Randomly Perturb Some Parameter Value with Varying Magnitudes

Fan Wang

2021-07-06

Contents

1 Randomly Perturbing a Parameter

Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.1 Perturbation Normally with Log-Normal Magnitude Scaling Value

During estimation, we have some starting estimation parameter value. We want to do multi-start estimation by perturbing initial starting points. The perturbing process follows the rules specified below, implemented in the function below.

1. Select from 0 to 1, 0 closest to the existing parameter value, 1 very far from it.
2. Log normal distribution with 1st quartile = 0.185, and mu of normal = 0. The value from (1) correspond to a cumulative mass point of this log normal distribution.
3. Draw a value randomly from standard normal
4. Transform the randomly drawn value to current parameter scale with inverse z-score, the resulting value is the parameter of interest.

Test and implement the ideas above.

```r
# Step 0
ar_fl_original_param <- c(-100, -10, -1, -0.1, 0, 0.1, 1, 10, 100)
ar_fl_original_param <- c(-10, -0.1, 0, 0.1, 10)
# Step 1
ar_zero_to_one_select <- seq(1e-3, 1 - 1e-3, length.out = 11)
# Step 2
# Assume mean of normal = 0, with sdlog = 2, 25th percentile is 0.185
fl_sdlog <- 2.5
fl_p25_logn <- qlnorm(0.25, meanlog = 0, sdlog = fl_sdlog)
# Step 3
# random draw, for now fix at positive number, which means to "randomly" expand
fl_draw_znorm <- 1
# Step 4
mt_collect <- matrix(
  data = NA,
  nrow = length(ar_zero_to_one_select),
  ncol = length(ar_fl_original_param)
)
it_col_ctr <- 0
for (fl_original_param in ar_fl_original_param) {
  it_col_ctr <- it_col_ctr + 1
  # inverse z-score
  ar_logn_coef_of_var <- qlnorm(1-ar_zero_to_one_select, meanlog = 0, sdlog = fl_sdlog)
  ar_logn_sd <- fl_original_param/ar_logn_coef_of_var
  ar_param_perturbed <- fl_draw_znorm * ar_logn_sd + fl_original_param
  # fill matrix
  mt_collect[, it_col_ctr] <- (ar_param_perturbed)
}
# Out to table
ar_st_varnames <- c("zero_one_scalar", paste0("ori_val=", ar_fl_original_param))
# Combine to tibble, add name col1, col2, etc.
tb_collect <- as_tibble(cbind(ar_zero_to_one_select, mt_collect)) %>%
  rename_all(~ c(ar_st_varnames))
# Display
kable(tb_collect) %>% kable_styling_fc()
```

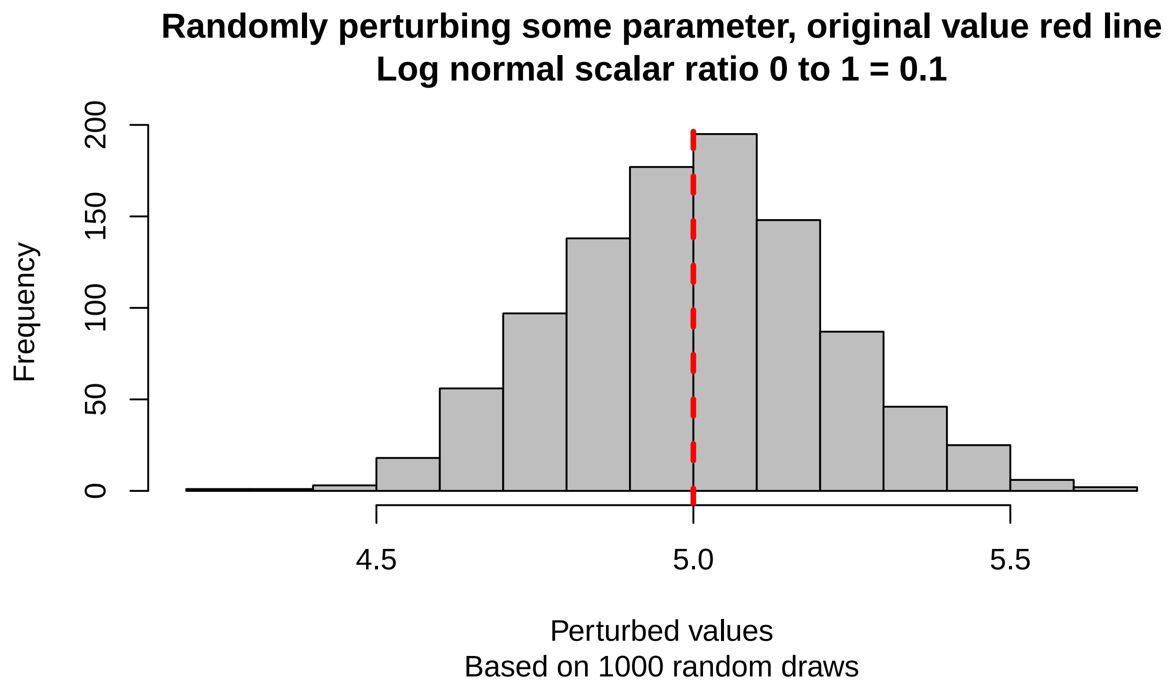
zero_one_scalar	ori_val=-10	ori_val=-0.1	ori_val=0	ori_val=0.1	ori_val=10
0.0010	-10.00441	-0.1000441	0	0.1000441	10.00441
0.1008	-10.41068	-0.1041068	0	0.1041068	10.41068
0.2006	-11.22616	-0.1122616	0	0.1122616	11.22616
0.3004	-12.70326	-0.1270326	0	0.1270326	12.70326
0.4002	-15.31489	-0.1531489	0	0.1531489	15.31489
0.5000	-20.00000	-0.2000000	0	0.2000000	20.00000
0.5998	-28.81508	-0.2881508	0	0.2881508	28.81508
0.6996	-46.99235	-0.4699235	0	0.4699235	46.99235
0.7994	-91.55561	-0.9155561	0	0.9155561	91.55561
0.8992	-253.49612	-2.5349612	0	2.5349612	253.49612
0.9990	-22665.67969	-226.6567969	0	226.6567969	22665.67969

