Data Analysis and Simulation for Ratio and Regression Estimation

Longhai Li

October 2022

Contents

1 Functions and packages for Analyzing Data

```
## ydata --- observations of the variable of interest
## xdata --- observations of the auxilliary variable
## N --- population size
## xbarU --- population mean of auxilliary variable

## the output is the estimate mean or total (est.total=TRUE)
srs_reg_est <- function (ydata, xdata, xbarU, N = Inf, est.total = FALSE)
{
```

```r
  n <- length (ydata)
  lmfit <- lm (ydata ~ xdata)
  Bhat <- lmfit$coefficients
  efit <- lmfit$residuals
  SSe <- sum (efit^2) / (n - 2)
  yhat_reg <- Bhat[1] + Bhat[2] * xbarU
  se_yhat_reg <- sqrt ((1-n/N) * SSe / n)
  mem <- qt (0.975, df = n - 2) * se_yhat_reg
  output <- c(yhat_reg, se_yhat_reg, yhat_reg - mem, yhat_reg + mem)

  if (est.total) {
      if(!is.finite(N)) stop("N must be finite for estimating population total" )
      output <- output * N
  }

  names (output) <- c("Est.", "S.E.", "ci.low", "ci.upp" )
  output
}


## ydata --- observations of the variable of interest
## xdata --- observations of the auxilliary variable
## N --- population size

## the output is the ratio of ybarU/xbarU
srs_ratio_est <- function (ydata, xdata, N = Inf)
{
  n <- length (xdata)
  xbar <- mean (xdata)
  ybar <- mean (ydata)
  B_hat <- ybar / xbar
  d <- ydata - B_hat * xdata
  var_d <- sum (d^2) / (n - 1)
  sd_B_hat <- sqrt ((1 - n/N) * var_d / n) / xbar
  mem <- qt (0.975, df = n - 1) * sd_B_hat
  output <- c (B_hat, sd_B_hat, B_hat - mem, B_hat + mem )


  names (output) <- c("Est.", "S.E.", "ci.low", "ci.upp" )
  output
}

## sdata --- a vector of original survey data in a domain
## N --- population size
## n --- total sample size (not the sample size in the domain)
## to find total, multiply domain size N_d to the estimate returned by this function
srs_domain_mean_est <- function (sdata, n, N = Inf)
{
    n_d <- length (sdata)
    ybar <- mean (sdata)
    se.ybar <- sqrt((1 - n / N)) * sd (sdata) / sqrt(n_d)
    mem <- qt (0.975, df = n_d - 1) * se.ybar
    c (Est. = ybar, S.E. = se.ybar, ci.low = ybar - mem, ci.upp = ybar + mem)
```

```
}

## sdata --- a vector of original survey data
## N --- population size
## to find total, multiply N to the estimate returned by this function

srs_mean_est <- function (sdata, N = Inf)
{
    n <- length (sdata)
    ybar <- mean (sdata)
    se.ybar <- sqrt((1 - n / N)) * sd (sdata) / sqrt(n)
    mem <- qt (0.975, df = n - 1) * se.ybar
    c (Est. = ybar, S.E. = se.ybar, ci.low = ybar - mem, ci.upp = ybar + mem)
}

## to find total, multiply N to the estimate returned by this function
## for poststratification, use nh = n * Nh/N
str_mean_estimate <- function (ybarh, sh, nh, Nh)
{
    N <- sum (Nh)
    Pi_h <- Nh/N
    ybar <- sum(ybarh * Pi_h)
    seybar <- sqrt(sum((1-nh/Nh)*Pi_h^2*sh^2/nh))
    mem <- 1.96 * seybar
    c(Est. = ybar, S.E. = seybar, ci.low = ybar - mem, ci.upp = ybar + mem)
}
```

