VizTest: Optimal Confidence Intervals for Visual Testing

by William Poirier and David A. Armstrong

Abstract When presented with several confidence intervals plotted together, readers have a natural inclination to try and compare the bounds visually to test for the statistical significance of their differences. Comparing confidence bounds this way to perform a pairwise significance test yields erroneous results. The VizTest R package provides researchers with optimal confidence intervals for visual representation of pairwise tests. It is natively compatible with basic regression model outputs and includes a function that creates usable objects from other outputs, such as : 1) multilevel regression; 2) predicted probabilities; and 3) descriptive quantities. Pairwise tests are performed natively via normal theory or simulations, but users could choose to provide the results of alternative procedures directly. Multiplicity adjustments are natively implemented. We provide thorough examples of these different applications to facilitate the integration of the package into the users' workflows.

1 Introduction

Routinely, researchers convey the findings of their research through the visual display of estimates and confidence intervals (Few 2012; Kastellec and Leoni 2007). When readers engage these displays, they want to do some or all of the following three tasks: 1) evaluate whether estimates are different from zero, 2) evaluate the relative variability of estimates, and 3) compare estimates with each other.

Assume for the purpose of this discussion we have estimates b_j for $j = \{1, \ldots, J\}$ with a $J \times J$ variance-covariance matrix V. With desired type I error rate α, we could make $(1-\alpha) \times 100\%$ confidence intervals with $b_j \pm z_{\left(1-\frac{\alpha}{2}\right)} \sqrt{V_{J,J}}$.[1] Testing $H_0 : \beta_j = 0$ against a two-sided alternative is as easy as identifying whether zero lies in the 95% confidence interval. If zero is outside the interval, we reject H_0. Evaluating the similarity of the relative variability of the estimates is slightly more involved, requiring the reader to compare the widths of the confidence intervals—$2 \times z_{\left(1-\frac{\alpha}{2}\right)} \sqrt{V_{J,J}}$. While this task is cognitively more complicated—estimating the lengths of each line and calculating the ratio, all required information is present in the visualization. The real problem lies in the third task—evaluating the difference between two estimates. Ideally, readers could compare the upper and lower bounds of the confidence intervals of two estimates in order to know whether they are different from each other. That is, they could engage in a *visual testing* exercise. However, using the overlap (or lack thereof) in $(1-\alpha) \times 100\%$ confidence intervals as an indication of the (in)significance of the difference between the two estimates at the α level will often result in erroneous inferences (Browne 1979).

This is a problem that has been well understood for nearly a century. Both Dice and Laraas (1936) and Simpson and Roe (1939) recognized that $(1-\alpha) \times 100\%$ confidence intervals did not naturally equate to tests at the α level. Their scholarship tried to identify rules of thumb for identifying differences that were either significant or not depending on the relative size of the standard errors of the estimates. This approach continued into the late 20th century where Browne (1979) developed the same type of rules for other displays (e.g., $\pm 1SE$ bars) and provided more formalization around the problem of differing sampling variances.

Moving on from rules of thumb, Tukey (1991) recognized that under the relatively restrictive assumption of equal standard errors and uncorrelated estimates 84% confidence intervals will overlap roughly 95% of the time under the null hypothesis. Goldstein and Healey (1995) proposed a more general approach that would identify the appropriate confidence level for tests at any level under the same assumptions adopted by Tukey. Payton, Greenstone, and Schenker (2003) extend this idea to the non-normal context.

Extending this line of scholarship, Afshartous and Preston (2010) show that the probability that a pair of intervals overlaps under the null hypothesis is a function of θ, the ratio of their standard errors, ρ the correlation of the estimates and Z_γ, the value of z on the standard normal distribution that puts $\frac{\gamma}{2}$ of the distribution to its right. For any desired overlap in confidence intervals under the null, we can calculate the appropriate value of Z_γ. Specifically, Afshartous and Preston (2010) show the following:

[1] The relevant normal distribution quantile can easily be replaced with the one from the Student's t distribution.

$$Z_\gamma = \left[\frac{F^{-1}\left(\frac{\alpha}{2}\right)}{\frac{\theta}{\sqrt{\theta^2+1-2\rho\theta}} + \frac{\frac{1}{\theta}}{\sqrt{1+\theta^{-2}-2\rho\theta^{-1}}}} \right]$$

$$\Pr(\text{Overlap}) = 2\left(1 - F\left(Z_\gamma \frac{\theta}{\sqrt{\theta^2+1-2\rho\theta}} + \frac{\frac{1}{\theta}}{\sqrt{1+\theta^{-2}-2\rho\theta^{-1}}}\right)\right)$$

Radean (2023) further developed this idea to the non-normal context where draws from a (quasi-)Bayesian simulation may be available. While more general, this solution and the one proposed by Afshartous and Preston (2010) are only guaranteed to work on a single pair of intervals. Generally speaking, all pairs of estimates in a larger collection of estimates will have neither the same ratio of standard errors nor the same correlation. This means that finding a single value of Z_γ that provides the right type I error rate for all pairs of estimates may be impossible. Further, even if we *could* find such a solution, it may not be "best" or optimal for visual testing. We propose an entirely different solution to the problem as discussed below (Armstrong II and Poirier Forthcoming).

2 Searching for inferential confidence intervals

All previous solutions, up to and including the most recent intervention in this literature Radean (2023) use the probability of overlap in the intervals to do the test. That is, the visualization and the test are inextricably linked. We propose a solution that estimates pairwise tests of all estimates and then performs a grid search over a set of predefined, reasonable values to find a confidence level γ, such that (non-)overlaps in the $\gamma \times 100\%$ confidence intervals correspond as closely as possible with the (in)significant differences. That is, we decouple the visualization from the testing. The algorithm we propose is as follows:

1. Conduct all pairwise tests between estimates. The researcher uses her preferred method to distinguish significant or interesting differences from insignificant or uninteresting ones. These tests could be accomplished with or without multiplicity adjustments, robust standard errors or any other adjustment the researcher deems necessary. The tests could also include a reference estimate of zero, ensuring that all univariate tests against zero are respected by the procedure. This procedure produces a vector, s_{ij} that indicates whether estimate b_i is significantly different from estimate b_j (in which case $s_{ij} = 1$, or zero otherwise).

