Behavior Research Methods (2023) 55:474–490
https://doi.org/10.3758/s13428-022-01808-5

DIY bootstrapping: Getting the nonparametric bootstrap confidence interval in SPSS for any statistics or function of statistics (when this bootstrapping is appropriate)

Shu Fai Cheung[1] · Ivan Jacob Agaloos Pesigan[1] · Weng Ngai Vong[1]

Accepted: 24 January 2022 / Published online: 15 March 2022

Abstract
Researchers can generate bootstrap confidence intervals for some statistics in SPSS using the `BOOTSTRAP` command. However, this command can only be applied to selected procedures, and only to selected statistics in these procedures. We developed an extension command and prepared some sample syntax files based on existing approaches from the Internet to illustrate how researchers can (a) generate a large number of nonparametric bootstrap samples, (b) do desired analysis on all these samples, and (c) form the bootstrap confidence intervals for selected statistics using the `OMS` commands. We developed these tools to help researchers apply nonparametric bootstrapping to any statistics for which this method is appropriate, including statistics derived from other statistics, such as standardized effect size measures computed from the *t* test results. We also discussed how researchers can extend the tools for other statistics and scenarios they encounter.

Keywords Bootstrapping · Effect sizes · Confidence intervals

Bootstrapping is a useful technique for making inferences about parameter estimates for which analytic solutions for confidence intervals may be difficult to derive or are not yet developed (Efron & Hastie, 2016). Bootstrapping is also useful for making inferences on robust estimators for data with outliers and/or severely nonnormal data (Mair & Wilcox, 2020). On some platforms, it is easy to do bootstrapping. For example, in R (R Core Team, 2021), researchers can use the *boot* package to do bootstrapping for any statistics, as long as they can write a function to compute these statistics (Canty & Ripley, 2021). SPSS, another popular statistical package, also has a bootstrapping function added. However, unlike R, at the time of writing, SPSS only supports selected statistics in selected procedures. Even though we use R in most of our work, migrating to R or other tools, or just using these tools for forming the confidence intervals of some statistics, may not be a cost-effective solution for some researchers because they may use SPSS for some practical reasons, such as working with a team with SPSS integrated into the workflow. Helping researchers who use SPSS learn how to do bootstrapping on more statistics gives them more options in doing analysis. This can also help researchers learn more about the potential of the bootstrapping approach without the need to learn a new computing environment.

Approaches for doing bootstrapping using syntax commands in SPSS have been around on the Internet for a long time (e.g., Nichols, 1996). To help researchers using SPSS have nearly the same flexibility as in R, we present below an extension command and a few sample syntax files to illustrate how researchers can form confidence intervals by bootstrapping for (nearly) any statistics they can get in SPSS. We selected several common scenarios for illustration, and described how the approach can be extended to other scenarios. Our goal is to empower researchers to do bootstrapping in scenarios not covered in this paper and to develop tools for other statistics.

We need to stress that our goal is *not* to develop a general tool for forming bootstrap confidence intervals easily for any statistics. This is impossible given the diversity of scenarios. If there are existing tools that can form the desired bootstrap confidence interval of an analysis (e.g., PROCESS, Hayes, 2018, a useful SPSS macro developed for testing mediation and moderation effects), there is no

✉ Shu Fai Cheung
sfcheung@um.edu.mo

[1] Department of Psychology, Faculty of Social Sciences, University of Macau, Avenida da Universidade, Taipa, Macao SAR, China

reason to reinvent the wheel. Instead, our goal is to illustrate how forming bootstrap confidence intervals can be done using syntax commands. A certain level of understanding of SPSS syntax commands and defining macro is necessary. Nevertheless, we believe researchers who decide to use mainly SPSS should possess this skill because syntax commands also facilitate reproducible research. We believe our illustration can help researchers implement this approach in their situations when existing tools are not available and are not supported by the version of the `BOOTSTRAP` command they have access to, help them develop tools like PROCESS for other statistics and scenarios commonly found in their research areas.

In the following sections, we first briefly review the idea of bootstrapping. We then use standardized regression coefficients in multiple regression to illustrate how researchers can form the bootstrap confidence interval. We then illustrate how this approach can be applied to more complicated scenarios, such as forming the bootstrap confidence interval for a statistic that researchers need to compute from the results of an analysis themselves, for example, Hedges's g in t test.

A brief introduction to bootstrapping

Interval estimation

One common goal in data analysis is to estimate a parameter, such as a population correlation or a regression coefficient. In addition to using point estimation, which yields a value (the estimate) of this parameter, it is now common to report an interval estimate, which yields an interval enclosed by two values (Appelbaum et al., 2018; Pek & Flora, 2018). The popular intervals reported are the confidence intervals. For a 95% confidence interval, if we repeat a study many times, each with a new random sample from the population, we expect that 95% of the intervals computed from these samples will include the population value of the parameter being estimated. In addition to providing an interval estimate, a confidence interval can also be used for hypothesis testing (Greenland et al., 2016). For example, if we are testing a null hypothesis of zero correlation and the 95% confidence interval of a sample correlation does not include zero, we reject the null hypothesis at $100\% - 95\%$ or 5% level of significance (two-tailed). If the interval includes zero, we conclude that there is insufficient evidence to reject the null hypothesis.

Why bootstrapping

In some situations and for some statistics, closed-form formulas are available to compute the confidence limits directly and common statistical packages can report them (e.g., the confidence interval of an unstandardized regression coefficient, labeled B in SPSS). However, all such formulas are derived based on certain assumptions on the population distribution of data and the data generation model. For example, one popular way to form the confidence interval for a Pearson's r is to use Fisher's z transformation (Hotelling, 1953). This method assumes that the two variables have a bivariate normal distribution in the population (Gayen, 1951; Hawkins, 1989). If the assumption of a method is not tenable, then the method can lead to biased estimates of the sampling variance and result in suboptimal confidence intervals, with coverage probability lower than (too conservative) or higher than the nominal level, that is, 95% for a 95% confidence interval (e.g., Bishara & Hittner, 2012). In these situations, nonparametric bootstrapping can be a viable alternative because it does not make any assumptions on the population distribution (Bishara & Hittner, 2017; but note that nonparametric bootstrapping depends heavily on the distribution of data in a particular sample, as will be shown below; see also Efron & Hastie, 2016, Section 10.6, on nonparametric bootstrapping as a "very highly parametrized" bootstrapping). There are also situations in which an analytic solution is possible but complicated. For example, one analytic solution for forming the confidence interval of an indirect effect in mediation uses the distribution of the product of two random variables (e.g., MacKinnon et al., 2004). However, it assumes that the two variables are normally distributed, which is not a tenable assumption for standardized indirect effect (Craig, 1936). In these situations, bootstrapping is a common solution because it also does not require knowing the sampling distribution of the target statistic.

