German Tanks

Oliver d'Pug

Simulate a Number of Battles

The function **tanks** simulates **n_reps** battles with **n_obs** serial numbers recorded for each battle. The argument **n_tanks** is the number of tanks that the Germans had. The argument **fixedest** is the expert's best guess.

```
tanks <- function (n_tanks = 84, n_obs = 5, n_reps = 100, fixedest = 87)
{
    temp <- as.matrix(rep(n_tanks, n_reps))
    temp <- apply(temp, 1, sample, size = n_obs)
    temp <- t(apply(temp, 2, tanks.ests, fixedest))
    temp
}
```

Compute Estimates

The function **tank.ests** runs on a sample of observed serial numbers. Each of the vector values is the result of a student supplied estimator. These need to be changed to reflect the class' ideas.

```
tanks.ests <- function (x = stop("Argument 'x' is missing"), fixedest = 125)
{
    n <- length(x)
    nodd <- n %% 2 == 1
    ests <- rep(NA, 16)
    xbar <- mean(x)
    xmedian <- median(x)
    xn <- max(x)
    xvar <- var(x)
    xstddev <- sqrt(xvar)
    x1 <- min(x)
    rng <- c(-1,1)%*%range(x)
    xsum <- sum(x)
    xnm1 <- sort(x)[n-1]
```

```r
    ests[1] <- 2 * xbar
    ests[2] <- 2 * xmedian
    ests[3] <- xbar + xmedian
    ests[4] <- sum(x[c(n-1,n)])
    ests[5] <- nodd * (xmedian + x[(n+1)/2 + 1]) + !nodd * (xmedian + x[n/2 + 1])
    ests[6] <- xn + xmedian
    ests[7] <- xbar + 2*xstddev
    #
      y = sort(x)
      z = c(NA,y[-n])
    ests[8] <- xn + mean(y-z, na.rm = TRUE)
      z = c(0,y[-n])
    ests[9] <- xn + mean(y-z)
      z = c(1,y[-n])
    ests[10] <- xn + mean(y-z)
    ests[11] <- (3*xn - x1)/2
    ests[12] <- (5*xn - xbar)/3
    ests[13] <- 3*xmedian + rng
    ests[14] <- xn
    ests[15] <- xn * ((n+1)/n) - 1
    ests[16] <- fixedest
    names(ests) <- c("2*xbar", "2*m", "xbar + m", "x[n]+x[n-1]", "m + next biggest",
                     "x[n] + m", "xbar + 2s", "x[n]+mean(x -lag(x))", "x[n]+mean(x -lag(x) w
                     "x[n]+mean(x -lag(x) w/ 1)", "(3x[n]-x[1])/2", "(5*x[n]-xbar)/3", "3m +
                     "x[n]", "UMVUE", fixedest)
    ests

    #ests[1] <- xn
    #ests[2] <- 2 * xbar
    #ests[3] <- xn + x1
    #ests[4] <- xn + rng/2
    #ests[5] <- xbar + xstddev
    #ests[6] <- 2 * xmedian
    #ests[7] <- xn + xnm1
    #ests[8] <- 2*xn - x1
    #ests[9] <- xsum
    #ests[10] <- xn * ((n+1)/n) - 1
    #ests[11] <- fixedest
    #names(ests) <- c("x[n]","2*xbar", "x[n]+x[1]", "x[n]+R/2", "xbar+s", "2*m",
    #                  "x[n]+x[n-1]","2x[n]-x[1]","sum(x)", "UMVUE",fixedest)
    #ests
}
```

Compute Descriptive Statistics

The function **tanks.descriptives** computes descriptive statistics for each of the estimators in **tanks.ests**.

```
tanks.descriptives <- function (n = 84, obs = 5, reps = 100, fixedest = 87)
{
    temp <- tanks(n, obs, reps, fixedest)
    means <- apply(temp, 2, mean)
    stddevs <- sqrt(apply(temp, 2, var))
    medians <- apply(temp, 2, median)
    bias <- means - n
    mse <- bias^2 + stddevs^2
    t(cbind(means, stddevs, medians, bias, mse))
}
```

