Robust Methods of Performing One-Way RM ANOVA in R

Wilcox's WRS package (Wilcox & Schönbrodt, 2014) provides several methods of testing one-way repeated measures data. The rmanova() command can trim means and the rmanovab() can trim means and perform bootstrap procedures on data where only one variable contains repeated measures. Both of these commands want as input an entire dataset. As always, Wilcox's commands need data in the "wide" form, but since we already have those forms all we will need to do to get the data ready is to remove any variables that are not going to be used in the analysis, such as the column that indexes which participant the data came from.

With the Murphy (2004) data in the murphy.wide dataset form, we could test the hypothesis that there was no difference between verb similarity for regular verbs. That hypothesis would involve only one repeated measure. Notice that we cannot add in a between-group independent variable (such as group). Here's how I would prepare the data for use in one of these Wilcox commands:

```
> names(murphy.wide)
[1] "group"         "reg_proto"     "reg_int"       "reg_distant"
[5] "irreg_proto"   "irreg_int"     "irreg_distant" "participant"
```

murphy.wrs<-murphy.wide[,2:4] #specify all rows (blank before comma), and columns 2 through 4

#if you needed to get non-adjacent columns, simple use c: murphy.wide[, c(2, 4, 6)]

names(murphy.wrs)

```
[1] "reg_proto"    "reg_int"      "reg_distant"
```

With the data in the correct form I can run an RM ANOVA with 20% trimmed means like this:

```
rmanova(murphy.wrs, tr=.2, grp=0) #did you open the WRS package?
```

The argument grp is used if you have several levels of your variable that you want to pick out and compare. For example, if you had 5 levels but only wanted to compare levels 2, 4 and 5, just write grp=c(2,4,5). Here I don't need it so I wrote 0. The output is a little scattered so I've edited it together here.

```
[1] "The number of groups to be compared is"
[1] 3
Warning in cor(xvec, yvec) : the standard deviation is zero
$test          $df         $siglevel
[1] 9.722222   [1]  1 35   [1] 0.003630193
$tmeans                          $ehat      $etil
[1] 5.000000 5.000000 4.666667   [1] 0.5    [1] 0.5
```

There is a warning in the output (the original is 5 lines long but says the same thing 5 times!) that I don't have any standard deviation in my score. This is because with means trimming, I've cut off the little bit of variation there was in the regular verbs for the Prototypical and Intermediate verbs, as you can see by the scores in $tmeans. Thus, this dataset is probably not a good one to use with means trimming (you can try it with tr = .1 or 0 and see that no warnings appear), but anyway, the results of this test show that there was a statistical difference for verb similarity in the regular verbs (remember, I only used

regular verbs here), $F_{1,35} = 9.72$, $p = .004$. The last two values, $ehat and $etil, are used by the algorithm to adjust the degrees of freedom due to means trimming, and thus probably do not need to be reported. To perform post-hoc tests on this same data, since I have 3 levels of variables and don't know where the differences lie, I can use the following command (I'm going to cut the trimming down to 10% because of my lack of variation problem):

```
rmmcp(murphy.wide, tr=.1)
```

The results show test values, *p*-values, and confidence intervals for the comparisons. These are interpreted in the normal way and show differences between the Prototypical and Distant, and also between the Intermediate and Distant verb similarities, but the lower level bound of the CI is extremely close to zero and I would probably argue that there were no practically interesting differences (see the pairdepb( ) command below for an example of how to analyze the results).

To try out the bootstrapped command, let's take Lyster's (2004) data, which we also have in the "wide" form. We'll again need to cut it down first to only those variables we want analyzed.

```
names(lyster.wide)
```

```
[1] "participant"    "cond"           "prebinary"      "post1binary"
[5] "post2binary"    "pretaskcompl"   "post1taskcompl" "post2taskcompl"
[9] "binarygain1"    "binarygain2"    "compgain1"      "compgain2"
```

I only want to analyze the repeated measure of Time on the Completion (cloze) task, so I'll pick out columns 6, 7 & 8.

```
lyster.wrs<-lyster.wide[, 6:8]
```

In addition, let's say that I decided I really didn't want the Comparison (control) group in there, so we want to cut out those rows that have the Comparison group. This is tricky because the Comparison group is repeated several times over the Time variable, so we can't just, say, cut out the last however number of rows that belong to Comparison. To subset the dataset, we need to know the exact name of the level we want to get rid of in the rows (and I'll need to go back to the lyster.wide since it has the information about Conditions in it):

```
> levels(lyster.wide$cond)
[1] "FFIrecast"  "FFIprompt"  "FFIonly"    "Comparison"
```

OK, got it, now let's cut down the dataset to one that only has rows that are *not* those of Comparison. The "!-" symbol means "not equal to." While I'm at it, I'll combine it with only including columns 6 through 8.

```
lyster.wrs<-lyster.wide[lyster.wide$cond!="Comparison", 6:8]
```

Run the str( ) command again to verify that your dataset has been cut down. Mine now has 129 observations whereas the original lyster.wide had 180.

