

# Package ‘robustrank’

July 15, 2020

**LazyLoad** yes

**LazyData** yes

**Version** 2019.9-10

**Title** Robust Rank-Based Tests

**Author** Youyi Fong <youyifong@gmail.com>

**Maintainer** Youyi Fong <youyifong@gmail.com>

**Depends** R (>= 3.2.0), kyotil

**Suggests** RUnit, VGAM, copula, mvtnorm, pracma

## Description

Implements two-sample tests for paired data with missing values (Fong, Huang, Lemos and McElrath 2018, *Biostatistics*, <doi:10.1093/biostatistics/kxx039>) and modified Wilcoxon-Mann-Whitney two sample location test, also known as the Fligner-Policello test.

**License** GPL-2

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-07-15 09:40:05 UTC

## R topics documented:

choose.test . . . . .	2
dat.mtct.rob . . . . .	2
mod.wmw.test . . . . .	3
multinom.test . . . . .	4
mw.mw.2.perm . . . . .	5
pair.wmw.test . . . . .	6
pm.wilcox.test . . . . .	7
robustrank . . . . .	8
sim.partially.matched . . . . .	8
wmw.paired.replicates.test . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

choose.test

*Make Recommendations on the Most Powerful Test to Use*

---

**Description**

Performs simulations to compare the power of different tests

**Usage**

```
choose.test(Xpaired, Ypaired, Xextra = NULL, Yextra = NULL, mc.rep = 1000)
```

**Arguments**

Xpaired

Ypaired

Xextra

Yextra

mc.rep

**Examples**

```
# There are unpaired observations from both samples
dat=sim.partially.matched(m=20,n.x=40,n.y=5,distr="normal",
  params=c(loc.2=.8,rho=.1,scale.2=1),seed=1)
choose.test(dat$X, dat$Y, dat$Xprime, dat$Yprime)

## There are unpaired observations from only one sample
#dat=sim.partially.matched(m=20,n.x=0,n.y=10,distr="normal",
#  params=c(loc.2=.5,rho=.8,scale.2=1),seed=1)
#choose.test(dat$X, dat$Y, dat$Xprime, dat$Yprime)
```

---

dat.mtct.rob*Example Dataset*

---

**Description**

from MTCT correlates study, C-section only

**Usage**

```
data("dat.mtct.rob")
```

**Format**

A data frame with 55 observations on the following 2 variables.

y a numeric vector

V3\_BioV3B\_500 a numeric vector

**References**

Fong and Huang (2016) Modified Wilcoxon-Mann-Whitney Test and Power against Strong Null.

---

mod.wmw.test	<i>Modified Wilcoxon-Mann-Whitney Test</i>
--------------	--

---

**Description**

Also known as the Fligner-Policello test.

**Usage**

```
mod.wmw.test(X, Y, alternative = c("two.sided", "less", "greater"),
             correct = TRUE, perm = NULL, mc.rep = 10000, method =
             c("combine", "comb2", "fp", "wmw", "fplarge", "nsm3"),
             verbose = FALSE, mode = c("test", "var"), useC = TRUE)
```

**Arguments**

X	Samples from population 1.
Y	Samples from population 2.
alternative	Direction of the alternative hypothesis.
correct	Whether to do continuity correction.
perm	Boolean, whether to do permutation to get p-value or use normal approximation. See details.
mc.rep	Default number of replicates when doing permutation. See details.
method	For development.
verbose	For development. Print some debug info.
mode	For development.
useC	For development. Run C or R implementation.

**Details**

When perm is null, we will compute permutation-based p values if either sample size is less than 20 and compute normal approximation-based p values otherwise. When doing permutation, if the possible number of combinations is less than mc.rep, every possible configuration is done.

**Value**

A p value for now.

