

## **Supplementary Data**

### **Survival Analysis as a Basis to Test Hypotheses When Using Quantitative Ordinal Scale Disease Severity Data**

K.S. Chiang<sup>1</sup>, Y.M. Chang<sup>2</sup>, H.I. Liu<sup>3</sup>, J.Y. Lee<sup>4</sup>, M. El Jarroudi<sup>5</sup> and C.H. Bock<sup>6</sup>

<sup>1</sup> Division of Biometrics, Department of Agronomy, National Chung Hsing University, Taichung, Taiwan

<sup>2</sup> Department of Statistics, Tunghai University, Taichung 407, Taiwan

<sup>3</sup> Bachelor Program in Industrial Artificial Intelligence, Ming Chi University of Technology, New Taipei City 243, Taiwan

<sup>4</sup> Department of Statistics, Feng Chia University, Taichung 407, Taiwan

<sup>5</sup> Department of Environmental Sciences and Management, Université de Liège, 185 Avenue de Longwy, 6700 Arlon, Belgium

<sup>6</sup> USDA-ARS-SEFTNRL, 21 Dunbar Road, Byron, GA 31008, USA

Co-corresponding authors:

K.S. Chiang, email: [kucst@nchu.edu.tw](mailto:kucst@nchu.edu.tw)

Clive H. Bock, email: [clive.bock@usda.gov](mailto:clive.bock@usda.gov)

This document serves as a brief guide for the use of the R analysis program included in the study "Survival Analysis as a Basis to Test Hypotheses When Using Quantitative Ordinal Scale Disease Severity Data." It is divided into three sections: (1) an overview of the sample data, (2) an introduction to the program, and (3) an examination of the analysis performed on the sample data, including interpretation of the results.

## Overview of the sample data

Referring to the citrus canker data statistics (mean: 9.06% and standard deviation (SD): 9.96%) as an example, we generate data that includes four different treatments, each with a sample size of 30. The mean severity of each treatment are 9.06% (treatA), 14.06% (treatB), 4.06% (treatC), 19.06% (treatD), respectively. In addition, considering the assessment results of the Horsfall-Barratt (HB) scale (classes of 1 to 12, representing 0, 0<sup>+</sup>-3, 3<sup>+</sup>-6, 6<sup>+</sup>-12, 12<sup>+</sup>-25, 25<sup>+</sup>-50, 50<sup>+</sup>-75, 75<sup>+</sup>-88, 88<sup>+</sup>-94, 94<sup>+</sup>-97, 97<sup>+</sup>-100, 100% disease), the disease severity can be recorded by class interval mid-point or class value. The recorded values of this simulated data are as follows:

Obs	treatA		treatB		treatC		treatD	
	Mid-point	Class value	Mid-point	Class value	Mid-point	Class value	Mid-point	Class value
1	18.5	5	18.5	5	1.5	2	18.5	5
2	9	4	4.5	3	4.5	3	18.5	5
3	1.5	2	1.5	2	0	1	9	4
4	18.5	5	9	4	9	4	18.5	5
5	18.5	5	9	4	0	1	18.5	5
6	9	4	18.5	5	0	1	37.5	6
7	9	4	9	4	9	4	37.5	6
8	1.5	2	18.5	5	0	1	9	4
9	18.5	5	9	4	0	1	37.5	6
10	1.5	2	9	4	4.5	3	9	4
11	1.5	2	37.5	6	1.5	2	4.5	3
12	4.5	3	9	4	0	1	18.5	5
13	9	4	18.5	5	9	4	18.5	5
14	4.5	3	18.5	5	0	1	37.5	6
15	1.5	2	18.5	5	0	1	9	4
16	1.5	2	1.5	2	1.5	2	37.5	6
17	37.5	6	37.5	6	18.5	5	18.5	5
18	1.5	2	1.5	2	1.5	2	37.5	6
19	1.5	2	4.5	3	0	1	18.5	5
20	9	4	18.5	5	4.5	3	9	4

