Journal of Statistical Software

January 2013, Volume 52, Issue 1. *http://www.jstatsoft.org/*

The cg Package for Comparison of Groups

Bill Pikounis
Johnson & Johnson

John Oleynick
Johnson & Johnson

Abstract

In research of medicines, the comparison of treatments, test articles, conditions, administrations, etc., is very common. Studies are completed, and the data are then most often analyzed with a default mixture of equal variance t tests, analysis of variance, and multiple comparison procedures. But even for an implicit, presumed one-factor linear model to compare groups, more often than not there is the added need to accommodate data which is better suited for expression of multiplicative effects, potential outliers, and limits of detection. Base R and contributed packages provide all the pieces to develop a comprehensive strategy to account for these needs. Such an approach includes exploration of the data, fitting models, formal analysis to gauge the magnitude of effects, and checking of assumptions. The **cg** package is developed with those goals in mind, using a flow of wrapper functions to guide the full analysis and interpretation of the data. Examples from our non-clinical world of research will be used to illustrate the package and strategy.

Keywords: strategy, one factor, resistance, robustness, censoring, R.

1. Introduction

In our experiences as statisticians in research & development of pharmaceuticals, comparisons are sought in a large portion of experiments and studies. Any statistical evaluation of data towards that purpose virtually always leads to a t test of some kind. The concept of "t test" here is defined as the calculation of the ratio of some estimate to its corresponding estimate of its variability, and then using a t distribution as a reference for generating a p value or similar quantity to claim statistical significance.

For the common situation of continuous outcomes when comparisons are of interest, most data analysis is done without the aid of a statistician. This is certainly true in our direct experiences as pre-clinical/non-clinical statisticians in the aforementioned research & development environment of pharmaceuticals. Along with that, researchers primarily rely on available software for learning about and practicing statistics and doing quantitative assessment. We

feel this has a direct impact on how well or how poorly information and value is extracted from data to answer the scientific questions of interest.

Even for the most familiar data structure of a one-way layout of groups, the capability of comparisons in software packages may not consist of more than the omnibus F test. Most have at least follow-up pairwise comparisons, with perhaps a choice of various multiple comparisons procedures. The better ones also provide graphs of individual values. These are all important parts of the data evaluation, and should all be performed for a complete evaluation and subsequent interpretations of the data.

The **cg** package is developed with the overall goal of a comprehensive analysis of data when comparison of groups is a primary interest. It is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=cg`. A flow of wrapper functions are contained within it to guide the full analysis and interpretation of the data. The wrapper functions encompass functions in base R (R Development Core Team 2012) and other packages. While the **cg** package can be used by someone fluent in R, it is also developed with the intent of integration with front-end graphical user interfaces. No such integrated interfaces are yet established within our company.

Data examples contained in the **cg** package are introduced in Section 2 below. Section 3 discusses features of the package, along with the specific data analysis issues they were developed to address. Section 4 illustrates those features on the two data sets introduced in Section 2.

Much of the philosophy behind the **cg** package is expounded in a chapter of Millard and Krause (2001), written by the first author of this article. See Pikounis (2001) in particular. Therefore, context about the choices of techniques in the **cg** package are only briefly touched upon, as the focus is on the use of the package. Detailed commentary can be found in the aforementioned Pikounis (2001) reference.

2. Data examples

Two data sets will be used as running examples throughout this manuscript. Both are included in the **cg** package.

2.1. Canine prostate volume data

The purpose of this experiment was to evaluate the effect of a physiological dose of estradiol on prostate growth in dogs using ultrasound. See Rhodes *et al.* (2000) for details. Comparisons amongst all five groups are of interest. Assuming the package is loaded via

```
R> library("cg")
```

for this and all following code, the data frame within the **cg** package is shown with

```
R> canine
```

```
      AE     E1     E2    CC     NC
1  9.132 10.356 37.200 1.975  9.301
2 10.070  6.313 12.639 3.125 13.531
3 20.077 21.708 16.791 4.433 12.840
4 14.691 12.651 36.996 6.154 14.336
5 23.698 15.464 22.808 4.175 25.102
```

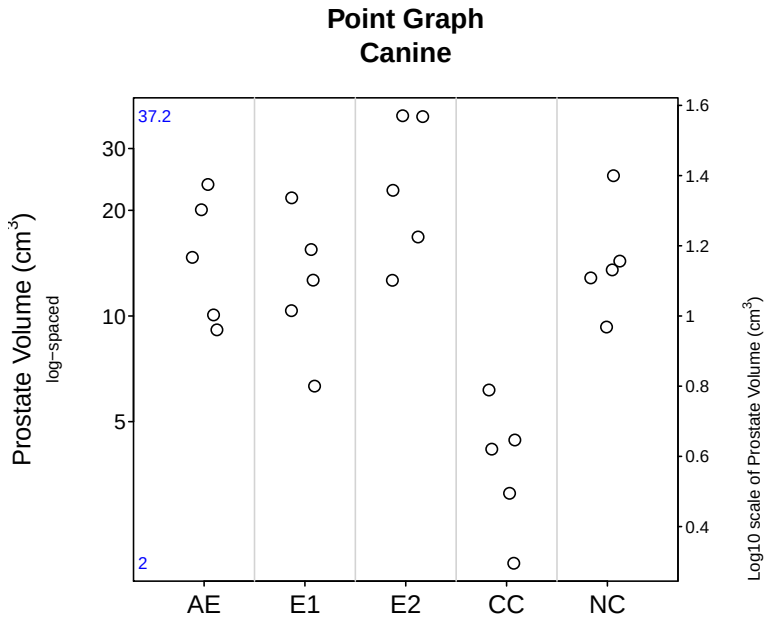

Figure 1: Point graph of the canine prostate volume data.

and Figure 1 shows these values. The calls in the **cg** package to create Figure 1 are covered in the applications in Section 4. The five groups are defined as

`AE`: Castration plus estradiol and androstanediol.

`E1`: Castration plus low dose estradiol.

`E2`: Castration plus high dose estradiol.

`CC`: Castration alone.

`NC`: No castration (normal controls).

The measured outcome in the data set is prostate volume. Units of cubic centimeters is the standard in this research field (cm^3). There are 5 observations in each of the five groups.

2.2. GM-CSF cytokine data

GM-CSF is an acronym for Granulocyte macrophage colony stimulating factor, a type of cytokine that is important in the growth of white blood cells. It is one of the outcomes measured in the experiment described in Shealy *et al.* (2010). Therapeutic inhibition of it may be beneficial in cases where too many white blood cells are produced, such as arthritis. In other situations where white blood cell counts are low, stimulation of it is desired. In Shealy *et al.* (2010), GM-CSF is evaluated in the context of inflammation, and measured in units of picograms per milliliter.

The data frame within the **cg** package is shown with

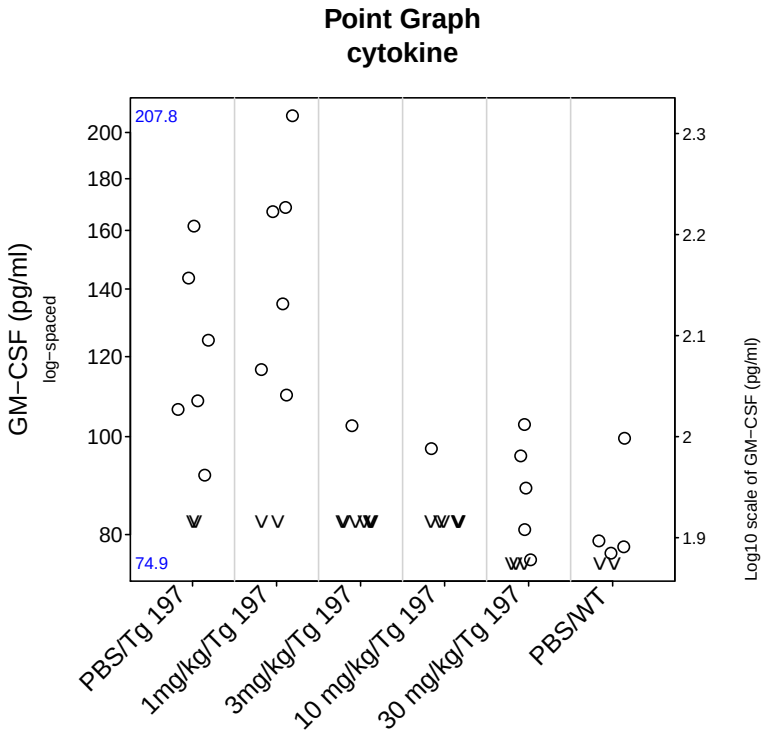

Figure 2: Granulocyte macrophage colony stimulating factor (GM-CSF) cytokine data.

```
R> print(gmcsfcens, row.names = FALSE)
```

