# Bootstrapping ziCPUE data

*Lucas A. Nell*

*March 31, 2016*

## Initial setup

The following packages are required:

```
library('dplyr')
library('readxl')
library('ggplot2')
library('boot')
library('broom')
library('RColorBrewer')
```

Read, clean, and add necessary columns to the table in `brett_example.xlsx`.

```
CPUEdf <- read_excel("brett_example.xlsx",1) %>%
  rename(Area = `Effort m^2`, Count = SumOfSPEC) %>%
  filter(Year %in% 1993:2013, !is.na(Year), Month == 10, !is.na(Month)) %>%
  select(Year, Month, Station, Count, Area) %>%
  mutate(CPUE = (Count/Area)*100) %>%
  arrange(Year)
```

## Bootstrap Resampling

For seamless bootstrapping with `dplyr`, the package `broom` provides the function `boostrap`. However, this function samples from all rows regardless of grouping. I've hacked the version of the function found here to sample within a given grouping as such: If, for example, you group by a variable called `type` and in the original sample, `types` 1, 2, and 3 have sample sizes of 10, 20, and 30, respectively. This function would randomly sample exactly 10 items from the original dataset (with replacement) that are of `type` 1, 20 of `type` 2, and 30 of `type` 3.

```
stratBootstrap <- function(df, m, group) {

  # Inner function to sample row numbers from an input vector of sample sizes
  # for each group
  stratSample <- function(group_n){
    ends <- cumsum(group_n)
    starts <- c(1, ends[-length(ends)] - 1)
    bootRows <- unlist(lapply(seq(length(ends)), function(i){
      sample(seq(starts[i], ends[i]), replace = TRUE)}))
    return(bootRows)
  }

  df <- df %>% arrange_(group)

  n <- nrow(df)
```

```
  group_n <- (df %>%
    group_by_(group) %>%
    summarize(n = n()))[['n']]

  attr(df, "indices") <- replicate(m, stratSample(group_n) - 1,
                                   simplify = FALSE)
  attr(df, "drop") <- TRUE
  attr(df, "group_sizes") <- rep(n, m)
  attr(df, "biggest_group_size") <- n
  attr(df, "labels") <- data.frame(replicate = 1:m)
  attr(df, "vars") <- list(quote(replicate))
  class(df) <- c("grouped_df", "tbl_df", "tbl", "data.frame")

  return(df)
}
```

Now for the actual bootstrapping. We'll do 10,000 simulations and calculate the 95% CI via the "percentile" method.

```
set.seed(9721)
CPUEboot <- CPUEdf %>%
  stratBootstrap(1e4, group = 'Year') %>%
  do(summarize(group_by(., Year), CPUE = mean(CPUE))) %>%
  # To strip attributes:
  as.data.frame %>% as.tbl

# Summary table
CPUEbootSumm <- CPUEboot %>%
  group_by(Year) %>%
  summarize(low = quantile(CPUE, probs = 0.025),
            mid = quantile(CPUE, probs = 0.5),
            high = quantile(CPUE, probs = 0.975))

CPUEbootSumm
```

```
## Source: local data frame [19 x 4]
##
##     Year         low         mid        high
##    (dbl)       (dbl)       (dbl)       (dbl)
## 1   1993  4.83832211  9.58010489 16.83885783
## 2   1994  6.99210643 15.83355396 26.44557596
## 3   1995  8.39890051 20.89424272 35.98938845
## 4   1996  1.31134895 24.82473582 61.64871335
## 5   1997  3.57124248  6.70801146 11.27221312
## 6   1999  1.65468194  4.36716710  7.85672566
## 7   2000  0.20093735  0.47319168  0.81233606
## 8   2001  0.41001564  0.80018388  1.28325638
## 9   2002  0.02824304  0.09308116  0.18038384
## 10  2003  0.00000000  0.02769080  0.06139001
## 11  2004  0.33760247  0.95487542  1.76591515
## 12  2005 12.99019720 26.96280502 47.85944003
## 13  2006  0.35316891  0.52012957  0.76636280
## 14  2007  2.74382176  5.31447841  8.78115557
## 15  2008  5.14752570  8.38709651 12.52172003
```

```
## 16  2009  6.21371226 10.32210104 15.49383447
## 17  2010  0.29392622  0.75528146  1.40376504
## 18  2012  0.00000000  0.00000000  0.00000000
## 19  2013  0.00000000  0.02905816  0.06399798
```