2. Find the inferential confidence level(s). With the results from all the pairwise tests in $\boldsymbol{s}$, we find $(1-\gamma)$[2] as the solution to:

$$\underset{(1-\gamma)}{\arg\max} \sum_{j=2}^{J} \sum_{i=1}^{j-1} I(s_{ij} = s_{ij}^*) \tag{1}$$

where, s_{ij}^* is 0 if $(1-\gamma) \times 100\%$ confidence intervals overlap for b_i and b_j and 1 if they do not. Hence, we find the value or values of γ for which the agreement between pairwise tests (s_{ij}) and visual tests (s_{ij}^*) is maximized.

3. Pick the inferential confidence level that is most useful. If multiple γ are found to be equivalent in step 2, we should try to identify which is "best". While we discuss some alternatives below, the idea is that it should be as easy as possible to do two things—identify where intervals do not overlap for those estimates that are statistically different from each other and identify where intervals do overlap for those estimates that are not statistically different from each other.

If a γ level is identified where **all** the pairwise tests match the visual tests, then using that level in the corresponding visualization would solve the aforementioned issue—readers would be able to easily identify which pairs of estimates are statistically distinguishable from each other and which are not. This is implemented in the **VizTest** package described below along with several use cases that will allow researchers to deploy the algorithm described above in a wide array of real-world analytical situations.

3 The VizTest package

The **VizTest** package allows the user to find the optimal confidence level for visual testing in many different scenarios. The `viztest()` function is a generic that currently dispatches one of two methods—

[2]We call $(1-\gamma)$ the inferential confidence level, the confidence level that best visually represents the underlying pairwise tests.

the `default` method which performs a normal theory test for pairwise difference and the `vtsim` method which operates on realized samples from a distribution (e.g., a matrix of samples from a Bayesian posterior distribution).

- `obj`: the object that contains the estimates.
- `test_level`: the type 1 error rate (α) for the pairwise tests (default at 0.05).
- `range_levels`: the range of γ to try (default from 0.25 to 0.99).
- `level_increment`: step size between values of `range_levels` (default at 0.01).
- `adjust`: multiplicity adjustment (`holm`, `hochberg`, `hommel`, `bonferroni`, `BH`, `BY`, `fdr` or `none`) to use when computing the p-values for the pairwise tests. This is not implemented for the `sim` method. The default is to do no adjustment.

- `cifun`: the method used to calculate the confidence/credible interval for simulation results. Either "quantile" (default) or "hdi" for highest density region.
- `include_intercept`: Logical whether to include the intercept in the pairwise tests (intercept excluded by default).
- `include_zero`: Logical whether univariate null hypothesis test should be included for each estimate (`TRUE` by default).
- `sig_diffs`: Optional logical vector indicating, for each pair of estimates, whether or not there is a significant difference. This is `NULL` by default.
- `...` Currently, no other arguments are passed down to other internal functions.

The package has dependencies on three other packages. First, **dplyr** is used for different data wrangling tasks all throughout the implementation of the procedure. Second, **ggplot2** is use as the main driver of the plot method. Finally, **HDInterval** is used for simulation results where the user specifies the use of highest density region (HDI) to calculate the confidence/credible interval.

For the default method, normal theory tests are performed. Specifically, we use R's `combn()` function to identify all pairwise combinations of observation ids. For example, for a vector of estimates (call it `est` with five values) and a variance-covariance matrix (call it `v`, which is 5×5), we define `combs <- combn(length(est),2)` which would identify the ten different pairwise combinations. We reorder `ests` so they are from largest to smallest and reorder the rows and columns of `v` accordingly, too. We then make a matrix that calculates the pairwise differences in the following way:

```
D <- matrix(0, nrow= length(est), ncol = ncol(combs))
D[cbind(combs[1,], 1:ncol(D))] <- -1
D[cbind(combs[2,], 1:ncol(D))] <- 1
```

This produces a matrix, `D`, such that each column corresponds with a single pairwise test. We can then calculate the pairwise differences and their standard errors with:

```
diffs <- est %*% D
v_diffs <- t(D) %*% v %*% D
se_diffs <- sqrt(diag(v_diffs))
```

Finally, we calculate z-statistics for all pairwise differences and their p-values using the standard normal CDF. The vector s above, is then defined as $2 \times Pr(|z| > Z) <$ `test_level` for the vector of z-statistics from the pairwise tests.

Next, we calculate the lower and upper confidence bounds for all test values of γ. This produces a `length(est)`$\times$`diff(range_levels)/level_increment` matrix of lower (`L`) and upper (`U`) confidence bounds. Since the values of `est` are ordered from largest to smallest, if $L_i > U_j$ then the confidence intervals for stimulus i and j do not overlap for any values of $i < j \leq$ `length(est)`. Likewise, if $L_i < U_j$, the two intervals do overlap. Thus, each column of the resulting matrix `L[combs[1,],] > U[combs[2,], ]` contains a vector s^* that can be used to evaluate correspondence with s.

For the simulation method, `est` is a # draws$\times$ `length(est)` matrix where we calculate pairwise differences of the estimates using the `D` matrix defined above: `diffs <- est %*% D`. We then calculate the probability that each pairwise difference is greater than zero: `p_diffs <- apply(diffs, 2, \(x)mean(x > 0))`. Then, to treat credible values greater than zero and credible values smaller than zero the same, we calculate `p_diffs <- ifelse(p_diffs > .5, 1-p_diffs, p_diffs)`. We create the s vector as `p_diffs < test_level`. We use `cifun()` at level γ to calculate the lower and upper bounds of each estimate and proceed in the same fashion as above to define s^* and evaluate the correspondence between s and s^*.