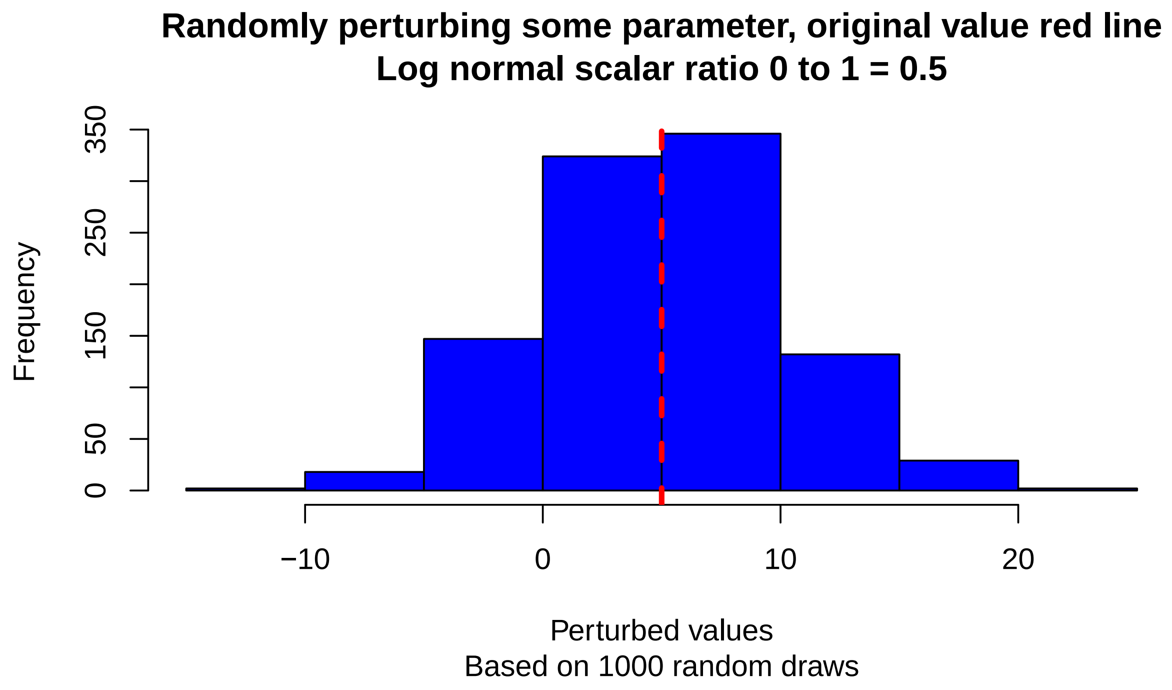
Implement the above idea with a function.

