

# PIMs as classical distribution free tests

Nick Sabbe

October 16, 2014

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Wilcoxon-Mann-Whitney</b>	<b>2</b>
<b>3</b>	<b>Kruskal-Wallis</b>	<b>3</b>
<b>4</b>	<b>Mack-Skillings</b>	<b>4</b>
<b>5</b>	<b>Brown-Hettmansperger</b>	<b>6</b>
<b>6</b>	<b>Jonckheere-Terpstra</b>	<b>7</b>
<b>7</b>	<b>Mack-Wolfe</b>	<b>9</b>

## 1 Introduction

While Thas et al. (2012) introduces PIMs, they were shown to be extensions of the known distribution free tests, like Wilcoxon in `rankpaperref`.

A general estimator for the variance under the null hypothesis has been created (*varianceestimator.H0*), but for these special cases, some simplified formulas exist, which have been provided through the *simplified\** set of functions.

For each of the code sections below, it is easily checked that the simplified formulas give the same result as the generic code, and that the links between the known distribution free tests and PIM are confirmed.

The code also (on the final line of each block) the `classical.test` function that has been provided. This function uses the PIM to calculate the matching test statistic. The advantage (although not shown in this vignette) is that another way of calculating the (co)variances can be specified. In particular, and as indicated in `rankpaper`, the `varianceestimator.sandwich` can be passed along to get Wald-style tests for the same null hypotheses.

## 2 Wilcoxon-Mann-Whitney

Code to check the equivalence (note this includes a legacy implementation):

```
> library(pim)
> set.seed(1)
> wmw1<-demo.WilcoxonMannWhitney()
> wmw1$pim2<-pim(y~F(x)-1, data=wmw1$dta, link="identity", poset=lexiposet,
+                                                       varianceestimator=
+                                                       interpretation="re
> wmw1$pim2
```

Call:

```
pim(formula = y ~ F(x) - 1, data = wmw1$dta, link = "identity",
    poset = lexiposet, interpretation = "regular", varianceestimator = varianceestimator,
    keep.data = TRUE, verbosity = 0)
```

Coefficients:

```
x_L_R_1_2
0.7937182
```

```
> #Simplified formulas
> simplifiedpimestimation.pairwisecoefficients(wmw1$dta, out="y", group="x")$
```

```
[1] 0.7937182
```

```
> simplifiedpimestimation.pairwisecovariance(wmw1$dta, out="y", group="x")
```

```
      1 - 2
1 - 2 0.003572439
```

```
> #From applying generic code:
> wmw1$pim2$coefficients
```

```
x_L_R_1_2
0.7937182
```

```
> wmw1$pim2$vcov[1,1]
```

```
[1] 0.003572439
```

```
> #Standardized WMW based on wilcoxon test
> wmw1$legacy<-legacy.WilcoxonMannWhitney(data=wmw1$dta, out="y", group="x")
> wmw1$legacy$statistic
```

```

      W
-4.91415

>      wmw1$legacy$conversion(wmw1$pim2$coefficients, wmw1$pim2$vcov)

x_L_R_1_2
  4.91415

>      classical.test(test="WilcoxonMannWhitney", data=wmw1$dta, out="y", group="x")

x_L_R_1_2
  4.91415

```

### 3 Kruskal-Wallis

Code to check the equivalence (note this includes a legacy implementation):

```

>      kw1<-demo.KruskalWallis()
>      kw1$pim3<-pim(y~F(x)-1, data=kw1$dta, link="identity", poset=fullposet, int
+                                     varianceestimator=v
>      kw1$pim3

```

Call:

```

pim(formula = y ~ F(x) - 1, data = kw1$dta, link = "identity",
    poset = fullposet, interpretation = "marginal", varianceestimator = varianceestimat
    keep.data = TRUE, verbosity = 0)

```

Coefficients:

```

      x_R_1      x_R_2      x_R_3
0.2480435 0.4335366 0.7366667

```

```

>      #Simplified formulas (lemma 1)
>      simplifiedpimestimation.marginalcoefficients(kw1$dta, out="y", group="x")

```

```

      1      2      3
0.2480435 0.4335366 0.7366667

```

```

>      simplifiedpimestimation.marginalcovariance(kw1$dta, out="y", group="x")

```

```

      1      2      3
1  0.0028177536 -0.0008416667 -0.0008416667
2 -0.0008416667  0.0012111789 -0.0008416667
3 -0.0008416667 -0.0008416667  0.0014962963

