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]  
   
 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/ 0)",   
 "x[n]+mean(x -lag(x) w/ 1)", "(3x[n]-x[1])/2", "(5\*x[n]-xbar)/3", "3m + range(x)",   
 "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()  
}

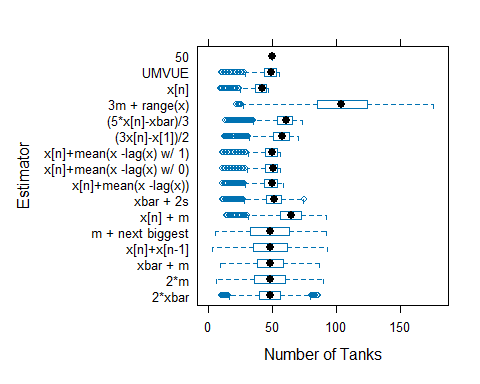
## Compare our Estimators

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

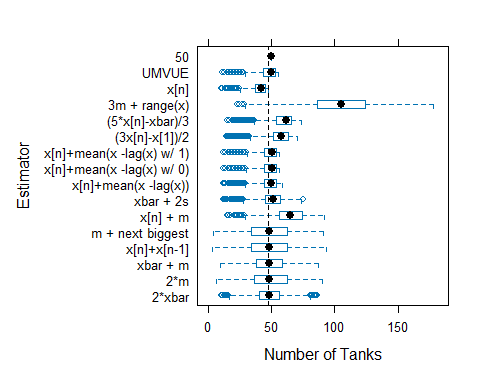
### Compute descriptive stats  
 tanks.descriptives(n = 47, obs = 5, reps = 10000, fixedest = 50)

2\*xbar 2\*m xbar + m x[n]+x[n-1] m + next biggest x[n] + m  
means 47.88520 47.87300 47.87910 47.65990 47.82310 63.85250  
stddevs 11.61528 17.02381 13.89358 18.96896 18.50423 12.77533  
medians 48.00000 48.00000 47.80000 48.00000 48.00000 65.00000  
bias 0.88520 0.87300 0.87910 0.65990 0.82310 16.85250  
mse 135.69839 290.57218 193.80443 360.25688 343.08414 447.21582  
 xbar + 2s x[n]+mean(x -lag(x)) x[n]+mean(x -lag(x) w/ 0)  
means 50.417969 47.898825 47.899200  
stddevs 9.190688 7.856860 7.699107  
medians 51.438429 49.750000 50.400000  
bias 3.417969 0.898825 0.899200  
mse 96.151254 62.538142 60.084808  
 x[n]+mean(x -lag(x) w/ 1) (3x[n]-x[1])/2 (5\*x[n]-xbar)/3 3m + range(x)  
means 47.699200 55.881650 58.54580 103.74080  
stddevs 7.699107 9.506414 9.52518 26.86245  
medians 50.200000 57.500000 60.93333 104.00000  
bias 0.699200 8.881650 11.54580 56.74080  
mse 59.765128 169.255612 224.03454 3941.10936  
 x[n] UMVUE 50  
means 39.916000 46.899200 50  
stddevs 6.415922 7.699107 0  
medians 42.000000 49.400000 50  
bias -7.084000 -0.100800 3  
mse 91.347116 59.286408 9

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



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



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



Note that is UMVUE for when the are i.i.d. DU(1,).