21	1.5	2	1.5	2	0	1	18.5	5
22	9	4	37.5	6	9	4	18.5	5
23	1.5	2	9	4	1.5	2	18.5	5
24	9	4	4.5	3	1.5	2	4.5	3
25	18.5	5	1.5	2	1.5	2	18.5	5
26	4.5	3	18.5	5	9	4	37.5	6
27	9	4	4.5	3	1.5	2	18.5	5
28	1.5	2	18.5	5	4.5	3	18.5	5
29	1.5	2	9	4	1.5	2	18.5	5
30	4.5	3	18.5	5	1.5	2	37.5	6

## Introduction to the program

We introduce a functional R program named "CompMuCens", which allows for the comparison of multiple treatments means of plant disease severity data through nonparametric survival analysis using class- or interval-based data from a quantitative ordinal scale (QOS). The program is user-friendly and requires only that users input their experiment data in the format of the provided sample data before running the analysis function. The code for the program is as follows:

```
CompMuCens <- function(dat, scale, grade=T, ckData=F){
  #Check if the required packages are installed in the working environment
  if (class(try(library(interval)))=="try-error") {
    if (class(try(library(Icens)))=="try-error") {
      if (!require("BiocManager", quietly = TRUE))
        install.packages("BiocManager")
      #Installing the Icens package, which is required by the Interval package
      BiocManager::install("Icens")
    }
    #Installing the interval package
    install.packages ("interval", repos="http://cloud.r-project.org")
  }
  if (class(try(library(dplyr)))=="try-error") {
    #Installing the dplyr package.
    install.packages("dplyr", repos="http://cloud.r-project.org")
  }
  # Converting data into a censored data format
  if (grade==T) {
```

```

#####class value
if (scale[1]==0) {
  if (scale[length(scale)-1]==100) {
    Scale=unique(scale)
    dat$Slow=ifelse(dat$x==1, 0,
                    ifelse(dat$x==length(scale), 100,
                            suppressWarnings( #Suppress Warnings
                                              as.numeric(lapply(dat$x,
function(x)Scale[x-1])) ) ) )
    dat$Sup=ifelse(dat$x==1, 0,
                   ifelse(dat$x==length(scale), 100,
                           suppressWarnings( #Suppress Warnings
                                              as.numeric(lapply(dat$x,
function(x)Scale[x])) ) ) )
  } else {
    Scale=scale
    dat$Slow=ifelse(dat$x==1, 0,
                    suppressWarnings( #Suppress Warnings
                                      as.numeric(lapply(dat$x, function(x)Scale[x-
1])) ) )
    dat$Sup=ifelse(dat$x==1, 0,
                   suppressWarnings( #Suppress Warnings
                                      as.numeric(lapply(dat$x, function(x)Scale[x])) ) )
  }
} else {
  if (scale[length(scale)-1]==100) {
    Scale=c(0,unique(scale))
    dat$Slow=ifelse(dat$x==length(scale), 100,
                    suppressWarnings( #Suppress Warnings
                                      as.numeric(lapply(dat$x, function(x)Scale[x])) ) )
    dat$Sup=ifelse(dat$x==length(scale), 100,
                   suppressWarnings( #Suppress Warnings
                                      as.numeric(lapply(dat$x, function(x)Scale[x+1])) ) )
  } else {
    if (scale[length(scale)]==100){
      Scale=c(0,scale)
    } else {
      Scale=c(0,scale,100)
    }
  }
}

```

```

    }
    dat$Slow=as.numeric(lapply(dat$x, function(x)Scale[x]))
    dat$Sup=as.numeric(lapply(dat$x, function(x)Scale[x+1]))
  }
}
outPrintD <- dat %>% rename(ClassValue=x)
dat <- dat %>% select (treatment, Slow, Sup)
} else {
#####NPE
if (scale[1]==0) {
  if (scale[length(scale)-1]==100) {
    Scale=unique(scale)
    dat$Slow=ifelse(!dat$x %in% c(0,100),
                    suppressWarnings( #Suppress Warnings
                    as.numeric(lapply (dat$x, function(x)
Scale[max(which(x>Scale))])),
                    dat$x)
    dat$Sup=as.numeric(lapply(dat$x, function(x)
Scale[min(which(x<=Scale))]) )
  } else {
    Scale=scale
    dat$Slow=ifelse(!dat$x==0,
                    suppressWarnings( #Suppress Warnings
                    as.numeric(lapply (dat$x, function(x)
Scale[max(which(x>Scale))])),
                    dat$x)
    dat$Sup=as.numeric(lapply(dat$x, function(x)
Scale[min(which(x<=Scale))]) )
  }
} else {
  if (scale[length(scale)-1]==100) {
    Scale=c(0,unique(scale))
    dat$Slow=ifelse(!dat$x %in% c(0,100),
                    suppressWarnings( #Suppress Warnings
                    as.numeric(lapply (dat$x, function(x)
Scale[max(which(x>Scale))])),
                    dat$x)
    dat$Sup=ifelse(!dat$x==0,