The common implementation

Bootstrapping, proposed over four decades ago (Efron, 1979; see Efron & Hastie, 2016, for an introduction), has many different variants. Unless stated otherwise, by *bootstrapping*, we refer to nonparametric bootstrapping, a variant commonly used in behavioral research. The procedure is simple. Suppose a researcher has a sample of n cases. The following steps will be repeated for a large number of times, say, 5000 times:

1. Draw n cases from the sample, *with replacement*. That is, once a case is drawn, this case is "placed back" to the sample, available for the next draw. Therefore, a case can be drawn more than once. The resulting sample with n cases is a bootstrap sample.
2. The target statistic is computed on this bootstrap sample.

3. The bootstrap estimate is stored, and the bootstrap sample is discarded. (This rarely mentioned step is included for reasons presented below.)

After repeating the steps 5000 times, there will be 5000 bootstrap estimates of the target statistic. The distribution of these 5000 estimates is the empirical sampling distribution of the target statistic, which is used to form the bootstrap confidence intervals.

Common types of bootstrap confidence intervals

To form a confidence interval, a simple and popular method is the *percentile confidence interval*. For a 95% confidence interval, the 2.5th percentile and the 97.5th percentile are found from the distribution. This interval encloses the middle 95% of the distribution. In general, a $100(1-\alpha)\%$ percentile confidence interval is formed by finding the $100(\alpha/2)$th and $100(1-\alpha/2)$th percentiles, where $\alpha=.05$ for a 95% confidence interval.

Another type of bootstrap confidence interval is the *normal theory* (NT) bootstrap confidence interval (Padilla & Divers, 2015; also called *standard confidence interval* in Efron & Tibshirani, 1993). Instead of using percentiles, this method first computes the bootstrap estimate of the standard error of the target statistic, denoted as SE_{xb}. This estimate is simply the standard deviation of the bootstrap estimates (5000 bootstrap estimates in the above example). The NT bootstrap confidence interval is then computed by $[x-z_{\alpha/2}\mathrm{SE}_{xb}, x+z_{\alpha/2}\mathrm{SE}_{xb}]$, where x is the point estimate in the original sample, and $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile in the standard normal distribution, that is, about 1.96 for $\alpha=.05$. This is an *approximate* confidence interval (Efron & Tibshirani, 1993) but has been found to have more accurate coverage probability than the percentile confidence interval in some situations (e.g., Padilla et al., 2012, on Cronbach's alpha).

There are other types of bootstrap confidence interval, such as the bias-corrected-accelerated (BCa) confidence interval (Efron, 1987). Because the percentile confidence interval usually performs satisfactorily and is the default method in some existing tools (e.g., PROCESS, Hayes, 2018), we will focus on percentile confidence intervals in this manuscript. We will also compute the NT bootstrap confidence interval in examples where this method was found to work better in previous studies.

An alternative implementation

Though the idea of bootstrapping is simple, there are different ways to implement it in a program. One common implementation of bootstrapping is as illustrated above: resample, estimate, store the statistic and discard the bootstrap sample, repeat. This ensures that the memory usage is small. If the number of bootstrap samples increases, only the storage of the bootstrap estimate increases, which is just a vector of numbers with its length equal to the number of bootstrap samples. This minimizes the memory usage of this computing intensive method.

However, bootstrapping can also be implemented this way:

1. Generate B bootstrap samples and store them in *one dataset*, with a grouping variable to indicate which bootstrap samples a case belongs to.
2. Loop over the B bootstrap samples to compute the target statistic for each sample.

We call this approach the one-pass approach because the repetition occurs within each step, rather than across all steps. This approach is storage inefficient in both memory usage and hard disk storage. For example, for R, in the default installation, all objects are stored in the memory. Therefore, if 5000 bootstrap samples are pregenerated, the size of this object is about 5000 times the size of the source sample. However, this approach also has four advantages. First, the same set of bootstrap samples can be used for several analyses, allowing for analysis that combines the results from several separate analyses conducted on the same sample, such as indirect effects or difference between independent R^2s. Second, this approach allows researchers to share the source bootstrap samples for others to reproduce the results without the need to regenerate the bootstrap samples.[1] Even with as many as 10,000 bootstrap samples, due to the duplication in information, the file size after compression is not large. Third, in some software packages, such as SPSS, it allows researchers to apply bootstrapping to analysis that does not natively support bootstrapping because Step 2 in this approach is just a multiple-group analysis, available in most software packages. Last, in SPSS, a dataset is not stored entirely in the memory and so memory inefficiency is not a major concern. Therefore, this is the approach we adopted in the present manuscript.[2]

[1] Reproducibility can also be done by sharing the seed and *exactly* how the resampling is conducted. However, this may not be possible in some situations, e.g., when the random number generators changed in a platform. Researchers without relevant technical knowledge may have no way to be certain that the samples they generated are indeed the samples used in a study.

[2] The original 3-step approach, though memory and storage efficient, can be slow in some software packages. For example, the *OMS Bootstrapping* macro adopts this approach. The speed is slow because the overhead in repeating each loop is high.

