paircompviz: An **R** Package for Visualization of Multiple Pairwise Comparison Test Results

Michal Burda
University of Ostrava

Abstract

The aim of this paper is to describe an R package for visualization of the results of multiple pairwise comparison tests. Having data that capture some treatments, multiple comparisons test for differences between all pairs of them. The **paircompviz** package provides a function for visualization of such results in Hasse diagram, a graph with significant differences as directed edges between vertices representing the treatments.

Keywords: multiple, pairwise, comparison, test, visualization, Hasse diagram, R.

```
>   library(paircompviz)
```

1. Introduction

Visualization is an important tool for exploration, analysis and presentation of data. A clear and vivid figure or diagram can improve significantly the readibility of the presented facts. In this paper, an implementation of a novel technique for visualization of multiple comparison test results is presented. *Multiple comparison* tests (or *pairwise* tests) occur in testing for differences between all pairs of k treatments (**?**). It is a common fact that for k treatments, a batch of $\binom{k}{2} = \frac{k(k-1)}{2}$ tests has to be performed to compare all pairs.

Typically, the pairwise comparison tests are performed on treatment means, but they may be also applied on other statistics such as variances, quantiles, or proportions.

Multiple comparison procedure usually results in a triangular matrix M of p-values, where the cell m_{ij} of matrix M has p-value of test for difference between treatments i and j. Comparisons with p-value below some given threshold α (usually $\alpha = 0.05$) are considered statistically significant.

Even for as small number of treatments (e.g. $k = 5$), it is rather inconvenient to perceive the results directly. An appropriate visualization may significantly improve the understandability. A common approach for visualization of multiple comparisons is a *line* or *letter diagram* (**??**).

The main ideas of these diagrams are briefly described on the following example. Assume treatments $T_1, \ldots, T_5$ such that T_1 and T_5 is the only significantly different pair. The line diagram is constructed as follows: (1) Sort the treatments accordingly to their tested values (e.g. mean) into increasing order. (2) Draw a line under all largest groups of treatments such that no pair of treatments in that group is significantly different. Since there are two

$$
\begin{array}{ccccc}
T_1 & T_2 & T_3 & T_4 & T_5 \\
\hline
\end{array}
\qquad
\begin{array}{ccccc}
T_1 & T_2 & T_3 & T_4 & T_5 \\
a & a & a & a & \\
 & b & b & b & b
\end{array}
$$

(a) Line diagram   (b) Letter diagram
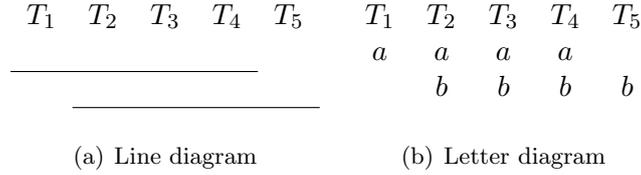
Figure 1: Diagrams of treatments $T_1, \ldots, T_5$ such that only treatments T_1 and T_5 are significantly different, i.e. treatments underlined with the same line or sharing a common letter are not (significantly) different.

$$
\begin{array}{ccccc}
T_1 & T_2 & T_3 & T_4 & T_5 \\
a & & & a & \\
b & b & & & \\
 & & c & c & c \\
 & d & d & & d
\end{array}
$$
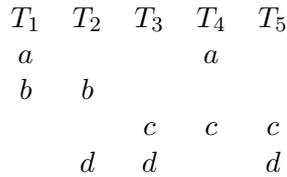
Figure 2: An example of letter diagram for which the corresponding line diagram can not be constructed.

groups of pairwise-non-significant treatments, the resulting line diagram contains two lines – see Figure **??**.

Some authors (e.g. **?**) prefer to use letters instead of lines, because it is sometimes not possible to represent all statements of significance by a diagram with non-interrupted lines. The general idea is to replace each line of the line diagram with a letter – see letter diagram in Figure **??** that is equivalent to line diagram in Figure **??**.

Consider now the only significant treatments be pairs $(T_1, T_5), (T_1, T_3), (T_2, T_4)$. A corresponding letter diagram is depicted in Figure **??**. For that case, a line diagram does not exist since it is not possible to draw all lines without interrupting them (**??**).

