Introducing the PrevCorr package

cheungngo

26 February 2021

This package is developed to allow high-throughput statistical calculation for prevalence-correlate studies. Many useful functions were included, and users can produce hundreds clumpsy confusion tables at a time, do hundreds of multivariate logistic regressions at a time, or hundreds of tests for multicollinearity at a time, etc. With the use of this package, the time for statistical analysis would be much much reduced, and the results would be readily reproducible and repeatable.

Preparing the environment

```
### Installing the package

install.packages("devtools") # devtools required to install packages from github
library(devtools)

devtools::install_github("cheungngo/PrevCorr") # installing the "PrevCorr" package
```

```
### Loading the library

library(PrevCorr)
```

Format of dataframe required

```
### Loading the data
# Data not included in the package

# separate files for dependent and independent variables
library(readr)
dependent_var <- read_csv("~/dependent_var.csv")
masterdata_A <- read_csv("~/masterdata_A.csv")
masterdata_B <- read_csv("~/masterdata_B.csv")
```

```
### Format of the dataframe for dependent variables

# Need to follow the rules to let the functions work
# each row for one subject
# for binary outcome, 0 for no; 1 for yes (i.e. 0 for no condition, 1 for condition present)
# functions written in this package are mostly for binary outcome, please contact author for
 other possible variations

dependent_var[1:10,c(1, 14)] # a portion of dataframe
```

```
## # A tibble: 10 x 2
##    Dx_ISSM_PE_before Dx_PEDT_11
##                <dbl>      <dbl>
##  1                 0          0
##  2                 1          0
##  3                 0          1
##  4                 1          0
##  5                 0          0
##  6                 0          1
##  7                 0          0
##  8                 0          0
##  9                 0          1
## 10                 0          0
```

```
### Format of the dataframe for independent variables

# Need to follow the rules to let the functions work
# each row for one subject

# for categorical variables (i.e. education, welfare, marital, etc), just use numbers to repr
esent each category; if binary => 1 and 2
# for continuous variables (i.e. age, sex_no_per_mo, sex_partner_no, etc), can simply input t
he numbers
# inputing "0" represents no entry, but it might be regarded as an extra category for categor
ical variables; for continuous variables, we have functions to include or exclude those "0"s
 as sometimes "0" means zero but not no entry

masterdata_A[1:10,c(14, 16:17, 13, 18:19)] # a portion of dataframe
```

```
## # A tibble: 10 x 6
##    education welfare marital   age sex_no_per_mo sex_partner_no
##        <dbl>   <dbl>   <dbl> <dbl>         <dbl>          <dbl>
##  1         4       1       2    34             3              1
##  2         1       1       3    65             1              1
##  3         3       3       3    59             0              1
##  4         3       1       2    56             2              1
##  5         2       2       2    35             0              1
##  6         1       2       3    46             0              0
##  7         4       1       1    48             2              1
##  8         4       1       1    37             0              0
##  9         2       3       2    63             0              1
## 10         3       1       2    62             0              1
```