2 Ratio Estimation for cherry.csv dataset

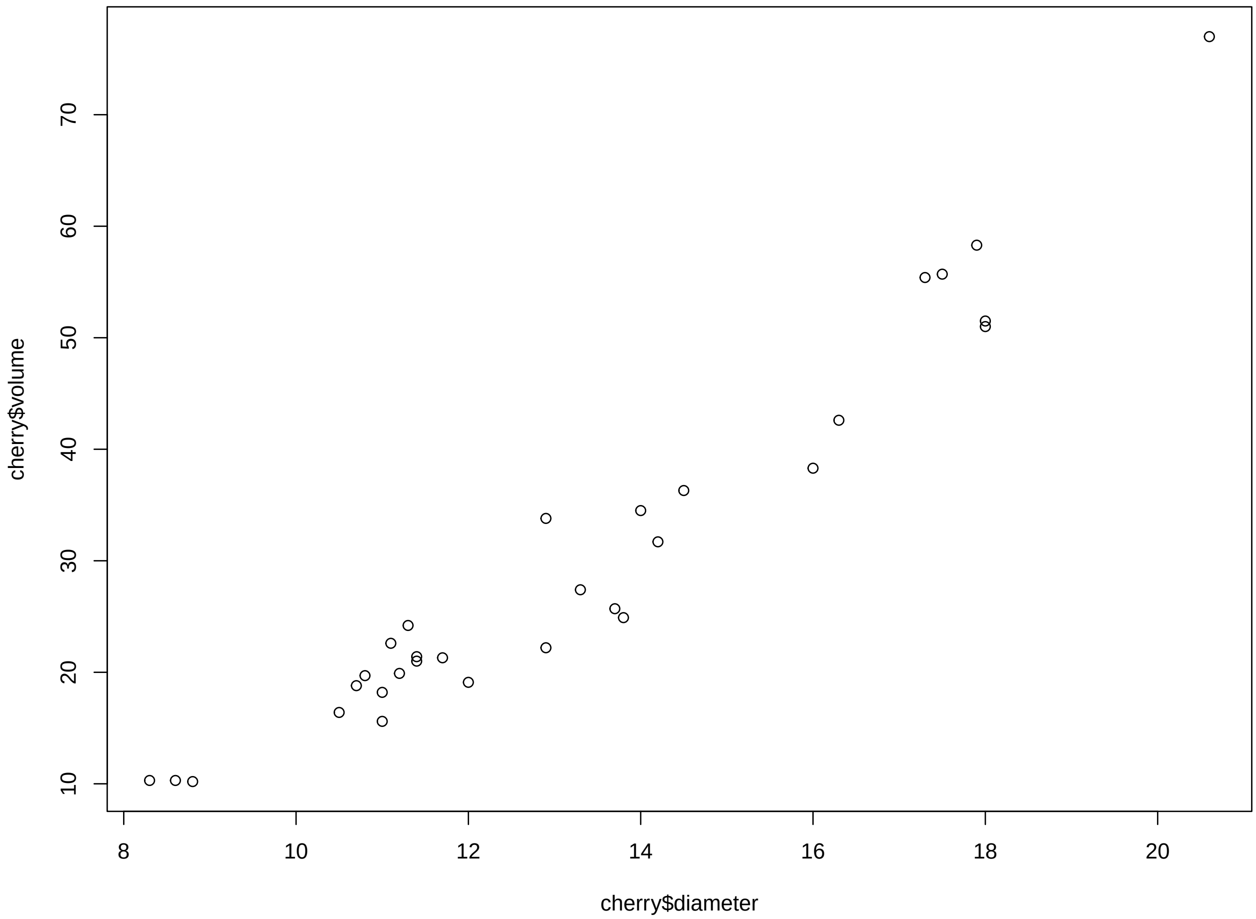
2.1 Importing data

```
cherry <- read.csv ("data/cherry.csv", header = T)
cherry
```

```
##    diameter height volume
## 1       8.3     70   10.3
## 2       8.6     65   10.3
## 3       8.8     63   10.2
## 4      10.5     72   16.4
## 5      10.7     81   18.8
## 6      10.8     83   19.7
## 7      11.0     66   15.6
## 8      11.0     75   18.2
## 9      11.1     80   22.6
## 10     11.2     75   19.9
## 11     11.3     79   24.2
## 12     11.4     76   21.0
## 13     11.4     76   21.4
## 14     11.7     69   21.3
## 15     12.0     75   19.1
## 16     12.9     74   22.2
## 17     12.9     85   33.8
## 18     13.3     86   27.4
```

```
## 19      13.7     71   25.7
## 20      13.8     64   24.9
## 21      14.0     78   34.5
## 22      14.2     80   31.7
## 23      14.5     74   36.3
## 24      16.0     72   38.3
## 25      16.3     77   42.6
## 26      17.3     81   55.4
## 27      17.5     82   55.7
## 28      17.9     80   58.3
## 29      18.0     80   51.5
## 30      18.0     80   51.0
## 31      20.6     87   77.0
```

```
plot (cherry$volume ~ cherry$diameter)
```




```
summary (lm(cherry$volume ~ 0+cherry$diameter))
```

```
##
## Call:
## lm(formula = cherry$volume ~ 0 + cherry$diameter)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.104  -8.470  -6.199   1.883  27.129
##
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## cherry$diameter   2.4209     0.1253   19.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.493 on 30 degrees of freedom
## Multiple R-squared:  0.9256, Adjusted R-squared:  0.9231
## F-statistic: 373.1 on 1 and 30 DF,  p-value: < 2.2e-16
```

2.2 SRS estimate

```r
N <- 2967
## estimating the mean of volume
srs_mean_volume <- srs_mean_est(cherry$volume, N = N)
srs_mean_volume
```

```
##      Est.      S.E.    ci.low    ci.upp
## 30.170968  2.936861 24.173098 36.168837
```

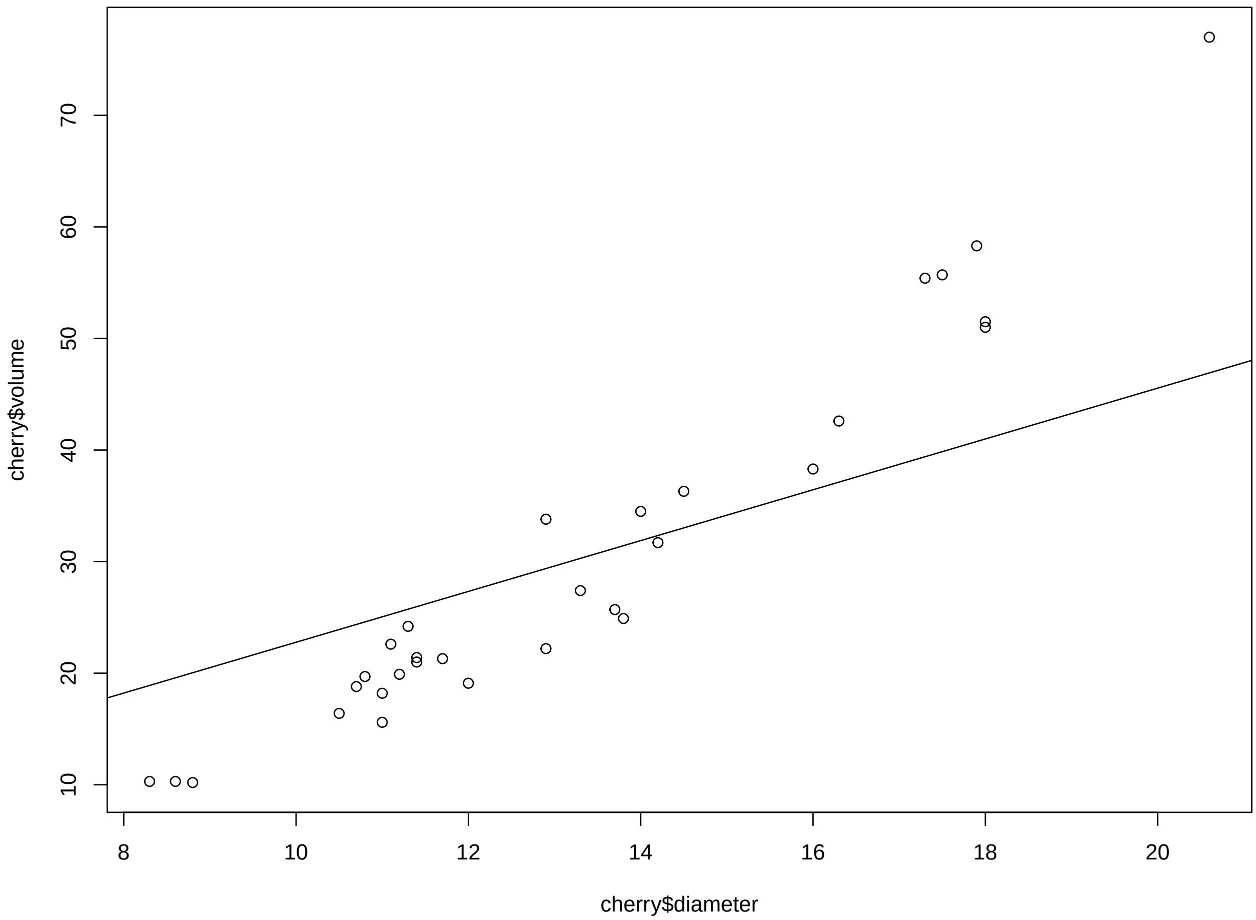
```r
## estimating the total of volume
srs_total_volume <- srs_mean_est(cherry$volume, N = N) * N
srs_total_volume
```

```
##       Est.       S.E.     ci.low     ci.upp
##  89517.261   8713.665  71721.583 107312.940
```