Bootstrapping in SPSS

Existing command and its limitations

Bootstrapping has been available in SPSS for several years as an add-on module. Since version 27, it is available in the base version. This function is easy to use and has options other than nonparametric bootstrapping (e.g., resampling residuals). It can generate bootstrap confidence intervals for some commonly reported statistics, such as correlation coefficients and unstandardized regression coefficients. However, there are cases in which researchers may need to form the bootstrap confidence intervals for statistics not yet supported, such as R^2 and standardized regression coefficients (labeled *beta* in SPSS), or statistics that are functions of other statistics, such as indirect effects in mediation models and effect size measures in mean comparison. Some specialized tools are needed for this situation, such as the PROCESS macro for indirect effects (Hayes, 2018).

Combining existing methods in the Internet

Other methods have also been proposed to do bootstrapping in SPSS. For example, the macro commands at the website *Ryan's SPSS Tools* (Raynald, n.d.) illustrate how bootstrap confidence intervals can be formed for various statistics. The macro *OMS Bootstrapping* on that page also illustrates that bootstrapping can be done using randomly generated frequency weights to simulate resampling, and then using the `OMS` command to collect and export the results from a table to an SPSS data file. This approach can be extended to forming bootstrap confidence intervals for any statistics that are reported in an output table. The one-pass approach has also been proposed on the Internet for SPSS (e.g., Nichols, 1996). This approach can be done using built-in SPSS syntax commands.

However, despite the existence of these methods, they were rarely used in published papers. Some of them are customized for a specific statistic and researchers may not know how to extend the method to other statistics. Some of them are also slow because they did not use the one-pass approach and so resampling is done for each statistic. In this manuscript, we developed an SPSS extension command to implement the resampling step of one-pass approach and presented sample syntax commands for the second step of this approach for some commonly reported statistics. We also demonstrated how to write simple macro commands to reduce the process to just a few lines of commands. Researchers can use the extension command and adapt the sample syntax commands to form the bootstrap confidence intervals for any statistics reported in SPSS, and even for statistics not reported in SPSS but can be computed from other reported results. Our goal is to empower researchers to apply this approach to their situations.

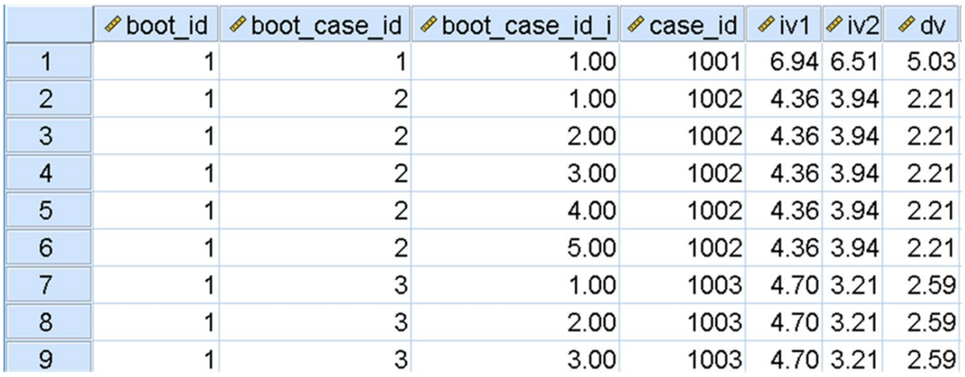
DIY bootstrapping in SPSS

We call this approach *DIY bootstrapping* because, except for the first step (generating *B* bootstrap samples), researchers do the remaining steps themselves using built-in SPSS commands. Once a researcher understands the approach, only a quick search of the documentation is needed to apply the approach to other statistics. We first illustrate how to implement this approach for standardized regression coefficients. We then illustrate this approach for selected statistics that are commonly reported in behavioral research. All files in the following sections are available from the OSF page (https://osf.io/2twf5/).

A numerical example: Standardized regression coefficients

The dataset used in this example is `2iv_regression.sav`, which has 100 cases and three variables, `dv`, `iv1`, and `iv2`, and a case identification number, `case_id`. We will demonstrate how to find the bootstrap confidence interval for standardized regression coefficients, called *beta*s in SPSS, in a multiple regression analysis. Although betas were frequently reported in behavioral studies, finding the unbiased standard error and forming a confidence interval with the desired coverage probability is actually not simple (see Yuan & Chan, 2011, on why the formulas presented in some textbooks are incorrect). Several approaches have been proposed (e.g., Jones & Waller, 2013). In addition to using analytic solutions, bootstrapping is a possible solution, especially because the distributions of the two predictors in this example deviate substantially from normal distributions (`iv1` was generated from an exponential distribution, and `iv2` was generated from a χ^2 distribution [$df = 3$]). Multivariate normal distribution of predictors is not required for estimating the standard errors and forming the confidence intervals of the unstandardized regression coefficients in ordinary least squares (OLS) estimation (Fox, 2016). However, to form the confidence intervals of the standardized regression coefficients, the distribution of the predictors needs to be taken into account (Yuan & Chan, 2011). Given the popularity of reporting standardized regression coefficients in behavioral research, we selected this statistic as the first scenario. If we do a multiple regression to predict `dv` by `iv1` and `iv2`, the betas of `iv1` and `iv2` are .534 and .568, respectively. We will illustrate how to form the percentile bootstrap confidence intervals for these two estimates in SPSS.[3]

[3] The syntax file for this example is `diy_approach_illustration.sps`.



	boot_id	boot_case_id	boot_case_id_i	case_id	iv1	iv2	dv
1	1	1	1.00	1001	6.94	6.51	5.03
2	1	2	1.00	1002	4.36	3.94	2.21
3	1	2	2.00	1002	4.36	3.94	2.21
4	1	2	3.00	1002	4.36	3.94	2.21
5	1	2	4.00	1002	4.36	3.94	2.21
6	1	2	5.00	1002	4.36	3.94	2.21
7	1	3	1.00	1003	4.70	3.21	2.59
8	1	3	2.00	1003	4.70	3.21	2.59
9	1	3	3.00	1003	4.70	3.21	2.59