## Plotting

These are just some asthetic pieces that I often employ. They're included here in case others are interested.

```
# Minimal ggplot2 theme
plotTheme <- function(base_size = 10, base_family = 'Helvetica') {
  theme_minimal(base_size = base_size, base_family = base_family) %+replace%
    theme(
      strip.text = element_text(face = 'bold'),
      panel.grid.minor = element_blank(),
      panel.grid.major.x = element_blank(),
      panel.grid.major.y = element_line(color = 'gray50', size = 0.125, linetype = 3),
      panel.border = element_rect(fill = NA, color = "gray50"),
      axis.ticks = element_line(color = "gray50"),
      axis.ticks.length = unit(2, 'points'),
      legend.position = 'none'
    )
}

# Color palette for unique values in an input vector
myPalette <- function(inputVector){
  outPalette <- inputVector %>%
    unique %>%
    length %>%
    colorRampPalette(brewer.pal(8,  'Dark2'))(.)
  return(outPalette)
}
```

#### Base plot

All following plots are based off this base one. Having no `geoms`, it just stores the data and some aesthetics that we'll pass on to plots later.

```
bootPlot <- CPUEdf %>%
  ggplot(aes(x = factor(Year), y = CPUE, color = factor(Year))) +
  plotTheme() +
  scale_x_discrete('Year', breaks = seq(min(CPUEdf$Year), max(CPUEdf$Year))) +
  scale_color_manual(values = myPalette(CPUEdf$Year))
```

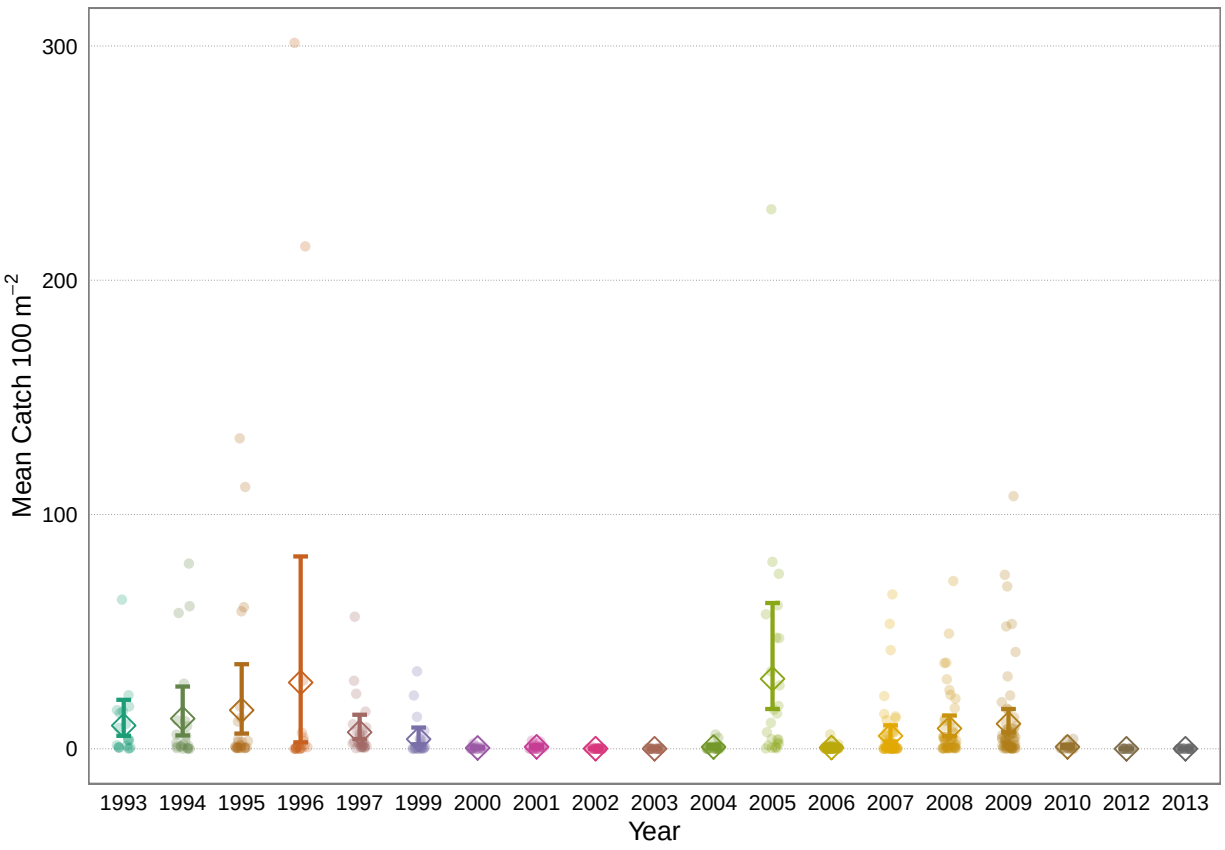
#### Bootstrapping via `boot` in `ggplot2`