PBS/Tg 197	1mg/kg/Tg 197	3mg/kg/Tg 197	10 mg/kg/Tg 197	30 mg/kg/Tg 197	PBS/WT
143.535	116.515	<82.5	97.31	<74.94	<74.94
108.51	207.785	<82.5	<82.5	75.53	76.68
124.575	109.94	102.525	<82.5	88.94	78.86
91.6	168.595	<82.5	<82.5	<74.94	99.63
161.575	166.99	<82.5	<82.5	102.805	<74.94
<82.5	<82.5	<82.5	<82.5	95.71	77.8
<82.5	135.34	<82.5	<82.5	80.91	
106.4	<82.5	<82.5		<74.94	

and Figure 2 shows these values. The individual group values in the data frame are of mode character, since some of them are represented as left-censored values such as `<82.5`. These are due to limits of detection in the measurement capabilities of the assay. Note that two of the groups have less than 8 observations, and the corresponding cells in the data frame actually contain empty quote `""` values.

The six groups are defined as

`PBS/Tg 197`: Phosphate buffered saline control group.

`1mg/kg/Tg 197`: 1 mg/kg dose.

`3mg/kg/Tg 197`: 3 mg/kg dose.

`10/mg/kg/Tg 197`: 10 mg/kg dose.

`30/mg/kg/Tg 197`: 30 mg/kg dose.

`PBS/WT`Phosphate buffered saline control group of wild-type mice.

The first five groups have transgenic (Tg197) mice subjects, a well established model to induce arthritis. The sixth group `PBS/WT` did not have arthritis induced. The various doses of the inner four groups are administrations of *golimumab*, a monoclonal antibody therapy.

The data frame format of the two above example data sets are not standard R convention. However, they do represent a familiar way for scientists to store data in spreadsheet software. Microsoft Excel, for example, requires this format or its transpose for use of its `Anova:Single Factor Data Analysis` function. Commensurate with the intent to integrate **cg** with a front-end graphical user interface within our own research organization, this one-column-per-group format provides a familiar way to represent data for an analysis run.

The calls in the **cg** package to create Figure 2 are covered in the applications in Section 4. Section 3 discusses handling of these limit-of-detection values as left-censored for the overall evaluation. For now we just point out that the down-pointed arrowheads in the graph signify these limit-of-detection observations.

3. Features

3.1. Strategy

Figure 3 summarizes the workflow design of the **cg** package for comprehensive data analysis of a one-way layout. The `prepare` and `fit` portions represent actual **cg** calls in the analysis sequence, and are described in Section 3.2. The possible output falls into four categories:

Exploratory: Data graphs and descriptive summaries.

Formal analysis: Estimates, comparisons, and significance tests.

Diagnostics: Assessment of fitted model(s).

Sample size estimation: Guidance for subsequent studies.

The exploratory and diagnostics portions critically depend on good data graphs. The Formal Analysis focuses on magnitudes and expressions of effects from a model or models fitted to the data. The Sample Size Estimation portion utilizes the estimated variability from the data to calculate power for a potential subsequent study.

3.2. Specifics

This section describes general features of the **cg** package. Section 4 illustrates the features with specific examples.

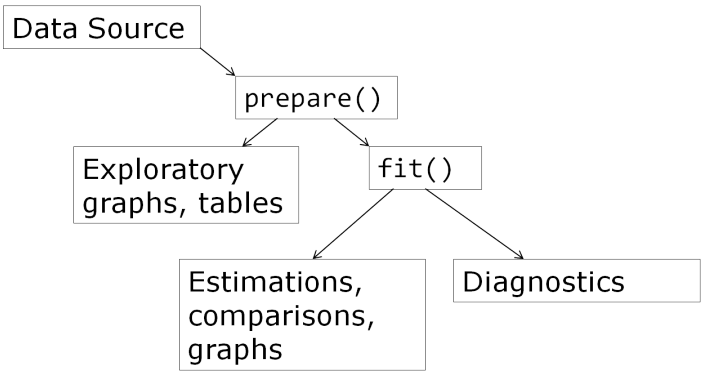

Figure 3: Strategy of **cg** evaluation.

Data preparation

The `prepare` function wraps the `prepareCGOneFactorData` function in the **cg** package. The first argument `dfr` takes a data frame that is a type of `format`, the second argument. The value of `format` is either `"listed"` or `"groupcolumns"`. Sufficiently unique abbreviations such as `format = "g"` for `format = "groupcolumns"` can be used.

The canine prostate volume data shown earlier in Section 2.1 follows the `"groupcolumns"` format. As mentioned at the end of Section 2, we have found this a more conventional way for scientists to represent such one-factor data structure since it is required by software such as **Excel** or **GraphPad Prism** (**GraphPad** Software 2010).

The `"listed"` version of the same data is also a data set in the **cg** package, shown here as abbreviated output.

```
R> canine.listfmt

   grpf  endpt
1    AE  9.132
2    AE  10.07
...
11   E2   37.2
...
24   NC 14.336
25   NC 25.102
```

For data sets that have censored observations, both formats can be used as well. The example seen in Section 2.2 follows the `format = "groupcolumns"` type. The use of the `<` symbol for left-censored observations, or analogously the `>` symbol for right-censored observations, will make the input data frame of mode character. We acknowledge that this format as illustrated by the data frame `gmcsfcens` is not a standard format to represent censored observations. It is one we would hope that could be adopted by scientists to represent such observations as an extension of the `groupcolumns` format. The `prepareCGOneFactorData` function ensures the needed conversion to numeric data and the status representation of censored or complete observations following the `Surv` function conventions in the **survival** package (Therneau and Lumley 2010).

The `format = "listed"` specification handles either two, three, or four column versions. In all cases the first column must have the group level values that define the factor. The help file for `prepareCGOneFactorData` details all the possibilities for the remaining data columns. Appendix A also illustrates the processing of these alternative data frame input formats.

`two columns`: The `<` or `>` representations should be used for the second column values, such as seen with the `"groupcolumns"` format in Section 2.2. See the data frame `gmcsfcens.listfmt1` and its help file in the **cg** package. Alternatively, a single "threshold" number could be assigned to the `leftcensor` or `rightcensor` arguments in a `prepareCGOneFactorData` call to create censored values for observations that are equal to or less than or greater than these thresholds.

`three columns`: The second column contains the response values, and the third is the status according to `Surv` in the **survival** package: `0` = right-censored, `1` = no censoring, and `2` = left-censored. See the data frame `gmcsfcens.listfmt2` and its help file in the **cg** package. As in the previous item, `leftcensor` or `rightcensor` will need to be specified if right-censored or left-censored observations occur, but not both.

`four columns`: The second and third columns need to have numeric response information, and the fourth column needs to have censoring status. This is the most general representation, where any combination of left-censoring, right-censoring, and even interval-censoring is permitted, following the convention in the **survival** package, with `type = "interval"` in the `Surv` call. See the data frame `gmcsfcens.listfmt3` in the **cg** package.

All of the input data formats lead to a `cgOneFactorData` object that has methods to perform exploratory evaluations, model fitting and subsequent formal analyses, fit diagnostics, and sample size calculations. A log transformation is associated with the data object by default through `logscale = TRUE` in the `prepareCGOneFactorData` call. The actual application of the logarithmic transformation to the data, and back transformation to original scale, is actually done in the methods. This choice of `logscale = TRUE` as the default comes from the dominant frequency of positive valued, right skewed endpoints in our experience. There is also a dominant desire by scientific colleagues to express differences in terms of percent change and ratios. Most of the time statistical assumptions of Gaussian errors and equal variances are more reasonably met as well.