2.3 Step-by-step calculation with Ratio Estimation

2.3.1 Estimating B and calculating residuals

```r
## input
ydata <- cherry$volume
xdata <- cherry$diameter
N <- 2967

## calculation
n <- length (xdata)
xbar <- mean (xdata)
ybar <- mean (ydata)
B_hat <- ybar / xbar ## ratio estimate


plot (cherry$volume ~ cherry$diameter)
abline (a = 0, b = B_hat)
```




```
d <- ydata - B_hat * xdata ## errors
data.frame (cherry, d = d)
```

```
##    diameter height volume          d
## 1       8.3     70   10.3 -8.6018505
## 2       8.6     65   10.3 -9.2850499
## 3       8.8     63   10.2 -9.8405162
## 4      10.5     72   16.4 -7.5119795
## 5      10.7     81   18.8 -5.5674458
## 6      10.8     83   19.7 -4.8951790
## 7      11.0     66   15.6 -9.4506452
## 8      11.0     75   18.2 -6.8506452
## 9      11.1     80   22.6 -2.6783784
## 10     11.2     75   19.9 -5.6061115
## 11     11.3     79   24.2 -1.5338447
## 12     11.4     76   21.0 -4.9615778
## 13     11.4     76   21.4 -4.5615778
## 14     11.7     69   21.3 -5.3447772
## 15     12.0     75   19.1 -8.2279766
## 16     12.9     74   22.2 -7.1775749
## 17     12.9     85   33.8  4.4224251
## 18     13.3     86   27.4 -2.8885074
## 19     13.7     71   25.7 -5.4994400
## 20     13.8     64   24.9 -6.5271731
## 21     14.0     78   34.5  2.6173606
```

```
## 22     14.2     80   31.7 -0.6381057
## 23     14.5     74   36.3  3.2786949
## 24     16.0     72   38.3  1.8626978
## 25     16.3     77   42.6  5.4794984
## 26     17.3     81   55.4 16.0021670
## 27     17.5     82   55.7 15.8467008
## 28     17.9     80   58.3 17.5357682
## 29     18.0     80   51.5 10.5080351
## 30     18.0     80   51.0 10.0080351
## 31     20.6     87   77.0 30.0869735
```

2.3.2 Estimating SE of B

```r
## estimating S^2_e
var_d <- sum (d^2) / (n - 1) ## variance of errors
sd_B_hat <- sqrt ((1 - n/N) * var_d / n) / xbar ## SE for B
mem <- qt (0.975, df = n - 1) * sd_B_hat ## margin error for B

## output
output_B <- c (B_hat, sd_B_hat, B_hat - mem, B_hat + mem )
names (output_B) <- c("Est.", "S.E.", "ci.low", "ci.upp" )
output_B
```

```
##     Est.     S.E.   ci.low   ci.upp
## 2.277331 0.130786 2.010231 2.544432
```

2.3.3 Estimating the mean volume of wood

```r
mean_diameters <- 41835/N
output_B * mean_diameters
```

```
##      Est.      S.E.    ci.low    ci.upp
## 32.110603  1.844097 28.344455 35.876750
```

2.3.4 Estimating the total volume of wood

```r
t_diameters <- 41835
output_B * t_diameters
```

```
##       Est.       S.E.     ci.low     ci.upp
##  95272.159   5471.434  84097.999 106446.318
```

2.4 Ratio estimation with a function

2.4.1 Estimating the ratio of volume to diameter

```r
B_v2d <- srs_ratio_est (ydata = cherry$volume, xdata = cherry$diameter,  N = 2967)
B_v2d
```

```
##     Est.     S.E.   ci.low   ci.upp
## 2.277331 0.130786 2.010231 2.544432
```

2.4.2 Estimating the mean of volume

```
xbarU <- 41835/N
ratio_mean_volume <- srs_ratio_est (ydata = cherry$volume, xdata = cherry$diameter, N = 2967) * xbarU
ratio_mean_volume
```

```
##       Est.      S.E.    ci.low    ci.upp
## 32.110603  1.844097 28.344455 35.876750
```

2.4.3 Estimating the total of volume

```
total_diameters <- 41835
srs_ratio_est (ydata = cherry$volume, xdata = cherry$diameter, N = 2967) * total_diameters
```

```
##        Est.       S.E.     ci.low     ci.upp
##   95272.159   5471.434  84097.999 106446.318
```

