BACS HW 4

Question 1) Let's reexamine how to *standardize* data: subtract the mean of a vector from all its values, and divide this difference by the standard deviation to get a vector of standardized values.

a) Create a normal distribution (mean=940, sd=190) and standardize it (let's call it `rnorm_std`)

i) What should we expect the mean and standard deviation of `rnorm_std` to be, and why?

ii) What should the distribution (shape) of `rnorm_std` look like, and *why*?

iii) What do we generally call distributions that are normal and standardized?

b) Create a standardized version of `minday` discussed in question 3 (let's call it `minday_std`)

i) What should we expect the mean and standard deviation of `minday_std` to be, and *why*?

ii) What should the distribution of `minday_std` look like compared to `minday`, and *why*?

Question 2) Install the compstatslib package from Github (see class notes) and *run the `plot_sample_ci()` function that simulates samples drawn randomly from a population. Each sample is a horizontal line with a dark band for its 95% CI, and a lighter band for its 99% CI, and a dot for its mean. The population mean is a vertical black line. Samples whose 95% CI includes the population mean are blue, and others are red.*

a) Simulate 100 samples (each of size 100), from a normally distributed population of 10,000:

```
plot_sample_ci(num_samples = 100, sample_size = 100, pop_size=10000,
               distr_func=rnorm, mean=20, sd=3)
```

i) How many samples do we *expect* to NOT include the population mean in its 95% CI?

ii) How many samples do we *expect* to NOT include the population mean in their 99% CI?

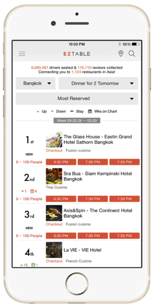
b) Rerun the previous simulation with the same number of samples, but larger sample size (`sample_size=300`):

i) Now that the size of each sample has increased, do we expect their 95% and 99% CI to become wider or narrower than before?

ii) This time, how many samples (out of the 100) would we *expect* to NOT include the population mean in its 95% CI?

c) If we ran the above two examples (a and b) using a uniformly distributed population (specify parameter `distr_func=runif` for `plot_sample_ci`), how do you *expect* your answers to (a) and (b) to change, and *why*?

Question 3) The company EZTABLE has an online restaurant reservation platform that is accessible by mobile and web. Imagine that EZTABLE would like to start a promotion for new members to make their bookings earlier in the day.



We have a *sample* of data about their new members, in particular the date and time for which they make their first ever booking (i.e., the booked time for the restaurant) using the EZTABLE platform. Here is some sample code to explore the data:

```
bookings <- read.table("first_bookings_datetime_sample.txt", header=TRUE)
bookings$datetime[1:9]
  [1] 4/16/2014 17:30  1/11/2014 20:00  3/24/2013 12:00 ...
  18416 Levels: 1/1/2012 17:15 1/1/2012 19:00 ... 9/9/2014 19:30
hours  <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$hour
mins   <- as.POSIXlt(bookings$datetime, format="%m/%d/%Y %H:%M")$min
minday <- hours*60 + mins
plot(density(minday), main="Minute (of the day) of first ever booking", col="blue", lwd=2)
```

a) What is the "average" booking time for new members making their first restaurant booking? (use `minday`, which is the absolute minute of the day from 0-1440)

- i) Use traditional statistical methods to estimate the *population mean* of `minday`, its *standard error*, and the *95% confidence interval* (CI) of the sampling means
- ii) Bootstrap to produce 2000 new samples from the original sample
- iii) Visualize the means of the 2000 bootstrapped samples
- iv) Estimate the *95% CI of the bootstrapped means* using the quantile function

b) By what time of day, have half the new members of the day already arrived at their restaurant?

- i) Estimate the *median* of `minday`
- ii) Visualize the *medians* of the 2000 bootstrapped samples
- iii) Estimate the *95% CI of the bootstrapped medians* using the quantile function

Here's a hint if you are having trouble figuring out how to do bootstrapping for this week's assignment, you may start by considering this function:

```
compute_sample_mean <- function(sample0) {
  resample <- sample(sample0, length(sample0), replace=TRUE)
  mean(resample)
}
```

If you run `compute_sample_mean(minday)` then you will get a new sample's mean from `minday`. You could then use the `replicate()` function to run this function 2000 times for bootstrapping.