When zero is an endpoint value the log scale is undefined, so two approaches are offered by **cg**. The first is to replace the zeroes with a small value determined from the data that is less than the minimum of non-zero values. This "zero score" is determined by the concept of arithmetic-logarithmic scaling as discussed in Tukey, Ciminera, and Heyse (1985), and is only enabled for data with no censoring. A numeric value could also be specified for the `zeroscore` argument, and this would enable use for data sets that have censored observations. The second approach is to add a constant to all the endpoint values. This could be supplied with a numeric value to the `addconstant` argument, or estimated. The `addconstant = "simple"` estimation approach is just `0.0001 * (max(x) - min(x))` of the `x` endpoint vector; see Chambers and Hastie (1992, p. 68). The `addconstant = "VR"` approach relies on the `logtrans` function from the **MASS** package and is detailed in Venables and Ripley (2002, pp. 171–172).

Exploratory

Once the `cgOneFactorData` object has been created, there are three graphical methods and one tabular method for it in the Exploratory category. The `pointGraph` method produces a one dimensional scatter plot in the same spirit of the `stripchart` function in the **graphics** package. It takes care of log spacing the y-axis with presentation of tick marks in the original scale on the left axis margin, and `log10` values on the right axis margin. Points are represented as open circles and jittered to alleviate overlap. Minimum and Maximum values are displayed in the top and bottom left corners just inside the plot region. If there are censored observations, they are represented with `<` and `>` characters that are turned downward for left-censored observations, and upward for right-censored observations. See earlier Figure 2. A `boxplot` method also portrays the axis, minimum, maximum and extreme censored values the same way. For the quantile estimates of the box when there are censored observations, Kaplan-Meier type estimates are used as proposed by Gentleman and Crowley (1991). The third graphical method is called `kmGraph`, and produces non-parametric survival or cumulative distribution step line functions.

A `descriptiveTable` method creates a summary table object of quantiles, means and variability estimates for each of the groups. An additional print method produces moderate formatting of the table. If censored values are present in any of the groups, columns on the number of complete and censored observations are added to the table. For all of the above methods, labels for titles and axes are drawn from the `cgOneFactorData` object settings slot components created by the `prepareCGOneFactorData` call, such as `analysisname`, `endptname`, and `endptunits`.

Formal analysis

Model fit objects are created with the `fit` method call on a `cgOneFactorData` object. By default, the classical least squares fit through `lm` will be performed. A resistant & robust model fit via `rlm` from the **MASS** package will also be carried out. Except for the specification of `method = "MM"` and the allowance of the number of iterations, the call of `rlm` uses its default arguments. If there are censored data observations, then an accelerated failure time model is fit via the `survreg` function in the **survival** package. The `survreg` call is made with `dist = "gaussian"`. By default the "robust sandwich" estimator for the variance-covariance matrix of the estimated group means is used but can be overriden in the `fit` call. The `type` argument in the `fit` method can be explicitly specified, and this is required in the case where a fit permits unequal variances amongst the groups. This unequal variance case, designated by `type = "uv"`, uses the `gls` function with the appropriate `varWeights` specification from the **nlme** package (Pinheiro, Bates, DebRoy, Sarkar, and R Development Core Team 2009).

The `fit` call creates a `cgOneFactorFit` object. Methods for the objects include graphs and tables that focus on estimation of group means and differences amongst those means. Wald-type "t tests" of significance and corresponding confidence intervals are calculated and presented. Expressions and magnitudes of effects and differences are automatically handled if the log scale is used, in the sense of presentation of percent differences, geometric means, and axis spacing and tickmark back-transformations. When adjustment for multiple comparisons is requested, critical quantiles and probabilities are generated from the **multcomp** package (Hothorn, Bretz, and Westfall 2008). Appendix B demonstrates some balanced and unbalanced group sample size configurations for determining critical points when either the

well-known Tukey's honest significant difference test for all possible pairwise comparisons, or Dunnett's test for comparison of all groups to one control, would be applied. The `glht` function in the **multcomp** package aligns very well with these approximations of the multivariate t distribution associated with the generated family of comparisons; see Appendix B. Of course, **multcomp** handles any other custom built families of comparisons, and is also appealing to use with other models such as accelerated failure time parametric fits to censored data. The needed calculation time for the critical point is a matter of a few seconds on today's computers. We feel this is a worthy tradeoff since the use of the **multcomp** generated critical points relieves the burden of which multiple comparison procedure to choose.

It is acknowleged that the use of degrees of freedom for Wald-type "t test" comparisons based on the resistant & robust model fits of the **cg** package are empirical approximations. But similar to the above reasoning for the use of **multcomp** for multiplicity adjustment, this has proven in practice to be worthwhile, especially in contrast to the well-intentioned but flawed use of outlier tests and rules by scientists to exclude points in the absence of any experiment-based criteria.

Diagnostics

The main methods which work on a `cgOneFactorFit` object are called `varianceGraph` and `qqGraph`, to focus on the equal variance and Gaussian error model assumptions, respectively. When a resistant & robust fit is part of the object, weighted residuals and un-weighted residuals versions of graphs can be created. Smoothing trend lines for the equal variance graphs are added with the conventional `lowess` function. For censored data residuals, an experimental ad hoc approach uses cubic smoothing splines from the **VGAM** package (Yee 2010), which are designed to accommodate censored data. Also for resistant & robust fits, there is a `downweightedTable` method which can list individual observations that are downweighted more than a specified threshold. These weight estimates come from the `rlm` fit portion of the object.

Sample size estimation

Companion `samplesizeTable` and `samplesizeGraph` methods take a `cgOneFactorFit` object and use the variability estimates from the classical least squares and resistant & robust fits to estimate prospective sample sizes. The sample size calculation is based on the global F test statistic, so two or more groups may be specified. See for example Fleiss (1986, Appendix A, pp. 371–376). There is no method currently implemented for censored data. In addition to the usual required inputs of power, significance level, and postulated differences, the method coordinates the expression of percent differences based on log-scale analyses.

4. Applications

4.1. Canine prostate volume data

Figure 1 showed the canine prostate volume data introduced in Section 2.1. Let us prepare the `cgOneFactorData` class object with the `canine` data frame and the following call:

```
R> canine.data <- prepareCGOneFactorData(canine, format = "groupcolumns",
```

```
+     analysisname = "Canine", endptname = "Prostate Volume",
+     endptunits = expression(plain(cm)^3), digits = 1, logscale = TRUE)
```

The `digits` argument controls the display of output decimal places for method calls on the `canine.data` object, as will be shown shortly. Figure 1 is produced by the call `pointGraph(canine.data)`. An exploratory summary table is presented next.

```
R> descriptiveTable(canine.data)
```

```
Descriptive Table of Canine
Endpoint: Prostate Volume
   n  Min 25%ile Median 75%ile  Max Mean StdDev StdErr GeoMean SEGeoMean
AE 5  9.1   10.1   14.7   20.1 23.7 15.5    6.3    2.8    14.5       2.7
E1 5  6.3   10.4   12.7   15.5 21.7 13.3    5.8    2.6    12.3       2.5
E2 5 12.6   16.8   22.8   37.0 37.2 25.3   11.4    5.1    23.2       5.0
CC 5  2.0    3.1    4.2    4.4  6.2  4.0    1.6    0.7     3.7       0.7
NC 5  9.3   12.8   13.5   14.3 25.1 15.0    6.0    2.7    14.2       2.3
```

In the descriptive table we see moderate formatting with the `analysisname` and `endptname` character values drawn from the `canine.data` object, and all entries in the display rounded to one decimal place due to the initial `digits = 1` setting. Note the presence of the last two columns that refer to geometric means and their estimated standard errors, while the previous three columns present the arithmetic means, their standard deviations, and their standard errors.

Classical least squares and resistant & robust fits to the data are accomplished with

```
R> canine.fit <- fit(canine.data)
```

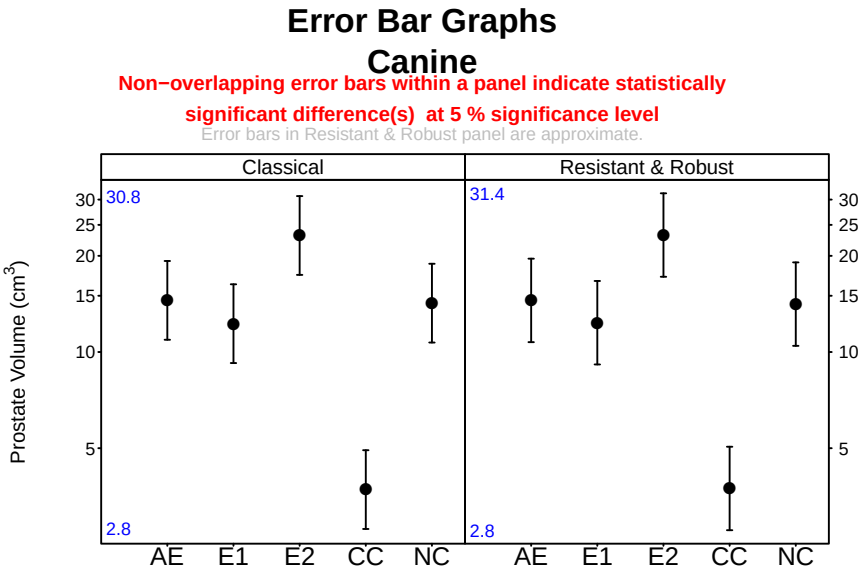

Figure 4: Canine prostate volume data error bar graph.