To have better control of bootstrapping within `ggplot2` plots, the following function allows you to use the `boot` package for bootstrapping and CI calculations. The `ciMethod` parameter takes the following inputs, inherited from the `boot.ci` function: "norm","basic", "stud", "perc", or "bca". If parallel bootstrapping is desired, on Mac OSX, add `parallel = "multicore"` and on Windows add `parallel = "snow"` to the `boot` function call; also add `ncpus = x`, where `x` is the number of available cores.

```
ggBootCI <- function(inputData, numSims = 1e4, ciMethod = 'bca'){
  # List names in the `boot.ci` output for these methods differ from their input names
  if (ciMethod %in% c('norm', 'stud', 'perc')){
    outListMethod <- gsub('norm', 'normal', ciMethod) %>%
      gsub('stud', 'student', .) %>%
      gsub('perc', 'percent', .)
  } else {
    outListMethod <- ciMethod
  }
  # Inner function to compute mean and variance of the sampling distribution of the mean
  # for each bootstrap replication
  bootMean <- function(inputData, ind){
    m <- mean(inputData[ind])
    n <- length(ind)
    v <- var(inputData[ind]) / n
    return(c(m, v))
  }

  bootstraps <- boot(inputData, statistic = bootMean, R = numSims)

  # If all resamples return the same value, output df will simply be that value x3:
  if (diff(range(bootstraps$t)) == 0){
    result <- range(bootstraps$t)[1] %>%
      data.frame(ymin = ., y = ., ymax = .)
  } else { # If not, then run `boot.ci`:
    boot.ciOutput <- boot.ci(bootstraps, type = ciMethod)
    bootCI <- rev(boot.ciOutput[[outListMethod]])[c(2,1)]
    result <- data.frame(ymin = bootCI[1],
                         y = median(bootstraps$t),
                         ymax = bootCI[2])
  }

  return(result)
}
```

Now we'll employ the above function, using 1,000 simulations and calculating the 95% CI via the "bca" method: bias-corrected and accelerated bootstrapping.