Regardless of the method, we identify the level(s) of γ that produce the greatest correspondence between s and s^*. Below, we describe some of the function arguments in greater detail as well as some of the subsequent generic methods (i.e., `print()` and `plot()`) which are defined for objects returned by `viztest()`.

The obj argument

The obj argument can be of two different broad types. The default method of viztest() uses coef() and vcov() to extract the estimates and their variance-covariance matrix, respectively. Obviously, this will not work as intended for all analytical situations of interest. To that end, we provide a function make_vt_data() that can take a vector of estimates and either a vector of standard errors (in the case of independent estimates) or a variance-covariance matrix in the case of non-independent samples. If a vector of standard errors is provided, a variance-covariance matrix will be produced by multiplying the vector of standard errors by the appropriately sized identity matrix. The final argument of make_vt_data() is an argument identifying the type of input data. If type = "est_var" the function assumes that estimates contains point estimates and variances either a vector of standard errors or a variance-covariance matrix. In this case, the function returns an object of class vtcustom. We have defined coef() and vcov() methods for .vtcustom objects so that viztest() will work as expected in this case. We anticipate two primary use-cases for this function, though others will likely arise, too.

1. Users may want to replace the default variance-covariance matrix with a robust alternative. In that case, calling make_vt_data() on the original estimates and the robust variance-covariance matrix will allow the pairwise tests to be calculated with the robust variance-covariance matrix.

2. It is possible that the user would want to make a plot with a collection of descriptive statistics that do not derive from a single object. In that case, assuming they are independent, the vector of estimates and standard errors could be provided to make_vt_data() to evaluate all statistics at the same time.

For simulation results, users can also use the make_vt_data() function. In this case, the estimates argument is the draws×estimates matrix of simulated values. The variances argument remains NULL and type is set to "sim". This produces an object of class .vtsim, that can be passed to viztest. Note, that if type = "sim", then even if variances is non-null, the argument will be disregarded.

The sig_diffs argument

We provide the option for the user to input the vector s directly with the sig_diffs argument. The specification of this argument is a bit complex. While it is just a logical vector, care must be taken to ensure that the order of the values is appropriate. The complexity here comes from the fact that internally the estimates and variance-covariance matrix are rearranged in decreasing order of the estimates. To aid in the process, we provide a function make_diff_template() that takes three arguments—estimates, the same estimated values that will be passed to viztest(), include_zero and include_intercept, which should be set to the same values as they are in the viztest() function (for convenience, they have the same default values in both places). The function simply reorders the estimates and identifies the pairwise combinations the same way that viztest() does. It returns a data frame that contains the names (taken from names(estimates)) of the estimates for the larger and smaller estimate in each pair. The resulting data frame could be merged with estimates or any other values with the same names as the estimates that can help identify the significance or credibility of pairwise differences. If this is provided, it overrides the calculation of the pairwise tests in viztest(). We show an example of this below.

viztest() returned values

The viztest() function returns a list containing 8 elements:

- tab: a table returning the results of the tests for all values of γ. This table is composed of 4 columns:
  - level: the tested γs.
  - psame: the percentage of agreement between the pairwise test and the visualization test (s_{ij} and s^*_{ij}) for corresponding levels of γ (higher is better).
  - pdiff: the percentage of statistically significative differences in pairwise tests (s_{ij}).
  - easy: the easiness measure for each γ (higher is better).
- pw_test: s_{ij}, a Boolean vector of significance of pairwise tests.
- ci_tests: s^*_{ij}, a Boolean vector of significance of visual CI tests.
- combs: the matrix describing all the tested pairs.
- param_names: the vector of covariates' names in descending order.

- `L`: the matrix of lower bounds for each covariate (rows) across all tested γ (columns).
- `U`: the matrix of upper bounds for each covariate (rows) across all tested γ (columns).
- `est`: a sub-list containing the original estimates (`est`), their standard errors (`se`), and the variable's names (`vbl`).

Cognitive easiness

In the event that there are multiple values of γ that produce identical correspondence between s and s^*, we should choose the one that is "best". We define best here as one that produces results that are most easily engaged by viewers. To do this, we consider the two most difficult cases to discern:

1. The pair of estimates whose difference is statistically different from zero and has the smallest distance between the lower and upper confidence bounds. Presumably, if the viewer can identify the non-overlap in these confidence intervals, they could do the same for all other statistically different pairs where the ends of their confidence intervals will be more distant.
2. The pair of estimates whose difference is not statistically significant and has the smallest overlap in their confidence intervals. Presumably, if the viewer can identify the overlap in these confidence bounds, they could do the same for all other statistically insignificant pairs whose confidence intervals will overlap more.

Ideally, the number that maximizes the distance in the confidence bounds for test 1 above and the overlap in confidence bounds for test 2 above will be the best value. If we define $\sim o$ as the distance between the lower and upper bounds for test 1 above and o as the overlap in the bounds for test 2, we define easiness as $\sim o \times o$. As this number departs from its maximum, it will either make it easier to discern non-overlaps at the expense of being able to discern overlapping intervals, or vice versa. We calculate this measure as "easiness" for all solutions. See Appendix 4 in the supplementary material of Armstrong II and Poirier (Forthcoming) for more details.

We turn now to some use cases that will also allow us to describe the output from the `print()` and `plot()` methods that are defined for the results.