Useful functions from this package

```
### You can view the documentation of all functions in the package using the following codes

help(package = "PrevCorr")
```

```
### Getting the column number

# First you need to get the column numbers for your concerned variables
# you can simply visualize using names(#concerned dataframe)

names(dependent_var) # this serves an example
```

```
##  [1] "Dx_ISSM_PE_before"       "Dx_ISSM_PE_after"
##  [3] "Dx_ISSM_lifelong_before" "Dx_ISSM_lifelong_after"
##  [5] "Dx_ISSM_acquired_before" "Dx_ISSM_acquired_after"
##  [7] "subjective_before"       "subjective_after"
##  [9] "DSMV_before"             "DSMV_after"
## [11] "Dx_PLED_1"               "Dx_PLED_2"
## [13] "PEDT_total"              "Dx_PEDT_11"
## [15] "Dx_PEDT_10"              "Dx_PEDT_9"
## [17] "IELT"                    "concealment"
```

```
### Serial Shapiro-Wilk test

# Shapiro-Wilk test is performed to see if the variables were normally distributed, thus to g
uide subsequent tests (i.e. t-test or Wilcoxon)

ind = c(13, 55:57, 19, 18, 97, 98) # these numbers represents the column number of the concer
ned variable in the dataframe

serial_shapiro(masterdata_A, ind) # this gives all the results for the concerned variables; n
ote that "masterdata_A" is the dataframe concerned
```

```
##      [,1]             [,2]    [,3]
## [1,] "age"            "0.955" "1.54507135039505e-06"
## [2,] "BW"             "0.968" "5.34221961533313e-05"
## [3,] "BH"             "0.995" "0.674126537443528"
## [4,] "BMI"            "0.975" "0.000476755032356344"
## [5,] "sex_partner_no" "0.325" "3.08884728641776e-28"
## [6,] "sex_no_per_mo"  "0.675" "9.8097104515497e-21"
## [7,] "PHQ9_total"     "0.926" "2.79825819495604e-09"
## [8,] "GAD7_total"     "0.94"  "4.76833338612357e-08"
```

```
# serial_shapiro(data, cols)
# data : the dataframe concerned
# cols : the columns of the variable in the dataframe concerned
```

```
### Frequency of occurence for multiple variables

# This is to calculate the frequency of occurences of many categorical variables at a time

indb = c(3:5,2) # these numbers represents the column number of the concerned variable in the
dataframe; of course there could be more

serial_foo(masterdata_B, indb)
```

```
##       [,1]                     [,2]
##  [1,] "PSY_comorbidity_type_0" "203 (88.65%) "
##  [2,] "PSY_comorbidity_type_1" "10 (4.37%) "
##  [3,] "PSY_comorbidity_type_2" "2 (0.87%) "
##  [4,] "PSY_comorbidity_type_3" "4 (1.75%) "
##  [5,] "PSY_comorbidity_type_4" "4 (1.75%) "
##  [6,] "PSY_comorbidity_type_5" "6 (2.62%) "
##  [7,] "Antidepressant_type_0"  "28 (12.23%) "
##  [8,] "Antidepressant_type_1"  "101 (44.1%) "
##  [9,] "Antidepressant_type_2"  "60 (26.2%) "
## [10,] "Antidepressant_type_3"  "40 (17.47%) "
## [11,] "SSRI_type_0"            "87 (37.99%) "
## [12,] "SSRI_type_1"            "29 (12.66%) "
## [13,] "SSRI_type_2"            "55 (24.02%) "
## [14,] "SSRI_type_3"            "21 (9.17%) "
## [15,] "SSRI_type_4"            "15 (6.55%) "
## [16,] "SSRI_type_5"            "21 (9.17%) "
## [17,] "SSRI_type_6"            "1 (0.44%) "
## [18,] "PSY_comorbidity_bool_1" "204 (89.08%) "
## [19,] "PSY_comorbidity_bool_2" "25 (10.92%) "
```

```
# serial_foo(data, cols)
# data : the dataframe concerned
# cols : the columns of the variable in the dataframe concerned
```

```
### Median and IQR

# This is to calculate the median and IQR for continuous variables (that were not of normal d
istributions)

ind = c(13, 55:57, 19, 18, 97, 98) # these numbers represents the column number of the concer
ned variable in the dataframe

serial_miqr(masterdata_A, ind)
```

```
##      [,1]             [,2]
## [1,] "age"            "53 (15)"
## [2,] "BW"             "72.4 (15.3)"
## [3,] "BH"             "1.723 (0.083)"
## [4,] "BMI"            "24.2 (4.5)"
## [5,] "sex_partner_no" "1 (0)"
## [6,] "sex_no_per_mo"  "2 (3)"
## [7,] "PHQ9_total"     "7 (9)"
## [8,] "GAD7_total"     "7 (9)"
```

```
# serial_miqr(data, cols)
# data : the dataframe concerned
# cols : the columns of the variable in the dataframe concerned
```

```
### A series of confusion tables

# This is to produce the confusion tables for categorical variables

ind = c(14:17) # these numbers represents the column number of the concerned variable in the
 dataframe

serial_tab(dependent_var, # the dataframe for dependent variables
           masterdata_A, # the dataframe for independent variables
           1, # 1 being the column number of the dependent variable concerned
           ind)
```

```