Letter diagrams can be quite complicated even for such simple setting as in the previous example. Since letter diagrams visualize groups of non-significant treatments, it may be sometimes hard to immediately determine the significantly different pairs of treatments.

Therefore the package **paircompviz**, a novel R implementation of the visualization technique based on *Hasse diagrams*, is presented in this paper. Unlike letter diagrams, Hasse diagrams explicitly depict the significant pairs in an easy-understandable way. Moreover, additional visual information can be introduced within the diagrams, such as treatment means, sample sizes or p-values. The author believes that Hasse diagrams provide a vivid and easily understandable alternative to letter diagrams for visualization of the all-pairwise comparison test results.

2. Hasse Diagrams

Before describing the proposed visualization technique, some important notions have to be defined.

A *partial order* is a binary relation $Q \subseteq P \times P$, which is reflexive, antisymmetric, and transitive. A *partially ordered set* is a set P equipped with a partial order relation Q.
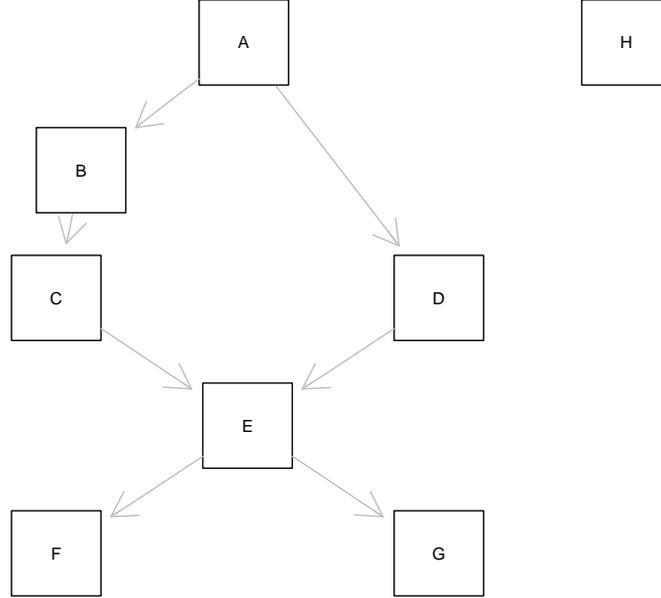
Figure 3: An illustrative Hasse diagram (see the example in section **??**).

Hasse diagram is a simple picture of a finite partially ordered set. It is a graph with each member of a set P being represented as a vertex. A directed edge is drawn downwards from vertex y to vertex x if and only if $(x, y) \in Q$, $x \neq y$, and there is no such z that $(x, z) \in Q \wedge (z, y) \in Q$. Each edge must be connected to exactly two vertices: its two endpoints. In other words, $x, y \in Q$ iff y is positioned above x and there exists a path from y to x.

Example. Consider Hasse diagram in Fig. **??**. As can be seen, node A is greater than all other nodes except H, because there exists a path from A to any other node except H. Nodes C and D are incomparable, although they are both greater than E, F and G. Node H is incomparable with any other node. The whole partial order relation is: $\Big\{ (B, A), (C, B),$ $(C, A), (D, A), (E, C), (E, B), (E, A), (E, D), (F, E), (F, C), (F, B), (F, A), (F, D), (G, E),$ $(G, C), (G, B), (G, A), (G, D), (A, A), (B, B), (C, C), (D, D), (E, E), (F, F), (G, G), (H, H) \Big\}.$

3. Visualization of All-Pairwise Comparison Tests

Consider now an all-pairwise comparison test τ of statistic ψ on k treatments $T_1, \ldots, T_k$. Let $\psi(T_i)$ denote a value of statistic ψ on treatment T_i. For instance, if ψ would be a mean, then $\psi(T_i)$ would denote the mean value of treatment T_i. Let $\tau(T_i, T_j) < \alpha$ denote statistically significant difference between treatments T_i and T_j at the significance level α (e.g. τ could be a two sample Student's t–test).

If a relation

$$H = \Big\{ (T_i, T_j) \mid i = j \vee \Big(\psi(T_i) < \psi(T_j) \ \wedge \ \tau(T_i, T_j) < \alpha \Big) \Big\} \tag{1}$$

is a partial order on $\{T_1, \ldots, T_k\}$, it can be depicted as a Hasse diagram.