Differences amongst the groups can then be shown as in Figure 4 with

```
R> errorBarGraph(canine.fit)
```

This last call provides a concise summary of the group means and potentially significant differences via the method of Andrews, Snee, and Sarner (1980). Since the sample sizes are equal across the groups, the multiple of the standard error derived by the method permits quick visualization of which group means would be found to be statistically significantly different from each other in all possible pairwise comparisons. The Firth and De Menezes (2004) approach of quasi-variances is similar in spirit, and applies widely to R fit classes such as `glm`. For the `canine` data, the `plot.qv` method from the **qvcalc** package (Firth 2010) would map to the `errorBarGraph` error bar lengths by scaling each of the corresponding quasi-standard errors by $\sqrt{2}$.

Since there were no outlying points apparent in Figure 1, the least squares and resistant & robust results line up closely. In some cases the area to judge between overlap and non-overlap of error bars will be difficult, and reference to the actual results would be necessary. The `comparisonsTable` method shows the numeric details of the differences:

```
R> comparisonsTable(canine.fit, model = "olsonly")
```

```
Comparisons Table of Canine
Endpoint: Prostate Volume
Percent Differences (A vs. B)

Least Squares Model Fit
95% Confidence (alpha of 0.05)
```

	estimate	se	lowerci	upperci	pval	geomeanA	seA	geomeanB	seB
E1 vs. AE	-15	23	-52	49	0.544	12.3	2.4	14.5	2.8
E2 vs. AE	60	43	-10	182	0.101	23.2	4.5	14.5	2.8
CC vs. AE	-74	7	-86	-55	<0.001	3.7	0.7	14.5	2.8
NC vs. AE	-2	27	-44	73	0.942	14.2	2.7	14.5	2.8
AE vs. E1	18	32	-33	109	0.544	14.5	2.8	12.3	2.4
E2 vs. E1	89	51	7	233	0.030	23.2	4.5	12.3	2.4
CC vs. E1	-70	8	-83	-47	<0.001	3.7	0.7	12.3	2.4
NC vs. E1	16	32	-34	105	0.593	14.2	2.7	12.3	2.4
AE vs. E2	-37	17	-64	11	0.101	14.5	2.8	23.2	4.5
E1 vs. E2	-47	14	-70	-7	0.030	12.3	2.4	23.2	4.5
CC vs. E2	-84	4	-91	-72	<0.001	3.7	0.7	23.2	4.5
NC vs. E2	-39	17	-65	8	0.088	14.2	2.7	23.2	4.5
AE vs. CC	291	106	122	590	<0.001	14.5	2.8	3.7	0.7
E1 vs. CC	231	90	87	483	<0.001	12.3	2.4	3.7	0.7
E2 vs. CC	524	170	254	1001	<0.001	23.2	4.5	3.7	0.7
NC vs. CC	283	104	117	576	<0.001	14.2	2.7	3.7	0.7
AE vs. NC	2	28	-42	80	0.942	14.5	2.8	14.2	2.7
E1 vs. NC	-14	23	-51	52	0.593	12.3	2.4	14.2	2.7
E2 vs. NC	63	44	-8	187	0.088	23.2	4.5	14.2	2.7
CC vs. NC	-74	7	-85	-54	<0.001	3.7	0.7	14.2	2.7

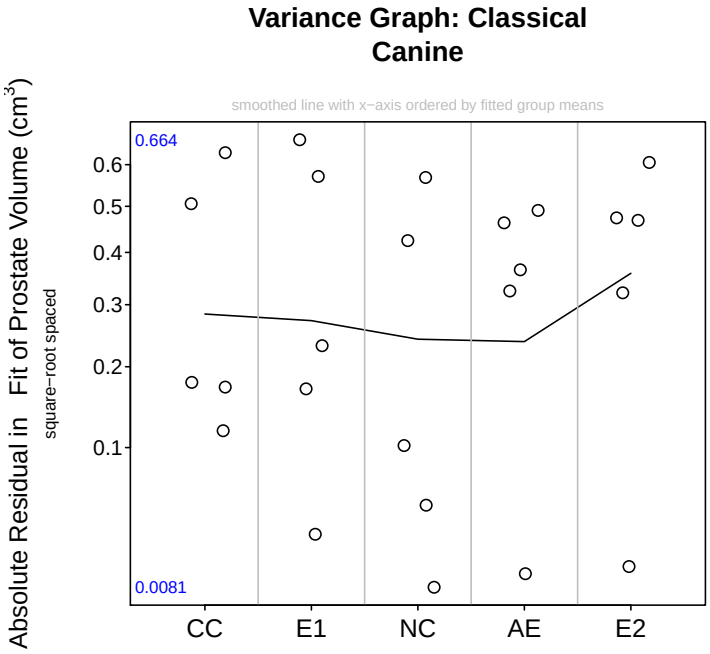

Figure 5: Canine prostate volume data equal variance graph.

The E1 vs. E2 comparison, which for example looks close to overlap in Figure 4, is indeed significant at the conventional 5% level since the p value is $P = 0.030$. Note how the above table displays the group differences in terms of percent, and the group means are shown in the original scale, as the method takes care of the back-transformations from the log-scale of the model fit to geometric means and subsequent evaluations. Omission of the `model = "olsonly"` specification in the call would have produced the analogous resistant & robust table version as well. All methods that operate on `cgOneFactorFit` objects that have both fits have the same capability of inclusion or exclusion.

More examples follow for the diagnostic graphs:

```
R> varianceGraph(canine.fit, model = "olsonly")
```

The groups are ordered by increasing mean in Figure 5. The use of the square root of the absolute residual to focus the evaluation on the equal variance assumption comes from a concept by John Tukey (Cleveland 1993).

```
R> qqGraph(canine.fit, model = "extended")
```

The middle and right panels of the Q-Q graph in Figure 6 respectively use unweighted and weighted residuals from the resistant & robust fit. There is not much to distinguish amongst the three panels for this data set, as the use of the log transformation shows the underlying model assumptions to be reasonably met.

If another similar study was planned, the sample size methods available in **cg** could be used to create a table and graph that show estimated required sample sizes.

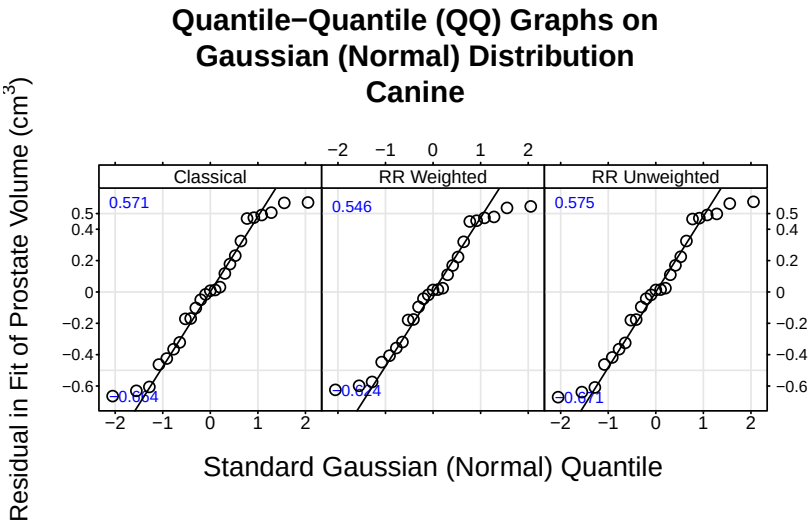

Figure 6: Canine prostate volume data graph of quantile-quantile Gaussian errors.

```
R> canine.samplesize <- samplesizeTable(canine.fit,
+    direction = "increasing", mmdvec = c(10, 25, 50, 75, 100))

Sample Size Table from Canine
Endpoint: Prostate Volume
Percent Increasing Differences
80% Power and 5% Significance Level

Least Squares Based
Variability Estimate (Log scale) of 0.4303
2 Groups
    n per group N Total
 10         321     642
 25          60     120
 50          19      38
 75          11      22
100           8      16

Resistant & Robust Based
Variability Estimate (Log scale) of 0.4559
2 Groups
    n per group N Total
 10         361     722
 25          67     134
 50          21      42
 75          12      24
100           8      16

R> samplesizeGraph(canine.samplesize)
```