4 Use cases

In this section, we present four use cases where **VizTest** comes in handy. First, an example using a regular regression model. Second, we demonstrate how to use the function with multilevel regression objects. Third, we show how to create a model object compatible with the function such that we may visualize predicted probabilities and finally descriptive quantities. The code chunk below loads the different packages used in these examples. **wooldridge** and **carData** for their datasets; and **dplyr**, **tidyr**, **ggplot2**, **patchwork**, **lme4**, and **marginaleffects** for data wrangling and analysis.

```
#### 0. Initialization ####
# remotes::install_github('davidaarmstrong/VizTest')
library(VizTest)

# Datasets from
library(wooldridge)
library(carData)

# For wrangling and analysis
library(dplyr)
library(tidyr)
library(ggplot2)
library(patchwork)
library(lme4)
library(marginaleffects)
```

Objects where `coef()` and `vcov()` work

For this example, we use the `gpa1` dataset from the **wooldridge** package. We create a regression model using `lm`. Models created from `lm` and `glm` can directly be used in `viztest` as we do here. In fact, any object that has a defined `coef()` method that returns estimates and a `vcov()` method that returns the variance-covariance matrix of the estimates will work this way. We also make explicit

the default values of the `test_level`, `range_levels`, and `level_increment`. The `print()` method for `viztest` objects will print a few different pieces of information. First, it prints a table that identifies the smallest, middle, largest and easiest values of inferential confidence intervals that maximize the correspondence between the pairwise tests and the (non-)overlaps of the confidence intervals. In some cases, this might identify a single value or very small range. In other cases, it could identify quite a large range. If the `missed_tests` argument is `TRUE` (its default), then the printed results will either indicate that all tests have been accounted for or it will identify which tests are not accommodated by the inferential confidence intervals. The output below indicates that the optimal CI is at the 72% level and that this yields one error: the coefficient for `alcohol` is not significantly different from zero (see the `Insig` designation in the `pw_test` column), but its inferential CI does not include zero (see the `No` designation in the `ci_olap` column). This means that no one inferential CI can visually represent all pairwise test *and* all univariate tests against zero.

```
#### 4.1 Regular object ####
model1 <- lm(colGPA~skipped+alcohol+PC+male+car+job20,data=gpa1)

# Parsing to viztest
viztestObj <- viztest(model1, test_level = 0.05, range_levels = c(0.25,0.99),
    level_increment = 0.01)

# Print
print(viztestObj)

#>
#> Correspondents of PW Tests with CI Tests
#>   level    psame     pdiff         easy  method
#> 1  0.72 0.952381 0.4285714 -0.001287547  Lowest
#> 2  0.72 0.952381 0.4285714 -0.001287547  Middle
#> 3  0.72 0.952381 0.4285714 -0.001287547 Highest
#> 4  0.72 0.952381 0.4285714 -0.001287547 Easiest
#>
#> Missed Tests for Lowest Level (n=1 of 21)
#>    bigger smaller pw_test ci_olap
#> 7 alcohol    zero   Insig      No
#>
#> Missed Tests for Middle Level (n=1 of 21)
#>    bigger smaller pw_test ci_olap
#> 7 alcohol    zero   Insig      No
#>
#> Missed Tests for Highest Level (n=1 of 21)
#>    bigger smaller pw_test ci_olap
#> 7 alcohol    zero   Insig      No
#>
#> Missed Tests for Easiest Level (n=1 of 21)
#>    bigger smaller pw_test ci_olap
#> 7 alcohol    zero   Insig      No
```

We see the ultimate goal of this analysis as making a visualization that permits readers to do all relevant tests visually. To that end, we have created a `plot()` method for these objects as well.

The `plot` method for the object then yields panel A of Figure 1. By default, it plots the estimates and their optimal CI using the `geom_pointrange` function from **ggplot2**. In addition to the `viztest` object, the method takes 5 arguments:

- `ref_lines`: one of `all`, `ambiguous`, `none`, or a vector of coefficient names (they must match the names included in the `viztest` object). `all` will plot vertical dotted lines along the upper bound of the lowest coefficient to the most distant coefficient with overlapping confidence intervals. `ambiguous` will draw these lines but only for the overlapping bounds that are near to each other. `none`, the default value, won't draw reference lines. There is also the option of manually feeding the argument a vector of coefficient names for which the user wants reference lines drawn.
- `viz_diff_thresh`: threshold value for identifying ambiguity that will be used if `ref_lines = "ambiguous"`. Default at 0.02.
- `make_plot`: a Boolean value indicating whether to directly plot the results of the object (`TRUE`, the default) or to return a `dataframe` that can be used to make a custom visualization.

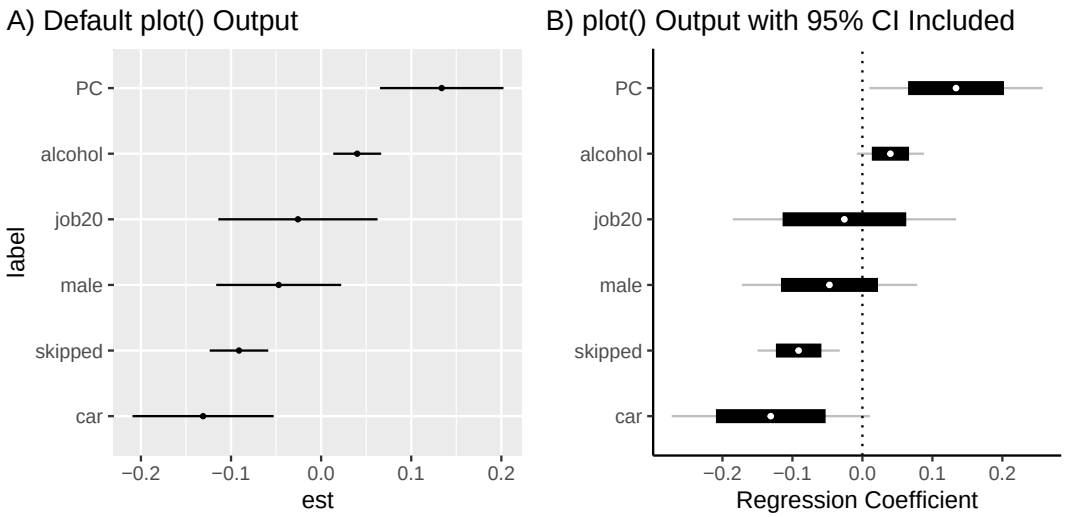

Figure 1: Plots of Regression Output with Inferential Confidence Intervals

- `level`: allows users to choose the CI level they want, either through a numerical value or one of the following: `ce` for cognitively easiest, `max` for highest, `min` for smallest, and `median` for the middle value. The default is to return the cognitively easiest.
- `trans`: a function that transforms the coefficients and their CI. Default is the identity function `I`, i.e. no transformation. Note, non-linear transformations do not trigger a recalculation of the test statistics, they simply result in a transformation of the estimate and the ends of the confidence bounds.

```
f1 <- plot(viztestObj)
```