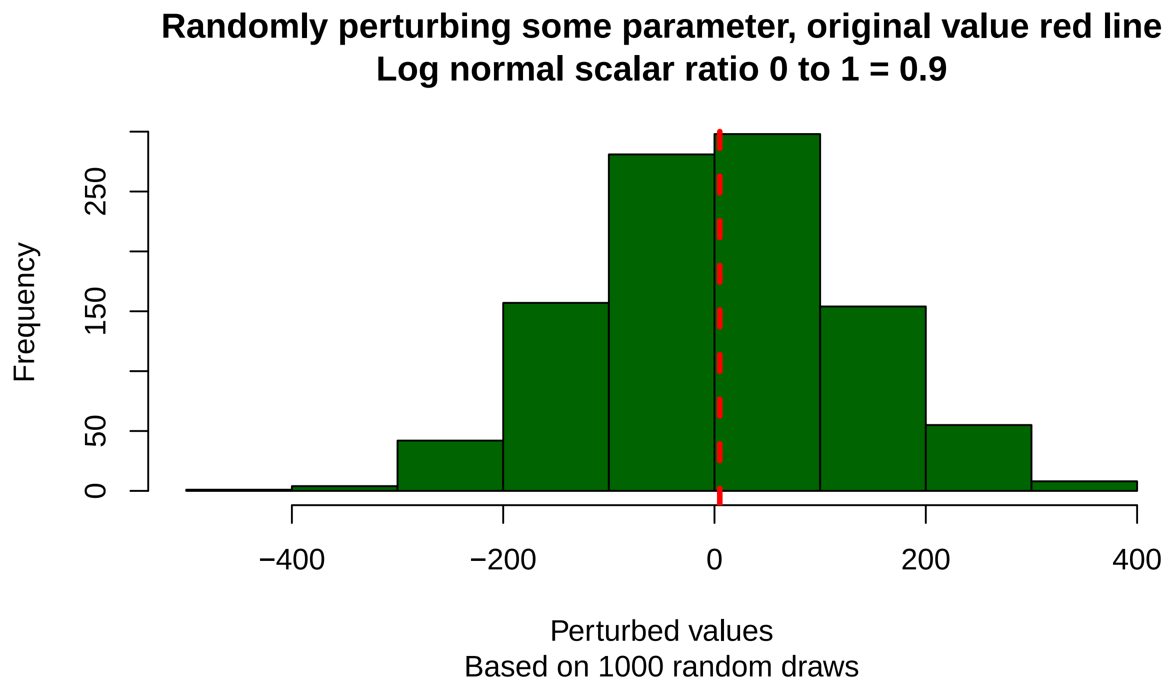
```r
ffi_param_logn_perturber <- function(
  param_original=5, scaler_0t1=0.5,
  it_rand_seed=1, fl_sdlog=2.5, fl_min_quantile=1e-3) {
  #' @param float original current parameter value to be perturbed
  #' @param scaler_0t1 float, must be between 0 to 1, 0 means don't scale much, 1 mean a lot
  #' @param it_rand_seed integer randomly sperturbing seed
  #' @param fl_sdlog float the sdlog parameter
  #' @param fl_min_quantile float minimum quantile point (and 1 - max) to allow for selecting 0 and 1 f

  # Draw randomly
  set.seed(it_rand_seed)
  fl_draw_znorm <- rnorm(1)
  # logn value at quantile
  scaler_0t1 <- scaler_0t1*(1-fl_min_quantile*2) + fl_min_quantile
  logn_coef_of_var <- qlnorm(1-scaler_0t1, meanlog = 0, sdlog = fl_sdlog)
  # Coefficient of variation
  ar_logn_sd <- param_original/logn_coef_of_var
  # Invert z-score
  param_perturbed <- fl_draw_znorm * ar_logn_sd + param_original

  return(param_perturbed)
}
```

Test the function with differently randomly drawn parameters, and visualize.

```r
# Start image
# Loop over different scalars
param_original <- 5
ar_scaler_0t1 <- c(0.1, 0.5, 0.9)
ar_color_strs <- c("gray", "blue", "darkgreen")
it_scalar_ctr <- 0
for (scaler_0t1 in ar_scaler_0t1) {
  it_scalar_ctr <- it_scalar_ctr + 1

  # Generate differently perturbed parameters
  ar_param_perturbed <- c()
  for (it_rand_seed in seq(1, 1000)) {
    param_perturbed <- ffi_param_logn_perturber(
      param_original, scaler_0t1, it_rand_seed=it_rand_seed)
    ar_param_perturbed <- c(ar_param_perturbed, param_perturbed)
  }

  # Line through origin
  par(mfrow = c(1, 1))
  hist(ar_param_perturbed, col = ar_color_strs[it_scalar_ctr],
       ylab = "", xlab = "", main = "")
  # Original parameter line
  abline(
    v = param_original,
    col = "red", lwd = 3, lty = 2
  )
  # Titles
  title(
    main = paste0(
      "Randomly perturbing some parameter, original value red line\n",
      "Log normal scalar ratio 0 to 1 = ", scaler_0t1
    ),
    sub = paste0(
      "Based on 1000 random draws"
    ),
    xlab = "Perturbed values", ylab = "Frequency"
  )
}
```


Randomly perturbing some parameter, original value red line
Log normal scalar ratio 0 to 1 = 0.1
Frequency
0
50
100
150
200
4.5
5.0
5.5
Perturbed values
Based on 1000 random draws


Randomly perturbing some parameter, original value red line
Log normal scalar ratio 0 to 1 = 0.5
Frequency
0
50
150
250
350
−10
0
10
20
Perturbed values
Based on 1000 random draws


Randomly perturbing some parameter, original value red line
Log normal scalar ratio 0 to 1 = 0.9
Frequency
0
50
150
250
−400
−200
0
200
400
Perturbed values
Based on 1000 random draws