**References**

manuscript in preperation

**Examples**

```
# Example 4.1, Hollander, Wolfe and Chicken (2014) Nonparametric Statistics
X <- c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46)
Y <- c(1.15, 0.88, 0.90, 0.74, 1.21)
mod.wmw.test(X, Y, method="wmw", alternative="greater")
mod.wmw.test(X, Y, method="combine", alternative="greater", verbose=1)

# Section 4.1 Problem 1, Hollander et al.
X=c(1651,1112,102.4,100,67.6,65.9,64.7,39.6,31.0)
Y=c(48.1,48.0,45.5,41.7,35.4,34.3,32.4,29.1,27.3,18.9,6.6,5.2,4.7)
mod.wmw.test(X, Y, method="wmw")
mod.wmw.test(X, Y, method="combine", verbose=1)

# Section 4.1 Problem 5, Hollander et al.
X=c(12 ,44 ,34 ,14 ,9 ,19 ,156,23 ,13 ,11 ,47 ,26 ,14 ,33 ,15 ,62 ,5 ,8 ,0 ,154,146)
Y=c(37,39,30,7,13,139, 45,25,16,146,94,16,23,1,290,169,62,145,36, 20, 13)
mod.wmw.test(X, Y, method="wmw", alternative="less")
mod.wmw.test(X, Y, method="combine", alternative="less", verbose=1)

# Section 4.1 Problem 15, Hollander et al.
X=c(0.19,0.14,0.02,0.44,0.37)
Y=c(0.89,0.76,0.63,0.69,0.58,0.79,0.02,0.79)
mod.wmw.test(X, Y, method="wmw")
mod.wmw.test(X, Y, method="combine", verbose=1)

# Table 4.7, Hollander et al.
X=c(297,340,325,227,277,337,250,290)
Y=c(293,291,289,430,510,353,318)
mod.wmw.test(X, Y, method="wmw", alternative="less")
mod.wmw.test(X, Y, method="combine", alternative="less", verbose=1)
```

---

multinom.test

*Multinom Test*

---

**Description**

Perform multinom test.

**Usage**

```
multinom.test(X, Y, alternative = c("two.sided", "less", "greater"),
  correct = FALSE, perm = NULL, mc.rep = 10000, method =
  c("exact.2", "large.0", "large", "exact", "exact.0",
  "exact.1", "exact.3"), verbose = FALSE, mode =
  c("test", "var"), useC = TRUE)
```

**Arguments**

X  
Y  
alternative  
correct  
perm  
mc.rep  
method  
verbose  
mode  
useC

---

mw.mw.2.perm

*A Test that Combines WMW for Paired Data and WMW for Unpaired Data*


---

**Description**

Use permutation-based reference distribution to obtain p values for a test that combines WMW for paired data and WMW for unpaired data

**Usage**

```
mw.mw.2.perm(X, Y, Xprime, Yprime, .corr, mc.rep = 10000,
  alternative = c("two.sided", "less", "greater"), verbose = FALSE)
```

**Arguments**

X  
Y  
Xprime  
Yprime  
.corr  
mc.rep  
alternative  
verbose

---

 pair.wmw.test

 WMW test for paired data
 

---

### Description

Performs a WMW-type test of the strong null for paired data.

### Usage

```
pair.wmw.test(X, Y, alternative = c("two.sided", "less", "greater"),
  correct = TRUE, perm = NULL, mc.rep = 10000, method =
  c("exact.2", "large.0", "large", "exact", "exact.0",
  "exact.1", "exact.3"), verbose = FALSE, mode =
  c("test", "var"), p.method = NULL, useC = TRUE)
```

### Arguments

X	Sample 1.
Y	Sample 2.
alternative	Alternative hypothesis.
correct	Whether to apply continuity correction.
perm	Whether to use permutation distribution or normal approximation to find p-value. See details.
mc.rep	Number of Monte Carlo replicates for permutation test.
method	Choices of test statistics.
verbose	Print debug message when positive.
mode	For development used.
useC	For development used.
p.method	Method for obtaining p values.

### Details

When perm is NULL, if  $(\min(m,n) \geq 20)$  normal approximation is used to find p value, otherwise permutation test is used. When permutation test is used, if the number of possible permutations is less than mc.rep, a test statistic is computed for all permutations; otherwise, Monte Carlo is done.

### Value

P value for now.

### References

Under prep.

**Examples**

```

dat=sim.partially.matched(m=15,n.x=0,n.y=20,distr="mixnormal",params=c(p.1=0.3,p.2=0.3),seed=1)
X=dat$X; Y=dat$Y
pair.wmw.test(X, Y, perm=TRUE, method="large.0", verbose=1)
pair.wmw.test(X, Y, perm=FALSE, method="large.0", verbose=1)

```

---

pm.wilcox.test

*Wilcoxon test for Partially Matched Two Sample Data*


---

**Description**

Performs rank-based two sample test for partially matched two sample data by combining information from matched and unmatched data

**Usage**

```

pm.wilcox.test(Xpaired, Ypaired, Xextra = NULL, Yextra = NULL,
  alternative = c("two.sided", "less", "greater"),
  method = c("SR-MW", "MW-MW", "all"), mode = c("test",
  "var", "power.study"), useC = FALSE, correct = NULL,
  verbose=FALSE)

```

**Arguments**

Xpaired	
Ypaired	
Xextra	
Yextra	
alternative	
method	String. SR-MW is recommended, all is for development only.
mode	String. Do not change it to var, for development only.
useC	Boolean. Do not set it to TRUE, for development only.
verbose	
correct	Continuity correction.

**Details**

If Xpaired and Ypaired have NAs, the corresponding unpaired data in Ypaired and Xpaired will be combined with Yextra and Xextra.