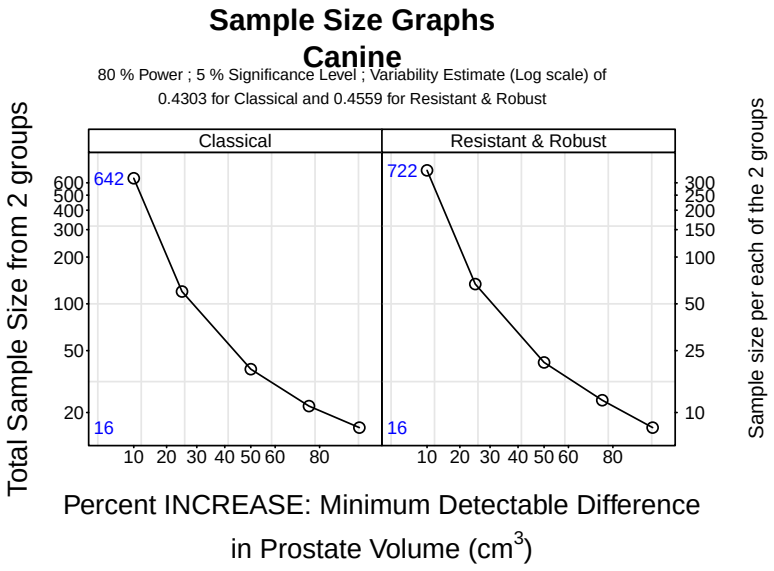

Figure 7: Canine prostate volume data sample size graph.

While the Figure 7 graph is essentially a portrayal of the table, it has been a useful conceptual aid for us in collaborations with scientists. It reinforces, for example, that smaller sample sizes require larger true underlying differences. As in earlier figures, there is summary text in the margins to clarify the content and/or interpretation of the graph.

4.2. GM-CSF cytokine data

Recall in Section 2.2 that this cytokine data is represented with left-censored observations to handle limits-of-detection in the measurement process. The data frame object printed there is `gmcsfcens`. Figure 2 showed the point graph version created by **cg**, where these left-censored values are shown with left-down arrow symbols. The call

```
R> gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens,
+    format = "groupcolumns", analysisname = "cytokine",
+    endptname = "GM-CSF (pg/ml)", logscale = TRUE, digits = 1)
```

creates the `cgOneFactorData` class object, and `pointGraph(gmcsfcens.data)` produces the Figure 2 point graph. Appendix A shows some alternative data frame input formats and code to create the object.

Section 3.2 mentioned the suggested (Gentleman and Crowley 1991) use of Kaplan-Meier type quantile estimates of the distribution when there are censored data. The `boxplot` method in **cg** wraps the `survfit` functionality from the **survival** package. This handles the potential of left-censored observations in addition to the canonical Kaplan-Meier case of right-censored observations, as summarized in the `survfit.formula` help page.

From the call

```
R> boxplot(gmcsfcens.data)
```

some of the groups in Figure 8 exhibit the inability to estimate quantiles when too many censored data observations are present. Note the rotation of the group labels on the x-

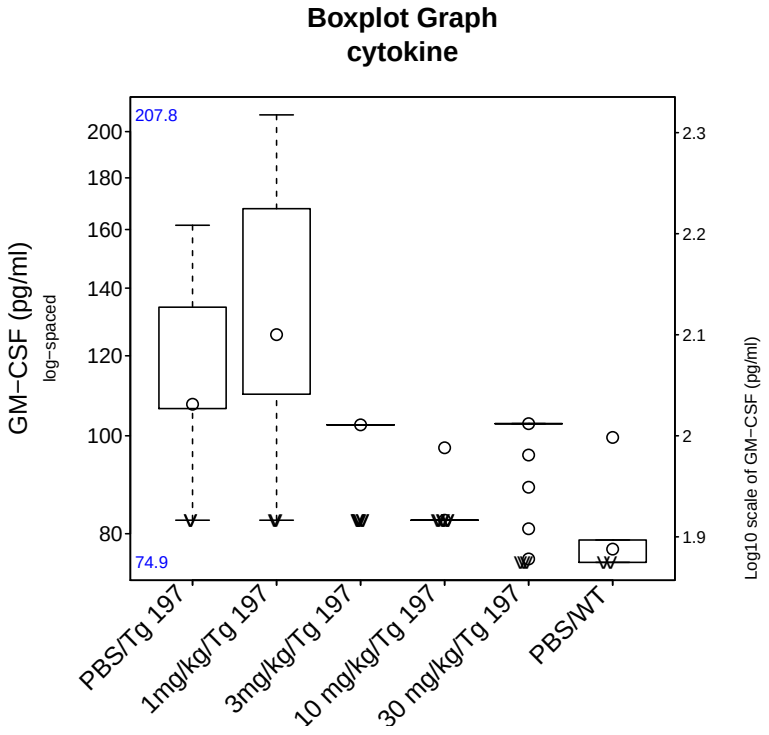

Figure 8: GM-CSF cytokine data boxplot graph.

axis. The rotation of 45° degrees comes from the method since it detects that the default representation on the device would create overlapped labels.

The `descriptiveTable` method prints a summary description of the data. It is shown here as purposely wrapped due to text width constraints:

```
R> descriptiveTable(gmcsfcens.data)

Descriptive Table of cytokine
Endpoint: GM-CSF (pg/ml)
                n ncensored ncomplete   Min 25%ile Median 75%ile   Max
PBS/Tg 197      8         2         6 <82.5  106.4  107.5  134.1 161.6
1mg/kg/Tg 197   8         2         6 <82.5  109.9  125.9  167.8 207.8
3mg/kg/Tg 197   8         7         1 <82.5  102.5  102.5  102.5 102.5
10 mg/kg/Tg 197 7         6         1 <82.5   82.5   82.5   82.5  97.3
30 mg/kg/Tg 197 8         3         5 <74.9  102.8  102.8  102.8 102.8
PBS/WT          6         2         4 <74.9   74.9   77.2   78.9  99.6

                Mean StdDev StdErr GeoMean SEGeoMean
PBS/Tg 197      <NA>   <NA>   <NA>    <NA>      <NA>
1mg/kg/Tg 197   <NA>   <NA>   <NA>    <NA>      <NA>
3mg/kg/Tg 197   <NA>   <NA>   <NA>    <NA>      <NA>
10 mg/kg/Tg 197 <NA>   <NA>   <NA>    <NA>      <NA>
30 mg/kg/Tg 197 <NA>   <NA>   <NA>    <NA>      <NA>
PBS/WT          <NA>   <NA>   <NA>    <NA>      <NA>
```

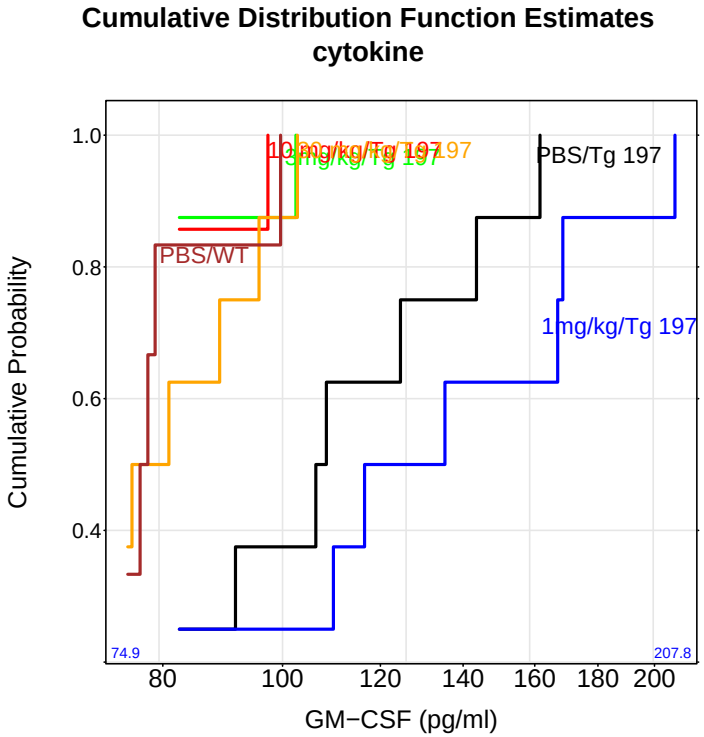

Figure 9: GM-CSF empirical distribution function graph.