Function `paircomp` of the **paircompviz** package is intended to create Hasse diagrams from the result H of multiple comparisons. Graph rendering is performed with the **Rgraphviz** package (**?**).

As an example, consider a `cholesterol` dataset from the **multcomp** package by **?**. It contains measures of cholesterol response (`cholesterol$response`) to 5 treatments (`cholesterol$trt`).

```
>    library(multcomp)
>    head(cholesterol)

     trt response
1 1time    3.8612
2 1time   10.3868
3 1time    5.9059
4 1time    3.0609
5 1time    7.7204
6 1time    2.7139
```

Let us perform a pairwise Student's t test on `response` grouped by `trt`:

```
>    pairwise.t.test(cholesterol$response, cholesterol$trt)

        Pairwise comparisons using t tests with pooled SD

data:  cholesterol$response and cholesterol$trt

       1time    2times   4times   drugD
2times 0.06400  -        -        -
4times 0.00023  0.06870  -        -
drugD  2.8e-07  0.00053  0.06870  -
drugE  1.1e-12  2.1e-09  2.7e-06  0.00139

P value adjustment method: holm
```

We can see many statistically significant results (for significance level $\alpha = 0.05$), however, it is not immediately clear, which group response is greater than other etc. Let us depict the results in a Hasse diagram (see Fig. **??**):

```
>    paircomp(obj=cholesterol$response, grouping=cholesterol$trt, test="t")
```

The `obj` argument is for a response vector and `grouping` is for a grouping factor.

The `test` argument specifies the type of pairwise tests. Possible values are: `"t"`, `"prop"`, and `"wilcox"`, for t test, proportion test and Wilcoxon rank sum test, respectively. See also **stats** package functions `pairwise.t.test`, `pairwise.prop.test` and `pairwise.wilcox.test` for details.

**Pairwise comparisons on cholesterol$response by cholesterol$trt
using t tests with pooled SD and holm p−value adjustment**
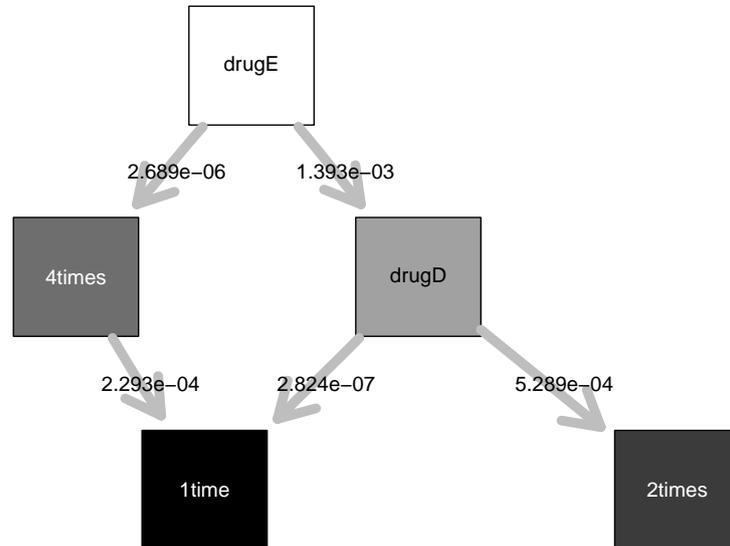
Figure 4: Hasse diagram from pairwise t test on the `cholesterol` data

Other function arguments are: `level`, `main`, `visualize`, `result`, `draw` and `compress`. The last argument is described separately in section **??**.

The `level` argument is for setting the significance threshold α (default is 0.05), `main` specifies the diagram title. If `result` is `TRUE`, the test statistics are also returned from the `paircomp` function. Diagram drawing can be disabled by setting `draw` to `FALSE`.

The `visualize` argument is a vector of string flags that enable drawing of some additional graphical information into the diagram. The meaning of the flags depends on the type of the test being used (t test, Wilcoxon test, or proportion test). If `"position"` is present in the vector, the vertices background color is derived from mean, median, or proportion, respectively, `"size"` enables vertices sizes to correspond to sample sizes, and finally, the presence of `"pvalue"` enables p-value labels along with edges as well as varying edge thickness corresponding to p-value of the comparison test.

