PIMs as classical distribution free tests

Nick Sabbe

November 30, 2012

Contents

1 Introduction

While ?) introduces PIMs, they were shown to be extensions of the known distribution free tests, like Wilcoxon in TODO: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 TODO ref 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=pairwisepose
>           wmw1$pim2
```

```
Call:
pim(frm = y ~ F(x) - 1, data = wmw1$dta, link = "identity", poset = pairwiseposet,
    varianceestimator = varianceestimator.H0(), keep.data = TRUE,
    verbosity = 0)

Coefficients:
x_R_L_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_R_L_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_R_L_1_2
  4.91415

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

x_R_L_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, for
>         kw1$pim3

Call:
pim(frm = y ~ F(x) - 1, data = kw1$dta, link = "identity", poset = fullposet,
    force.marginal = TRUE, varianceestimator = varianceestimator.H0(),
    keep.data = TRUE, verbosity = 0)

Coefficients:
    x_R_1     x_R_2     x_R_3
0.2863043 0.4330952 0.7207143

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

        1         2         3
0.2863043 0.4330952 0.7207143

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

              1             2             3
1  0.0028177536 -0.0008416667 -0.0008416667
2 -0.0008416667  0.0011623016 -0.0008416667
3 -0.0008416667 -0.0008416667  0.0015630952

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

    x_R_1     x_R_2     x_R_3
0.2863043 0.4330952 0.7207143

>         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  0.0011623016 -0.0008416667
x_R_3 -0.0008416667 -0.0008416667  0.0015630952

>         #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
                  34.97031

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

         [,1]
[1,] 34.97031

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

         [,1]
[1,] 34.97031
```

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
>         mss1$pim1

Call:
pim(frm = y ~ F(x) - 1, data = mss1$dta, link = "identity", blocking.variables = "b",
    poset = fullposet, force.marginal = TRUE, varianceestimator = varianceestimator.H0(
    keep.data = TRUE, verbosity = 0)

Coefficients:
    x_R_1     x_R_2     x_R_3
0.2858333 0.5225000 0.6916667

>         #Simplified formulas (lemma 4)
>         simplifiedpimestimation.marginalcoefficients(mss1$dta, out="y", group="x",

        1         2         3
0.2858333 0.5225000 0.6916667

>         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
0.2858333 0.5225000 0.6916667

>         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] 38.60581

>         mss1$legacy$conversion(mss1$pim1$coefficients, mss1$pim1$vcov)

         [,1]
[1,] 38.60581

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

         [,1]
[1,] 38.60581
```

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, var
>         bh1$pim1

Call:
pim(frm = y ~ F(x) - 1, data = bh1$dta, link = "identity", poset = fullposet,
    varianceestimator = varianceestimator.H0(), keep.data = TRUE,
    verbosity = 0)

Coefficients:
x_R_L_1_2 x_R_L_1_3 x_R_L_2_3
0.6454545 0.8746929 0.8081081

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

```
[1] 0.6454545 0.8746929 0.8081081

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

             1 - 2       1 - 3        2 - 3
1 - 2  0.005387205 0.002525253 -0.002777778
1 - 3  0.002525253 0.004845755  0.002252252
2 - 3 -0.002777778 0.002252252  0.005105105

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

x_R_L_1_2 x_R_L_1_3 x_R_L_2_3
0.6454545 0.8746929 0.8081081

>         bh1$pim1$vcov

             x_R_L_1_2   x_R_L_1_3    x_R_L_2_3
x_R_L_1_2  0.005387205 0.002525253 -0.002777778
x_R_L_1_3  0.002525253 0.004845755  0.002252252
x_R_L_2_3 -0.002777778 0.002252252  0.005105105

>         #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
                  62.44696

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

         [,1]
[1,] 62.44696

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

         [,1]
[1,] 62.44696
```

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=pairwiseposet,
>         jt1$pim1

Call:
pim(frm = y ~ F(x) - 1, data = jt1$dta, link = "identity", poset = pairwiseposet,
    varianceestimator = varianceestimator.H0(), keep.data = TRUE,
    verbosity = 0)

Coefficients:
x_R_L_1_2 x_R_L_1_3 x_R_L_1_4 x_R_L_2_3 x_R_L_2_4 x_R_L_3_4
0.7925926 0.9025641 0.9916667 0.7065527 0.9733796 0.9314904

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

[1] 0.7925926 0.9025641 0.9916667 0.7065527 0.9733796 0.9314904

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

            1 - 2       1 - 3       1 - 4        2 - 3        2 - 4
1 - 2  0.008847737  0.005555556 0.005555556 -0.003086420 -0.003086420
1 - 3  0.005555556  0.008974359 0.005555556  0.003205128  0.000000000
1 - 4  0.005555556  0.005555556 0.008333333  0.000000000  0.002604167
2 - 3 -0.003086420  0.003205128 0.000000000  0.006410256  0.003086420
2 - 4 -0.003086420  0.000000000 0.002604167  0.003086420  0.005787037
3 - 4  0.000000000 -0.003205128 0.002604167 -0.003205128  0.002604167
            3 - 4
1 - 2  0.000000000
1 - 3 -0.003205128
1 - 4  0.002604167
2 - 3 -0.003205128
2 - 4  0.002604167
3 - 4  0.005909455