```

```

>      #From applying generic code:
>      kw1$pim3$coefficients

      x_R_1      x_R_2      x_R_3
0.2480435 0.4335366 0.7366667

>      kw1$pim3$vcov

      x_R_1      x_R_2      x_R_3
x_R_1  0.0028177536 -0.0008416667 -0.0008416667
x_R_2 -0.0008416667  0.0012111789 -0.0008416667
x_R_3 -0.0008416667 -0.0008416667  0.0014962963

>      #Standardized KW based on Kruskal-Wallis test
>      kw1$legacy<-legacy.KruskalWallis(data=kw1$dta, out="y", group="x")
>      kw1$legacy$statistic

Kruskal-Wallis chi-squared
      43.45664

>      kw1$legacy$conversion(kw1$pim3$coefficients, kw1$pim3$vcov)

      [,1]
[1,] 43.45664

>      classical.test(test="KruskalWallis", data=kw1$dta, out="y", group="x")$stat

      [,1]
[1,] 43.45664

```

## 4 Mack-Skillings

Code to check the equivalence (note this includes a legacy implementation):

```

>      mss1<-demo.MackSkillings()
>      mss1$pim1<-pim(y~F(x)-1, data=mss1$dta, link="identity", blocking.variables=
+                                                                    poset=fullposet, i
+                                                                    varianceestimator=
>      mss1$pim1

```

Call:

```

pim(formula = y ~ F(x) - 1, data = mss1$dta, link = "identity",
     blocking.variables = "b", poset = fullposet, interpretation = "marginal",
     varianceestimator = varianceestimator.H0(), keep.data = TRUE,

```

```

    verbosity = 0)

Coefficients:
    x_R_1    x_R_2    x_R_3
0.2425000 0.5283333 0.7291667

>      #Simplified formulas (lemma 4)
>      simplifiedpimestimation.marginalcoefficients(mss1$dta, out="y", group="x",
    1          2          3
0.2425000 0.5283333 0.7291667

>      simplifiedpimestimation.marginalcovariance(mss1$dta, out="y", group="x", bl
    1          2          3
1  0.0014351852 -0.0007175926 -0.0007175926
2 -0.0007175926  0.0014351852 -0.0007175926
3 -0.0007175926 -0.0007175926  0.0014351852

>      #From applying generic code:
>      mss1$pim1$coefficients
    x_R_1    x_R_2    x_R_3
0.2425000 0.5283333 0.7291667

>      mss1$pim1$vcov
    x_R_1    x_R_2    x_R_3
x_R_1  0.0014351852 -0.0007175926 -0.0007175926
x_R_2 -0.0007175926  0.0014351852 -0.0007175926
x_R_3 -0.0007175926 -0.0007175926  0.0014351852

>      #Standardized MS based on Mack-Skillings test
>      mss1$legacy<-legacy.MackSkillings(data=mss1$dta, out="y", group="x", block=
>      mss1$legacy$statistic

[1] 55.56839

>      mss1$legacy$conversion(mss1$pim1$coefficients, mss1$pim1$vcov)
    [,1]
[1,] 55.56839

>      classical.test(test="MackSkillings", data=mss1$dta, out="y", group="x", blo
    [,1]
[1,] 55.56839

```

## 5 Brown-Hettmansperger

Code to check the equivalence (note this includes a legacy implementation):

```
> bh1<-demo.BrownHettmansperger()
> bh1$pim1<-pim(y~F(x)-1, data=bh1$dta, link="identity", poset=fullposet,
+                                                     varianceestimator=v
+                                                     interpretation="reg
> bh1$pim1
```

Call:

```
pim(formula = y ~ F(x) - 1, data = bh1$dta, link = "identity",
    poset = fullposet, interpretation = "regular", varianceestimator = varianceestimator,
    keep.data = TRUE, verbosity = 0)
```

Coefficients:

```
x_L_R_1_2 x_L_R_1_3 x_L_R_2_3
0.6990196 0.8733660 0.7462963
```

```
> #Simplified formulas (lemma 4)
> simplifiedpimestimation.pairwisecoefficients(bh1$dta, out="y", group="x")$b
```

```
[1] 0.6990196 0.8733660 0.7462963
```

```
> simplifiedpimestimation.pairwisecovariance(bh1$dta, out="y", group="x")
```

```
      1 - 2      1 - 3      2 - 3
1 - 2 0.005310458 0.002450980 -0.002777778
1 - 3 0.002450980 0.004833878 0.002314815
2 - 3 -0.002777778 0.002314815 0.005169753
```

```
> #From applying generic code:
> bh1$pim1$coefficients
```

```
x_L_R_1_2 x_L_R_1_3 x_L_R_2_3
0.6990196 0.8733660 0.7462963
```

```
> bh1$pim1$vcov
```

```
      x_L_R_1_2      x_L_R_1_3      x_L_R_2_3
x_L_R_1_2 0.005310458 0.002450980 -0.002777778
x_L_R_1_3 0.002450980 0.004833878 0.002314815
x_L_R_2_3 -0.002777778 0.002314815 0.005169753
```

```

>      #Standardized BH based on Brown-Hettmansperger test
>      bh1$legacy<-legacy.BrownHettmansperger(data=bh1$dta, out="y", group="x")
>      bh1$legacy$statistic

Kruskal-Wallis chi-squared
          55.44186

>      bh1$legacy$conversion(bh1$pim1$coefficients, bh1$pim1$vcov)

          [,1]
[1,] 55.44186

>      classical.test(test="BrownHettmansperger", data=bh1$dta, out="y", group="x")

          [,1]
[1,] 55.44186

