235       1.00000000     -0.9892966
## recentnegative    -0.13221064      -0.98929657      1.0000000
```

The inputs for the `incprops()` function can be summarized as a transformation of either the population totals or the population proportions. In the case of estiamted population totals, we consider `{prev, prop_recent} = F(n_recent, n_not_recent, n_negative)`, and the covariance of `{prev, prop_recent}` is estimated by application of the delta method.

```
F <- function(totals){  # totals = {n_recent, n_not_recent, n_negative}
  c(prev        = sum(totals[1:2]) / sum(totals),
    prop_recent = unname(totals[1] / sum(totals[1:2])))
}
```

```r
dF <- function(totals){
  cbind(prev        = unname(c(totals[3], totals[3], -sum(totals[1:2])) / sum(totals)^2),
        prop_recent = unname(c(totals[2], -totals[1], 0) / sum(totals[1:2])^2))
}

estF <- F(totals) # {prevalence, prop_recent}
estF_V <- t(dF(totals)) %*% vcov(totals) %*% dF(totals)
estF
```

```
##        prev prop_recent
##  0.09137367  0.03136152
```

```r
sqrt(diag(estF_V)) # standard errors of {prevalence, prop_recent}
```

```
##        prev prop_recent
## 0.003434146 0.005455410
```

```r
cov2cor(estF_V)    # correlation of {prevalence, prop_recent}
```

```
##                    prev prop_recent
## prev         1.00000000 -0.08313673
## prop_recent -0.08313673  1.00000000
```

Note that the estimates and standard errors are the same as those estiamted above through direct application of `svymean()`. For outputs consisting of population proportions, the transformation is simpler because it does not depend on the proportion negative and this input can be omitted.

```r
G <- function(props){  # props = {prop_recent, prop_not_recent, prop_negative}
  c(prev        = sum(props[1:2]),
    prop_recent = unname(props[1] / sum(props[1:2])))
}
dG <- function(props){
  cbind(prev        = c(1, 1, 0),
        prop_recent = unname(c(props[2], -props[1], 0) / sum(props[1:2])^2))
}

estG <- G(props) # {prevalence, prop_recent}
estG_V <- t(dG(props)) %*% vcov(props) %*% dG(props)

estG
```

```
##        prev prop_recent
##  0.09137367  0.03136152
```

```r
sqrt(diag(estG_V)) # standard errors of {prevalence, prop_recent}
```

```
##        prev prop_recent
## 0.003434146 0.005455410
```

```r
cov2cor(estG_V)    # correlation of {prevalence, prop_recent}
```

```
##                    prev prop_recent
## prev         1.00000000 -0.08313673
## prop_recent -0.08313673  1.00000000
```

Incomplete recency testing

Suppose that some proportion of HIV positive tests did not undergo recency testing. Then the proportion recent is calculated among only those who underwent recency testing whilst those not tested for recent infection are included in the prevlance calculation. The formulas for the transformation are easily updated to reflect this.

First simulate a 10% proportion of HIV positive samples that did not undergo recency testing.

```
head(data)
```

```
##     stratum cluster totpop      prev      incid   weight weight_norm
## 1         A       1    985 0.1095561 0.01095561 32.83333   0.9842102
## 1.1       A       1    985 0.1095561 0.01095561 32.83333   0.9842102
## 1.2       A       1    985 0.1095561 0.01095561 32.83333   0.9842102
## 1.3       A       1    985 0.1095561 0.01095561 32.83333   0.9842102
## 1.4       A       1    985 0.1095561 0.01095561 32.83333   0.9842102
## 1.5       A       1    985 0.1095561 0.01095561 32.83333   0.9842102
##     hivstatus recentstatus   recent
## 1           0            0 negative
## 1.1         0            0 negative
## 1.2         0            0 negative
## 1.3         0            0 negative
## 1.4         0            0 negative
## 1.5         0            0 negative
```

```
data$recent2 <- factor(data$recent, c("recent", "not recent", "no lag", "negative"))
data$recent2[data$hivstatus == 1 & rbinom(nrow(data), 1, 0.1) == 1] <- "no lag"

table(data$recent)
```

```
##
##     recent not recent   negative
##         35       1050      10915
```

```
table(data$recent2)
```

```
##
##     recent not recent     no lag   negative
##         33        943        109      10915
```

```
des <- svydesign(~cluster, strata=~stratum, data=data, weights=~weight_norm)

totals2 <- svytotal(~recent2, des)
props2 <- svymean(~recent2, des)
totals2
```

```
##                       total      SE
## recent2recent        31.331   5.694
## recent2not recent   954.639  42.248
## recent2no lag       110.514  11.053
## recent2negative   10903.516 172.599
```

```
props2
```

```
##                         mean     SE
## recent2recent     0.0026109 0.0005
## recent2not recent 0.0795532 0.0032
## recent2no lag     0.0092095 0.0009
```

```
## recent2negative   0.9086263 0.0034
```

Update transformations to account for proportion not tested.

```
Fstar <- function(totals){ # totals = {n_recent, n_not_recent, n_not_tested, n_negative}
  c(prev        = sum(totals[1:3]) / sum(totals),
    prop_recent = unname(totals[1] / sum(totals[1:2])))
}

dFstar <- function(totals){
  cbind(prev        = unname(c(totals[4], totals[4], totals[4], -sum(totals[1:3])) / sum(totals)^2),
        prop_recent = unname(c(totals[2], -totals[1], 0, 0) / sum(totals[1:2])^2))
}

estF2 <- Fstar(totals2) # {prevalence, prop_recent}
estF2_V <- t(dFstar(totals2)) %*% vcov(totals2) %*% dFstar(totals2)
estF2
```

```
##        prev prop_recent
##  0.09137367  0.03177669
```

```
sqrt(diag(estF2_V)) # standard errors of {prevalence, prop_recent}
```

```
##        prev prop_recent
## 0.003434146 0.005758756
```

```
cov2cor(estF2_V)    # correlation of {prevalence, prop_recent}
```

```
##                    prev prop_recent
## prev         1.00000000 -0.09463748
## prop_recent -0.09463748  1.00000000
```

And now the case of population proportions

```
Gstar <- function(props){  # props = {prop_recent, prop_not_recent, prop_not_tested, prop_negative}
  c(prev        = sum(props[1:3]),
    prop_recent = unname(props[1] / sum(props[1:2])))
}
dGstar <- function(props){
  cbind(prev        = c(1, 1, 1, 0),
        prop_recent = unname(c(props[2], -props[1], 0, 0) / sum(props[1:2])^2))
}

estG2 <- Gstar(props2) # {prevalence, prop_recent}
estG2_V <- t(dGstar(props2)) %*% vcov(props2) %*% dGstar(props2)

estG2
```

```
##        prev prop_recent
##  0.09137367  0.03177669
```

```
sqrt(diag(estG2_V)) # standard errors of {prevalence, prop_recent}
```

```
##        prev prop_recent
## 0.003434146 0.005758756
```

```
cov2cor(estG2_V)    # correlation of {prevalence, prop_recent}
```

```
##                    prev prop_recent
## prev         1.00000000 -0.09463748
```

```
## prop_recent -0.09463748  1.00000000
```