Fig. 1 The generated bootstrap samples

Step 1: Generate *B* bootstrap samples

Because this step does not depend on what the target statistic is, we developed a simple extension command, `generate nonpar bootsamples`, to integrate the aforementioned methods to conduct this step.[4] This extension command can be downloaded from the aforementioned OSF page. Once downloaded, select *Extension* and then *Install Local Extension Bundles …*, and select the downloaded file to install this command.[5] This command needs the *Integration Plug-In for Python*, which is installed by default along with SPSS.[6] Although this step can also be conducted by built-in SPSS commands as in the macros mentioned above, we found it easier to implement the resampling step using Python functions, to make the whole process as automatic as possible. The command will automatically find the number of cases (sample size) in the active dataset, while other approaches require researchers to input this value manually. Therefore, users do not need to change the syntax even if the sample size changed. Once installed, a researcher only needs to open the dataset to be resampled, and run a command similar to the following one[7]:

```
        generate nonpar bootsamples b = 5000
         /options outfile = 'H:/temp/2iv_regression_bootsamples.sav'
                  seed = 53243.
```

This command has two required arguments, `b` is the number of bootstrap samples, *B* (5000 in the example), `outfile` in the subcommand `/options` is the file name of the output data file to store the bootstrap samples (`'H:/temp/2iv_regression_bootsamples.sav'` in the example). The optional argument `seed` in `/OPTIONS` is the seed for the random number generator (53243 in the example). This seed ensures that the same set of bootstrap samples will be generated every time the command is run. Users can also generate the command above using a custom dialog installed with the extension (located in the Data menu, titled "Generate Bootstrap Samples").[8]

After running this command with the source dataset opened and active, output similar to the following one will appear in the SPSS output file:

```
Source data file:
 H:/gbootdatext/examples/2iv_regression.sav
Data file with 100 cases in each copy:
 H:/temp/2iv_regression_bootsamples.sav
```

The command reports the locations of the active data file and the generated file. Users can open the latter file

[4] A similar command, GSD (Harding & Cousineau, 2016), has been developed but for a different purpose. Moreover, our command uses Python functions to do the resampling, while GSD uses SPSS built-in commands.

[5] A real time recording of installing the extension command and running the syntax commands in the file is available from the OSF page: https://osf.io/95rhx/

[6] We used only the standard libraries installed with Python 3.4, the version installed along with SPSS 26. Therefore, users of SPSS 26 or later version do not need to do any additional steps once SPSS is installed. They also do not need administrative privileges to their computers, which usually are not available for computers managed by their institutions.

[7] Although it is conventional to use uppercase for SPSS commands, we used lowercase in the syntax command examples and the syntax files due to better readability.

[8] Watch the video on OSF on how to use the custom dialog: https://osf.io/3bdmj/

and verify that the bootstrap samples are correctly generated (Fig. 1). In this file, boot_id, boot_case_id, and boot_case_id_i, are automatically generated. The bootstrap samples are identified by boot_id, ranging from 1 to *B* (5000 in this example). Case identification number, boot_case_id, is automatically generated, unique for each case in the source data file. In this example, the cases with boot_case_id equal to 1, 2, and 3 were drawn once, five times, and thrice, respectively. The variable boot_case_id is the unique identification number for each draw of each case (e.g., 1 to 5 for the five draws of the case with boot_case_id equal 2). In each bootstrap sample, there are 100 cases, the sample size of the source data file. The total number of cases in this dataset is 5000 × 100 or 500,000 cases.

This step only needs to be conducted once. The file can be used for finding the bootstrap confidence intervals of any statistics to be computed on this sample. To facilitate data sharing, this file can be compressed and shared in a data repository (e.g., *Open Science Framework*). If file size is a concern, researchers can also just share a file with the bootstrap sample identification number (boot_id) and the case identification number in the source data file (case_id in this example), which are sufficient for other researchers to reproduce the bootstrap samples if the source data file is also shared.

Step 2: Do the analysis *B* times

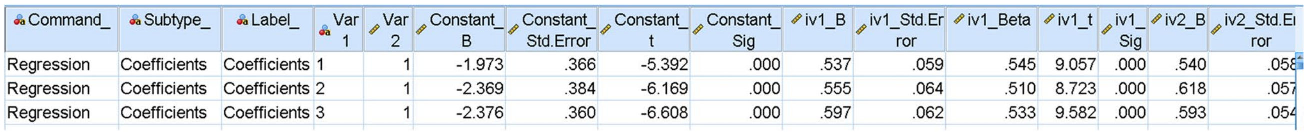
With the dataset of *B* bootstrap samples generated above, it is easy to do a regression analysis *B* times in SPSS using split file by boot_id, boot_id being the variable that identifies each bootstrap sample. However, to efficiently store these *B* sets of results, we need to use the OMS command. This command can save the content of selected output tables to an external file, such as regression coefficient tables. This is an example:

```
split file by boot_id.
oms /destination viewer = no.
oms /select tables
    /if commands = ["Regression"] subtypes = ["Coefficients"]
 /destination format = sav
             outfile = "H:/temp/regression_coefficients.sav"
 /columns dimnames = ["Variable" "Statistics"].
regression
 /dependent dv
 /method = enter iv1 iv2.
omsend.
split file off.
```

This is a typical block in the DIY approach. First, we turn on the split file mode. The line oms /destination viewer = no suppresses the results in the output window, such that SPSS will not print a large table with 5000 sets of results. The second OMS command direct the results of some output tables to an external SPSS data file. In /if, the commands option specifies the name of the procedure ("Regression" in the example), and the subtypes option specifies the type of the table in this procedure. Because the betas are reported in the "Coefficients" table, the subtypes option is set to "Coefficients".[9] The outfile option in the destination subcommand specifies the name of the data file to store the results ("H:/temp/regression_coefficients.sav" in the example). To store the results of each bootstrap result in one row, we need to specify the dimnames option in the /column subcommand. This is the most complicated part for this command. However, the sample syntax commands we provided above and in the following sections should be sufficient for most typical scenarios. If the regression analysis to be conducted has only one model, then dimnames = ["Variable" "Statistics"] ensures that the resulting data file will have one row for each bootstrap sample. For researchers who prefer using the graphical user interface, a video demonstration is available on the OSF page to illustrate how to generate the aforementioned syntax commands using nearly only the pull-down menus and dialog boxes (https://osf.io/3bdmj/).

