Robu_Wild

Zhipeng

3/10/2017

robu_wild() gets the hypothesis test result (p-value) of coefficients in a meta-regression model with "Wild bootstrap" algorithm. This function is developed based on function robu(). It has identical arguments to robu() and three additional ones, Replication_num, coef_index and six_pts.

Replication_num indicates the number of bootstrap replicatioins to be performed. The default value is 1000 means it will generate 1000 bootstrap sample sets for estimation.

coef_index indicates which coefficient's p-value will be estimated by Wild bootstrap. By default, it is 1, which means the p-value of the first coefficient in the model is estimated by wild-bootstrap.

six_pts indicates whether 2-point distributioin or 6-point distribution is used to perturb the residuals.

```
robu_Wild    <-function(formula, data, studynum,var.eff.size, userweights,
                        modelweights = c("CORR", "HIER"), rho = 0.8,
                        small = TRUE, Replication_num =1000, coef_index=1, six_pts = F, ...) {
```

The first 500 lines code is COPIED from function robu(). That part code does exactly the same work as robu() does. I omitted that part code.

Wild bootstrap algorithm starts from here.

Firstly, I extracted some necassary information from the results I get from above. For example, covariate matrices, residuals, etc.

```
  ##############################################################################################

  ###Data needed to generate bootstrap samples
  Study_id <- X.full[,1]
  X_matrx <- Xreg
  Y_matrx <- data.full$effect.size
  Y_hat_matrx <- data.full$pred.r
  coef_est <- b.r
  residual_matrx <- data.full$e.r
  residual_by_group <- by(data = residual_matrx, INDICES = Study_id,FUN = function(x){x})
  group_num <- length(residual_by_group)
  outcome_name <- as.character(ml[[2]])
```

Secondly, I created a function to generate a bootstrap sample set from the restricted model. At first, the bootstrap sample set is generated from full model. However, it does not work very well, so I changed to restricted model.

This function is called Get_Bootdata(), and I generate a list of bootstrap sample sets by it.

```
  ###Function to generate bootstrap samples

  Get_Bootdata <- function(){
    if(six_pts == F){

      RandomVector_list <-as.list(rbinom(n = group_num,size = 1,0.5)*2 -1)

    }else{
```

```
    Sixth_points <- c(- sqrt(3/2), -1, -sqrt(1/2), sqrt(1/2), 1, sqrt(3/2))
    RandomVector_list <- Sixth_points[ceiling(runif(n =group_num) *6)]
  }


  Boot_residual_by_group <- Map(f = function(residual, randomVec){residual*randomVec},
                                residual_by_group, RandomVector_list)

  Boot_residual <- unlist(Boot_residual_by_group)

  Boot_b <- b.r
  Boot_b[coef_index] <- 0
  Boot_Y <- Xreg %*% Boot_b + Boot_residual
  Boot_data <- data
  Boot_data[outcome_name] <-Boot_Y
  return(Boot_data)
}
#Get Bootstrap data list
Boot_data_list <- replicate(n = Replication_num, expr = Get_Bootdata(),simplify = F)
```

Thirdly, I created a function called Get_BootResult() to get t-score values from each boostrap sample set. Eventually, the t-score values are saved in a vector called Boot_t_val_vec.

Finally, the wild bootstrap p-value estimation is got by comparing the original t-score and the bootstarp t-scores.

```
#Function to calculate bootstrap result
Get_BootResult <- function(Boot_data){
  cl[[1]] <-  quote(robu)
  cl$data <-quote(Boot_data)
  new_result <- eval(cl)
  return(new_result)
}
#Get bootstrap result list.
BootResult_list <- lapply(X = Boot_data_list,FUN = Get_BootResult)

Boot_t_val_list <- lapply(X = BootResult_list, FUN = function(x){x$reg_table["t"][coef_index,]})

Boot_t_val_vec <-unlist(Boot_t_val_list)

p_val_Wild <- sum(abs(Boot_t_val_vec) >abs(t[coef_index,]))/Replication_num

res <- list(data.full = data.full, X.full = X.full, reg_table = reg_table,
            mod_label = mod_label, mod_notice = mod_notice, modelweights =
              modelweights, mod_info = mod_info, user_weighting =
              user_weighting, ml = ml, cl = cl, N = N, M = M, k = k,
            k_list = k_list, p = p, X = X, y = y, Xreg = Xreg, b.r = b.r,
            VR.r = VR.r, dfs = dfs, small = small, data = data, labels =
              labels, study_orig_id = study_orig_id)

class(res) <- "robu"
res
return(list(CRVE_result = res,
            Wild_boot_result_list = BootResult_list,
            Wild_boot_t_val_vec = Boot_t_val_vec,
```

```
              Wild_boot_p_val = p_val_Wild))
}
```

The output of this function is a list contains the regular robu() output of the original data, robu() outputs of each bootstrap sets, the t-value vector and the Wild bootstrap p-value.