There are a few different ways that we could account for tests not accommodated by the inferential confidence intervals. We could simply discuss them in the note to the figure. In this case, we would simply add a note to the effect: "Even though the `alcohol` inferential confidence interval does not include zero, it is not statistically different from zero in a proper pairwise test." In this particular case (or any case where tests against zero are not captured), the 95% confidence intervals will capture those tests (since they are univariate tests against a point null hypothesis). We could simply add the 95% confidence intervals to the plot. The easiest way to do this would be to save the plot data by adding the argument `make_plot=FALSE` and add in the additional confidence intervals, as shown in panel B of Figure 1.

```
reg_plot_data <- plot(viztestObj, make_plot=FALSE) %>%
  mutate(lwr95 = est - qnorm(.975)*se,
         upr95 = est + qnorm(.975)*se)

f2 <- ggplot(reg_plot_data, aes(y = label, x=est)) +
  geom_linerange(aes(xmin=lwr95, xmax=upr95), color="gray75") +
  geom_linerange(aes(xmin=lwr, xmax=upr), color="black", linewidth=3) +
  geom_point(color="white", size=.75) +
  geom_vline(xintercept=0, linetype=3) +
  theme_classic() +
  labs(x="Regression Coefficient", y="")
```

Replacing or Adding to Default Confidence Intervals

In Armstrong II and Poirier (Forthcoming), we follow the convention in the literature and suggest replacing the default (e.g., 95%) confidence intervals with the ones that best accomplish the visual testing goal. This is the reasoning that many are using for presenting 84% intervals—not because they are inherently interested in 84% confidence, but because they facilitate pairwise tests at the 5% level.

We recognize that for many, trying to publish a paper with, say, 72% confidence intervals and convincing reviewers that they are not doing something disingenuous will be a heavy lift. In these cases, there is no harm in also presenting the 95% intervals. The only real cost is increased cognitive

load required to decode the graph owing to the two different sets of intervals. If they are well explained, they should provide sufficient information to both allay fears of reviewers and provide readers with the ability to perform any arbitrary set of pairwise tests. In Figure 1, we show the default intervals replaced in panel A and embellished with the inferential confidence intervals in panel B.

Multilevel regression objects

For this example we use the World Value Survey dataset from **carData** and produce a multilevel regression model using `lmer()` from the **lme4**. Here, the results have class `lmerMod` which is not readily usable by `viztest`. For `lmerMod` and `merMod` objects, `coef()` returns a matrix of coefficients with level-2 observations in the rows and variables in the columns. Thus, `coef()` is defined, but produces output that is not compatible with `viztest()`. To proceed, we need to create a custom object; we can do this with `make_vt_data()`. First, we extract the fixed effects from the model and assign them cleaner names for the subsequent plot. Second, we extract the variance-covariance matrix using `vcov` which in the case of `lmerMod` objects outputs an object of class `dpoMatrix`. The `make_vt_data()` function does a test of whether the input variances inherit the "matrix" class. This fails for `dpoMatrix` objects, so we recast the object as a `matrix` using `as.matrix`. Now that we have both a named vector of coefficients and its corresponding variance-covariance matrix, we create a compatible object fed to `viztest()`.

```
#### 4.2 Multilevel regression objects ####
data(WVS, package='carData')

# Poverty variable as a scale from -1 to 1
NewWVS <- WVS %>%
  mutate(povertyNum = as.numeric(poverty)-2)

# The model
model2 <- lmer(povertyNum ~ age + religion +
                 degree + gender + (1 | country), NewWVS)

# Creating custom object
## Extracting fixed effect coefficients and naming them
named_coef_vec <- fixef(model2)
coefNames <- c("(Intercept)","Age","Religious","Has a degree","Male")
names(named_coef_vec) <- coefNames

## Extracting vcov matrix
#### As matrix necessary to overwrite special object from lme4
vcov <- as.matrix(vcov(model2))

eff_vt <- make_vt_data(named_coef_vec, vcov)

# Parsing to viztest
viztestObj <- viztest(eff_vt,test_level = 0.05,
                      range_levels = c(0.25,0.99),level_increment = 0.01)

# Print
print(viztestObj)

#>
#> Correspondents of PW Tests with CI Tests
#>   level psame pdiff          easy  method
#> 1  0.46     1   0.7 -0.0256354927  Lowest
#> 2  0.68     1   0.7 -0.0036862387  Middle
#> 3  0.90     1   0.7 -0.0536560657 Highest
#> 4  0.66     1   0.7 -0.0005905607 Easiest
#>
#> All  10  tests properly represented for by CI overlaps.
```

Any inferential confidence level from 0.46 to 0.90 properly represents all pairwise tests. We then plot the estimates in Figure 2, specifying that we want to display the cognitively easiest level with all reference line.

```
# Plotting
```