After setting up the OMS commands, users include analysis commands as if the analysis is to be conducted in the original sample. For example, the command regression above is just a typical regression analysis with one model, predicting dv by iv1 and iv2. Users can also use the commands generated automatically by the dialog box in the SPSS graphical user interface. The only requirement is that these are the same commands used in the original sample to get the point estimate of the target statistics. With

[9] SPSS has an OMS Control Panel to find the command and subtype identifiers of SPSS procedures (https://www.ibm.com/docs/en/spss-statistics/26.0.0?topic=oms-command-subtype-identifiers-command).



Command_	Subtype_	Label_	Var 1	Var 2	Constant_ B	Constant_ Std.Error	Constant_ t	Constant_ Sig	iv1_B	iv1_Std.Er ror	iv1_Beta	iv1_t	iv1_ Sig	iv2_B	iv2_Std.Er ror
Regression	Coefficients	Coefficients	1	1	-1.973	.366	-5.392	.000	.537	.059	.545	9.057	.000	.540	.058
Regression	Coefficients	Coefficients	2	1	-2.369	.384	-6.169	.000	.555	.064	.510	8.723	.000	.618	.057
Regression	Coefficients	Coefficients	3	1	-2.376	.360	-6.608	.000	.597	.062	.533	9.582	.000	.593	.054

Fig. 2 The data file with the regression coefficients for each bootstrap sample

Table 1 The variable labels in the regression coefficients data file

	Name	Label
1	Command_	
2	Subtype_	
3	Label_	
4	Var1	
5	Var2	
6	Constant_B	(Constant) Unstandardized Coefficients B
7	Constant_Std.Error	(Constant) Unstandardized Coefficients Std. Error
8	Constant_t	(Constant) t
9	Constant_Sig	(Constant) Sig.
10	ivl_B	ivl Unstandardized Coefficients B
11	ivi_Std.Error	ivl Unstandardized Coefficients Std. Error
12	ivi_Beta	ivl Standardized Coefficients Beta
13	ivl_t	
14	ivl_Sig	ivl Sig.
15	iv2_B	iv2 Unstandardized Coefficients B
16	iv2_Std.Error	iv2 Unstandardized Coefficients Std. Error
17	iv2_Beta	iv2 Standardized Coefficients Beta
18	iv2_t	iv2 t
19	iv2_Sig	iv2 Sig.

split file on, this analysis will be repeated *B* times, once for each bootstrap sample. All results will be discarded, except that the tables selected by the previous `OMS` command (i.e., the table with the betas in this example) will be stored in the destination file.

After the commands for the analysis, `omsend` is used to end all active `OMS` commands, and `split file off` is used to turn off the split file mode. We recommend keeping the block of code for an analysis self-contained, starting with the `split file` command and `OMS` commands, and ending with the `omsend` command and the `split file off` command. This makes the code easier to read and debug, especially if users run the code line-by-line interactively, or form the bootstrap confidence intervals for several different sets of analysis.

After running this block of command, SPSS will loop over the *B* bootstrap samples and run the regression analysis once for each sample. The output window should have no results because all tables except for the selected tables (the tables with the regression coefficients) are discarded. This process may take 30 seconds to a few minutes on some computers, depending on the time to do the analysis in one sample. After the commands finished running, users can open the output data file (`"H:/temp/regression_coefficients.sav"` in the example) and verify the results (Fig. 2).

A lot of variables are stored in this data file because the `OMS` command will store all the information in the selected tables. However, most of them can be ignored. Researchers only need to identify the columns with the target statistics. The variable names may be difficult to read but the variables are usually labeled clearly (Table 1). In this example, the standardized regression coefficients estimated in each sample are stored in `iv1_Beta` and `iv2_Beta`, labeled as *iv1 Standardized Coefficients Beta* and *iv2 Standardized Coefficients Beta*, respectively. The labels are usually generated based on the labels in the original output tables.

Step 3: Form the bootstrap confidence interval

Once we know which variables store the target statistics, the percentile confidence intervals can be easily formed by