Since all groups have at least one censored observation, there is no simple distribution-free estimate of the mean. The last five columns are therefore filled with "not available" values. Similarly the distributions could be examined as shown in Figure 9 with

```
R> kmGraph(gmcsfcens.data, distfcn = "cumulative")
```

Figure 9 arguably is a better representation of the distributions than the boxplot graph in Figure 8.

Fitting a model and proceeding with comparisons works analogously to the `canine` data example. The output below is purposely wrapped to fit inside the margins of this manuscript:

```
R> gmcsfcens.fit <- fit(gmcsfcens.data)
R> gmcsfcens.comps <- comparisonsTable(gmcsfcens.fit,
+    type = "allgroupstocontrol", refgrp = "PBS/Tg 197")

Comparisons Table of cytokine
Endpoint: GM-CSF (pg/ml)
Percent Differences (A vs. B)

Accelerated Failure Time Model Fit
with Sandwich Variance-Covariance Estimate
95% Confidence (alpha of 0.05)
```

```
                                estimate se lowerci upperci  pval
1mg/kg/Tg 197 vs. PBS/Tg 197         17 20     -17      65 0.364
3mg/kg/Tg 197 vs. PBS/Tg 197        -42 11     -61     -14 0.008
10 mg/kg/Tg 197 vs. PBS/Tg 197      -42 11     -61     -14 0.009
30 mg/kg/Tg 197 vs. PBS/Tg 197      -27 10     -45      -5 0.023
PBS/WT vs. PBS/Tg 197               -30 10     -47      -7 0.013

                                geomeanA  seA geomeanB  seB
1mg/kg/Tg 197 vs. PBS/Tg 197       123.2 16.8    105.3 11.5
3mg/kg/Tg 197 vs. PBS/Tg 197        60.6 10.6    105.3 11.5
10 mg/kg/Tg 197 vs. PBS/Tg 197      61.0 10.6    105.3 11.5
30 mg/kg/Tg 197 vs. PBS/Tg 197      76.3  6.7    105.3 11.5
PBS/WT vs. PBS/Tg 197               74.1  6.4    105.3 11.5
```

The comparisons table can be portrayed graphically with the call as follows, and the result shown in Figure 10:

```
R> comparisonsGraph(gmcsfcens.comps)
```

The PBS/Tg 197 group serves as the reference (control) and all other groups are compared to it. (Note the specification of `refgrp = "PBS/Tg 197"` in the `comparisonsTable` call.) The comparisons graph focuses on the differences, so a zero reference line is added to indicate significant differences for those comparisons where the reference line is not crossed.

If a multiplicity adjustment is desired, then the `mcadjust = TRUE` setting is added to the call,

```
R> comparisonsTable(gmcsfcens.fit, type = "allgroupstocontrol",
+    refgrp = "PBS/Tg 197", mcadjust = TRUE)

Some time may be needed as the critical point from the multcomp::summary.glht
function call is calculated. Please wait...

... Done. Critical point from Accelerated Failure Time fit is calculated.

Comparisons Table of cytokine
Endpoint: GM-CSF (pg/ml)
Percent Differences (A vs. B)

Accelerated Failure Time Model Fit
with Sandwich Variance-Covariance Estimate
95% Confidence (alpha of 0.05), Multiplicity Adjusted
                                estimate se lowerci upperci  pval
1mg/kg/Tg 197 vs. PBS/Tg 197         17 20     -25      84 0.837
3mg/kg/Tg 197 vs. PBS/Tg 197        -42 11     -66      -3 0.034
10 mg/kg/Tg 197 vs. PBS/Tg 197      -42 11     -65      -3 0.036
30 mg/kg/Tg 197 vs. PBS/Tg 197      -27 10     -49       4 0.093
PBS/WT vs. PBS/Tg 197               -30 10     -51       1 0.056
```

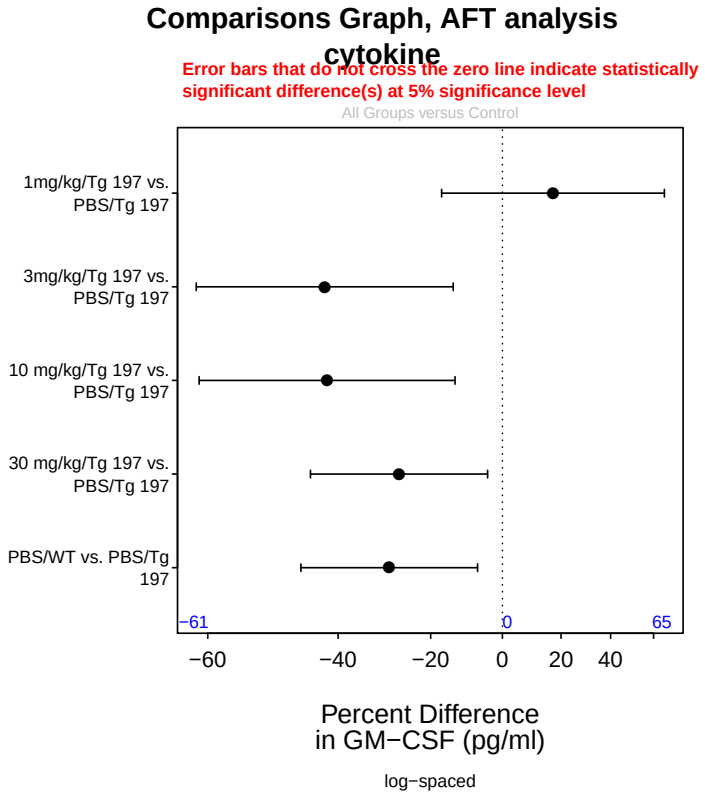

Figure 10: GM-CSF cytokine data comparisons graph.

```
                                geomeanA  seA geomeanB  seB
1mg/kg/Tg 197 vs. PBS/Tg 197       123.2 16.8    105.3 11.5
3mg/kg/Tg 197 vs. PBS/Tg 197        60.6 10.6    105.3 11.5
10 mg/kg/Tg 197 vs. PBS/Tg 197      61.0 10.6    105.3 11.5
30 mg/kg/Tg 197 vs. PBS/Tg 197      76.3  6.7    105.3 11.5
PBS/WT vs. PBS/Tg 197               74.1  6.4    105.3 11.5
```

and provides the above table of results, with a predisplayed status message as the calculation of the critical point from the **multcomp** package takes a few seconds.

Diagnostic graphs analogous to those shown in Section 4.1 for the `canine` data can be created with `varianceGraph` and `qqGraph` calls. The output from these are not shown here. As mentioned earlier, they are experimental methods for censored data and require caution to determine their usefulness when there is substantive censoring present.

5. Summary

The **cg** package arranges tools written in R into a strategy of comprehensive data evaluation for the comparison of groups in a one-factor data structure where the endpoint is continuous and possibly censored. The workflow of data preparation and subsequent exploratory and formal analyses aims for ease of use of modern methods. Presentation of results through informative graphs and tables is also a guiding design principle of the package. Continuous

refinement of the package is intended, along with the addition of comparison concepts for other frequently generated data structures in medical research such as paired difference, crossover, and longitudinal data designs.

Acknowledgments

The authors thank the three anonymous reviewers for helpful comments and suggestions to improve the manuscript and the software.

References

Andrews HP, Snee RD, Sarner MH (1980). "Graphical Display of Means." *The American Statistician*, **34**(4), 195–199.

Chambers JM, Hastie TJ (1992). *Statistical Models in S*. Chapman & Hall, London.

Cleveland WS (1993). *Visualizing Data*. Hobart Press, Summit, NJ.

Firth D (2010). ***qvcalc**: Quasi Variances for Factor Effects in Statistical Models*. R package version 0.8-7, URL http://CRAN.R-project.org/package=qvcalc.