There may be also other arguments passed to the `paircomp` function. They are transparently forwarded to the underlying pairwise comparison tests functions (`pairwise.t.test`, `pairwise.prop.test`, or `pairwise.wilcox.test`). For instance, a different p-value adjustment method may be required. For that, a `p.adjust.method` argument exists in the underlying functions:

```
>   paircomp(InsectSprays$count, InsectSprays$spray, test="t",
+           compress=TRUE, p.adjust.method="bonferroni")
```

See help pages for more details.

3.1. Displaying Pairwise Comparisons Provided by the multcomp Package

Another method for pairwise comparisons is provided by the **multcomp** package (**?**). Moreover, that package allows to depict box-plots enriched with *compact letter display* that was discussed in section **??**. The **multcomp** package provides Tukey's test for multiple comparisons by the `glht` function. Results from that function may also be passed to `paircomp` in order to generate a Hasse diagram from it. An example on `car90` dataset from the **rpart** package (**?**) follows:

```
>   library(rpart)        # for car90 dataset
>   library(multcomp)     # for glht() and cld() functions
>   aovR <- aov(Price ~ Type, data = car90)
>   glhtR <- glht(aovR, linfct = mcp(Type = "Tukey"))
>   paircomp(glhtR, compress=FALSE)
```

Compare Hasse diagram in Fig. **??** with compact letter display in Fig. **??**:

```
>   cldR <- cld(glhtR)
>   par(mar=c(4, 3, 5, 1))
>   plot(cldR)
```

3.2. Edge Compression

Sometimes, the pairwise comparison tests may yield in such bipartite setting that each pair of nodes of some two node subsets is inter-connected (without any intra-edge between nodes of the same subset).

More specifically, let U, V be non-empty sets of vertices, $U \cup V \subseteq \{T_1, T_2, \ldots, T_n\}$, $U \cap V = \emptyset$, such that for each $u \in U$ and $v \in V$ there exists an edge from u to v. (The number of such edges equals $|U| \cdot |V|$.) Starting from $|U| > 2$ and $|V| > 2$, the Hasse diagram may become too complicated and hence confusing. Therefore, a `compress` argument exists in the `paircomp` function that enables "compression" of the edges in such a way that a new "dot" node w is introduced and $|U| \cdot |V|$ edges between sets U and V are replaced with $|U| + |V|$ edges from set U to node w and from node w to set V. See figures **??** and **??** for resulting diagrams of the following commands:

```
>   paircomp(InsectSprays$count, InsectSprays$spray, test="t",
+            compress=FALSE)

>   paircomp(InsectSprays$count, InsectSprays$spray, test="t",
+            compress=TRUE)
```

4. Notes on Partial Orderness of Multiple Comparisons

The ability of visualizing all-pairwise comparisons in Hasse diagram relies on whether relation H (see equation **??**) is a partial order. Generally, it is not true for all possible data. For instance, consider artificial data, `brokentrans`[1], that is a part of the **paircompviz** package:

---

[1] `brokentrans` is an artificial dataset that was generated from random numbers in order to provide an example of broken partial orderness with the *t*-test.
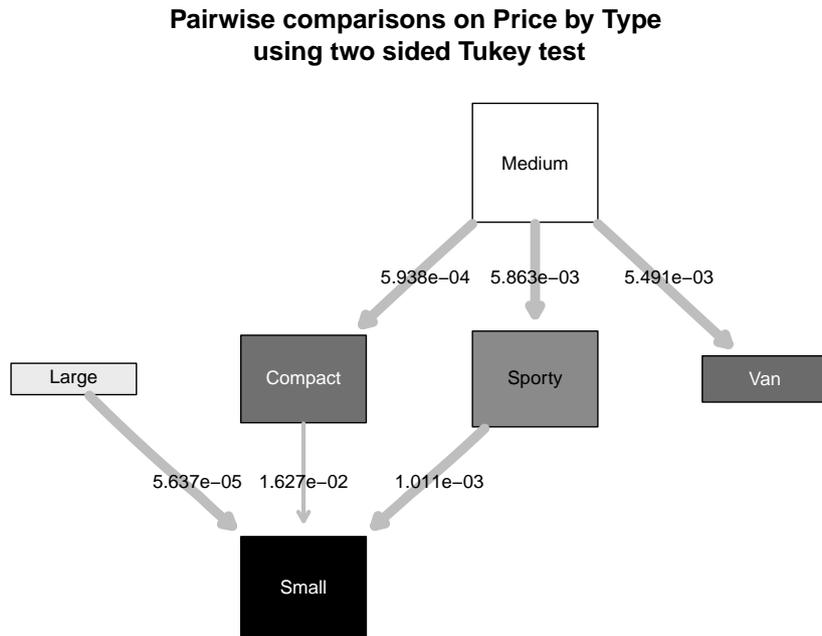
Figure 5: Hasse diagram from results of the `glht` function. Corresponding compact letter display is depicted in Fig. **??** below.
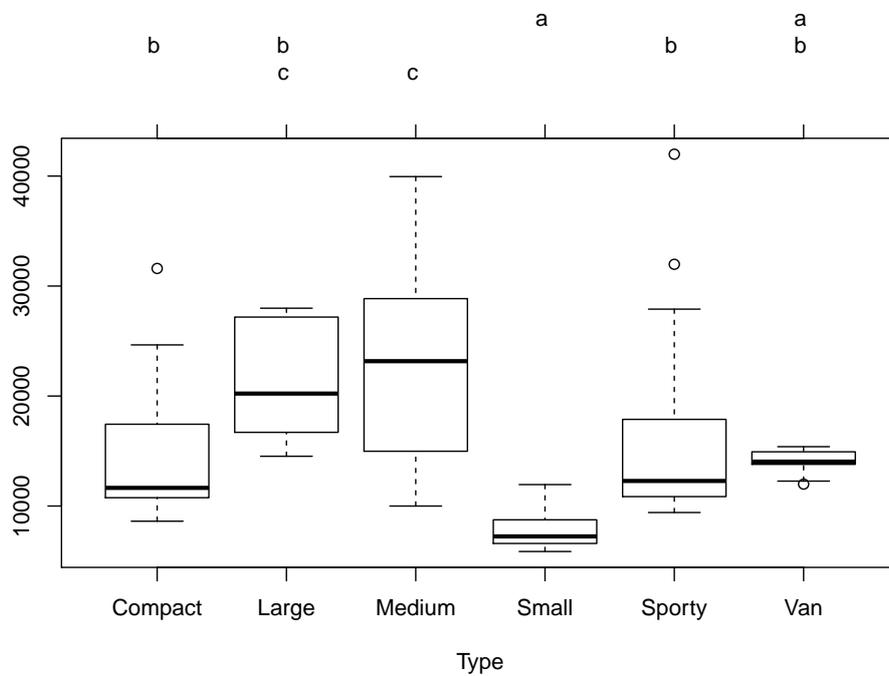

Figure 6: Box-plot with compact letter display on top, as generated by the **multcomp** package. Please compare with the equivalent Hasse diagram in Fig. **??**.

**Pairwise comparisons on InsectSprays$count by InsectSprays$spray
using t tests with pooled SD and holm p–value adjustment**
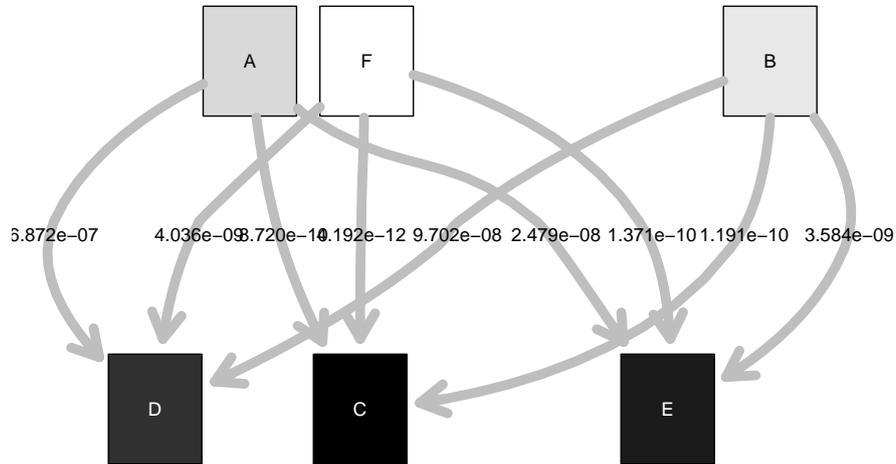
Figure 7: Hasse diagram without compressed edges, i.e. generated by function `paircomp` with argument `compress=FALSE` (compare with Fig. **??**)