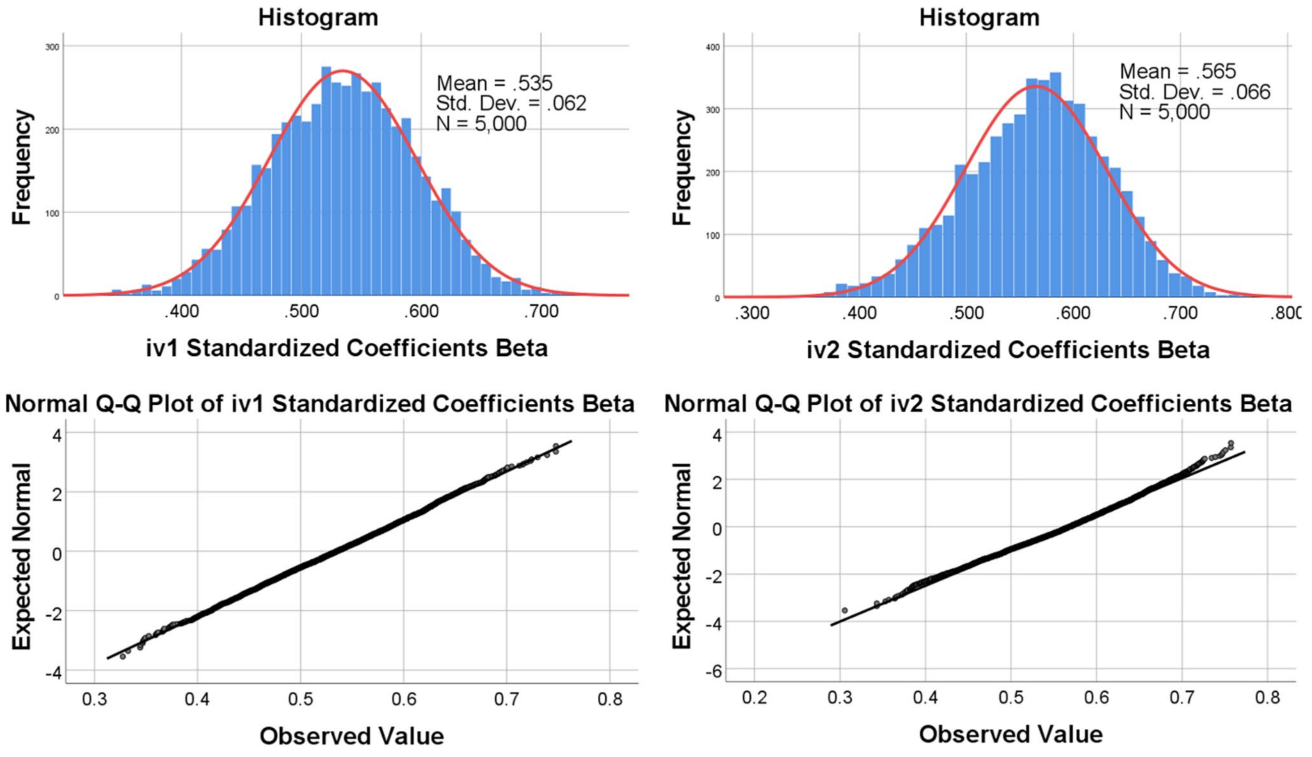


Fig. 3 The histograms and normal Q-Q plots of 5000 bootstrap estimates of betas in the example (normal curves with the same means and SDs overlaid in the histograms)

built-in SPSS procedures. We recommend using the `EXAMINE` command (named `Explore` in the pull-down menu) to request the percentiles, basic descriptive statistics including skewness and kurtosis, histograms, and normal Q-Q plots because, in addition to getting the percentiles to form the interval, researchers should also examine the empirical distribution of the bootstrap estimates (Rousselet et al., 2021):

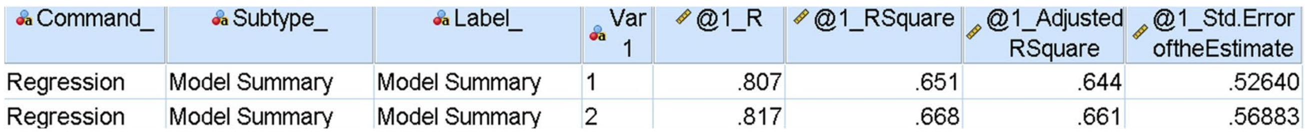
```
examine
  variables = iv1_Beta iv2_Beta
 /plot histogram npplot
 /percentiles(2.5, 97.5)
 /statistics descriptives.
```

In this example, the 95% bootstrap percentile confidence intervals of the betas of `iv1` (.534) and `iv2` (.568) are .409 to .658 and .426 to .684, respectively. If the textbook formula (Yuan & Chan, 2011) is used, the 95% confidence intervals of the betas of `iv1` and `iv2` are .416 to .651 and .450 to .685, respectively. The two confidence intervals are similar for `iv1`, while the bootstrap percentile confidence interval for `iv2` is wider than the textbook-formula confidence interval.

The histograms and the normal Q-Q plots (requested by the subcommand `/plot`) can be used to examine the distributions of the bootstrap estimates. If the excess kurtosis is high, the distribution has heavy tail(s) (extreme values in one or both ends). This suggests that a larger number of bootstrap samples may be needed to have stable estimates of the confidence limits because these limits are located at the two tails of the distribution. The histograms and normal Q-Q plots in this example are shown in Fig. 3. The distributions of both betas are close to normal, although that of `iv2` beta is slightly negatively skewed (skewness = −.271). The excess kurtosis values of bootstrap `iv1` betas and `iv2` betas are .030 and .072, respectively.

Other examples

In this section, we will illustrate how DIY bootstrapping can be used to form the bootstrap confidence intervals for other statistics. Some cases were selected to illustrate how DIY bootstrapping can be carried out in more complicated scenarios, such as forming the confidence interval for statistics which are computed from other statistics (e.g., Hedges's *g* computed from *t* test results). These scenarios illustrate how readers can extend the approach to other scenarios not covered here. Many of the steps are similar to the previous



Command_	Subtype_	Label_	Var1	@1_R	@1_RSquare	@1_Adjusted RSquare	@1_Std.Error oftheEstimate
Regression	Model Summary	Model Summary	1	.807	.651	.644	.52640
Regression	Model Summary	Model Summary	2	.817	.668	.661	.56883

Fig. 4 The data file with the model summary results for each bootstrap sample

example. Therefore, we only highlight the differences from the previous example. The complete SPSS syntax files and data files for these examples can be found in the `examples` folder at the OSF page.

Adjusted R^2

A commonly reported statistic in multiple regression is adjusted R^2. Although there are formulas for forming the confidence interval for R^2 or adjusted R^2, they usually assume the predictors are multivariate normal and/or the sample size is large (Algina, 1999). However, nonnormal predictors are common in behavioral research, such as dichotomous variables (e.g., gender), skewed variables (e.g., age), or even dummy variables. Bootstrap confidence interval is one possible solution for adjusted R^2 (Ohtani, 2000).