2.4.4 Percentage of Variance Reduction

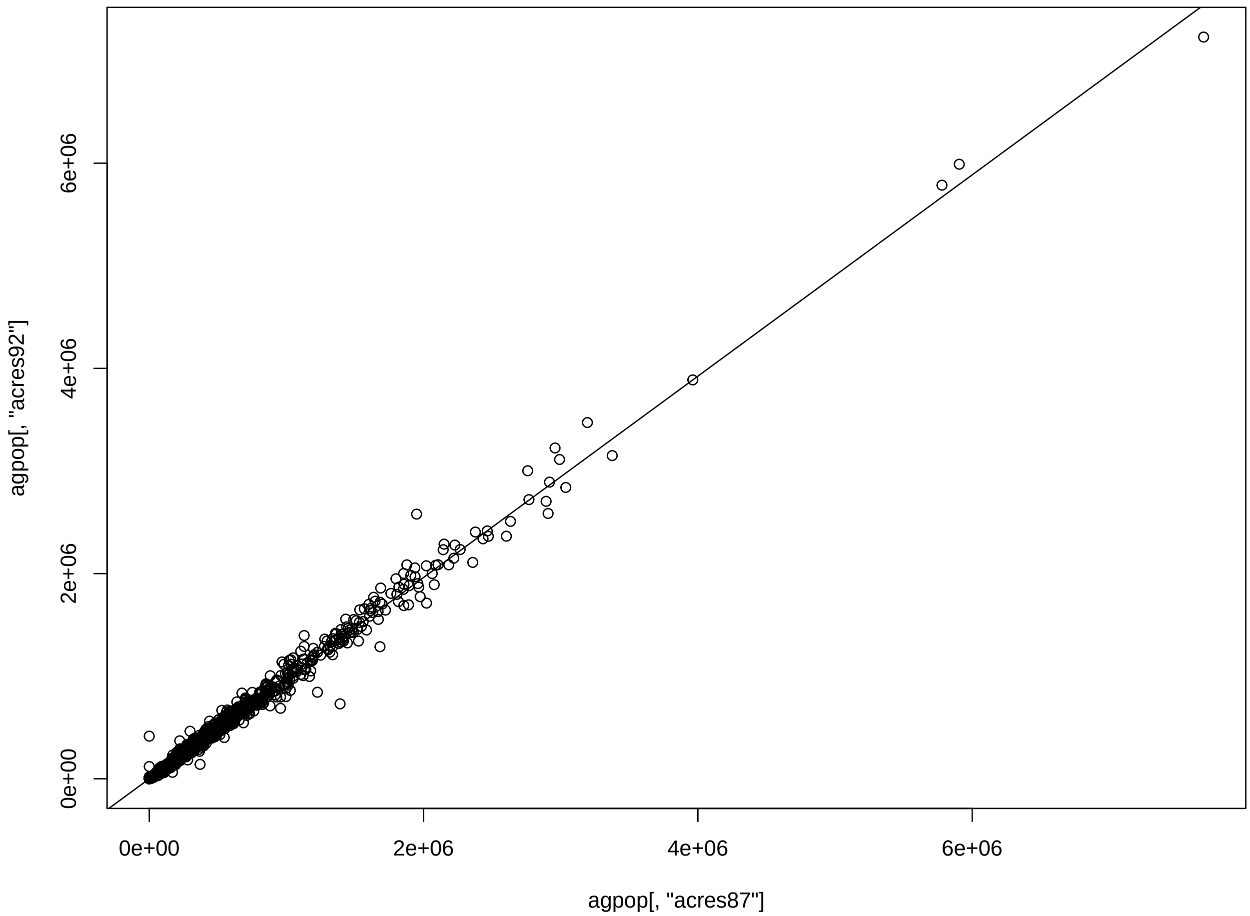
```
cat(1-(ratio_mean_volume[2]/srs_mean_volume[2])^2)
```

```
## 0.6057237
```

3 Simulation study with agpop.csv

3.1 Information of population data

```
agpop <- read.csv ("data/agpop.csv")
agpop <- agpop[agpop$acres92 != -99, ] ## remove those counties with na

# sample size
n <- 300
# population size
N <- nrow (agpop)

# true values that we want to estimate
tyU <- sum (agpop [,"acres92"])
# suppose known for ratio estimate
txU <- sum (agpop [,"acres87"])
B <- tyU/txU
plot (agpop [, "acres87"], agpop [, "acres92"])
abline(a=0,b=B)
```




```r
# expected reduction of variance of ratio estimate to srs

1-var(agpop$acres92-B*agpop$acres87)/var(agpop$acres92)
```

```
## [1] 0.9917355
```

```r
# linear model output

summary(lm(agpop$acres92 ~ agpop$acres87))
```

```
##
## Call:
## lm(formula = agpop$acres92 ~ agpop$acres87)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -642399   -8256    -692    5593  656458
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -2.121e+03  8.648e+02  -2.453   0.0142 *
## agpop$acres87  9.878e-01  1.626e-03 607.388   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38560 on 3057 degrees of freedom
```

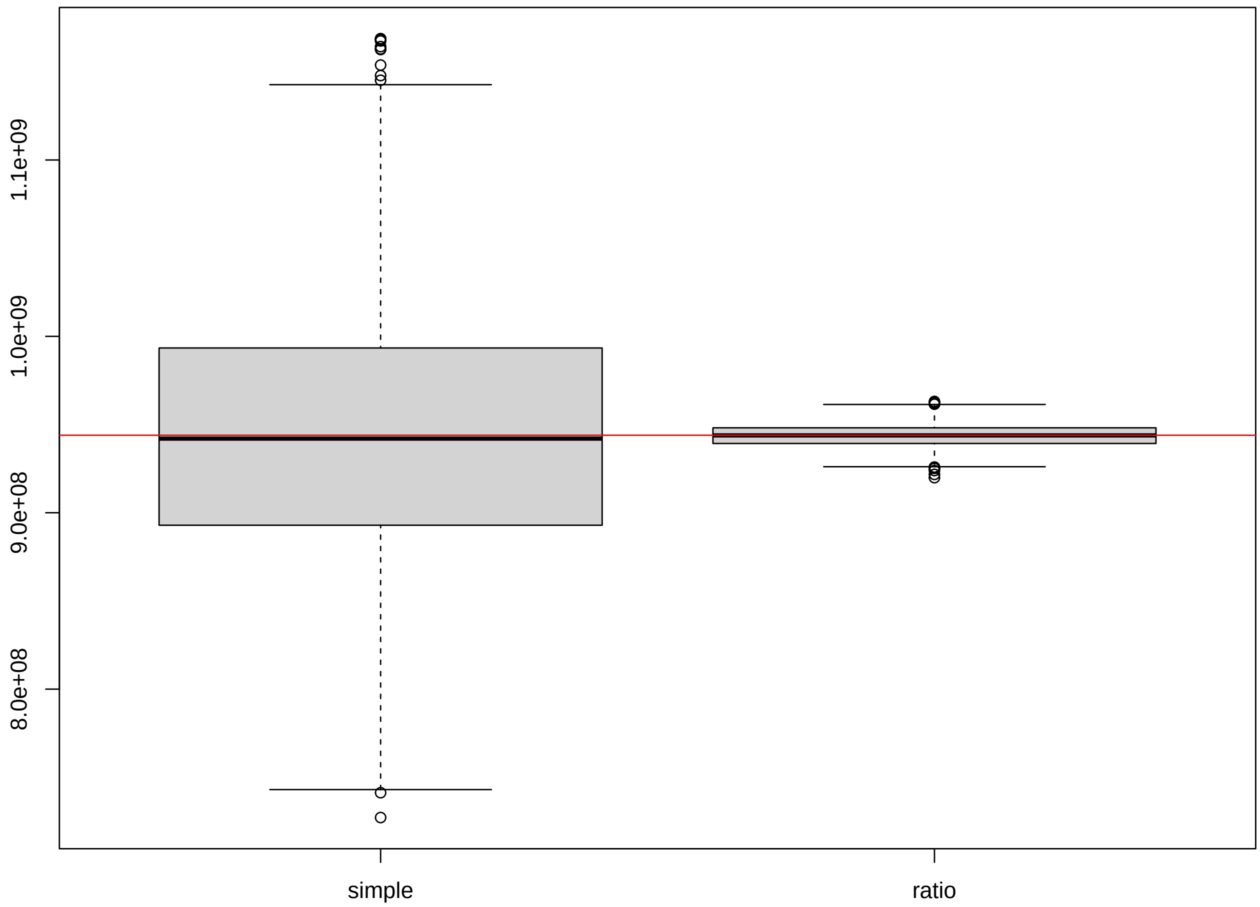
```
## Multiple R-squared:  0.9918, Adjusted R-squared:  0.9918
## F-statistic: 3.689e+05 on 1 and 3057 DF,  p-value: < 2.2e-16
```

