Data Extraction Evaluation

Matthew Leonawicz

December 24, 2014

1 R code: Statistical sampling for spatial data extraction

1.1 Setup

1.1.1 Packages

```
no.knit <- if ("knitr" %in% names(sessionInfo()$otherPkgs)) FALSE else TRUE
library(raster)
library(microbenchmark)
library(ggplot2)
library(reshape2)
```

1.1.2 Additional setup

```
setwd("C:/github/DataExtraction/data")
# testfile <-
# 'Z:/Base_Data/ALFRESCO_formatted/ALFRESCO_Master_Dataset/ALFRESCO_Model_Input_Datasets/AK_CAN_Inputs/C
testfile <- "tas_mean_C_AR5_GFDL-CM3_rcp60_01_2062.tif"

r <- readAll(raster(testfile))  # force into memory so I/O time does not confound extraction time
v <- getValues(r)  # numeric vector
dat.ind <- Which(!is.na(r), cells = T)
d <- v[dat.ind]  # numeric vector of data values (drop NAs)
nd <- length(dat.ind)
```

1.1.3 Examine population mean

```
# continue indexing v since this is how it will tend to occur in practice
# take mean of all cells
mean(v, na.rm = T)
# take mean of only the data cells
mean(v[dat.ind])
# take mean of only the data cells using sum and known length
sum(v[dat.ind])/nd
# take mean of data cells with .Internal
.Internal(mean(v[dat.ind]))
# take mean of data cells with .Internal sum, known length
.Primitive("sum")(v[dat.ind])/nd
```

1.1.4 Sampling setup

```
mean.pop <- sum(d)/nd
mean.pop.out <- round(mean.pop, 1)  # round to one decimal place for temperature data
discrete.out <- round(seq(mean.pop, mean.pop + 0.4, by = 0.1) - 0.2, 1)
# median.pop <- median(d) median.pop.out <- round(median.pop, 1) # round to
# one decimal place for temperature data
bounds.round <- mean.pop.out + c(-0.05, 0.05)  # within rounding distance of the rounded population mean
bounds.signif <- mean.pop + c(-0.1, 0.1)  # bounds on the unrounded population mean at the significant d
# Use sample mean
n <- 1e+05
m <- 100
keep <- seq(1000, n, by = 1000)  # burn in and thin to facilitate visualization

set.seed(47)
d.sub <- replicate(m, sample(d, n, replace = F))
means <- data.frame(1:n, (1:n)/nd, apply(d.sub, 2, function(x, n) cumsum(x)/(1:n),
    n = n))
names(means) <- c("Size", "Percent_Sample", paste0("Sample_", c(paste0(0, 0:9),
    10:m)[1:m]))
means <- means[keep, ]
p <- data.frame(Size = keep, Percent_Sample = keep/nd, P_value = 1 - apply(means,
    1, function(x) length(which(x >= bounds.signif[1] & x < bounds.signif[2]))/length(x)))
means <- melt(means, id.vars = c("Size", "Percent_Sample"), variable.name = c("Sample"),
    value.name = "Mean")
p <- melt(p, id.vars = c("Size", "Percent_Sample"), variable.name = c("Type"),
    value.name = "Pval")

clr <- c(Samples = "gray", `Pop. mean +/- 1 sig. fig.` = "blue", `Rounded pop. mean` = "red",
    `Possible rounded values` = "black")
```

1.2 Sample mean by size

```
if (no.knit) png("../plots/mean_by_size.png", width = 2000, height = 1600, res = 200)
g <- ggplot(means, aes(x = Percent_Sample, y = Mean, group = Sample)) + theme_bw() +
    geom_line(aes(colour = "Samples")) + geom_hline(aes(yintercept = d, colour = "Rounded pop. mean"),
    data = data.frame(d = mean.pop.out)) + geom_hline(aes(yintercept = d, colour = c("Pop. mean +/- 1 si
    data = data.frame(d = bounds.signif)) + geom_hline(aes(yintercept = d, colour = "Possible rounded va
    , data = data.frame(d = discrete.out[2:5]), linetype = 2) + scale_colour_manual(name = "hello",
    values = clr) + theme(legend.position = "bottom") + labs(title = "Sample mean ~ sample size")
print(g)
if (no.knit) dev.off()
```

1.3 Significant figures

```
if (no.knit) png("../plots/pvalue_sigdig.png", width = 2000, height = 2000,
    res = 200)
g <- ggplot(p, aes(x = Percent_Sample, y = Pval, group = Type, colour = Type)) +
    theme_bw() + geom_line(colour = "black")
g <- g + geom_hline(aes(yintercept = 0.05, linetype = "P-value = 0.05"), colour = "red",
```

```
    linetype = 2) + annotate("text", 0.005, 0.05 * 1.2, label = "P-value = 0.05",
    size = 3)
g <- g + labs(title = "P(abs(sample mean - pop. mean) > 1 sig. digit | sample size)")
print(g)
if (no.knit) dev.off()
```