Plot Estimates

The individual estimates computed for the samples from each battle can be plotted. This allows us to compare location and dispersion statistics — center and spread. **tanks.plots2** is intended to be an "improved" version of **tanks.plots**. Both plots use traditional **lattice** boxplots and there is a ggplot plot as well.

```
### Load the lattice package
p_load(lattice)

tanks.plots <- function (n = 84, obs = 5, reps = 100, fixedest = 87)
{
    temp <- tanks(n, obs, reps, fixedest)
    tanknames <- attributes(temp)$dimnames[[2]]
    dims <- dim(temp)
    temp <- as.vector(t(temp))
    temp <- cbind(temp, rep(1:dims[2], dims[1]))
    bwplot(factor(temp[, 2], labels = tanknames) ~
        temp[, 1], xlab = "Number of Tanks",
        ylab = "Estimator")
}

tanks.plots2 <- function (n = 84, obs = 5, reps = 100, fixedest = 87)
{
    temp <- tanks(n, obs, reps, fixedest)
    tanknames <- attributes(temp)$dimnames[[2]]
    dims <- dim(temp)
    temp <- as.vector(t(temp))
    temp <- cbind(temp, rep(1:dims[2], dims[1]))
```

```
    bwplot(factor(temp[, 2], labels = tanknames) ~
        temp[, 1], xlab = "Number of Tanks",
        ylab = "Estimator", panel = function (x ,
            y , vref = n, ... )
        {
            panel.bwplot(x, y)
            panel.abline(v = vref, lty = 2)
        }, vref = n)
}

p_load(ggplot2)
p_load(tidyverse)
tanks.plots3 <- function (n = 84, obs = 5, reps = 100, fixedest = 87)
{
    temp <- tanks(n, obs, reps, fixedest)
    tanknames <- attributes(temp)$dimnames[[2]]
    dims <- dim(temp)
    temp <- as.vector(t(temp))
    temp <- cbind(temp, rep(1:dims[2], dims[1]))
    temp <- as_tibble(temp)
    names(temp) <- c("Estimate", "Estimator")
    temp$Estimator <- factor(temp$Estimator)
    ggplot(temp, aes(x=Estimator, y=Estimate)) +
      geom_boxplot(alpha=0.7) +
      stat_summary(fun=mean, geom="point", shape=20, size=5, color="red", fill="red") +
      theme(legend.position="none") +
      scale_fill_brewer(palette="Set1") +
      geom_hline(yintercept = n, alpha = 0.5, color = "blue", lty = 1) +
      coord_flip()
}

tanks.plots4 <- function (n = 84, obs = 5, reps = 100, fixedest = 87)
{
    temp <- tanks(n, obs, reps, fixedest)
    temp <- melt(temp)
    names(temp) <- c("RowID","Estimator","Estimate")
    ggplot(temp, aes(x=Estimator, y=Estimate)) +
      geom_boxplot(alpha=0.7) +
      stat_summary(fun=mean, geom="point", shape=20, size=5, color="red", fill="red") +
      theme(legend.position="none") +
      scale_fill_brewer(palette="Set1") +
      geom_hline(yintercept = n, alpha = 0.5, color = "blue", lty = 1) +
      coord_flip()
}
```