```

```

                                as.numeric(lapply(dat$x, function(x)
Scale[min(which(x<=Scale))])),
                                Scale[2])
    } else {
      if (scale[length(scale)]==100){
        Scale=c(0,scale)
      } else {
        Scale=c(0,scale,100)
      }
      dat$Slow=ifelse(!dat$x==0,
                      suppressWarnings( #Suppress Warnings
                                as.numeric(lapply (dat$x, function(x)
Scale[max(which(x>Scale))])),
                                dat$x)
      dat$Sup=ifelse(!dat$x==0,
                    as.numeric(lapply(dat$x, function(x)
Scale[min(which(x<=Scale))])),
                    Scale[2])
    }
  }
  outPrintD <- dat %>% rename(MidPoint=x)
  dat <- dat %>% select (treatment, Slow, Sup)
}

#Convert to Interval Data (To confirmation the scale input format)
outPrintD$intervals=as.character(Surv(outPrintD$Slow, outPrintD$Sup, type =
"interval2"))
outPrintD <- outPrintD %>% select(-Slow,-Sup)

# Extracting score statistics from each treatment group
mAll=ictest(Surv(Slow, Sup, type = "interval2") ~ treatment,
scores="wmw",data=dat)
#Create a tag data frame for score statistics
anaD=dat %>% mutate(score=mAll$scores) %>%
  group_by(treatment) %>% summarise(score=sum(score)) %>%
  arrange(desc(score)) %>%
  mutate (mk=row_number ()) %>% # "mk" being the descending order of the
score statistic for each treatment.

```

```

as.data.frame()

# Creating a data frame to store the results of the analysis
out=anaD %>% arrange(desc(mk)) %>% mutate (V1=treatment,
                                             V2=lead(treatment),
                                             V1n=mk,
                                             V2n=lead(mk),
                                             pvalue=NA,
                                             pvalue2=NA,
                                             conclusion="",
                                             conc1="") %>%
as.data.frame()

# Creating a copy of the original data
dat1=dat %>% rowwise() %>% #Run code row by row
  mutate (treat=which (anaD$treatment %in% treatment)) # "treat" being the
descending order of the score statistic for each treatment

# Pairwise comparison is performed in a loop.
# The significance level is adjusted based on the Bonferroni adjustment. For
example, if there are three treatments A, B, and C, and the order in which they are
administered is fixed, then we only need to compare A to B and B to C. We don't
need to compare A to C, so the total number of comparisons is 2. Therefore, the
significance level should be alpha/2.
for (i in c(1:(nrow(out)-1))) {
  # Because ictest() performs the comparison according to the order of
treatments, it needs to sort the treatments first.
  testD=dat1 %>% filter(treatment %in% c(out[i,4],out[i,5])) %>%
    select(-treatment) %>% arrange(treat)
  m22=icest(Surv(Slow, Sup, type = "interval2") ~ treat, scores="wmw",
            data=testD, alternative="greater")
  out[i,8]=m22$p.value
  if (out[i,8]>(0.05/(nrow(out)-1))) {
    m221=icest(Surv(Slow, Sup, type = "interval2") ~ treat, scores="wmw",
              data=testD, alternative="two.sided")
    out[i,9]=m221$p.value
    out[i,10]=ifelse(m221$p.value>(0.05/(nrow(out)-1)),
paste0(out[i,4],"=",out[i,5]), paste0(out[i,4],"<",out[i,5]))