Now let's run a one-way ANOVA with means trimming and bootstrapping (bootstrap-t) to answer the question of whether there was an effect for Time on the scores from the completion task:

```
rmanovab(lyster.wrs, tr=.1, alpha=.05, grp=0, nboot=2000)
```

The default for the number of bootstrap samples is 599, but you can change it. You can also change the default alpha level. The grp argument means the same thing it meant above. Here is the output (again, edited for space):

```
[1] "Taking bootstrap samples. Please wait."
[1] "The number of groups to be compared is"
[1] 3
$teststat       $crit
[1] 79.07014    [1] 3.361284
```

This output is smaller than that of the first command, and just gives us a test statistic (79.07) and the critical value for this statistic (3.36). Obviously, the test statistic is much larger than the critical value, which means that we can reject the null hypothesis and conclude that there are differences for different times. This is not much good to know without further information, so Wilcox has some further tests that will help us see where the differences lie. The following command performs post-hocs for the bootstrap-t one-way ANOVA used above:

```
pairdepb(lyster.wrs, nboot=2000)
```

```
$test
     Group Group      test        se
[1,]     1     2 -9.584736 0.7131592
[2,]     1     3 -7.701264 0.7248523
[3,]     2     3  2.664631 0.4702957

$psihat
     Group Group    psihat   ci.lower  ci.upper
[1,]     1     2 -6.835443 -8.5199772 -5.150909
[2,]     1     3 -5.582278 -7.2944325 -3.870125
[3,]     2     3  1.253165  0.1422917  2.364037

$crit
[1] 2.362073
```

We are most interested in looking at confidence intervals, so we will look at "ci.lower" and "ci.upper." Since the levels of the variable are pretest, immediate posttest and delayed posttest, we know that Group 1 vs. Group 2 is pretest vs. immediate posttest. Here the mean difference between the groups is 6.84 points, with a 95% CI of [-8.52, -5.15]. In a test where scores ranged from about 10 to 40, a minimum of 5 points of difference (the lower limit of the CI) seems like a good-sized effect. For the comparison between the pretest and the delayed posttest, the situation is similar, although not quite as strong: mean difference = 5.58, CI [-7.29, -3.87]. Lastly, the difference between the pretest and posttest is similar although the reversed signs alert us to the fact that there was a decline in scores from immediate to delayed posttest, mean difference = 1.25, CI [0.14, 2.36]. The lower limit of the CI is in fact so close to zero we might conclude that there is no difference in practical terms.

Robust Methods of Performing RM ANOVA with Between-group Independent Variables in R

In the previous section I cut down my sample datasets to fit into a one-way ANOVA mold to demonstrate some of Wilcox's commands, but that it not necessary as Wilcox

also has commands to analyze data that include one or more independent variables along with one or more repeated measure independent variable. Wilcox and other authors call this type of ANOVA a "between-within" ANOVA because it has between-group variables as well as within-group variables. You'll notice that the names of the commands use a "b" for between-group and a "w" for within-group variables.

The syntax for a command that analyzes one between-group variable and one within-group variable is:

bwtrim(J, K, x, tr=0.2, grp=c(1:p))

where J is the number of levels of the between-groups independent variable, K is the number of levels of the repeated-measures (within-group) variable, and x is the data in matrix or list mode. The argument grp is used if you have several levels of your variable that you want to pick out and compare. For example, if you had 5 levels but only wanted to compare levels 2, 4 and 5, just write grp=c(2,4,5). Obviously the trick to using the command is to make sure your data is in the right format.

For data in the "wide" format, such as the lyster.wide data, the command bw2list(x, grp.col, lev.col) can convert the data into the format the Wilcox command needs:

```
> names(lyster.wide)
 [1] "participant"    "cond"           "prebinary"      "post1binary"
 [5] "post2binary"    "pretaskcompl"   "post1taskcompl" "post2taskcompl"
 [9] "binarygain1"    "binarygain2"    "compgain1"      "compgain2"
```

```
Lyster1=bw2list(lyster.wide, 2, c(6:8))
```

where the first argument is the dataset, the second argument is the column indicating the between-group variable, and the third argument indicates which columns the within-group data is stored in. If you look at Lyster1 you'll see 6 lists, and I don't know where the number 6 comes from, but as it turns out, down below when I try it using lyster.long it comes out exactly the same way so I guess it's OK! Now in the correct format, we can run the trimmed means RM ANOVA:

```
bwtrim(4, 3, Lyster1, tr=.1)
```

where 4 = the number of levels of Condition (between-groups) and 3 = the number of levels of Time (within-groups).The output shows test statistics and significance levels for main effects and then interactions (output rearranged for space):

```
$Qa
[1] 14.63384

$Qa.siglevel
              [,1]
[1,] 1.627632e-07
```

```
$Qb
[1] 74.23411

$Qb.siglevel
     [,1]
[1,]    0
```

```
$Qab
[1] 14.31893

$Qab.siglevel
              [,1]
[1,] 7.089018e-11
```

You would interpret the output by noticing that the interaction between Group and Time has a test statistic of F = 14.31, with $p<.0005$. Both main effects are also statistical.