Firth D, De Menezes RX (2004). "Quasi-Variances." *Biometrika*, **91**(1), 65–80.

Fleiss JL (1986). *The Design and Analysis of Clinical Experiments*. John Wiley & Sons, New York.

Gentleman R, Crowley J (1991). "Graphical Methods for Censored Data." *Journal of the American Statistical Association*, **86**(415), 678–683.

GraphPad Software (2010). ***GraphPad Prism** version 5.04*. La Jolla, CA. URL http://www.graphpad.com/.

Hothorn T, Bretz F, Westfall P (2008). "Simultaneous Inference in General Parametric Models." *Biometrical Journal*, **50**(3), 346–363.

Millard SP, Krause A (2001). *Applied Statistics in the Pharmaceutical Industry*. Springer-Verlag.

Pikounis B (2001). "One-Factor Comparative Studies." In SP Millard, A Krause (eds.), *Applied Statistics in the Pharmaceutical Industry*, chapter 2, pp. 17–40. Springer-Verlag.

Pinheiro J, Bates D, DebRoy S, Sarkar D, R Development Core Team (2009). ***nlme**: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-96, URL http://CRAN.R-project.org/package=nlme.

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Rhodes L, Ding VDH, Kemp RK, Khan MS, Nakhla AM, Pikounis B, Rosner W, Saunders HM, Feeney WP (2000). "Estradiol Causes a Dose Dependent Stimulation of Prostate Growth in Castrate Beagle Dogs." *The Prostate*, **44**, 8–18.

Shealy D, Cai A, Staquet K, Baker A, Lacy E, Johns L, Vafa O, Gunn G, Tam S, Sague S, Wang D, Brigham-Burke M, Dalmonte P, Emmell E, Pikounis B, Bugelski P, Zhou H, Scallon B, Giles-Komar J (2010). "Characterization of Golimumab (CNTO148), a Human Monoclonal Antibody Specific for Human Tumor Necrosis Factor." *mAbs*, **2**, 428–439.

Therneau T, Lumley T (2010). ***survival**: Survival Analysis, Including Penalised Likelihood*. R package version 2.35-8, URL http://CRAN.R-project.org/package=survival.

TIBCO (2012). *S+ Statistical Analysis Software*. TIBCO, Seattle, WA. URL http://spotfire.tibco.com/products/s-plus/statistical-analysis-software.aspx.

Tukey JW, Ciminera JL, Heyse JF (1985). "Testing the Statistical Certainty of a Response to Increasing Doses of a Drug." *Biometrics*, **41**(1), 295–301.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York.

Yee TW (2010). "The **VGAM** Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34. URL http://www.jstatsoft.org/v32/i10/.

A. Alternative censored data input formats

Sections 3.2.1 and 4.2 discuss alternative data input format options to create `cgOneFactorData` objects for the cytokine data set and censored data sets in general. All the alternative formats presented in this section need to specified with `format = "listed"` in the `prepareCGOneFactorData` call.

The first alternative format is the two-column case. The second column `endpt` is of mode character. After the printed data frame, the call to create the `cgOneFactorData` object is given. The `leftcensor` argument can be left as default `NULL`.

```
R> head(gmcsfcens.listfmt1)

        grpf   endpt
1 PBS/Tg 197 143.535
2 PBS/Tg 197  108.51
3 PBS/Tg 197 124.575
4 PBS/Tg 197    91.6
5 PBS/Tg 197 161.575
6 PBS/Tg 197   <82.5

R> tail(gmcsfcens.listfmt1)

     grpf  endpt
40 PBS/WT <74.94
41 PBS/WT  76.68
42 PBS/WT  78.86
43 PBS/WT  99.63
44 PBS/WT <74.94
45 PBS/WT   77.8

R> gmcsfcens.listfmt1.data <- prepareCGOneFactorData(gmcsfcens.listfmt1,
+    format = "listed", analysisname = "cytokine",
+    endptname = "GM-CSF (pg/ml)", logscale = TRUE, digits = 1)
```

The second alternative format has three columns. Here the `endpt` column is numeric. As before the call to create the `cgOneFactorData` object is given just after the printed data frame. In this format the `leftcensor = TRUE` argument needs to be set.

```
R> head(gmcsfcens.listfmt2)

         grp   endpt status
1 PBS/Tg 197 143.535      1
2 PBS/Tg 197 108.510      1
3 PBS/Tg 197 124.575      1
4 PBS/Tg 197  91.600      1
5 PBS/Tg 197 161.575      1
6 PBS/Tg 197  82.500      0
```

```
tail(gmcsfcens.listfmt2)
```

```
      grp endpt status
40 PBS/WT 74.94      0
41 PBS/WT 76.68      1
42 PBS/WT 78.86      1
43 PBS/WT 99.63      1
44 PBS/WT 74.94      0
45 PBS/WT 77.80      1
```

```
R> gmcsfcens.listfmt2.data <- prepareCGOneFactorData(gmcsfcens.listfmt2,
+    format = "listed", analysisname = "cytokine",
+    endptname = "GM-CSF (pg/ml)", logscale = TRUE, digits = 1,
+    leftcensor = TRUE)
```

The third alternative format has four columns, and the `leftcensor` argument can be left as the default `NULL`.

```
R> head(gmcsfcens.listfmt3)
```

```
         grp  endpt1  endpt2 status
1 PBS/Tg 197 143.535 143.535      1
2 PBS/Tg 197 108.510 108.510      1
3 PBS/Tg 197 124.575 124.575      1
4 PBS/Tg 197  91.600  91.600      1
5 PBS/Tg 197 161.575 161.575      1
6 PBS/Tg 197  82.500  82.500      2
```

```
tail(gmcsfcens.listfmt3)
```

```
      grp endpt1 endpt2 status
40 PBS/WT  74.94  74.94      2
41 PBS/WT  76.68  76.68      1
42 PBS/WT  78.86  78.86      1
43 PBS/WT  99.63  99.63      1
44 PBS/WT  74.94  74.94      2
45 PBS/WT  77.80  77.80      1
```

```
R> gmcsfcens.listfmt3.data <- prepareCGOneFactorData(gmcsfcens.listfmt3,
+    format = "listed", analysisname = "cytokine",
+    endptname = "GM-CSF (pg/ml)", logscale = TRUE, digits = 1)
```

For all the above `gmcsfcens.listfmt*.data` objects, the available methods will work in the same way as shown and discussed in Section 4.2 for the `gmcsfcens.data` object.

B. Equivalence of critical points

With the help of the `qdunnett` function from TIBCO Spotfire S+ (TIBCO 2012), the Dunnett, Tukey, and **multcomp** (Hothorn *et al.* 2008) 95% confidence level based critical points were determined for various sample size and number of groups combinations. With a $3 \times 3 = 9$ item list determined by three sample size levels and three group levels, balanced and unbalanced configurations were created.

```
R> group.s <- c(3, 5, 10)
R> n.s <- c(3, 5, 10)
R> balanced.configs <- vector("list", length = length(group.s) * length(n.s))
R> i <- 0
R> for(g in group.s) {
+    for(n in n.s) {
+      i <- i + 1
+      balanced.configs[[i]] <- rep(n, g)
+    }
+  }
R> data.frame(configs = sapply(balanced.configs,
+    function(x) paste(x, collapse = ",")))

                        configs
1                         3,3,3
2                         5,5,5
3                       10,10,10
4                     3,3,3,3,3
5                     5,5,5,5,5
6                 10,10,10,10,10
7           3,3,3,3,3,3,3,3,3,3
8           5,5,5,5,5,5,5,5,5,5
9 10,10,10,10,10,10,10,10,10,10
```

The next step is to generate unbalanced configurations by adding perturbations to balanced sample sizes. Only those cases where all balanced case sample sizes are at least 4 are adapted.

```
R> unbalance.theconfigs <- function(x, delta) {
+    lapply(x[unlist(lapply(balanced.configs, function(x) {
+        all(x > 3)
+      }))], function(x) {
+        add <- sample((-delta):delta, size = length(x), replace = TRUE)
+        while(length(unique(add)) == 1) {
+          add <- sample((-delta):delta, size = length(x), replace = TRUE)
+        }
+        return(x + add)
+      })
+  }
R> set.seed(47)
```

```
R> mild.unbalanced.configs <- unbalance.theconfigs(balanced.configs, 1)
R> unbalanced.configs <- unbalance.theconfigs(balanced.configs, 2)
```