```

```

rm(m221)
} else { out[i,10]=paste0(out[i,4], ">", out[i,5]) }

if (i==1){ out[i,11]=out[i,10] } else {
  out[i,11]=substr(out[i,10], start=which( unlist(strsplit(out[i,10], split=""))
%in% c(">", "<", "=") ), stop=nchar(out[i,10]))
}
rm(testD, m22)
}

#Renaming the columns in the analysis results
colnames(out)[c(4,5,8,9)]=c("treat1", "treat2", "p-value for H0: treat1 ≤ treat2",
                             "p-value for H0: treat1 = treat2" )

#Creating the final output file
if (ckData==F){
  out1=list( U.Score=out[,c(1,2)],
            Hypothesis.test=out[-nrow(out), c(4,5,8,9)],
            adj.Signif=0.05/(nrow(out)-1),
            Conclusion=paste(out[-nrow(out),11], collapse = ""))
} else {
  out1=list( inputData=outPrintD,
            U.Score=out[,c(1,2)],
            Hypothesis.test=out[-nrow(out), c(4,5,8,9)],
            adj.Signif=0.05/(nrow(out)-1),
            Conclusion=paste(out[-nrow(out),11], collapse = ""))
}

return(out1)
}

```

The "CompMuCens" program utilizes the `ictest()` function from the *interval* package (Fay and Shaw 2010) to conduct nonparametric survival analysis. Additionally, it employs some functions from the *dplyr* package for data processing. To use the program, the user's R environment must have the *interval* and *dplyr* packages installed (Wickham et al. 2022). To make it user-friendly, the program is designed to automatically detect and install itself.



Arguments of the "CompMuCens" program are as follows:

**dat** A dataset containing two columns of data with specific names, "treatment" and "x," respectively. The "treatment" column records the treatment labels, and the "x" column records the assessment results. For example:

treatment	x
A	5
A	4
A	2
A	5
A	5
A	4
A	4

**scale** The information on the upper limit of each grade of the disease scale used in the assessment. Taking HB scale (categories representing 0, 0<sup>+</sup>-3, 3<sup>+</sup>-6, 6<sup>+</sup>-12, 12<sup>+</sup>-25, 25<sup>+</sup>-50, 50<sup>+</sup>-75, 75<sup>+</sup>-88, 88<sup>+</sup>-94, 94<sup>+</sup>-97, 97<sup>+</sup>-100, 100% disease) as an example, the code should be given as “scale = c(0,3,6,12,25,50,75,88,94,97,100,100) “.

**grade** logical; if TRUE (default), the analysis data is recorded by ordinal rating scores. Otherwise, the analysis data is recorded by mid-point of the severity range.

**ckData** logical; the default is "FALSE". To help users ensure accurate data conversion and avoid errors caused by incorrect scale input formats, this parameter offers the option to display both raw data and the resulting output simultaneously.

An object of the analysis result from the "CompMuCens" program, which is a list with the following components:

**inputData** If "TRUE" is entered for ckData, both the original data and the results of the interval conversion will be displayed.

**Score** Provide the score statistic for each treatment in order to simplify the procedure for pairwise comparisons of treatments. The first step of the analysis procedure, ictest(), is used to carry out the Wilcoxon-Mann-Whitney test. The score statistic is used to preliminarily determine the numerical magnitude relationship of treatment means.

<b>Hypothesis.test</b>	A data frame with four columns is presented. The first two columns contain the treatment labels for a two-treatment comparison, while the third and fourth columns show the <i>p</i> -values of the analysis results for each two-treatment comparison. If the null hypothesis is rejected in the first analysis, the second analysis will not be conducted and the fourth column will display "NA."
<b>adj.Signif</b>	Provide the corrected significance level values. Here the significance level is adjusted based on the Bonferroni adjustment.
<b>Conclusion</b>	Provide the final conclusion of the analysis.

### Examination of the analysis performed on the sample data

The example data is used to account for different scenarios when performing the analysis and illustrating the results of the analysis.