```

## 6 Jonckheere-Terpstra

Code to check the equivalence (note this includes a legacy implementation):

```

>      jt1<-demo.JonckheereTerpstra(force.balanced=FALSE)
>      jt1$pim1<-pim(y~F(x)-1, data=jt1$dta, link="identity", poset=lexiposet,
+                                     varianceestimator=varianceestimator,
+                                     interpretation="regular")
>      jt1$pim1

```

Call:

```

pim(formula = y ~ F(x) - 1, data = jt1$dta, link = "identity",
    poset = lexiposet, interpretation = "regular", varianceestimator = varianceestimator,
    keep.data = TRUE, verbosity = 0)

```

Coefficients:

```

x_L_R_1_2 x_L_R_1_3 x_L_R_1_4 x_L_R_2_3 x_L_R_2_4 x_L_R_3_4
0.7805556 0.9384615 1.0000000 0.8461538 1.0000000 0.8373626

```

```

>      #Simplified formulas (lemma 4)
>      simplifiedpimestimation.pairwisecoefficients(jt1$dta, out="y", group="x")$b

[1] 0.7805556 0.9384615 1.0000000 0.8461538 1.0000000 0.8373626

>      simplifiedpimestimation.pairwisecovariance(jt1$dta, out="y", group="x")

```

	1 - 2	1 - 3	1 - 4	2 - 3	2 - 4
1 - 2	0.009259259	0.005555556	0.005555556	-0.003472222	-0.003472222
1 - 3	0.005555556	0.008974359	0.005555556	0.003205128	0.000000000
1 - 4	0.005555556	0.005555556	0.008095238	0.000000000	0.002380952
2 - 3	-0.003472222	0.003205128	0.000000000	0.006810897	0.003472222
2 - 4	-0.003472222	0.000000000	0.002380952	0.003472222	0.005952381
3 - 4	0.000000000	-0.003205128	0.002380952	-0.003205128	0.002380952

  

	3 - 4
1 - 2	0.000000000
1 - 3	-0.003205128
1 - 4	0.002380952
2 - 3	-0.003205128
2 - 4	0.002380952
3 - 4	0.005677656

```
> #From applying generic code:
> jt1$pim1$coefficients
```

```
x_L_R_1_2 x_L_R_1_3 x_L_R_1_4 x_L_R_2_3 x_L_R_2_4 x_L_R_3_4
0.7805556 0.9384615 1.0000000 0.8461538 1.0000000 0.8373626
```

```
> jt1$pim1$vcov
```

	x_L_R_1_2	x_L_R_1_3	x_L_R_1_4	x_L_R_2_3	x_L_R_2_4
x_L_R_1_2	0.009259259	0.005555556	0.005555556	-0.003472222	-0.003472222
x_L_R_1_3	0.005555556	0.008974359	0.005555556	0.003205128	0.000000000
x_L_R_1_4	0.005555556	0.005555556	0.008095238	0.000000000	0.002380952
x_L_R_2_3	-0.003472222	0.003205128	0.000000000	0.006810897	0.003472222
x_L_R_2_4	-0.003472222	0.000000000	0.002380952	0.003472222	0.005952381
x_L_R_3_4	0.000000000	-0.003205128	0.002380952	-0.003205128	0.002380952

  

	x_L_R_3_4
x_L_R_1_2	0.000000000
x_L_R_1_3	-0.003205128
x_L_R_1_4	0.002380952
x_L_R_2_3	-0.003205128
x_L_R_2_4	0.002380952
x_L_R_3_4	0.005677656

```
> #Standardized JT based on Jonckheere-Terpstra test
> jt1$legacy<-legacy.JonckheereTerpstra(data=jt1$dta, out="y", group="x", ver
```

```
mu: 1824.5
sigsq: 25924.92
mainterm: 3302
```



```

>          jt1$legacy$statistic

[1] 9.176325

>          jt1$legacy$conversion(jt1$pim1$coefficients, jt1$pim1$vcov)

          [,1]
[1,] 9.176325

>          classical.test(test="JonckheereTerpstra", data=jt1$dta, out="y", group="x")

          [,1]
[1,] 9.176325