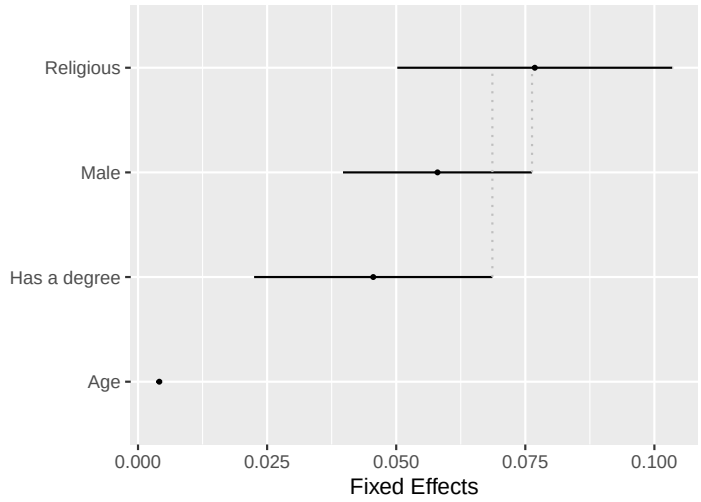

Figure 2: Multilevel regression objects — default output with all reference lines

```
plot(viztestObj,ref_lines = "all", level = "ce") +
  labs(y="", x="Fixed Effects")
```

Predicted probabilities

For this example, we use the `TitanicSurvival` dataset from **carData** to create predicted probabilities of survival based on a logistic regression model. We first create a variable corresponding to age categories and then run a model including an interaction between sex and this new categorical age variable. We use the `avg_predictions()` function from **marginaleffects** to produce predicted values for each pair of values of the interaction. One of the benefits of using the functions in **marginaleffects** is that they have defined `coef()` and `vcov()` methods which make them compatible with `viztest()`. However, one thing to note is that by default, the standard errors are calculated with the delta method and the confidence intervals are normal theory intervals around the estimated values. For predicted probabilities close to zero or one, users may want to use a different method. We show both below.

Normal Theory Intervals

Since the default behaviour for `avg_predictions()` is to use normal theory intervals, we do not need to intervene in the function. First, we can manage the data, estimate the model and then calculate the average predicted probabilities.

```
#### 4.3 Predicted probabilities ####
data(TitanicSurvival, package="carData")
NewTitanicSurvival <- TitanicSurvival %>%
  mutate(ageCat = case_when(age <= 10 ~ "0-10",
                            age > 10 & age <=18 ~ "11-18",
                            age > 18 & age <=30 ~ "19-30",
                            age > 30 & age <=40 ~ "31-40",
                            age > 40 & age <=50 ~ "41-50",
                            age >50 ~ "51+"))
# The model
model3 <- glm(survived~sex*ageCat+passengerClass,
              data=NewTitanicSurvival,family = binomial(link="logit"))

# Predicted values
mes <- avg_predictions(model3,
                variables = list(ageCat = levels(NewTitanicSurvival$ageCat),
                                 sex=levels(NewTitanicSurvival$sex)))
```

Next, we can find the appropriate inferential confidence levels. In this case, since we are considering predicted probabilities, we can exclude zero as the univariate test against zero is not particularly

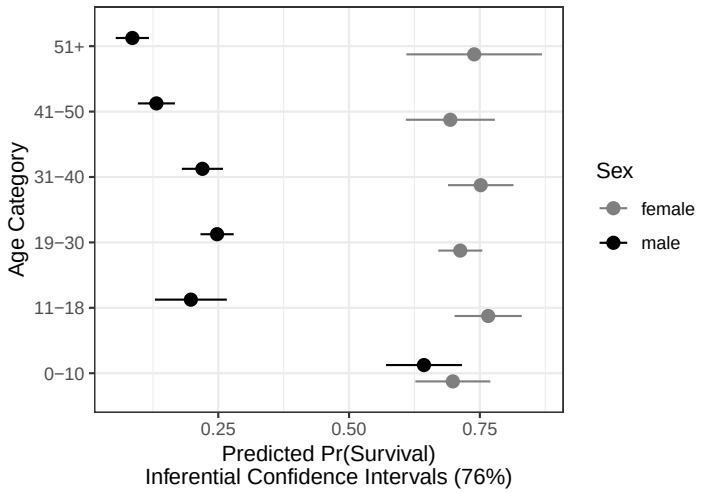

Figure 3: Average Predicted Probabilities — Plot with Normal Theory Inferential CIs

interesting. All 66 pairwise tests are accounted for by the (non-)overlaps in the inferential confidence intervals between 71% and 80%.

```
#>
#> Correspondents of PW Tests with CI Tests
#>   level psame     pdiff         easy  method
#> 1  0.71     1 0.6060606 -0.020384071  Lowest
#> 2  0.75     1 0.6060606 -0.002339719  Middle
#> 3  0.80     1 0.6060606 -0.024147874 Highest
#> 4  0.75     1 0.6060606 -0.002339719 Easiest
#>
#> All  66  tests properly represented for by CI overlaps.
```

One slight inconvenience of using the **marginaleffects** functions is that when multiple variables are involved in the prediction, the names of the estimates are not intuitive. So, instead of using the plot method for our visual testing result, we can simply make the appropriate confidence interval in the existing average prediction data.

```
mes <- mes %>%
  mutate(lwr76 = estimate - qnorm(.88)*std.error,
         upr76 = estimate + qnorm(.88)*std.error)

ggplot(mes, aes(y = ageCat, x=estimate, xmin = lwr76,
              xmax=upr76, colour=sex)) +
  geom_pointrange(position = position_dodge(width=.5)) +
  scale_colour_manual("Sex",values=c("gray50", "black")) +
  theme_bw() +
  labs(x="Predicted Pr(Survival)\nInferential Confidence Intervals (76%)",
       y="Age Category")
```

Using simulation method

The **marginaleffects** functions have an inference engine that will treat the input models as Bayesian (assuming ignorance priors) and sample from the multivariate normal distribution centred at the coefficient estimates with a variance-covariance matrix equal to the analytical variance-covariance matrix from the model. We can invoke this with the `inferences()` function.

```
ap_sim <- avg_predictions(model3,
                variables = list(ageCat = levels(NewTitanicSurvival$ageCat),
                                 sex=levels(NewTitanicSurvival$sex))) %>%
  inferences(method = "simulation", R=2500)
```