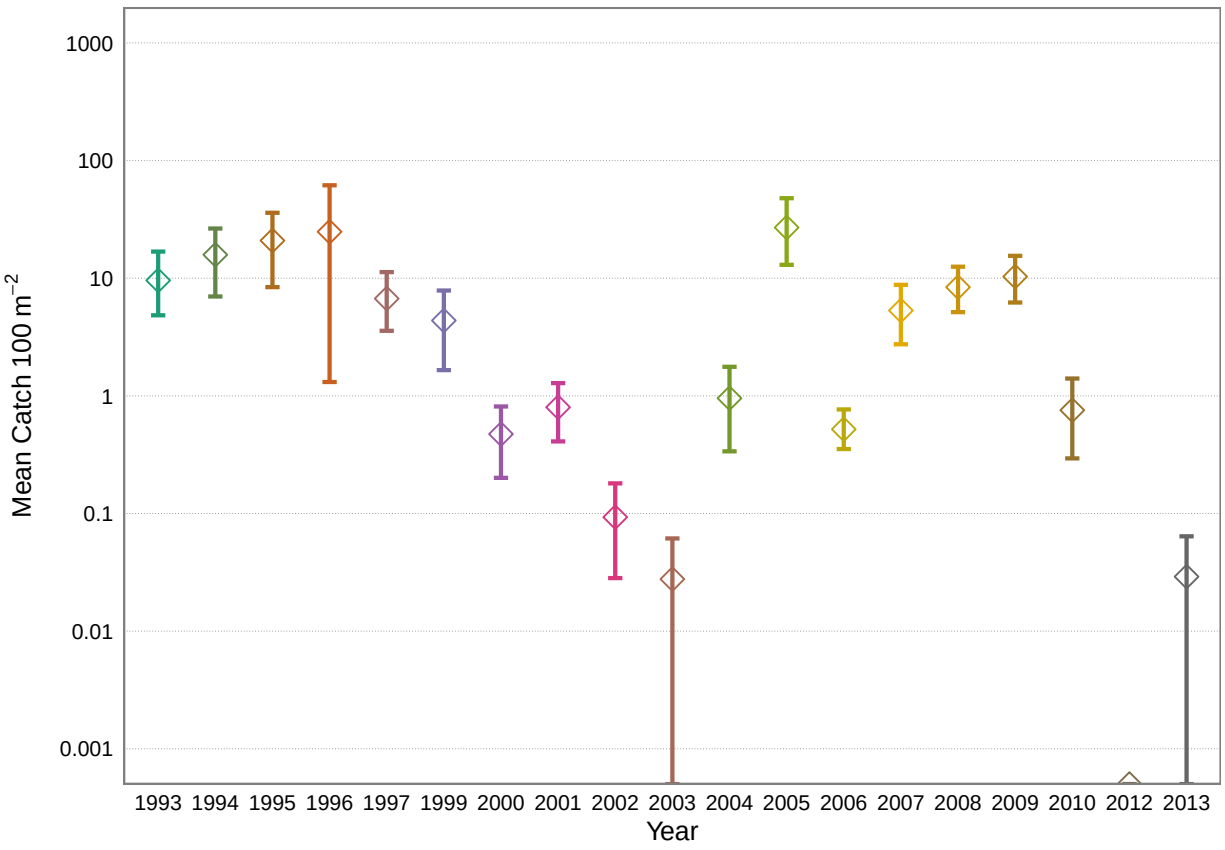
```
bootPlot +
  scale_y_continuous(expression(Mean ~ Catch ~ 100 ~ m^{-2})) +
  geom_point(position = position_jitter(width = 0.3, height = 0),
             alpha = 0.25, shape = 16) +
  stat_summary(fun.data = ggBootCI,
               fun.args = list(numSims = 1e3, ciMethod = 'bca'),
               geom = "errorbar", width = 0.25, size = 0.75) +
  stat_summary(fun.y = "mean", geom = "point", size = 3, shape = 23)
```




**Plot on $\log_{10}$ scale**

To create a plot on the $\log_{10}$ scale, we'll use the bootstrapping we did earlier via the `stratBootstrap` function, located in the `CPUEbootSumm` data frame. Trying to transform the y-axis while calculating bootstrapped CI on the fly proved a fruitless endeavor.

```
bootPlot +
  scale_y_continuous(expression(Mean ~ Catch ~ 100 ~ m^{-2}), trans = 'log10',
                     breaks = c(0.001, 0.01, 0.1, 1, 10, 100, 1000),
                     limits = c(0.001, 1000),
                     labels = function(n){format(n, scientific = FALSE,
                                                 drop0trailing = TRUE)}) +
  geom_errorbar(data = CPUEbootSumm, aes(ymin = low, y = mid, ymax = high,
                                         x = factor(Year), color = factor(Year)),
                width = 0.25, size = 0.75) +
  geom_point(data = CPUEbootSumm, aes(y = mid),
             shape = 23, size = 3)
```


1000
100
10
1
0.1
0.01
0.001
Mean Catch 100 m$^{-2}$
1993
1994
1995
1996
1997
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2012
2013
Year