Object dumps for use in S+ are created:

```
R> dump(c("balanced.configs","mild.unbalanced.configs",
+    "unbalanced.configs"), file = "p1Rdump.ssc")
```

Now within S+, the Dunnett critical points are determined. The first group is considered the control in each of the configurations. All of the next code block is performed with S+.

```
S> source("p1Rdump.ssc")
S> crit.dunnett <- function(x) {
+    lapply(x, function(x) {
+      qdunnett(p = 0.95, k = length(x), df = sum(x) - length(x),
+        nvec = x, control = 1)
+    })
+  }
S> balanced.dunnett <- crit.dunnett(balanced.configs)
S> mild.unbalanced.dunnett <- crit.dunnett(mild.unbalanced.configs)
S> unbalanced.dunnett <- crit.dunnett(unbalanced.configs)
S> dump(c("balanced.dunnett","mild.unbalanced.dunnett",
+    "unbalanced.dunnett"), file = "p1sscdump.R", oldStyle = TRUE)
```

Back to R, the **multcomp** based critical points for the various balanced and unbalanced data combinations are computed. Tukey critical points are first processed, then the **multcomp** counterparts to the `qtukey` and `qdunnett` functions.

```
R> crit.tukey <- function(x) {
+    lapply(x, function(x) {
+      qtukey(0.95, length(x), sum(x) - length(x))/sqrt(2)
+    })
+  }
R> balanced.tukey <- crit.tukey(balanced.configs)
R> mild.unbalanced.tukey <- crit.tukey(mild.unbalanced.configs)
R> unbalanced.tukey <- crit.tukey(unbalanced.configs)
R> crit.multcomp <- function(x, type) {
+    lapply(x, function(x) {
+      d <- data.frame(grp = factor(rep(1:length(x), x), 1:length(x)),
+        endpt = rnorm(sum(x)))
+      f <- aov(endpt ~ grp, data = d)
+      m <- glht(f, linfct = mcp(grp = type))
+      ca <- adjusted_calpha()
+      return(as.vector(ca(m, 0.95)))
+    })
+  }
R> balanced.tukey.multcomp <- crit.multcomp(balanced.configs, type = "Tukey")
R> mild.unbalanced.tukey.multcomp <- crit.multcomp(mild.unbalanced.configs,
```

```
+    type = "Tukey")
R> unbalanced.tukey.multcomp <- crit.multcomp(unbalanced.configs,
+    type = "Tukey")
R> balanced.dunnett.multcomp <- crit.multcomp(balanced.configs,
+    type = "Dunnett")
R> mild.unbalanced.dunnett.multcomp <- crit.multcomp(mild.unbalanced.configs,
+    type = "Dunnett")
R> unbalanced.dunnett.multcomp <- crit.multcomp(unbalanced.configs,
+    type = "Dunnett")
```

Now the equivalence of the critical points can be assessed. For the balanced configurations, all the values are equal to two decimal places.

```
R> data.frame(numgrps = rep(c(3, 5, 10), each = 3),
+    samplesizes = sapply(balanced.configs, function(x) {
+      paste(x, collapse = ",") }),
+    multcomp = unlist(balanced.tukey.multcomp),
+    crit.tukey = unlist(balanced.tukey))
```

	numgrps	samplesizes	multcomp	crit.tukey
1	3	3,3,3	3.066399	3.068274
2	3	5,5,5	2.669645	2.667864
3	3	10,10,10	2.479205	2.479418
4	5	3,3,3,3,3	3.291229	3.291082
5	5	5,5,5,5,5	2.992112	2.992375
6	5	10,10,10,10,10	2.842182	2.841450
7	10	3,3,3,3,3,3,3,3,3,3	3.540411	3.541108
8	10	5,5,5,5,5,5,5,5,5,5	3.347748	3.347806
9	10	10,10,10,10,10,10,10,10,10,10	3.248532	3.244427

```
R> data.frame(numgrps = rep(c(3, 5, 10), each = 3),
+    samplesizes = sapply(balanced.configs, function(x) {
+      paste(x, collapse = ",") }),
+    multcomp = unlist(balanced.dunnett.multcomp),
+    crit.dunnett = unlist(balanced.dunnett))
```

	numgrps	samplesizes	multcomp	crit.dunnett
1	3	3,3,3	2.862779	2.862750
2	3	5,5,5	2.502344	2.502367
3	3	10,10,10	2.333416	2.333411
4	5	3,3,3,3,3	2.890347	2.890480
5	5	5,5,5,5,5	2.651096	2.651030
6	5	10,10,10,10,10	2.531891	2.531277
7	10	3,3,3,3,3,3,3,3,3,3	2.945522	2.946282
8	10	5,5,5,5,5,5,5,5,5,5	2.811589	2.811754
9	10	10,10,10,10,10,10,10,10,10,10	2.740855	2.740937

For the mildly unbalanced configurations, the same equality patterns to two decimal places occurs.

```
R> data.frame(numgrps = rep(c(3, 5, 10), each = 2),
+    samplesizes = sapply(mild.unbalanced.configs, function(x) {
+      paste(x, collapse = ",") }),
+    multcomp = unlist(mild.unbalanced.tukey.multcomp),
+    crit.tukey = unlist(mild.unbalanced.tukey))

  numgrps               samplesizes multcomp crit.tukey
1       3                     6,5,6 2.617114   2.617280
2       3                  11,10,11 2.469243   2.469647
3       5                 5,5,5,6,4 2.991000   2.992375
4       5              11,9,10,10,11 2.838388   2.838914
5      10        5,4,4,5,4,5,5,4,6,5 3.360547   3.363127
6      10 9,9,10,11,11,10,10,9,11,10 3.242935   3.244427

R> data.frame(numgrps = rep(c(3, 5, 10), each = 2),
+    samplesizes = sapply(mild.unbalanced.configs, function(x) {
+      paste(x, collapse = ",") }),
+    multcomp = unlist(mild.unbalanced.dunnett.multcomp),
+    crit.dunnett = unlist(mild.unbalanced.dunnett))

  numgrps               samplesizes multcomp crit.dunnett
1       3                     6,5,6 2.460549     2.460552
2       3                  11,10,11 2.326261     2.326260
3       5                 5,5,5,6,4 2.651821     2.651838
4       5              11,9,10,10,11 2.536095     2.535441
5      10        5,4,4,5,4,5,5,4,6,5 2.830890     2.830814
6      10 9,9,10,11,11,10,10,9,11,10 2.729057     2.729033
```

The same pattern of equivalence, two decimal points, also occurs for a higher degree of imbalance amongst sample sizes.

```
R> data.frame(numgrps = rep(c(3, 5, 10), each = 2),
+    samplesizes = sapply(unbalanced.configs, function(x) {
+      paste(x, collapse = ",") }),
+    multcomp = unlist(unbalanced.tukey.multcomp),
+    crit.tukey = unlist(unbalanced.tukey))

  numgrps              samplesizes multcomp crit.tukey
1       3                   4,7,5 2.638074   2.640437
2       3                  11,8,9 2.489684   2.490830
3       5                5,7,4,4,6 2.975441   2.979037
4       5             12,9,12,8,11 2.834346   2.836488
5      10      3,3,3,6,3,3,5,5,6,5 3.386560   3.395242
6      10 10,9,8,10,12,8,12,8,10,8 3.246845   3.249230
```

```
R> data.frame(numgrps = rep(c(3, 5, 10), each = 2),
+    samplesizes = sapply(unbalanced.configs, function(x) {
+      paste(x, collapse = ",") }),
+    multcomp = unlist(unbalanced.dunnett.multcomp),
+    crit.dunnett = unlist(unbalanced.dunnett))

  numgrps              samplesizes multcomp crit.dunnett
1       3                   4,7,5 2.459891     2.459894
2       3                  11,8,9 2.351727     2.351733
3       5                5,7,4,4,6 2.639034     2.639161
4       5              12,9,12,8,11 2.538044     2.539125
5      10      3,3,3,6,3,3,5,5,6,5 2.803541     2.803859
6      10 10,9,8,10,12,8,12,8,10,8 2.751535     2.751009
```

Affiliation:

Bill Pikounis, John Oleynick
Johnson & Johnson
200 Great Valley Parkway
Malvern, PA 19355, United States of America
E-mail: cg@billpikounis.net
URL: http://billpikounis.net/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 52, Issue 1 *Submitted:* 2010-12-19
January 2013 *Accepted:* 2012-12-30