3.2 Simulation Studies

```r
nsim <- 2000
sim_rat <- data.frame (
    simple = rep (0, nsim), ratio = rep (0, nsim), B = rep (0, nsim))

for (i in 1:nsim)
{
  srs <- sample (N,n)
  y_srs <- agpop[srs, "acres92"]
  x_srs <- agpop[srs, "acres87"]

  # SRS estimate
  sim_rat$simple [i] <- mean (y_srs) * N
  # ratio estimate
  sim_rat$B [i] <- mean (y_srs) / mean (x_srs)
  sim_rat$ratio [i] <-  sim_rat$B [i] * txU
}

sim_rat[1:20,]
```

```
##        simple     ratio         B
## 1   825572413 944303128 0.9814095
## 2   994650838 936792246 0.9736035
## 3   763211415 948349550 0.9856149
## 4  1021784280 942586681 0.9796256
## 5   900020805 943547960 0.9806247
## 6   971200584 936981823 0.9738005
## 7  1059150709 954288942 0.9917877
## 8   863204741 941106670 0.9780874
## 9   932517582 939368272 0.9762807
## 10  973547357 956017962 0.9935847
## 11  864109175 938126389 0.9749901
## 12  829595314 938863565 0.9757562
## 13  929355116 953788532 0.9912676
## 14  943206737 944974778 0.9821075
## 15 1045243037 946151828 0.9833309
## 16  956421373 958355312 0.9960139
## 17  896965354 950007334 0.9873379
## 18  902881684 946490761 0.9836831
## 19  975085698 949493594 0.9868039
## 20  882554741 952674621 0.9901100
```

```r
boxplot (sim_rat [, 1:2])
abline (h = tyU, col = "red")
```




```
sim_var <- sapply (sim_rat[,1:2], var)

ratio_sim_var <- sim_var/sim_var[1]
percentage_reduction <- 1-ratio_sim_var
data.frame (sim_var, ratio_sim_var,percentage_reduction)
```

```
##             sim_var ratio_sim_var percentage_reduction
## simple 5.420187e+15   1.000000000            0.0000000
## ratio  4.206125e+13   0.007760111            0.9922399
```

4 Estimating Domain Means and Post-stratification with agsrs.csv example

4.1 Estimating Domain Means

```
agsrs <- read.csv("data/agsrs.csv")
n <- nrow (agsrs)
n
```

```
## [1] 300
```

```
## estimating domain means
region_agsrs_domain <- data.frame(rbind(
    NC = srs_domain_mean_est(agsrs$acres92[agsrs$region=="NC"], n=n, N = 3078),
    NE = srs_domain_mean_est(agsrs$acres92[agsrs$region=="NE"], n=n,N = 3078),
```

```
    S = srs_domain_mean_est(agsrs$acres92[agsrs$region=="S"], n=n,N = 3078),
    W = srs_domain_mean_est(agsrs$acres92[agsrs$region=="W"], n=n,N = 3078))
)
region_agsrs_domain
```

```
##        Est.     S.E.    ci.low   ci.upp
## NC 323416.4 30008.35 263662.18 383170.7
## NE 106549.0 28957.56  45453.89 167644.1
## S  246185.8 23503.62 199766.28 292605.4
## W  518977.6 70298.69 377206.82 660748.5
```

```
## Compared to the results of naively applying SRS estimation to each domain
region_agsrs_srs <- data.frame(rbind(
    NC = srs_mean_est(agsrs$acres92[agsrs$region=="NC"], N = 220),
    NE = srs_mean_est(agsrs$acres92[agsrs$region=="NE"], N = 3078),
    S = srs_mean_est(agsrs$acres92[agsrs$region=="S"], N = 1382),
    W = srs_mean_est(agsrs$acres92[agsrs$region=="W"], N = 422))
)
region_agsrs_srs
```

```
##        Est.     S.E.    ci.low   ci.upp
## NC 323416.4 25377.16 272884.05 373948.8
## NE 106549.0 30391.81  42427.89 170670.1
## S  246185.8 23264.01 200239.52 292132.2
## W  518977.6 70033.38 377741.86 660213.4
```

4.2 Post-stratification Analysis

```
nh <- tapply (1:nrow(agsrs), agsrs[,"region"], length)
n <- sum (nh)
n
```

```
## [1] 300
```

```
sh <- tapply (agsrs[, "acres92"], agsrs[,"region"], sd)
ybarh <- tapply (agsrs[, "acres92"], agsrs[,"region"], mean)
# create a vector with external information
Nh <- c(NC = 1054, NE = 220, S= 1382, W = 422)
```