1.4 Compute time by sample size

```
# compute time for means for different sample size
s005pct <- d.sub[1:round((nrow(d.sub) * 0.05)), 1]
s010pct <- d.sub[1:round((nrow(d.sub) * 0.1)), 1]
s025pct <- d.sub[1:round((nrow(d.sub) * 0.25)), 1]
s100pct <- d.sub[, 1]
```

1.5 Benchmarks

```
mb3 <- microbenchmark(sum(s005pct)/length(s005pct), sum(s010pct)/length(s010pct),
    sum(s025pct)/length(s025pct), sum(s100pct)/length(s100pct), times = 10000)
mb3

if (no.knit) png("../plots/benchmark3.png", width = 2000, height = 1600, res = 200)
autoplot(mb3) + theme_bw() + labs(title = "Compute time for mean by sample size",
    y = "Function")
if (no.knit) dev.off()
```

```
mb4 <- microbenchmark(sum(s005pct)/length(s005pct), mean(v, na.rm = T), sum(d)/length(d),
    mean(s005pct), times = 100)
mb4
med <- print(mb4)$median
names(med) <- print(mb4)$expr
med <- med[c(1, 4:2)]

if (no.knit) png("../plots/benchmark4.png", width = 2000, height = 1600, res = 200)
autoplot(mb4) + theme_bw() + labs(title = "Compute time for mean | sampling and/or function change",
    y = "Function")
if (no.knit) dev.off()
```

1.6 Benchmarks: median compute times

```
if (no.knit) png("../plots/benchmark4medians.png", width = 2000, height = 1000,
    res = 200)
ggplot(data.frame(x = names(med), y = med), aes(x = reorder(x, 1:length(x),
    function(z) z), y = y, colour = x)) + geom_bar(stat = "identity", size = 0.5,
    width = 0.9) + theme_bw() + theme(legend.position = "none", axis.ticks = element_blank(),
    axis.text.y = element_blank()) + scale_colour_manual(values = c("gray",
    "dodgerblue", "orange", "purple")[c(3, 1, 2, 4)]) + labs(title = "Compute time for mean | sampling a
    x = "Function +/- sampling", y = "Time [microseconds]") + annotate("text",
```

```
    x = (1:4) - 0.2, y = 20000, label = names(med), size = 4, hjust = 0, colour = c(rep("black",
        3), "white")) + coord_flip()
if (no.knit) dev.off()
```

```
if (no.knit) png("../plots/benchmark4medians2.png", width = 2000, height = 1000,
    res = 200)
ggplot(data.frame(x = names(med)[-4], y = med[-4]), aes(x = reorder(x, 1:length(x),
    function(z) z), y = y, colour = x)) + geom_bar(stat = "identity", size = 0.5,
    width = 0.9) + theme_bw() + theme(legend.position = "none", axis.ticks = element_blank(),
    axis.text.y = element_blank()) + scale_colour_manual(values = c("dodgerblue",
    "orange", "purple")[c(2, 1, 3)]) + labs(title = "Compute time for mean | sampling and/or function ch
    x = "Function +/- sampling", y = "Time [microseconds]") + annotate("text",
    x = (1:(4 - 1)) - 0.2, y = 125, label = names(med)[-4], size = 4, hjust = 0,
    colour = c(rep("black", 3 - 1), "white")) + coord_flip()
if (no.knit) dev.off()
```

1.7 Final benchmarks

```
mb <- microbenchmark(mean(v, na.rm = T), mean(v[dat.ind]), sum(v[dat.ind])/nd,
    .Internal(mean(v[dat.ind])), .Primitive("sum")(v[dat.ind])/nd, times = 100)
mb
if (no.knit) png("../plots/benchmark1.png", width = 2000, height = 1600, res = 200)
autoplot(mb) + theme_bw() + labs(title = "Comparisons of time to index data and compute mean",
    y = "Function")
if (no.knit) dev.off()
```

```
mb2 <- microbenchmark(mean(v[dat.ind]), sum(v[dat.ind])/nd, mean(d), sum(d)/nd,
    times = 1000)
mb2
if (no.knit) png("../plots/benchmark2.png", width = 2000, height = 1600, res = 200)
autoplot(mb2) + theme_bw() + labs(title = "Comparisons of time to compute mean",
    y = "Function")
if (no.knit) dev.off()
```