**Value**

An htest object.

**Examples**

```

set.seed(1)
z=rnorm(20, sd=0.5) # induces correlation between X and Y
X=rnorm(20)+z
Y=rnorm(20,mean=0.8)+z
X[1:10]=NA
boxplot(X,Y,names=c("X","Y"))

pm.wilcox.test(X,Y)
# for comparison
wilcox.test(X,Y,paired=TRUE)
wilcox.test(X,Y,paired=FALSE)# often a conservative test due to the correlation

# no paired data
Y1=Y
Y1[11:20]=NA
pm.wilcox.test(X,Y1)
# should match the following
wilcox.test(X,Y1,paired=FALSE)

# only 1 pair of matched data
Y1=Y
Y1[12:20]=NA
pm.wilcox.test(X,Y1)

```

---

robustrank

*robustrank*


---

**Description**

Please see the [Index](#) link below for a list of available functions.

---

sim.partially.matched *Simulate Paired, Independent, or Partially Matched Two-Sample Data*


---

**Description**

sim.partially.matched generates partially matched two-sample data. for Monte Carlo studies. r2sample is a wrapper for sim.partially.matched and generates independent two-sample data.



**Usage**

```

sim.partially.matched(m, n.x, n.y,
  distr = c("normal", "logistic", "student", "mixnormal", "gamma", "lognormal", "beta",
    "uniform", "hybrid1", "hybrid2", "doublexp"), params, seed)

r2sample(m, n,
  distr = c("normal", "logistic", "student", "mixnormal"), params, seed)

sim.paired.with.replicates(m, meanRatio, sdRatio, within.sd, type, hyp, distr, seed)

```

**Arguments**

m	Number of pairs.
n	Number of Ys.
n.x	Number of extra Xs.
n.y	Number of extra Ys.
distr	Distributions.
params	Named vector. See details.
seed	Seed for random number generator.
meanRatio	
sdRatio	
within.sd	
type	
hyp	

**Details**

If the distribution is in `c("normal", "student", "logistic")`, `params` should have three fields: `loc.2`, `rho` and `scale.2`; `loc.1` is set to 0 and `scale.1` is set to 1.

If the distribution is `mixnormal`, `params` should have three fields: `p.1`, `p.2` and `sd.n`.

If the distribution is `gamma`, `params` should have five fields: `loc.2`, `shape.1`, `shape.2`, `rate.1`, `rate.2` and `rho`.

For details on bivariate logistic distribution, see `rbilogistic`

**Value**

`sim.partially.matched` return a list with the following components:

X	m sample 1 that pair with Y
Y	m sample 2 that pair with X
Xprime	n.x sample 1
Yprime	n.y sample 2

r2sample returns a list with the following components:

X                    m sample 1 that are independent of Y  
Y                    n sample 2 that are independent of X

### Examples

```
dat=sim.partially.matched(m=10,n.x=5,n.y=4,distr="normal",
  params=c("loc.2"=0,"rho"=0,"scale.2"=1),seed=1)
X=dat$X; Y=dat$Y; Yprime=dat$Yprime
```

```
#dat=sim.partially.matched(m=10,n.x=5,n.y=4,distr="logistic",
#  params=c("loc.2"=0,"rho"=0,"scale.2"=1),seed=1)
#X=dat$X; Y=dat$Y; Yprime=dat$Yprime
```

---

wmm.paired.replicates.test

*WMW Paired Replicates Test*

---

### Description

Perform WMW paired replicates test.

### Usage

```
wmm.paired.replicates.test(X, Y, alternative = c("two.sided", "less", "greater"),
  correct = FALSE, perm = NULL, mc.rep = 10000, method =
  c("exact.2", "large.0", "large", "exact", "exact.0",
  "exact.1", "exact.3"), verbose = FALSE, mode =
  c("test", "var"), useC = TRUE)
```

### Arguments

X  
Y  
alternative  
correct  
perm  
mc.rep  
method  
verbose  
mode  
useC

# Index

- \* **Mann-Whitney U test**
  - mod.wmw.test, 3
- \* **Wilcoxon rank sum test**
  - mod.wmw.test, 3
- \* **distribution**
  - robustrank, 8
  
- choose.test, 2
  
- dat.mtct.rob, 2
  
- mod.wmw.test, 3
- multinom.test, 4
- mw.mw.2.perm, 5
  
- pair.wmw.test, 6
- pm.wilcox.test, 7
  
- r2sample(sim.partially.matched), 8
- robustrank, 8
  
- sim.paired.with.replicates
  - (sim.partially.matched), 8
- sim.partially.matched, 8
  
- wmw.paired.replicates.test, 10