Create nh proportional to Nh, instead of using observed nh

```
nh_post <- Nh/sum(Nh)* n

## regional summary for stratified sampling estimation
data.frame(cbind(ybarh, sh, nh, prop_obs=nh/sum(nh), Nh, nh_post, prop_post = nh_post/sum(nh_post)))
```

```
##       ybarh       sh  nh  prop_obs   Nh  nh_post  prop_post
## NC 323416.4 278970.0  78 0.2600000 1054 102.7290 0.34243015
## NE 106549.0 129320.2  18 0.0600000  220  21.4425 0.07147498
## S  246185.8 312941.3 160 0.5333333 1382 134.6979 0.44899285
## W  518977.6 490842.0  44 0.1466667  422  41.1306 0.13710201
```

```
## strata mean estimate
str_mean_estimate (ybarh, sh, nh_post, Nh)
```

```
##      Est.      S.E.    ci.low    ci.upp
## 300051.69  17760.26 265241.58 334861.79
```

```r
## compare with SRS estimate

srs_mean_est(agsrs$acres92, N = 3087)
```

```
##      Est.      S.E.    ci.low    ci.upp
## 297897.05  18901.41 260700.40 335093.69
```

```r
## true mean
mean(agpop$acres92)
```

```
## [1] 308582.4
```

5 Estimating Domain Means and Post-stratification with teacher example

5.1 Importing Data

```r
teacher <- read.csv ("data/college_teacher.csv")
head(teacher)
```

```
##   teacher gender
## 1       0      1
## 2       0      1
## 3       1      1
## 4       0      1
## 5       1      1
## 6       0      1
```

```r
table(teacher[,2:1])
```

```
##       teacher
## gender   0   1
##      1 156  84
##      2 120  40
```

```r
addmargins(table(teacher[,2:1]), 2)
```

```
##       teacher
## gender   0   1 Sum
##      1 156  84 240
##      2 120  40 160
```

```r
prop.table(table(teacher[,2:1]), margin = "gender")
```

```
##       teacher
## gender    0    1
##      1 0.65 0.35
##      2 0.75 0.25
```

5.2 Domain Summary

```r
n <- nrow (teacher)
yh <- tapply (teacher$teacher, INDEX = teacher$gender, FUN = mean); yh
```

```
##    1    2
## 0.35 0.25
```

```
sh <- tapply (teacher$teacher, INDEX = teacher$gender, FUN = sd); sh
```

```
##         1         2
## 0.4779664 0.4343722
```

```
nh_obs <- tapply (teacher$teacher, INDEX = teacher$gender, FUN = length)
prop_h_obs <- nh_obs/sum (nh_obs); prop_h_obs
```

```
##   1   2
## 0.6 0.4
```

```
data.frame(yh, sh, nh_obs, prop_h_obs)
```

```
##     yh        sh nh_obs prop_h_obs
## 1 0.35 0.4779664    240        0.6
## 2 0.25 0.4343722    160        0.4
```

5.3 Estimating Domain Means

```
Nh <- c(3000, 1000)
n <- nrow(teacher)
rbind(
    female=srs_domain_mean_est(subset(teacher, gender==1)$teacher, n=n, N=4000),
    male = srs_domain_mean_est(subset(teacher, gender==2)$teacher, n=n, N=4000)
)
```

```
##        Est.       S.E.    ci.low    ci.upp
## female 0.35 0.02926935 0.2923412 0.4076588
## male   0.25 0.03257792 0.1856587 0.3143413
```

5.4 Post-stratification Analysis

```
Nh <- c(3000, 1000)
prop_h_post <- Nh/sum (Nh); prop_h_post
```

```
## [1] 0.75 0.25
```

Create nh proportional to Nh, instead of using observed nh

```
nh_post <- Nh/sum (Nh) * n

## show the differences
data.frame(p_hat = yh, sh, nh_obs, prop_obs=prop_h_obs, nh_post, prop_post=prop_h_post)
```

```
##   p_hat        sh nh_obs prop_obs nh_post prop_post
## 1  0.35 0.4779664    240      0.6     300      0.75
## 2  0.25 0.4343722    160      0.4     100      0.25
```

```
## poststratification estimation of mean
str_mean_estimate (yh, sh, nh_post, Nh)
```

```
##       Est.       S.E.     ci.low     ci.upp
## 0.32500000 0.02217306 0.28154080 0.36845920
```

5.5 Comparing to the Analysis with SRS

```
srs_mean_est (teacher$teacher, N=sum(Nh))
```

```
##        Est.       S.E.     ci.low     ci.upp
## 0.31000000 0.02196545 0.26681751 0.35318249
```

6 Regression Estimation for the cherry.csv dataset

6.1 Importing data

```
cherry <- read.csv ("data/cherry.csv", header = T)
ydata <- cherry$volume
xdata <- cherry$diameter
t_diameters <- 41835
xbarU <- t_diameters/2967
N <- 2967
```