**Pairwise comparisons on InsectSprays$count by InsectSprays$spray
using t tests with pooled SD and holm p–value adjustment**
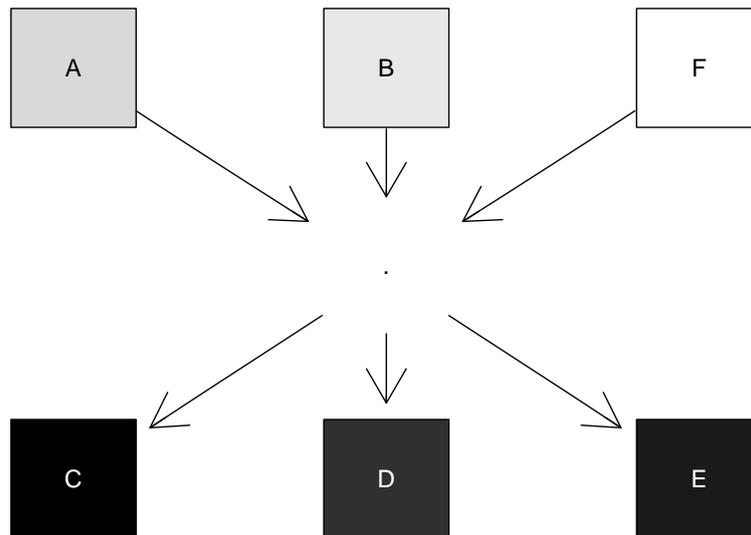
Figure 8: Hasse diagram with edges being compressed by introducing a "dot" edge, i.e. generated by function `paircomp` with argument `compress=TRUE` (compare with Fig. **??**)

Table 1: The distribution of the numbers of grouping factors' levels used in the experiment.

levels	3	4	5	6	7	8	9	10	11
frequency	1428	1087	137	81	50	28	64	134	19
levels	12	14	17	20	21	36	46	52	180
frequency	51	10	6	6	14	2	9	22	2

```
>   data("brokentrans")
>   tapply(brokentrans$x, brokentrans$g, mean)


        1         2         3
 8.240625 42.215852 95.239536


>   pairwise.t.test(brokentrans$x, brokentrans$g, pool.sd=FALSE)


        Pairwise comparisons using t tests with non-pooled SD

data:  brokentrans$x and brokentrans$g

  1       2
2 1.3e-11 -
3 3.5e-09 3.3e-10

P value adjustment method: holm
```

For `level` argument set to $\alpha = 10^{-9}$, we obtain $H = \{(1, 2), (2, 3), (1, 1), (2, 2), (3, 3)\}$, which is clearly not a partial order because of broken transitivity between 1 and 3. The `paircomp` function would stop with an error, in this case.

Although there do theoretically exist data such that its pairwise comparisons does not form a partial order and thus are not viewable in Hasse diagrams, it is rational to estimate the frequency of such settings in real life. For that purpose, a simple experiment was performed by utilizing datasets from packages **cluster** (**?**), **coin** (**?**), **datasets** (**?**), **lattice** (**?**), **MASS** (**?**), **multcomp** (**?**), **party**, **reshape2** (**?**), **rpart** (**?**), **strucchange** (**?**), **survival** (**?**) and **vcd** (**?**).

From that packages, all datasets were added to the experiment, if they contained a factor column with at least 3 levels. For all combinations of factor and numeric columns, various multiple comparison tests were performed and the cases with broken transitivity condition counted. Table **??** summarizes the frequencies of the numbers of grouping levels of datasets used in the experiment.

As can be seen in table **??**, among 3150 different test cases, only 3 of them, all made with the Wilcoxon rank sum test, suffered from broken transitivity that prevented the resulting Hasse diagram to exist. Table **??** provides details about that datasets.