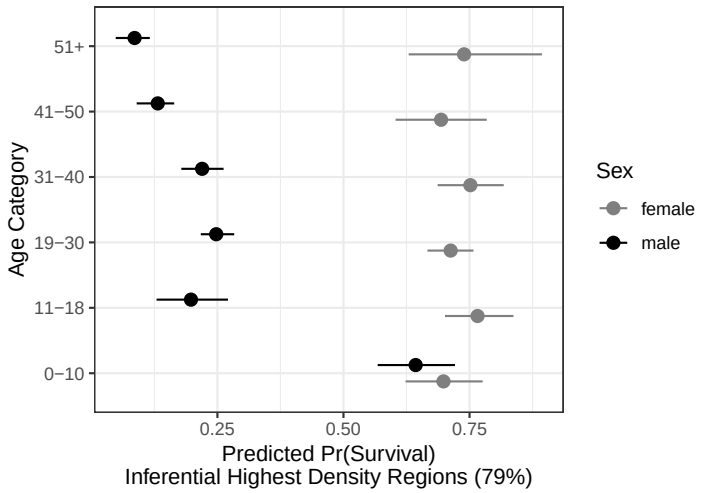

Figure 4: Average Predicted Probabilities — Plot with Simulation-based Inferential HDIs

The "posterior" attribute of the resulting object has the estimates-by-draws posterior simulation values. We transpose that matrix and add some column names. We then pass that to `make_vt_data()` with the argument `type="sim"` and call `viztest()`. Inferential highest density intervals between 76% and 82% all account for the pairwise tests perfectly. The 79% HDIs are easiest to evaluate.

```
post <- t(attr(ap_sim, "posterior"))
colnames(post) <- paste0("b", 1:ncol(post))
post_vt <- make_vt_data(post, type="sim")
vt_sim <- viztest(post_vt, cifun="hdi", include_zero=FALSE)
vt_sim

#>
#> Correspondents of PW Tests with CI Tests
#>   level psame     pdiff          easy  method
#> 1  0.76     1 0.6060606 -0.0001228599  Lowest
#> 2  0.79     1 0.6060606 -0.0064294949  Middle
#> 3  0.82     1 0.6060606 -0.0235205361 Highest
#> 4  0.76     1 0.6060606 -0.0001228599 Easiest
#>
#> All  66  tests properly represented for by CI overlaps.
```

While the `plot()` method works with simulation results as well, it is actually a bit easier to simply attach the appropriate credible intervals to the output from `avg_predictions()`.

```
hdis <- apply(post, 2, \(x)hdi(x, 0.79))
mes <- mes %>%
  arrange(ageCat) %>%
  mutate(lwr79 =hdis[1,],
         upr79 = hdis[2,])

ggplot(mes, aes(y = ageCat, x=estimate, xmin = lwr79,
              xmax=upr79, colour=sex)) +
  geom_pointrange(position = position_dodge(width=.5)) +
  scale_colour_manual("Sex",values=c("gray50", "black")) +
  theme_bw() +
  labs(x="Predicted Pr(Survival)\nInferential Highest Density Regions (79%)",
       y="Age Category")
```

VizTest and the Reference Category Problem

The reference category arises when a categorical variable with m different levels is operationalized in a model with $m - 1$ regressors (to prevent perfect collinearity). In the case of treatment contrasts (i.e.,

creating dummy variable regressors), the coefficients on each of the $m-1$ regressors represent the expected difference in the outcome for the group represented by the dummy variable relative to the excluded, or reference, category. For reasons similar to the visual testing problem, tests between two non-reference categories is more challenging (Firth 2003). If coefficients b_1 and b_2 are the coefficients whose difference is to be tested, we would need to make a t-statistic:

$$t = \frac{b_1 - b_2}{\sqrt{\text{var}(b_1) + \text{var}(b_2) - 2\text{cov}(b_1, b_2)}}$$

Normally, the requisite covariance terms are not printed in the model summary.

There are lots of solutions to the reference category problem. The most prominent of these are quasi-variances (Firth and Menzes 2004) and compact letter displays (Piepho 2004), though other solutions like mean-mean scatterplots (Hsu and Peruggia 1994) are also used from time-to-time. In graphical form, the reference category problem is just a special case of the visual testing problem where the desired comparisons are between a subset of the model parameters. To the extent that visual intervals preserve the univariate tests against zero and also account for all pairwise differences, they have solved the reference category problem. Our goal here is not to draw more comprehensive comparisons with solutions in this literature, but simply to note that the solution proposed here may be useful in solving another problem prevalent in statistical model presentation. Figure 4 shows one way the tools we develop here could be used to solve the reference category problem for age group in the Titanic survival model.

Descriptive quantities

For this example, we use the `CES11` dataset from **carData**, calculating the average importance given to religion per Canadian province in 2011. First, we create a numerical scale of the importance of religion from 0 to 1 and calculate the mean and sampling variance for each province. We then create a named vector of the averages per province and pass that vector along with the vector of sampling variances to `make_vt_data()`. Passing the newly formed object to `viztest` indicates that any inferential confidence level from 0.72 to 0.79 properly represents all pairwise tests.

```
#### 4.4 Descriptive quantities ####
data(CES11, package="carData")
NewCES11 <- CES11 %>%
  mutate(rel_imp=(as.numeric(importance)-1)/3) %>%
  group_by(province) %>%
  summarise(mean=mean(rel_imp),
            samp_var=var(rel_imp)/n())

# Creating a vtcustom object
## Vector for the means
means <- NewCES11$mean
names(means) <- NewCES11$province

vt_ces_data <- make_vt_data(means, NewCES11$samp_var)

# Passing to viztest
viztestCES <- viztest(vt_ces_data,
                      test_level = 0.05,
                      range_levels = c(0.25,0.99),
                      level_increment = 0.01,
                      include_zero=FALSE)

# Print
viztestCES

#>
#> Correspondents of PW Tests with CI Tests
#>   level psame     pdiff         easy  method
#> 1  0.72     1 0.6222222 -0.010528869  Lowest
#> 2  0.75     1 0.6222222 -0.001160104  Middle
#> 3  0.79     1 0.6222222 -0.012863144 Highest
#> 4  0.75     1 0.6222222 -0.001160104 Easiest
#>
```