If you started with a dataset in the "long" format, you will also need to massage the data into the correct form. Here we'll use the more familiar fac2list( ) command, where the first argument is the response variable and the second argument lists the columns where (in order) the between-groups variable is found and then the within-groups variable is found. Here's how I would do it for lyster.long:

```
> names(lyster.long)
[1] "participant" "cond"        "time"        "compscore"
```

```
Lyster2=fac2list(lyster.long[,4], lyster.long[,c(2, 3)])
```

When I look at this data I have 12 lists, which is more like what I thought I should have gotten above, since I have 4 × 3 levels. Then I can compare the 10% trimmed means with the command:

```
bwtrim(4, 3, Lyster2, tr=.1)
```

and the results are the same as above. A different command can perform a bootstrap-t analysis with trimmed means for the same data:

```
tsplitbt(4,3, Lyster2, tr=.1, alpha=.05, nboot=599)
```
#here I have left the default number of bootstraps

```
$Qa             $Qb             $Qab
[1] 14.63384    [1] 74.23411    [1] 14.31893
$crit.Qa        $crit.Qb        $crit.Qab
[1] 3.377547    [1] 3.309766    [1] 2.202999
```

As with the bootstrap command in the previous section of this paper, the results (rearranged for space) come back not with test statistics and *p*-values but rather with test values and critical values. If the test value for the interaction or main effect is greater than the critical value, you may reject the null hypothesis and assume a statistical effect for the interaction or main effect. For the Lyster dataset, we understand that the interaction between Condition and Time is statistical, as well as the main effects of Condition and Time. If you wanted to do further investigation you could break the analysis down to look at the effects of Condition at just one Time (this is a **simple interaction effect**, and you would go back to what I demonstrated in the first section of this paper).

The command for a design that includes two within-group independent variables and can trim means is:

```
wwtrim(J, K, x, grp=c(1:p), p=J*K, tr=.2, bop=F)
```

which would essentially act just like the bwtrim( ) command except that both J and K represent levels of within-group variables. From previous Wilcox commands I assume that p=J*K is a check on the number of levels and you don't need to worry about it (nor write it in your command), but I don't know what bop = F is.

Wilcox has commands for three-way designs. If you have a design with J × K × L factors where J and K are between-group variables and L is a within-group variable, you can use the function:

```
bbwtrim(J, K, L, x, grp=c(1:p), tr=.2)
```

To get your data into the correct format, use the `bbw2list(x, grp.col, lev.col)` command. This works the same way as we saw for the `bw2list( )` command above, where the first argument is the dataset, the second argument is the column indicating the between-group variable, and the third argument indicates which columns the within-group data is in. The only difference now is that the argument `grp.col` should have two values to indicate which columns contain data of the between-group factors.

For the analysis of one between-group variable and 2 within-group variables, just the name changes a bit:

```
bwwtrim(J, K, L, x, grp=c(1:p), tr=.2)
```

The Murphy (2004) dataset fills this description, so we'll try to get the data from that dataset into the right format and try the command out. The command `bw2list( )` we saw above functions with 2 within-group variables as well, working to get data in the "wide" form into the correct format. So let's take the `murphy.wide` dataset and work with it:

```
> names(murphy.wide)
[1] "group"          "reg_proto"      "reg_int"        "reg_distant"
[5] "irreg_proto"    "irreg_int"      "irreg_distant" "participant"
```

```
Murphy1<-bw2list(murphy.wide, 1, c(2,3,4,5,6,7))
```

In this command the first argument is the dataset, the second argument is the column indicating the between-group variable, and the third argument indicates which columns the within-group data is stored in, in order of the repeating Similarity variable (I could have just written 2:7 but by writing it out you see that you could list it in the correct order no matter what order the columns originally came in). Now the command should work:

```
bwwtrim(3, 2, 3, Murphy1, tr=.2)
```

There is one more command that Wilcox gives, which is obviously for three within-group variables:

```
wwwtrim(J, K, L, x, grp=c(1:p), tr=.2)
```

If my discussion is not satisfactory I refer the reader to Wilcox (2012) for more details (see Chapter 12). Wilcox does not give any commands for bootstrapping together with three-way designs.

Bibliography

Lyster, R. (2004). Differential effects of prompts and recasts in form-focused instruction. *Studies in Second Language Acquisition*, *26*(4), 399–432.

Murphy, V. A. (2004). Dissociable systems in second language inflectional morphology. *Studies in Second Language Acquisition*, *26*(3), 433–459.

Wilcox, R.R., & Schönbrodt, F.D. (2014). The WRS package for robust statistics in R (version 0.24) [Software]. Available from http://r-forge.r-project.org/projects/wrs/