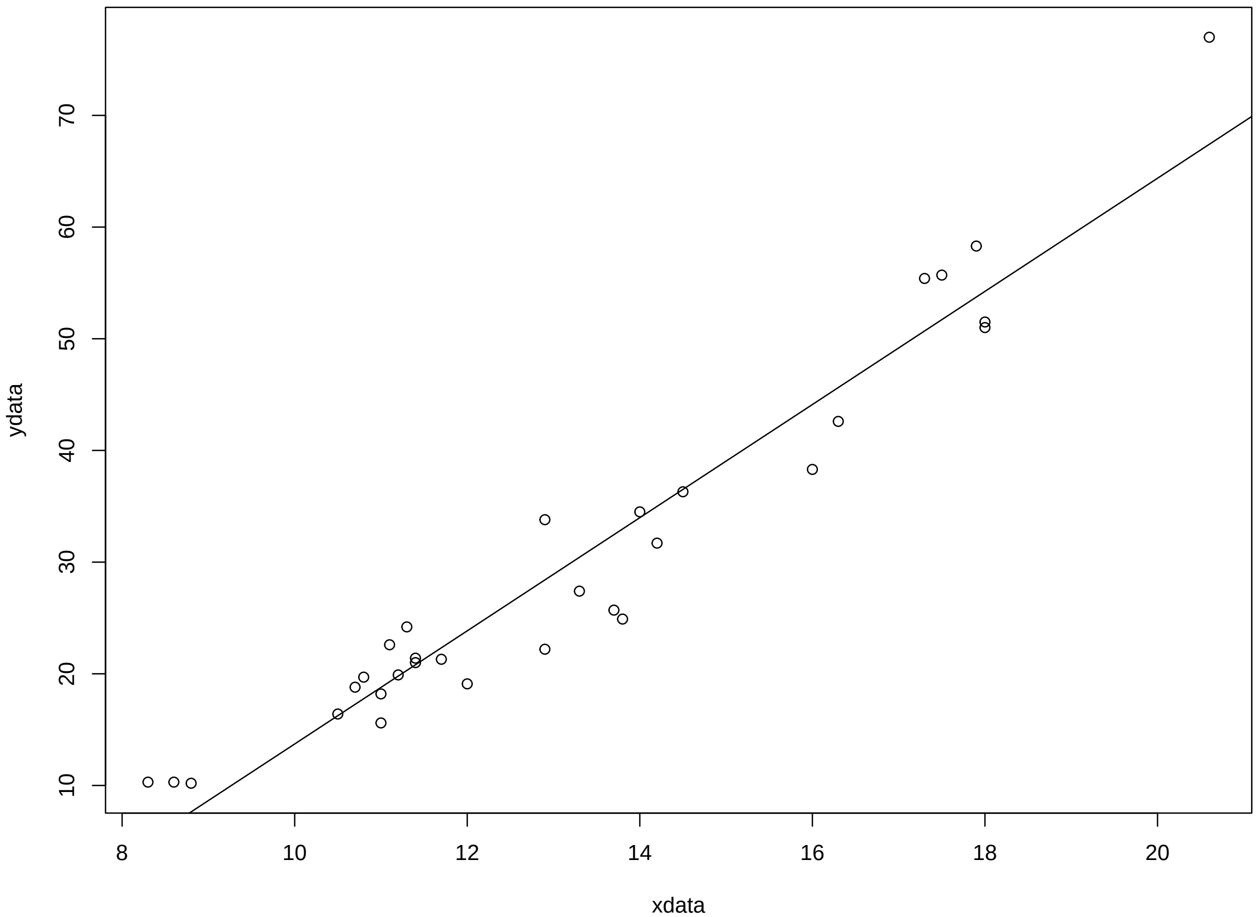
6.2 Step-by-step calculation

6.2.1 Fitting a linear regression model

```
n <- length (ydata)
lmfit <- lm (ydata ~ xdata)
summary (lmfit)
```

```
##
## Call:
## lm(formula = ydata ~ xdata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.065 -3.107  0.152  3.495  9.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
## xdata         5.0659     0.2474   20.48  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.252 on 29 degrees of freedom
## Multiple R-squared:  0.9353, Adjusted R-squared:  0.9331
## F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

```
plot (xdata, ydata)
abline (lmfit)
```




```r
Bhat <- lmfit$coefficients
efit <- ydata - (Bhat[1] + Bhat[2] * xdata)
data.frame (cherry, residual=efit) ## for visualization
```

```
##    diameter height volume   residual
## 1       8.3     70   10.3  5.1968508
## 2       8.6     65   10.3  3.6770939
## 3       8.8     63   10.2  2.5639226
## 4      10.5     72   16.4  0.1519667
## 5      10.7     81   18.8  1.5387954
## 6      10.8     83   19.7  1.9322098
## 7      11.0     66   15.6 -3.1809615
## 8      11.0     75   18.2 -0.5809615
## 9      11.1     80   22.6  3.3124528
## 10     11.2     75   19.9  0.1058672
## 11     11.3     79   24.2  3.8992815
## 12     11.4     76   21.0  0.1926959
## 13     11.4     76   21.4  0.5926959
## 14     11.7     69   21.3 -1.0270610
## 15     12.0     75   19.1 -4.7468179
## 16     12.9     74   22.2 -6.2060887
## 17     12.9     85   33.8  5.3939113
## 18     13.3     86   27.4 -3.0324313
## 19     13.7     71   25.7 -6.7587739
## 20     13.8     64   24.9 -8.0653595
```

```
## 21      14.0      78   34.5  0.5214692
## 22      14.2      80   31.7 -3.2917021
## 23      14.5      74   36.3 -0.2114590
## 24      16.0      72   38.3 -5.8102436
## 25      16.3      77   42.6 -3.0300006
## 26      17.3      81   55.4  4.7041430
## 27      17.5      82   55.7  3.9909717
## 28      17.9      80   58.3  4.5646292
## 29      18.0      80   51.5 -2.7419565
## 30      18.0      80   51.0 -3.2419565
## 31      20.6      87   77.0  9.5868168
```

```
SSe <- sum (efit^2) / (n - 2)
SSe
```

```
## [1] 18.0794
```

```
## SSe can be obtained directly from lm fitting output
anova(lmfit)
```

```
## Analysis of Variance Table
##
## Response: ydata
##           Df Sum Sq Mean Sq F value    Pr(>F)
## xdata      1 7581.8  7581.8  419.36 < 2.2e-16 ***
## Residuals 29  524.3    18.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6.2.2 Estimate the mean

```
yhat_reg <- Bhat[1] + Bhat[2] * xbarU
se_yhat_reg <- sqrt ((1-n/N) * SSe / n)
mem <- qt (0.975, df = n - 2) * se_yhat_reg
output <- c(yhat_reg, se_yhat_reg, yhat_reg - mem, yhat_reg + mem)
names (output) <- c("Est.", "S.E.", "ci.low", "ci.upp" )
output
```

```
##       Est.        S.E.     ci.low     ci.upp
## 34.4856287  0.7596795 32.9319097 36.0393476
```

6.2.3 Estimate the total

```
output * N
```

```
##       Est.        S.E.     ci.low     ci.upp
## 102318.860   2253.969  97708.976 106928.744
```

6.3 Regression estimation Using the function

6.3.1 Estimating the mean

```
reg_mean_volume <- srs_reg_est(ydata = cherry$volume, xdata = cherry$diameter,
                  xbarU=t_diameters/2967, N = 2967)
reg_mean_volume
```

```
##       Est.       S.E.     ci.low     ci.upp
## 34.4856287  0.7596795 32.9319097 36.0393476
```

6.3.2 Estimating the total

```r
t_diameters <- 41835
srs_reg_est(ydata = cherry$volume, xdata = cherry$diameter,
                 xbarU=t_diameters/2967, N = 2967, est.total = TRUE)
```

```
##       Est.       S.E.     ci.low     ci.upp
## 102318.860   2253.969  97708.976 106928.744
```