**Example 1.** Comparison of the mean values of the four treatments (class based values) is demonstrated using example disease severity data recorded by the HB class values. The R commands for this analysis are as follows:

```
# Example1
#Entering your data(ordinal rating scores)
trAs=c(5,4,2,5,5,4,4,2,5,2,2,3,4,3,2,2,6,2,2,4,2,4,2,4,5,3,4,2,2,3)
trBs=c(5,3,2,4,4,5,4,5,4,4,6,4,5,5,5,2,6,2,3,5,2,6,4,3,2,5,3,5,4,5)
trCs=c(2,3,1,4,1,1,4,1,1,3,2,1,4,1,1,2,5,2,1,3,1,4,2,2,2,4,2,3,2,2)
trDs=c(5,5,4,5,5,6,6,4,6,4,3,5,5,6,4,6,5,6,5,4,5,5,5,3,5,6,5,5,5,6)
#Data shaping into input format
inputData=data.frame(treatment=c(rep("A",30),rep("B",30),rep("C",30),
                                rep("D",30)),
                    x=c(trAs, trBs, trCs, trDs))
#Perform analysis using CompMuCens() function
CompMuCens(dat=inputData,
scale=c(0,3,6,12,25,50,75,88,94,97,100,100),ckData=T)
```

The outputs are listed as follows:

```
> #Perform analysis using CompMuCens() function
> CompMuCens(dat=inputData, scale=c(0,3,6,12,25,50,75,88,94,97,100,100),ckData=T)
$inputData
```

	treatment	ClassValue	intervals
1	A	5	[12, 25]
2	A	4	[ 6, 12]
3	A	2	[ 0, 3]
4	A	5	[12, 25]
5	A	5	[12, 25]
:	:	:	:
119	D	5	[12, 25]
120	D	6	[25, 50]

← 1

```
$U.Score
```

	treatment	score
1	D	-15.125000
2	B	-4.541667
3	A	4.225000
4	C	15.441667

← 2

```
$Hypothesis.test
```

	treat1	treat2	p-value for H0: treat1 ≤ treat2	p-value for H0: treat1 = treat2
1	D	B	0.0018767018	NA
2	B	A	0.0113215174	NA
3	A	C	0.0007456484	NA

← 3

```
$adj.Signif
```

[1] 0.01666667 ← 4

```
$Conclusion
```

[1] "D>B>A>C" ← 5

In the above screen, the red octagon-embedded numbers 1, 2, 3, 4, and 5 provide information about the original data and the results of interval conversion, the class statistics for each of the treatments, the *p*-values of the analysis results of each two-treatment comparison, the corrected significance values, and the final conclusion, respectively. Based on the final conclusion, it can be determined that the analysis results are consistent with the simulation conditions, thus indicating that the analysis results are correct.

**Example 2.** Comparison of the mean values of the four treatments (based on mid-point values) is demonstrated using the example data assessed using the mid-point value. The R commands for the analysis are as follows:

```
# example2
#Entering the data(ordinal rating scores)
trAm=c(18.5,9,1.5,18.5,18.5,9,9,1.5,18.5,1.5,1.5,4.5,9,4.5,1.5,1.5,37.5,1.5,
        1.5,9,1.5,9,1.5,9,18.5,4.5,9,1.5,1.5,4.5)
```

```

trBm=c(18.5,4.5,1.5,9,9,18.5,9,18.5,9,9,37.5,9,18.5,18.5,18.5,1.5,37.5,1.5,
        4.5,18.5,1.5,37.5,9,4.5,1.5,18.5,4.5,18.5,9,18.5)
trCm=c(1.5,4.5,0,9,0,0,9,0,0,4.5,1.5,0,9,0,0,1.5,18.5,1.5,0,4.5,0,9,1.5,1.5,
        1.5,9,1.5,4.5,1.5,1.5)
trDm=c(18.5,18.5,9,18.5,18.5,37.5,37.5,9,37.5,9,4.5,18.5,18.5,37.5,9,37.5,
        18.5,37.5,18.5,9,18.5,18.5,18.5,4.5,18.5,37.5,18.5,18.5,18.5,37.5)
#Data shaping into input format
inputData=data.frame(treatment=c(rep("A",30),rep("B",30),rep("C",30),
                                rep("D",30)),
                      x=c(trAm, trBm, trCm, trDm))
#Perform analysis using CompMuCens() function
CompMuCens(dat=inputData, scale=c(0,3,6,12,25,50,75,88,94,97,100,100),
grade=F)