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

x_R_L_1_2 x_R_L_1_3 x_R_L_1_4 x_R_L_2_3 x_R_L_2_4 x_R_L_3_4
0.7925926 0.9025641 0.9916667 0.7065527 0.9733796 0.9314904
```

```
>         jt1$pim1$vcov

            x_R_L_1_2   x_R_L_1_3  x_R_L_1_4   x_R_L_2_3   x_R_L_2_4
x_R_L_1_2  0.008847737  0.005555556 0.005555556 -0.003086420 -0.003086420
x_R_L_1_3  0.005555556  0.008974359 0.005555556  0.003205128  0.000000000
x_R_L_1_4  0.005555556  0.005555556 0.008333333  0.000000000  0.002604167
x_R_L_2_3 -0.003086420  0.003205128 0.000000000  0.006410256  0.003086420
x_R_L_2_4 -0.003086420  0.000000000 0.002604167  0.003086420  0.005787037
x_R_L_3_4  0.000000000 -0.003205128 0.002604167 -0.003205128  0.002604167
            x_R_L_3_4
x_R_L_1_2  0.000000000
x_R_L_1_3 -0.003205128
x_R_L_1_4  0.002604167
x_R_L_2_3 -0.003205128
x_R_L_2_4  0.002604167
x_R_L_3_4  0.005909455

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

mu: 1836.5
sigsq: 26044.92
mainterm: 3261

>         jt1$legacy$statistic

[1] 8.826753

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

         [,1]
[1,] 8.826753

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

         [,1]
[1,] 8.826753
```

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=pairwiseposet,
>         mw1$pim1
```

```
Call:
pim(frm = y ~ F(x) - 1, data = mw1$dta, link = "identity", poset = pairwiseposet,
    varianceestimator = varianceestimator.H0(), keep.data = TRUE,
    verbosity = 0)

Coefficients:
 x_R_L_1_2  x_R_L_1_3  x_R_L_1_4  x_R_L_2_3  x_R_L_2_4  x_R_L_3_4
0.76562500 1.00000000 0.93005952 0.98828125 0.68526786 0.02790179

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

[1] 0.76562500 1.00000000 0.93005952 0.98828125 0.68526786 0.02790179

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

             1 - 2        1 - 3        1 - 4        2 - 3        2 - 4
1 - 2  0.008897569  0.003472222 0.003472222 -0.005208333 -0.005208333
1 - 3  0.003472222  0.006184896 0.003472222  0.002604167  0.000000000
1 - 4  0.003472222  0.003472222 0.006572421  0.000000000  0.002976190
2 - 3 -0.005208333  0.002604167 0.000000000  0.007975260  0.005208333
2 - 4 -0.005208333  0.000000000 0.002976190  0.005208333  0.008370536
3 - 4  0.000000000 -0.002604167 0.002976190 -0.002604167  0.002976190
             3 - 4
1 - 2  0.000000000
1 - 3 -0.002604167
1 - 4  0.002976190
2 - 3 -0.002604167
2 - 4  0.002976190
3 - 4  0.005673363

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

 x_R_L_1_2  x_R_L_1_3  x_R_L_1_4  x_R_L_2_3  x_R_L_2_4  x_R_L_3_4
0.76562500 1.00000000 0.93005952 0.98828125 0.68526786 0.02790179

>         mw1$pim1$vcov

             x_R_L_1_2    x_R_L_1_3    x_R_L_1_4    x_R_L_2_3    x_R_L_2_4
x_R_L_1_2  0.008897569  0.003472222 0.003472222 -0.005208333 -0.005208333
x_R_L_1_3  0.003472222  0.006184896 0.003472222  0.002604167  0.000000000
x_R_L_1_4  0.003472222  0.003472222 0.006572421  0.000000000  0.002976190
x_R_L_2_3 -0.005208333  0.002604167 0.000000000  0.007975260  0.005208333
```

```
x_R_L_2_4 -0.005208333  0.000000000 0.002976190  0.005208333  0.008370536
x_R_L_3_4  0.000000000 -0.002604167 0.002976190 -0.002604167  0.002976190
             x_R_L_3_4
x_R_L_1_2  0.000000000
x_R_L_1_3 -0.002604167
x_R_L_1_4  0.002976190
x_R_L_2_3 -0.002604167
x_R_L_2_4  0.002976190
x_R_L_3_4  0.005673363

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

mu: 1280
sigsq: 19626.67
leftterm: 1568
rightterm: 871

>          mw1$legacy$statistic

[1] 8.272945

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

         [,1]
[1,] 8.272945

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

         [,1]
[1,] 8.272945
```