```

## 7 Mack-Wolfe

Code to check the equivalence (note this includes a legacy implementation):

```

>          mw1<-demo.MackWolfe(force.balanced=FALSE)
>          mw1$pim1<-pim(y~F(x)-1, data=mw1$dta, link="identity", poset=lexiposet,
+                               varianceestimator=varianceestimator,
+                               interpretation="regular")
>          mw1$pim1

```

Call:

```

pim(formula = y ~ F(x) - 1, data = mw1$dta, link = "identity",
    poset = lexiposet, interpretation = "regular", varianceestimator = varianceestimator,
    keep.data = TRUE, verbosity = 0)

```

Coefficients:

```

  x_L_R_1_2  x_L_R_1_3  x_L_R_1_4  x_L_R_2_3  x_L_R_2_4  x_L_R_3_4
0.635869565 1.000000000 0.898050975 1.000000000 0.883620690 0.006465517

```

```

>          #Simplified formulas (lemma 4)
>          simplifiedpimestimation.pairwisecoefficients(mw1$dta, out="y", group="x")$b

```

```

[1] 0.635869565 1.000000000 0.898050975 1.000000000 0.883620690 0.006465517

```

```

>          simplifiedpimestimation.pairwisecovariance(mw1$dta, out="y", group="x")

```

```

          1 - 2          1 - 3          1 - 4          2 - 3          2 - 4
1 - 2  0.009057971  0.003623188  0.003623188 -0.005208333 -0.005208333
1 - 3  0.003623188  0.006340580  0.003623188  0.002604167  0.000000000
1 - 4  0.003623188  0.003623188  0.006621689  0.000000000  0.002873563

```

```

2 - 3 -0.005208333  0.002604167  0.000000000  0.007975260  0.005208333
2 - 4 -0.005208333  0.000000000  0.002873563  0.005208333  0.008261494
3 - 4  0.000000000 -0.002604167  0.002873563 -0.002604167  0.002873563

```

```

      3 - 4

```

```

1 - 2  0.000000000
1 - 3 -0.002604167
1 - 4  0.002873563
2 - 3 -0.002604167
2 - 4  0.002873563
3 - 4  0.005567529

```

```

>      #From applying generic code:
>      mw1$pim1$coefficients

```

```

      x_L_R_1_2  x_L_R_1_3  x_L_R_1_4  x_L_R_2_3  x_L_R_2_4  x_L_R_3_4
0.635869565  1.000000000  0.898050975  1.000000000  0.883620690  0.006465517

```

```

>      mw1$pim1$vcov

```

```

      x_L_R_1_2  x_L_R_1_3  x_L_R_1_4  x_L_R_2_3  x_L_R_2_4
x_L_R_1_2  0.009057971  0.003623188  0.003623188 -0.005208333 -0.005208333
x_L_R_1_3  0.003623188  0.006340580  0.003623188  0.002604167  0.000000000
x_L_R_1_4  0.003623188  0.003623188  0.006621689  0.000000000  0.002873563
x_L_R_2_3 -0.005208333  0.002604167  0.000000000  0.007975260  0.005208333
x_L_R_2_4 -0.005208333  0.000000000  0.002873563  0.005208333  0.008261494
x_L_R_3_4  0.000000000 -0.002604167  0.002873563 -0.002604167  0.002873563

```

```

      x_L_R_3_4

```

```

x_L_R_1_2  0.000000000
x_L_R_1_3 -0.002604167
x_L_R_1_4  0.002873563
x_L_R_2_3 -0.002604167
x_L_R_2_4  0.002873563
x_L_R_3_4  0.005567529

```

```

>      #Standardized MW based on Mack-Wolfe test
>      mw1$legacy<-legacy.MackWolfe(data=mw1$dta, out="y", group="x",
+

```

```

mu: 1272
sigsq: 19541.33
leftterm: 1482
rightterm: 922

```

```

>      mw1$legacy$statistic

```

```
[1] 8.097843
```

```
> mw1$legacy$conversion(mw1$pim1$coefficients, mw1$pim1$vcov)
```

```
 [,1]
```

```
[1,] 8.097843
```

```
> classical.test(test="MackWolfe", data=mw1$dta, out="y", group="x",
```

```
+ levelP=as.character(0.05))
```

```
 [,1]
```

```
[1,] 8.097843
```

## References

O.~Thas, J.~De~Neve, L.~Clement, and J-P. Ottoy. Probabilistic index models (with discussion). *Journal of the Royal Statistical Society - Series B*, 74:1–29, 2012.