```

The outputs are listed as follows:

```

> #Perform analysis using CompMuCens() function
> CompMuCens(dat=inputData, scale=c(0,3,6,12,25,50,75,88,94,97,100,100), grade=F)
$U.Score
  treatment      score
1          D -15.125000
2          B  -4.541667
3          A   4.225000
4          C  15.441667

$Hypothesis.test
  treat1 treat2 p-value for H0: treat1 ≤ treat2 p-value for H0: treat1 = treat2
1      D      B          0.0018767018              NA
2      B      A          0.0113215174              NA
3      A      C          0.0007456484              NA

$adj.Signif
[1] 0.01666667

$Conclusion
[1] "D>B>A>C"

```

The analysis results are completely consistent with Example 1, as the only difference is in the data form (interval mid-point vs. class values). When recording the assessment result as the mid-point of the severity range, remember to change the parameter grade setting in the "CompMuCens" program to "grade=F."

In order to compare with the general paired t test. The pairwise.t.test() function of stats package is used to perform analysis on the data presented in Example 2 at the same time. R commands are as follows:

```
pairwise.t.test(inputData$x, inputData$treatment, "bonf")
```

The outputs are listed as follows:

```
> pairwise.t.test(inputData$x, inputData$treatment, "bonf")

Pairwise comparisons using t tests with pooled SD

data:  inputData$x and inputData$treatment

   A      B      C
B 0.1501  -      -
C 0.2578  0.0002  -
D 0.000000578474 0.0052 0.0000000000025

P value adjustment method: bonferroni
```


Based on the above analysis results, we cannot determine any difference between "treatment A and B" or "treatment A and C". Therefore, we cannot obtain consistent conclusions based on the mid-point simulation using a t-test compared to the interval censored data.

**Example 3.** The comparison of more treatment means is demonstrated in the "CompMuCens" program. To illustrate this, the Example 3 data is based on the Example 1 data, and two additional treatments are added (5th and 6th treatments). The assessment results for the 5th and 6th treatments were set to be the same as those for Treatment A and Treatment D, respectively. The R commands used for this demonstration are shown as follows:

```
# example 3
#Data shaping into input format
inputData=data.frame(treatment=c(rep("A",30),rep("B",30),rep("C",30),
                                rep("D",30), rep("E",30),rep("F",30)),
                    x=c(trAs, trBs, trCs, trDs, trAs, trDs))
#Perform analysis using CompMuCens() function
CompMuCens(dat=inputData, scale=c(0,3,6,12,25,50,75,88,94,97,100,100))
```

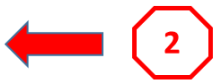
The outputs are listed as follows:

```
> #Perform analysis using CompMuCens() function
> CompMuCens(dat=inputData, scale=c(0,3,6,12,25,50,75,88,94,97,100,100))
$U.Score
  treatment      score
1         F -13.544444
2         D -13.544444
3         B  -2.611111
4         E   6.277778
5         A   6.277778
6         C  17.144444
```




```
$Hypothesis.test
  treat1 treat2 p-value for H0: treat1 ≤ treat2 p-value for H0: treat1 = treat2
1      F      D 0.5000000000 1.000000000
2      D      B 0.0018767018 NA
3      B      E 0.0113215174 0.02264303
4      E      A 0.5000000000 1.000000000
5      A      C 0.0007456484 NA
```

```
$adj.Signif
[1] 0.01
```



```
$Conclusion
[1] "F=D>B=E=A>C"
```



The red octagon-embedded 1 indicates that the null hypothesis cannot be rejected in the first paired treatment comparison, so further hypothesis of treatment comparison is conducted. Because the example 3 contains 6 treatments, the significance level was adjusted accordingly as indicated by the red octagon-embedded 2. Since the significance level was adjusted to be lower (compared to Example 1), the difference between treatment B and treatment E (i.e., treatment A) cannot be detected, so the final conclusion of the analysis is "F=D>B=E=A>C" as indicated by the red octagon-embedded 3.

## References

1. Fay, M. P., and Shaw, P. A. 2010. Exact and asymptotic weighted logrank tests for interval censored data: the interval R package. J. Stat. Softw. 36:1-34.
2. Wickham, H., François, R., Henry, L., and Müller, K. 2022. dplyr: A Grammar of Data Manipulation. R package version 1.0.8. <https://CRAN.R-project.org/package=dplyr>.