From that we can conclude, that although theoretically possible, broken partial orderness of multiple comparisons appears very seldom, in real data.

Table 2: Occurences of broken transitivity in multiple comparison tests performed on many datasets from various R packages.

test	broken	ok
pairwise comparisons for proportions	0	894
t-test without pooled SD	0	557
t-test with pooled SD	0	570
Tukey test	0	559
Wilcoxon rank sum test	3	567

Table 3: Settings at which broken transitivity appeared in the experiment.

package	dataset	obj	grouping	levels	test
vcd	Bundesliga	Year	HomeTeam	52	Wilcoxon rank sum test
vcd	Bundesliga	Year	AwayTeam	52	Wilcoxon rank sum test
MASS	Boston	tax	rad	9	Wilcoxon rank sum test

5. Direct Hasse Diagram Plot

The **paircompviz** package also provides a function, `hasse`, for direct visualization of a Hasse diagram defined by an adjacency matrix. Prior plotting, transitive edges have to be removed from the adjacency matrix. That can be done by using `transReduct` function. For instance, the Hasse diagram in Fig. **??** was generated by the subsequent code:

```
>    # create adjacency matrix
>    e <- matrix(c(0, 0, 0, 0, 0, 0, 0, 0,
+                  1, 0, 0, 0, 0, 0, 0, 0,
+                  1, 1, 0, 0, 0, 0, 0, 0,
+                  1, 0, 0, 0, 0, 0, 0, 0,
+                  1, 1, 1, 1, 0, 0, 0, 0,
+                  1, 1, 1, 1, 1, 0, 0, 0,
+                  1, 1, 1, 1, 1, 0, 0, 0,
+                  0, 0, 0, 0, 0, 0, 0, 0), nrow=8)
>    # remove transitive edges
>    e <- transReduct(e)
>    # draw Hasse diagram
>    hasse(e, v=LETTERS[1:8], main="")
```

The only required argument, `e`, must be an adjacency matrix with non-zero value e_{ij} if there has to be an edge from node i to node j. The values of matrix `e` are also used to determine edge line thickness.

Argument `v` is a vector of node labels, `elab` is a vector of edge labels, `ecol` for edge label colors, `ebg` for edge line colors, `vcol` and `vbg` for colors of node labels and background, `vsize` for node sizes, `fvlab`, `fvcol`, `fvbg fvsize` for dot node label (defaults to `"."`), foreground

and background colors and size, `febg` and `fesize` set the color and size of edges starting or ending in a dot node. There also may be a diagram title specified in the `main` argument. Edge compression described in section **??** can be turned on by setting the `compress` argument to `TRUE`.

6. Conclusion

The presented **paircompviz** package contains a function `paircomp` to draw a Hasse diagram from the results of multiple pairwise comparison tests. Nodes of the graph represent treatments, edges represent statistically significant difference. Optionally, the shape of the nodes and thickness of the edges respectively depict the size of the samples and strength of the differences.

The novel visualization method is compared to the existing compact letter display technique that is provided by the **multcomp** package. For large set of treatments (say more than 3), Hasse diagram combined with box-plot and compact letter display provides clear and useful additional visual information.

Hasse diagrams may be produced from the results of the *t*-test, Wilcoxon rank sum test or proportion test, that is realized by the **stats** package (**?**) functions `pairwise.t.test`, `pairwise.wilcox.test` or `pairwise.prop.test`, respectively. For convenience, `paircomp` also accepts results from **multcomp**'s (**?**) `glht` function that performs Tukey's test.

Hasse diagram can be constructed for results that partially order the treatments. Since it is theoretically not always the case, an experiment shows that pairwise comparisons violating partial orderness occur very seldom, in reality. From 567 trials of pairwise Wilcoxon rank sum test, partial order violation occured only 3 times. For other types of tests, the violation did not occur in any of more than 500 trials at all.

Acknowledgment

Affiliation:

Michal Burda
Institute for Research and Applications of Fuzzy Modeling
Centre of Excellence IT4Innovations
Division University of Ostrava
30. dubna 22, 701 03 Ostrava, Czech Republic
E-mail: michal.burda@osu.cz