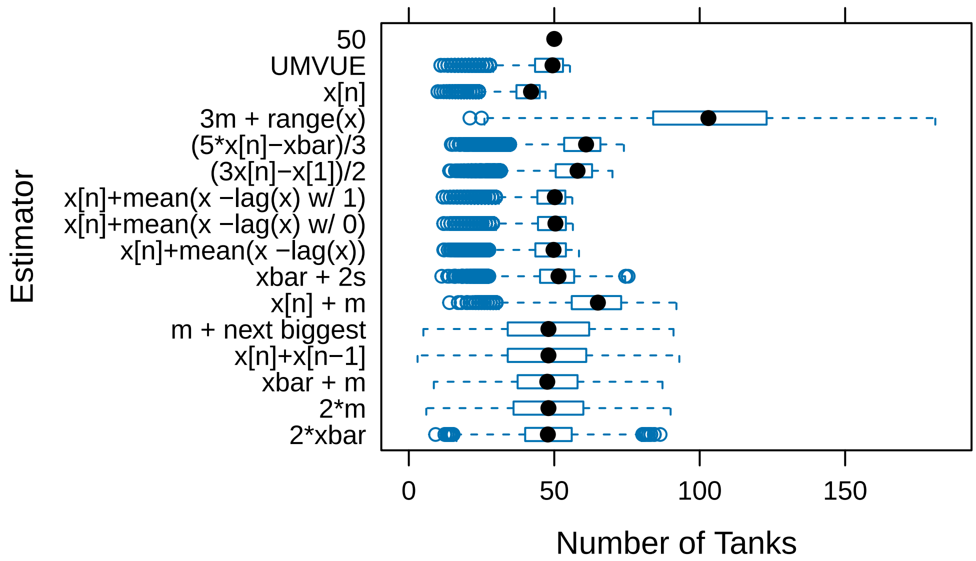
Compare our Estimators

The class' estimators can be compared using the functions defined above.

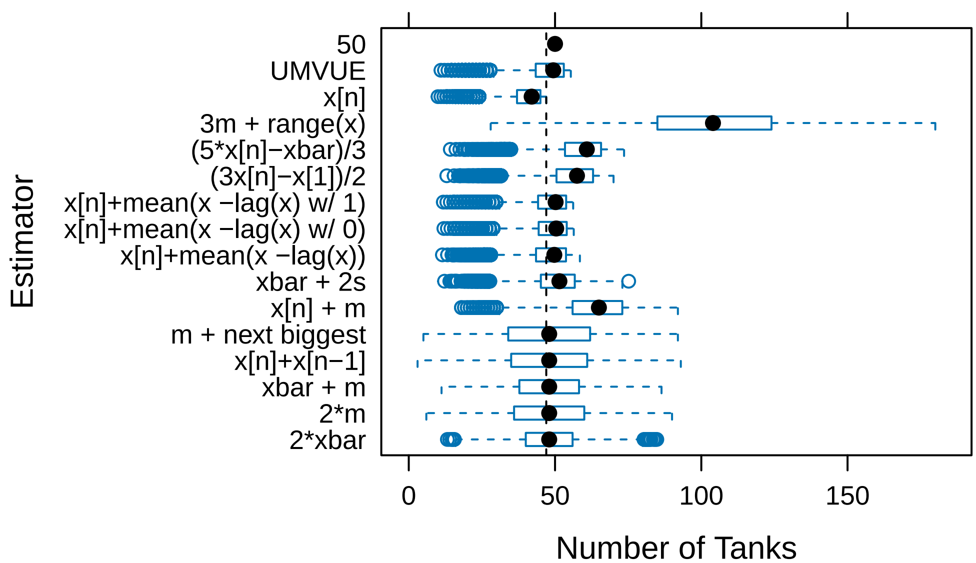
```r
### Compute descriptive stats
tanks.descriptives(n = 47, obs = 5, reps = 10000, fixedest = 50)
```

```
           2*xbar       2*m   xbar + m x[n]+x[n-1] m + next biggest  x[n] + m
means    48.02012  48.10260  48.06136     47.8870         48.05200  64.04400
stddevs  11.59744  16.95078  13.84946     19.0075         18.50449  12.62448
medians  48.00000  48.00000  48.00000     48.0000         48.00000  65.00000
bias      1.02012   1.10260   1.06136      0.8870          1.05200  17.04400
mse     135.54135 288.54473 192.93408    362.0719        343.52284 449.87534
        xbar + 2s x[n]+mean(x -lag(x)) x[n]+mean(x -lag(x) w/ 0)
means   50.532725            47.997775                  47.99124
stddevs  9.028678             7.676267                   7.52948
medians 51.446344            49.750000                  50.40000
bias     3.532725             0.997775                   0.99124
mse     93.997171            59.920636                  57.67562
        x[n]+mean(x -lag(x) w/ 1) (3x[n]-x[1])/2 (5*x[n]-xbar)/3 3m + range(x)
means                    47.79124       56.00285       58.651147     104.17420
stddevs                   7.52948        9.29459        9.301216      26.59443
medians                  50.20000       58.00000       60.933333     104.00000
bias                      0.79124        9.00285       11.651147      57.17420
mse                      57.31913      167.44071      222.261831    3976.15293
             x[n]    UMVUE 50
means   39.992700 46.99124 50
stddevs  6.274566  7.52948  0
medians 42.000000 49.40000 50
bias    -7.007300 -0.00876  3
mse     88.472437 56.69314  9
```