In this example, we illustrate how to form the bootstrap confidence interval for the adjusted R^2 in the previous example. Because the bootstrap samples have already been generated in the previous example, we can go directly to Step 2. The syntax commands for this task are similar to those in the previous example:

Table 2 The variable labels in the model summary data file

Name	Label
Command_	
Subtype_	
Label_	
Var1	
@1_R	Model 1 R
@1_RSquare	Model 1 R Square
@1_AdjustedRSquare	Model 1 Adjusted R Square
@1_Std.ErroroftheEstimate	Model 1 Std. Error of the Estimate

```
split file by boot_id.
oms /destination viewer = no.
oms /select tables
 /if commands = ["Regression"] subtypes = ["Model Summary"]
 /destination format = sav
              outfile = "H:/temp/model_summary.sav"
 /columns dimnames = ["Model" "Statistics"].
regression
 /dependent dv
 /method = enter iv1 iv2.
omsend.
split file off.
```

The only difference is the `OMS` command used to export the output. The `commands` argument is still `"Regression"` but the `subtypes` argument is `"Model Summary"`, the table in which the adjusted R^2 is reported. The argument `dimnames` is set to `["Model" "Statistics"]`. After running these commands, the bootstrap adjusted R^2 are stored in the output data file (Fig. 4 and Table 2). The adjusted R^2s are stored in the column `@1_AdjustedRSquare` in this example. The `EXAMINE` command can then be used to form the 95% bootstrap percentile confidence interval. The 95% bootstrap percentile confidence of the adjusted R^2 is .550 to .752. The distribution is slightly negatively skewed (skewness = −.317), with excess kurtosis .026.

If we know in advance that we want to form the bootstrap confidence intervals for both the adjusted R^2 and the standardized regression coefficients, we can have two `OMS` commands, one after the other, and then the regression command, such that the regression only needs to be run once for each bootstrap sample. Alternatively, users can include several `OMS` commands before the analysis command, to export all major tables in a regression analysis as shown below:

```
split file by boot_id.
oms /destination viewer = no.
oms /select tables
    /if commands = ["Regression"] subtypes = ["Model Summary"]
    /destination format = sav
                 outfile = "H:/temp/model_summary.sav"
 /columns dimnames = ["Model" "Statistics"].
oms /select tables
    /if commands = ["Regression"]
        subtypes = ["Coefficients"]
    /destination format = sav
                 outfile = "H:/temp/regression_coefficients.sav"
    /columns dimnames = ["Variable" "Statistics"].
regression
 /dependent dv
 /method = enter iv1 iv2.
omsend.
split file off.
```

The two `OMS` commands can be active at the same time because they process different tables. After running this block of commands, users can use one output data file to form the confidence intervals of standardized regression coefficients, and the other output data file to form the confidence intervals of adjusted R^2, without the need to run the 5000 regression analyses twice. This technique can be used for other analyses in which the confidence intervals for statistics in different tables will be formed.

R^2 Change

Another commonly reported statistic in behavioral research is R^2 change, the difference in R^2 between two regression models, one with one or more predictors added to the other model. For example, in the sample dataset `3iv_regresson.sav` with 200 cases, there are three predictors (`iv1`, `iv2`, and `iv3`) and one dependent variable (`dv`). Suppose the first model, Model 1, has one predictor, `iv1`, and the second model, Model 2, has two predictors added, `iv2` and `iv3`. The R^2s of Models 1 and 2 are .062 and .224, respectively. The R^2 change is .224 – .062 or .162, significant ($p < .001$). Like the previous example, the predictors are not normal, making existing analytic solutions inappropriate. Algina et al. (2010) proposed using bootstrap percentile confidence intervals to form interval estimates of R^2 changes. We will illustrate how to do this using DIY bootstrapping.

Step 1 is similar to that in previous example, generating the bootstrap samples:

```
generate nonpar bootsamples b = 5000
 /options outfile = "H:/temp/3iv_regression_bootsamples.sav"
          seed = 53243.
```

In Step 2, the bootstrap samples file is opened and the following commands are used to do the hierarchical regression analysis once for each bootstrap sample.

```
split file by boot_id.
oms /destination viewer = no.
oms /select tables
    /if commands = ["Regression"]
        subtypes = ["Model Summary"]
    /destination format = sav
              outfile = "H:/temp/model_summary_with_change.sav"
    /columns dimnames = ["Model" "Statistics"].
regression
 /statistics default change
 /dependent dv
 /method = enter iv1
 /method = enter iv2 iv3.
omsend.
split file off.
```

Table 3 A sample model summary table with more than one model

Model Summary					
Model	R	R Square	Adjusted R Square	Change Statistics	
				R Square Change	Sig. F Change
1	.249[a]	.062	.057	.062	.000
2	.473[b]	.224	.212	.162	.000

[a]Predictions: (Constant). iv1

[b]Predictions: (Constant), iv1, iv2, iv3

For multiple regression in SPSS, if there are more than two models, the model summary table will have several rows, one for each model (Table 3). The `OMS` command in the previous example can be used again in hierarchical regression analysis because `dimnames = ["Model" "Statistics"]` will automatically spread the rows for models across columns (variables). As shown in Table 4, the results for each model are prefixed by "@#_", where # is the model number. In this example, we want to form the bootstrap confidence interval for the R^2 change when adding `iv2` and `iv3` to a model with `iv1` only. Therefore, the statistic we need is `@2_RSquareChange`.

In Step 3, we simply use `EXAMINE` on `@2_RSquareChange`. Based on the 2.5th and 97.5th percentiles, the 95% percentile confidence interval of the R^2 change is .083 to .269. The distribution of the bootstrap estimates is slightly positively skewed (skewness .279, excess kurtosis .040).

Note that R^2 change will never be negative. This poses a problem for the percentile confidence interval because its lower limit can never be negative even if the population R^2 change is zero. Algina et al. (2010) proposed a modified percentile confidence interval for R^2 change. If the R^2 change is not significant at the corresponding level of significance (.05 for 95% level of confidence), the lower limit is changed to zero. Therefore, if the R^2 change is not significant, researchers can ignore the 2.5th percentile and set the lower limit to zero.