##              0              1
## education_1  "62 (29.11%)"  "8 (50%)"
## education_2  "78 (36.62%)"  "5 (31.25%)"
## education_3  "25 (11.74%)"  "1 (6.25%)"
## education_4  "48 (22.54%)"  "2 (12.5%)"
## employment_1 "4 (1.88%)"    "0 (0%)"
## employment_2 "131 (61.5%)"  "9 (56.25%)"
## employment_3 "34 (15.96%)"  "5 (31.25%)"
## employment_4 "44 (20.66%)"  "2 (12.5%)"
## welfare_1    "160 (75.12%)" "8 (50%)"
## welfare_2    "12 (5.63%)"   "3 (18.75%)"
## welfare_3    "34 (15.96%)"  "3 (18.75%)"
## welfare_4    "7 (3.29%)"    "2 (12.5%)"
## marital_1    "52 (24.41%)"  "4 (25%)"
## marital_2    "135 (63.38%)" "7 (43.75%)"
## marital_3    "24 (11.27%)"  "5 (31.25%)"
## marital_4    "2 (0.94%)"    "0 (0%)"
```

```
### A series of fisher exact tests

# Many fisher exact tests at a time

ind = c(14:17) # these numbers represents the column number of the concerned variable in the
 dataframe

serial_fisher(dependent_var, # the dataframe for dependent variables
              masterdata_A, # the dataframe for independent variables
              1, # 1 being the column number of the dependent variable concerned
              ind)
```

```
##            p-value grouped by Dx_ISSM_PE_before
## education                                 0.449
## employment                                0.492
## welfare                                   0.024
## marital                                   0.127
```

```
### Note that we also have functions for chi-square tests! Similar way of use
### serial_chisq()
```

```
### Serial MWW (wilcoxon) tests

# Many wilcoxon tests at a time

ind = c(13, 55:57, 19, 18, 97, 98) # these numbers represents the column number of the concer
ned variable in the dataframe

serial_wilcox(dependent_var, # the dataframe for dependent variables
              masterdata_A, # the dataframe for independent variables
              1, # 1 being the column number of the dependent variable concerned
              ind)
```

```
##      [,1]                 [,2]             [,3]     [,4]
## [1,] "Dx_ISSM_PE_before" "age"            "1240"   "0.069604954431562"
## [2,] "Dx_ISSM_PE_before" "BW"             "1309.5" "0.123159569561898"
## [3,] "Dx_ISSM_PE_before" "BH"             "1573.5" "0.610981699301078"
## [4,] "Dx_ISSM_PE_before" "BMI"            "1412.5" "0.254825608109225"
## [5,] "Dx_ISSM_PE_before" "sex_partner_no" "1771.5" "0.654566107361792"
## [6,] "Dx_ISSM_PE_before" "sex_no_per_mo"  "1801"   "0.700631598047947"
## [7,] "Dx_ISSM_PE_before" "PHQ9_total"     "1246"   "0.0729649742980929"
## [8,] "Dx_ISSM_PE_before" "GAD7_total"     "1332.5" "0.145733307144622"
##      [,5]                          [,6]
## [1,] "53 (16) "                    "57 (4.75) "
## [2,] "71.9 (15.4) "                "75.25 (10.95) "
## [3,] "1.722 (0.0820000000000001) " "1.7245 (0.09575) "
## [4,] "24.1 (4.7) "                 "24.85 (4.725) "
## [5,] "1 (0) "                      "1 (0) "
## [6,] "2 (3) "                      "2 (2.25) "
## [7,] "7 (8) "                      "9.5 (11.5) "
## [8,] "7 (9) "                      "10 (11.5) "
```

```
### Explaining the output table
# column 1: dependent var
# column 2: independent var
# column 3: wilcoxon stat
# column 4: p-value
# column 5 and 6: median and iqr for the respective group (grouped by dependent var = 0 and
 1)

### Note that for variables with normal distribution we can use serial_t() in the same packag
e
```

```
### Serial logistic regressions (with adjustments) for categorical variables at a time

adj = c(13, 42:46, 53, 61) # these numbers represents the column numbers of the concerned var
iable for adjustment in the dataframe; these would be repeated for each analysis for the conc
erned variables
ind = c(14, 16:17) # these numbers represents the column numbers of the concerned variable fo
r analysis in the dataframe

serial_logregka(dependent_var, # the dataframe for dependent variables
                masterdata_A, # the dataframe for independent variables
                1, # 1 being the column number of the dependent variable concerned
                ind,
                1, # The category for reference
                adj)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##                                    OR_b    V2
## education_1                           1  <NA>
## education_2             0.4 (0.1-1.5) 0.186
## education_3             0.6 (0.2-1.9) 0.367
## education_4             0.7 (0.3-1.2) 0.179
## welfare_1                             1  <NA>
## welfare_2              5.8 (1.2-27.6) 0.027
## welfare_3               1.3 (0.6-2.7) 0.492
## welfare_4               1.5 (0.8-2.8) 0.219
## marital_1                             1  <NA>
## marital_2                   0.2 (0-1)  0.05
## marital_3               1.2 (0.5-2.8) 0.622
## marital_4   26927788538073072 (0-Inf)     1
```

```
