simulation-study

Tony Ni

1/20/2021

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(Metrics) #package to help calculate mse
```

```
## Warning: package 'Metrics' was built under R version 4.0.3
```

```
library(NADA) #package with implementation of many methods
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'NADA'
```

```
## The following object is masked from 'package:stats':
##
##     cor
```

```
library(survival)
```

First, a function to help us generate our desired dataset, where users can specify the mean, sd, sample size (of each run), and censoring rate:

```
#function will generate a vector of numbers from the exponential
#distribution and censor them at the given rate
#function will take in arguments for 1) samplesize, 2) rate,
#3) censoring rate

generateEXP <- function(sampsize, r, censrate){
  true.value <- rexp(sampsize, rate = r)

  uncensored_df <- as.data.frame(true.value) %>%
    arrange(true.value)

  censored_df <- uncensored_df %>% #take the head(%) of data to be censored
    slice_head(n=nrow(uncensored_df)*censrate) %>%
    mutate(censored = TRUE)
```

```
  #full join original df and sliced df
  return_df <- full_join(uncensored_df, censored_df, by = "true.value")

  #replace NAs with FALSE
  return_df$censored <- replace_na(return_df$censored, replace = FALSE)

  return(return_df)
}
```

```
#function will generate a vector of numbers from the Weibull
#distribution and censor them at the given rate
#function will take in arguments for 1) samplesize, 2) rate,
#3) censoring rate

generateW <- function(sampsize, sh, sc, censrate){
  true.value <- rweibull(sampsize, shape = sh, scale = sc)

  uncensored_df <- as.data.frame(true.value) %>%
    arrange(true.value)

  censored_df <- uncensored_df %>% #take the head(%) of data to be censored
    slice_head(n=nrow(uncensored_df)*censrate) %>%
    mutate(censored = TRUE)

  #full join original df and sliced df
  return_df <- full_join(uncensored_df, censored_df, by = "true.value")

  #replace NAs with FALSE
  return_df$censored <- replace_na(return_df$censored, replace = FALSE)

  return(return_df)
}
```

What do we want our code to do?

We first need to generate simulated data sets with the following combinations of censoring rates and sample sizes: (censoring rates: 10, 30, 50, sample sizes: 10, 100, 1000)

For each censoring/sample size pair, we create a simulated dataset with those specifications and run the substitution method with a specified number of iterations (`iterations`). For each sample, we will obtain the true mean (when considering all values to be uncensored), the estimated mean (when considering the uncensored values and the imputed values), alongside the MSE of each dataset

the average mse is ultimately what we are interested in

```
iterations <- 1000 #number of iterations
censvalues <- c(0.10, 0.30, 0.50)
sampsizes <- c(10, 100, 1000)

df.tall <- data.frame(prop_cens = numeric(),
                      samplesize = numeric(),
                      iteration = numeric(),
                      method = character(),
                      true_mean = numeric(),
                      mean_complete = numeric(),
                      mean_method = numeric(),
```

```
                        true_sd = numeric(),
                        SE_complete = numeric(),
                        SE_method = numeric())
```

#Log-normal

```
## `summarise()` regrouping output by 'prop_cens', 'samplesize', 'method' (override with `.groups` argu
```

```
knitr::kable(df.ln)
```

prop_cens	samplesize	method	Avg_Mean	Bias	Variance	MSE
0.1	10	km	1.0092355	0.0092355	0.0228855	0.0229479
0.1	10	mle	0.9973831	-0.0026169	0.0227648	0.0227488
0.1	10	ros	0.9907731	-0.0092269	0.0219042	0.0219674
0.1	10	substitution	0.9809893	-0.0190107	0.0220108	0.0223502
0.1	100	km	1.0076570	0.0076570	0.0026283	0.0026843
0.1	100	mle	0.9982195	-0.0017805	0.0026295	0.0026301
0.1	100	ros	0.9985785	-0.0014215	0.0025839	0.0025833
0.1	100	substitution	0.9829313	-0.0170687	0.0025508	0.0028396
0.1	1000	km	1.0091259	0.0091259	0.0002448	0.0003278
0.1	1000	mle	0.9999640	-0.0000360	0.0002437	0.0002435
0.1	1000	ros	1.0012686	0.0012686	0.0002393	0.0002407
0.1	1000	substitution	0.9846354	-0.0153646	0.0002375	0.0004733
0.3	10	km	1.0501547	0.0501547	0.0275360	0.0300240
0.3	10	mle	0.9676480	-0.0323520	0.0243140	0.0253363
0.3	10	ros	0.9761882	-0.0238118	0.0237444	0.0242877
0.3	10	substitution	0.9371382	-0.0628618	0.0229813	0.0269099
0.3	100	km	1.0490450	0.0490450	0.0028810	0.0052836
0.3	100	mle	0.9737186	-0.0262814	0.0025227	0.0032109
0.3	100	ros	0.9999872	-0.0000128	0.0025943	0.0025917
0.3	100	substitution	0.9435821	-0.0564179	0.0024110	0.0055916
0.3	1000	km	1.0493913	0.0493913	0.0002613	0.0027005
0.3	1000	mle	0.9744596	-0.0255404	0.0002353	0.0008874
0.3	1000	ros	1.0037697	0.0037697	0.0002382	0.0002521
0.3	1000	substitution	0.9445950	-0.0554050	0.0002231	0.0032926
0.5	10	km	1.1480598	0.1480598	0.0371823	0.0590668
0.5	10	mle	0.9081187	-0.0918813	0.0223562	0.0307760
0.5	10	ros	0.9674303	-0.0325697	0.0281042	0.0291369
0.5	10	substitution	0.9077127	-0.0922873	0.0236772	0.0321705
0.5	100	km	1.1286680	0.1286680	0.0037229	0.0202747
0.5	100	mle	0.9035147	-0.0964853	0.0024048	0.0117118
0.5	100	ros	0.9960112	-0.0039888	0.0031582	0.0031710
0.5	100	substitution	0.9038640	-0.0961360	0.0025035	0.0117431
0.5	1000	km	1.1288505	0.1288505	0.0003475	0.0169496
0.5	1000	mle	0.9046487	-0.0953513	0.0002273	0.0093189
0.5	1000	ros	1.0051493	0.0051493	0.0002895	0.0003157
0.5	1000	substitution	0.9051573	-0.0948427	0.0002349	0.0092298