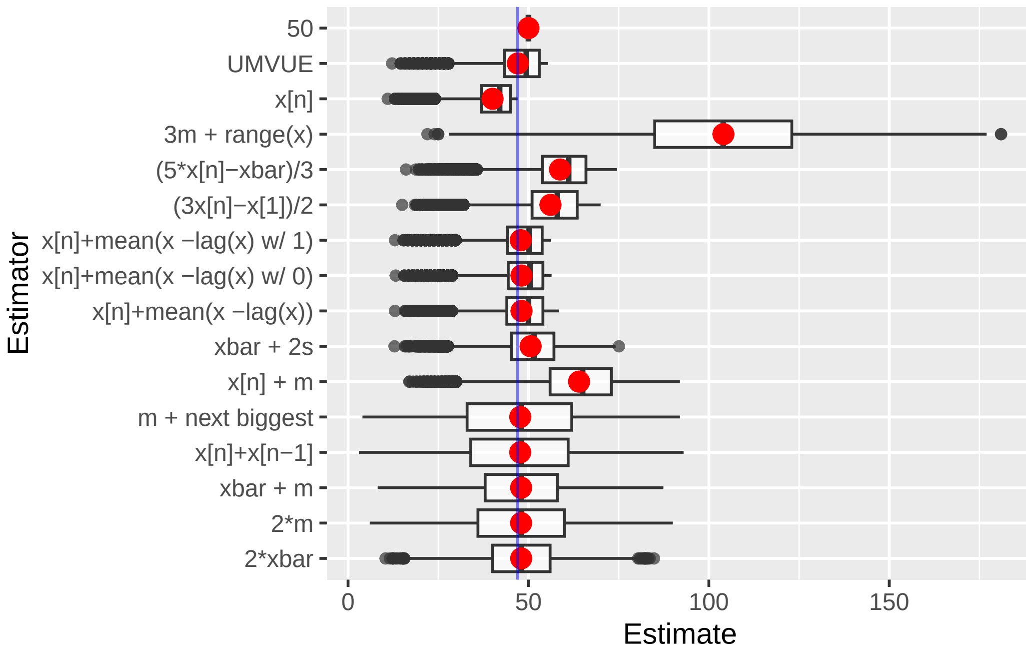
```r
### Plot the estimates from each of the estimators
tanks.plots(n = 47, obs = 5, reps = 10000, fixedest = 50)
```




```r
### Plot the estimates from each of the estimators
tanks.plots2(n = 47, obs = 5, reps = 10000, fixedest = 50)
```




```r
### Plot the estimates from each of the estimators
tanks.plots4(n = 47, obs = 5, reps = 10000, fixedest = 50)
```



Note that $\widehat{N} = X_{(n)}\frac{n+1}{n} - 1$ is UMVUE for N when the X_i are i.i.d. DU(1,N).