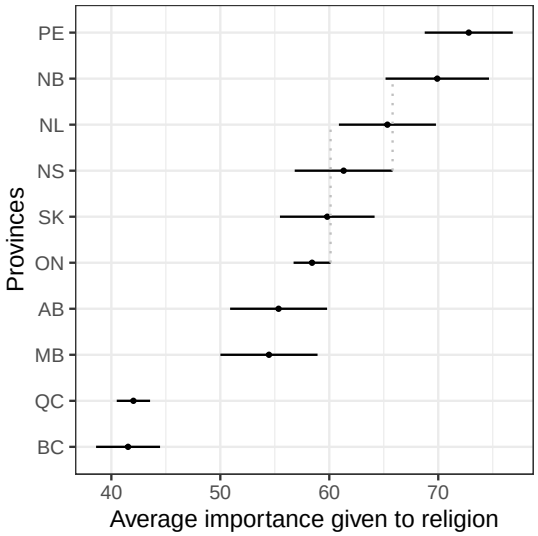

Figure 5: Descriptive quantities — default output with ambiguous reference lines

```
#> All  45  tests properly represented for by CI overlaps.
```

We then plot the estimates in Figure 5, where this time we create a function that will multiply the estimates and their confidence bounds by 100. This results in a display of the importance of religion per province on a scale from 0 to 100 instead of 0 to 1.

```
# plotting
plot(viztestCES, level = "ce", trans = \(x)x*100, ref_lines = "ambiguous")+
  labs(y="Provinces", x="Average importance given to religion") +
  theme_bw()
```

5 Conclusion

As Armstrong II and Poirier (Forthcoming) show, inferential confidence intervals can help solve the visual testing problem. By choosing the appropriate inferential confidence level, users can perform pairwise statistical tests accurately and reliably in most cases. The software described in this article allows users to calculate these inferential confidence levels in R for a wide class of objects using both normal-theory tests as well as tests on Bayesian posterior simulations. Users can interrogate these results with the packages `print()` method and plot the results either with the package's `plot()` method or via their graphing tool of choice.

References

Afshartous, David, and Richard A. Preston. 2010. "Confidence Intervals for Dependent Data: Equating Non-Overlap with Statistical Significance." *Computational Statistics and Data Analysis* 54: 2296–2305. https://doi.org/10.1016/j.csda.2010.04.011.

Armstrong II, David A., and William Poirier. Forthcoming. "Decoupling Visualization and Testing When Presenting Confidence Intervals." *Political Analysis*, Forthcoming. https://doi.org/10.1017/pan.2024.24.

Browne, Richard H. 1979. "On Visual Assessment of the Significance of a Mean Difference." *Biometrics* 35 (3): 657–65. https://doi.org/10.2307/2530259.

Dice, L. R., and H. J. Laraas. 1936. "A Graphic Method for Comparing Several Sets of Measurements." *Contributions from the Lab of Vertebrate Genetics*, no. 3: 1–3.

Few, Stephen. 2012. *Show Me the Numbers: Designing Tables and Graphs to Enlighten, 2nd Ed.* Burlingame, CA: Analytics Press.

Firth, David. 2003. "Overcoming the Reference Category Problem in the Presentation of Statistical Models." *Sociological Methodology* 33: 1–18. https://doi.org/10.1111/j.0081-1750.2003.t01-1-00125.x.

Firth, David, and Renee X. De Menzes. 2004. "Quasi-Variances." *Biometrika* 91 (1): 65–80. https://doi.org/10.1093/biomet/91.1.65.

Goldstein, Harvey, and Michael J. R. Healey. 1995. "The Graphical Presentation of a Collection of Means." *Journal of the Royal Statistical Society, Series A* 158 (1): 175–77. https://doi.org/10.2307/2983411.

Hsu, Jason C., and Mario Peruggia. 1994. "Graphical Representations of Tukey's Multiple Comparison Method." *Journal of Computational and Graphical Statistics* 3 (2): 143–61. https://doi.org/10.1080/10618600.1994.10474636.

Kastellec, Jonathan P., and Eduardo L. Leoni. 2007. "Using Graphs Instead of Tables in Political Science." *Perspectives on Politics* 5 (4): 755–71. https://doi.org/10.1017/S1537592707072209.

Payton, Mark E., Matthew H. Greenstone, and Nathaniel Schenker. 2003. "Overlapping confidence intervals or standard error intervals: What do they mean in terms of statistical significance?" *Journal of Insect Science* 3 (1): 34. https://doi.org/10.1093/jis/3.1.34.

Piepho, H. P. 2004. "An Algorithm for a Letter-Based Representation of All Pairwise Comparisons." *Journal of Computational and Graphical Statistics* 13: 456–66. https://doi.org/10.1198/1061860043515.

Radean, Marius. 2023. "The Significance of Differences Interval: Assessing the Statistical and Substantive Difference Between Two Quantities of Interest." *Journal of Politics* 85 (3): 969–83. https://doi.org/10.1086/723999.

Simpson, George G., and Anne Roe. 1939. *Quantitative Zoology, Revised Edition*. New York, NY: McGraw-Hill.

Tukey, John. 1991. "The Philosophy of Multiple Comparisons." *Statistical Science* 6 (1): 100–116. https://doi.org/10.1214/ss/1177011945.

William Poirier
Western University
Department of Political Science
London, Canada
https://williampo1.github.io/
ORCiD: 0000-0002-3274-1351
wpoirier@uwo.ca

David A. Armstrong
Western University
Department of Political Science
London, Canada
dave.armstrong@uwo.ca