# another example with a different category for reference
serial_logregka(dependent_var,
                masterdata_A,
                1,
                15, # different variable
                2, # different category for reference
                adj)
```

```
##                             OR_b    V2
## employment_2                   1  <NA>
## employment_1 2548398.3 (0-Inf) 0.996
## employment_3     2.1 (0.6-7.7)  0.27
## employment_4     0.4 (0.2-1.1) 0.065
```

```
### Explaining the output table
# column 1: variable name
# column 2: Odds ratio and CI
# column 3: p-value

### Note that there are functions for logistic regressions with continuous variables as well,
e.g.:
# serial_logregkacont(); which excludes "0" values from analysis
# serial_logregkacont0(); which includes "0" values from analysis
```

```
### VIFs for a series of multivariate logistic regressions (categorical variables)

adj = c(13, 42:46, 53, 61) # these numbers represents the column numbers of the concerned var
iable for adjustment in the dataframe; these would be repeated for each analysis for the conc
erned variables
ind = c(23:24, 47:49, 62:63) # these numbers represents the column numbers of the concerned v
ariable for analysis in the dataframe

serial_vif(dependent_var, # the dataframe for dependent variables
           masterdata_A, # the dataframe for independent variables
           1, # 1 being the column number of the dependent variable concerned
           ind,
           1, # The category for reference
           adj)
```

```
##                          var   age    DM hyperthyroidism vit_B_def prostitis
## smoking1                  NA    NA    NA              NA        NA        NA
## smoking2               1.064 1.131 1.062               1     1.000     1.045
## drinking1                 NA    NA    NA              NA        NA        NA
## drinking2              1.050 1.109 1.069               1     1.000     1.037
## multiple_sclerosis1       NA    NA    NA              NA        NA        NA
## multiple_sclerosis2    1.492 1.108 1.047               1     1.492     1.038
## peripheral_neuropathy1    NA    NA    NA              NA        NA        NA
## peripheral_neuropathy2 1.000 1.102 1.039               1     1.000     1.041
## CRF1                      NA    NA    NA              NA        NA        NA
## CRF2                   1.000 1.123 1.052               1     1.000     1.034
## SA_comb1                  NA    NA    NA              NA        NA        NA
## SA_comb2               1.079 1.102 1.047               1     1.000     1.066
## HSDD_comb1                NA    NA    NA              NA        NA        NA
## HSDD_comb2             1.245 1.131 1.081               1     1.000     1.040
##                        varicocele anti_androgen ED_comb
## smoking1                       NA            NA      NA
## smoking2                        1         1.090   1.114
## drinking1                      NA            NA      NA
## drinking2                       1         1.053   1.109
## multiple_sclerosis1            NA            NA      NA
## multiple_sclerosis2             1         1.026   1.090
## peripheral_neuropathy1         NA            NA      NA
## peripheral_neuropathy2          1         1.034   1.080
## CRF1                           NA            NA      NA
## CRF2                            1         1.034   1.084
## SA_comb1                       NA            NA      NA
## SA_comb2                        1         1.110   1.065
## HSDD_comb1                     NA            NA      NA
## HSDD_comb2                      1         1.066   1.139
```

```
### Note that there are functions for VIFs for logistic regressions with continuous variables
as well, e.g.:
# serial_vif_cont(); which excludes "0" values from analysis
# serial_vif_cont_0(); which includes "0" values from analysis
```

Output of results

```
### The output of results is simple
### For each analysis, pass the calculation to an object, i.e.

result = serial_logregka(dependent_var, masterdata_A, 1, ind, 1, adj)

### And then export the object with the following code:

write.csv(result, # the concerned object
          "~/testingresults.csv", # output filename
          row.names = T)

### the output file could be read by excel, and should be readily presentable as the final ta
ble in the dissertation with some standard formatting
```

The above demonstration does not include all the functions in the package. Other interesting functions include those for classifying anti-depressants into respective classes, calculating equivalent SSRI dosage, data wrangling, and functions to facilitate step-wise multivariate regressions. Please contact author for further queries.