6.3.3 Percentage of Variance Reduction

```r
cat(1-(reg_mean_volume[2]/srs_mean_volume[2])^2)
```

```
## 0.9330895
```

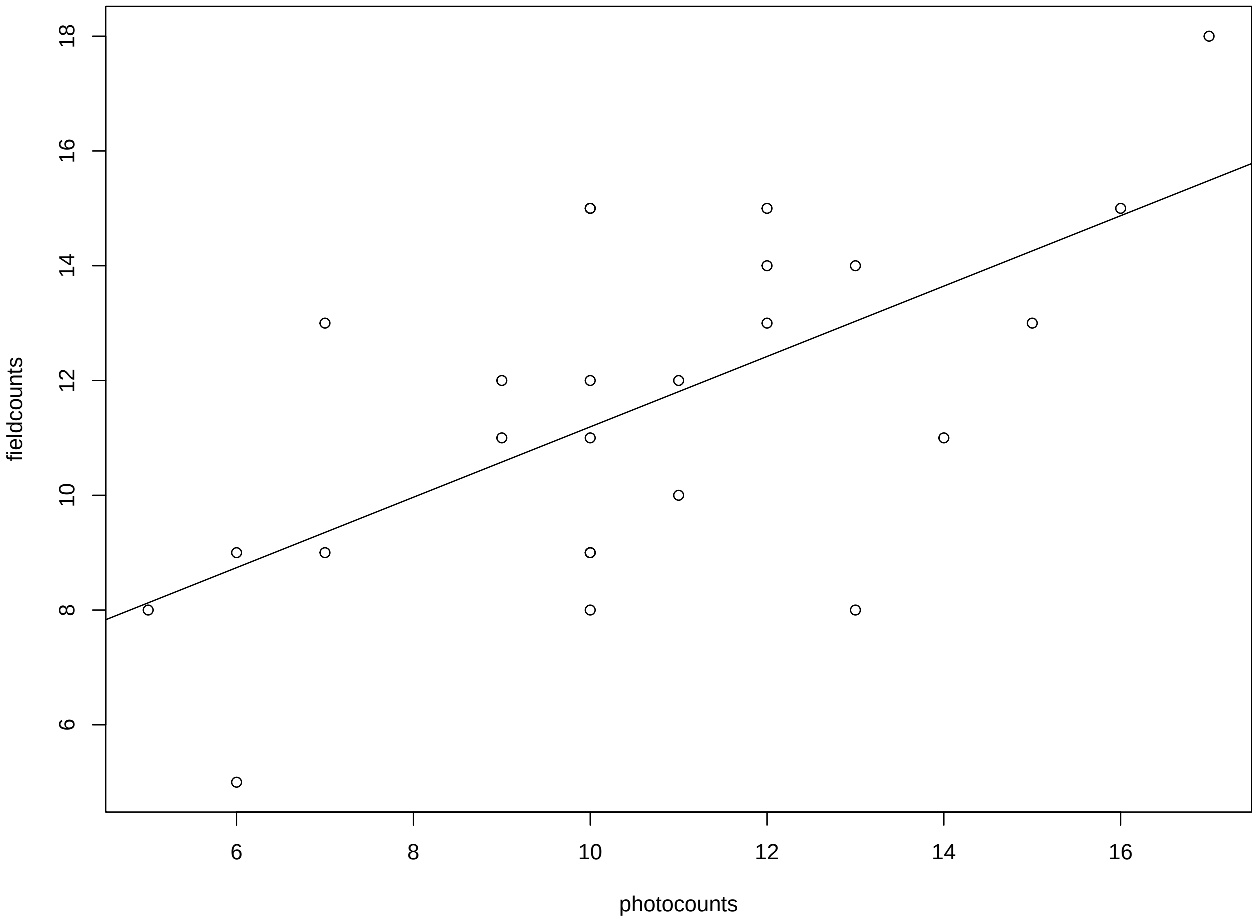
7 Regression estimation for photo counts of dead trees

To estimate the number of dead trees in an area, we divide the area into 100 square plots and count the number of dead trees on a photograph of each plot. Photo counts can be made quickly, but sometimes a tree is misclassified or not detected. So we select an SRS of 25 of the plots for field counts of dead trees. We know that the population mean number of dead trees per plot from the photo count is 11.3.

```r
photocounts <- c (10,12,7, 13,13, 6,17, 16, 15, 10, 14, 12, 10, 5,12, 10,
10, 9, 6, 11, 7, 9, 11, 10, 10)

fieldcounts <- c (15, 14, 9, 14, 8, 5, 18, 15, 13, 15, 11, 15, 12, 8, 13,
9, 11, 12, 9, 12, 13, 11, 10, 9, 8)


lmfit <- lm (fieldcounts ~ photocounts)
summary (lmfit)
```

```
##
## Call:
## lm(formula = fieldcounts ~ photocounts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0319 -1.8053  0.1947  1.4212  3.8080
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.0593     1.7635   2.869 0.008676 **
## photocounts   0.6133     0.1601   3.832 0.000854 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.406 on 23 degrees of freedom
## Multiple R-squared:  0.3896, Adjusted R-squared:  0.3631
## F-statistic: 14.68 on 1 and 23 DF,  p-value: 0.0008538
```

```
plot (photocounts, fieldcounts)
abline (lmfit)
```




```
# estimate for the mean number of dead trees per plot
srs_reg_est (ydata = fieldcounts, xdata = photocounts, xbarU = 11.3, N = 100)
```

```
##       Est.       S.E.     ci.low     ci.upp
## 11.9892920  0.4167579 11.1271625 12.8514215
```

```
# estimate for the total number of dead trees in the area
srs_reg_est (ydata = fieldcounts, xdata = photocounts, xbarU = 11.3, N = 100, est.total = TRUE)
```

```
##       Est.       S.E.     ci.low     ci.upp
## 1198.92920   41.67579 1112.71625 1285.14215
```

```
# compare to simple estimate
srs_mean_est (sdata = fieldcounts, N = 100)
```

```
##       Est.       S.E.     ci.low     ci.upp
## 11.5600000  0.5222069 10.4822180 12.6377820
```

```
# compare to ratio estimate
srs_ratio_est (ydata = fieldcounts, xdata = photocounts, N = 100) * 11.3
```

```
##       Est.       S.E.     ci.low     ci.upp
## 12.3233962  0.5121485 11.2663736 13.3804189
```