Mean difference: Hedges's *g*

When researchers compare two sample means, Hedges's *g* and Cohen's *d* are commonly reported effect sizes, the former being a less biased estimate of population standardized mean difference (Hedges, 1981). Cohen's *d* can be computed directly from the sample *t* in the *t* test of the two sample means:

$$d = t\sqrt{1/n_1 + 1/n_2},$$

where n_1 and n_2 are the sample sizes of the two samples. Hedges's g, the preferred statistic, can then be computed from Cohen's d:

$$g = d\left(1 - 3/\left[4\left(n_1 + n_2\right) - 9\right]\right)$$

Table 4 The variable labels in the model summary data file with more than one model

	Name	Label
1	Command_	
2	Subtype_	
3	Label_	
4	@1_R	Model 1 R
5	@1_RSquare	Model 1 R Square
6	@1_AdjustedRSquare	Model 1 Adjusted R Square
7	@1_Std.Errorofthe Estimat	Model 1 Std. Error of the Estimate
8	@1_RSquareChange	Model 1 Change Statistics R Square Change
9	@1_FChange	Model 1 Change Statistics F Change
10	@l_df1	Model 1 Change Statistics df1
11	@1_df2	Model 1 Change Statistics df2
12	@1_Sig.FChange	Model 1 Change Statistics Sig. F Change
13	@2_R	Model 2 R
14	@2_RSquare	Model 2 R Square
15	@2_AdjustedRSquare	Model 2 Adjusted R Square
16	@2_Std.ErroroftheEstimate	Model 2 Std. Error of the Estimate
17	@02_RSquareChange	Model 2 Change Statistics R Square Change
18	@2_FChange	Model 2 Change Statistics F Change
19	@2_df1	Model 2 Change Statistics df1
20	@2_df2	Model 2 Change Statistics df2
21	@2_Sig.FChange	Model 2 Change Statistics Sig. F Change

The dataset used in this example, mean_difference.sav, has a group variable, gp (1 = "Group A", 2 = "Group B"), and a dependent variable, dv_gp. Each group has 50 cases. The means for Group A and Group B are 49.70 and 55.74, respectively, and significantly different, $t(98) = 2.939$[10], $p = .004$. The Hedges's g is .588, a medium effect. Bootstrapping is a viable procedure to form the confidence interval of Hedges's g, especially when the population distributions are suspected to be nonnormal (Algina et al., 2006)[11].

To do bootstrapping, resampling needs to be done *within each group*. Otherwise, each bootstrap sample may have a different number of cases from each group. In this example, we will illustrate how bootstrap samples can be generated for subsamples, and then combine them.

These are the syntax commands:

```
temporary.
select if gp = 1.
generate nonpar bootsamples b = 5000
 /options outfile =
"H:/temp/mean_difference_gp1_bootsamples.sav" seed = 15324.

temporary.
select if gp = 2.
generate nonpar bootsamples b = 5000
 /options outfile =
"H:/temp/mean_difference_gp2_bootsamples.sav" seed = 43143.
```

To generate bootstrap samples for each group, we use one block of command for each group. The command generate nonpar bootsamples takes into account pending transformation before generating the bootstrap samples. In each block, we use TEMPORARY and SELECT IF such that the generate nonpar bootsamples command does the resampling only for the selected cases. Used alone, SELECT IF will delete cases not selected. However, with TEMPORARY run right before SELECT IF, the selection is effective only for operations up to the next procedure that will read the data and do the analysis.[12] Other than these two lines, the commands are similar to those used in previous examples. After running these blocks, the bootstrap samples file for Group A (gp = 1) is stored in "H:/temp/mean_difference_gp1_bootsamples.sav", and that for Group B (gp = 2) is stored in "H:/temp/mean_difference_gp2_bootsamples.sav". They can then be combined using the ADD FILES command:

```
add files
 /file = "H:/temp/mean_difference_gp1_bootsamples.sav"
 /file = "H:/temp/mean_difference_gp2_bootsamples.sav".
dataset name boot_samples window = front.
sort cases by boot_id.
save outfile "H:/temp/mean_difference_bootsamples.sav".
exe.
```

[10] We reversed the sign because SPSS computes the t by subtracting the second group (Group B) from the first group (Group A), and the mean difference by subtracting the mean of the second group (55.74) from the mean of the first group (49.70).

[11] Note that Algina et al. (2006) found that the percentile confidence interval, though performed satisfactorily in many conditions they examined, could perform worse than other procedures in some situations. This issue will be discussed in the Final Remarks section of this manuscript.

[12] Due to the implementation of the DataStep class in the SPSS Python module, the FILTER command cannot be used for this purpose. We have to use TEMPORARY and SELECT IF.

Table 5 The variable labels in the *t* test results data file

Name	Label
Command_	
Subtype_	
Label_	
Var1	
dv_gp_Equalvariancesassumed_F	dv_gp Equal variances assumed Levene's Test for Equality of Variances F
dv_gp_Equalvariancesassumed_Sig	dv_gp Equal variances assumed Levene's Test for Equality of Variances Sig
dv_gp_Equalvariancesassumed_t	dv_gp Equal variances assumed t-test for Equality of Means t
dv_gp_Equalvariancesassumed_df	dv_gp Equal variances assumed t-test for Equality of Means df
o'v_gp_Equalvariancesassumed_Sig.2tailed	dv_gp Equal variances assumed t-test for Equality of Means Sig. (2-tailed)
dv_gp_Equalvariancesassumed_MeanDifference	dv_gp Equal variances assumed t-test for Equality of Means Mean Difference
dv_gp_Equalvariancesassumed_Std.ErrorDifference	dv_gp Equal variances assumed t-test for Equality of Means Std. Error Difference
dv_gp_Equalvariancesassumed_Lower	dv_gp Equal variances assumed t-test for Equality of Means 95% Confidence Interval of the Difference Lower
dv_gp_Equalvariancesassumed_Upper	dv_gp Equal variances assumed t-test for Equality of Means 95% Confidence Interval of the Difference Upper
dv_gp_Equalvariancesnotassumed_t	dv_gp Equal variances not assumed t-test for Equality of Means t
dv_gp_Equalvariancesnotassumed_df	dv_gp Equal variances not assumed t-test for Equality of Means df
dv_gp_Equalvariancesnotassumed_Sig.2tailed	dv_gp Equal variances not assumed t-test for Equality of Means Sig. (2-tailed)
dv_gp_Equalvariancesnotassumed_MeanDifference	dv_gp Equal variances not assumed t-test for Equality of Means Mean Difference
dv_gp_Equalvariancesnotassumed_Std.ErrorDifference	dv_gp Equal variances not assumed t-test for Equality of Means Std. Error Difference
dv_gp_Equalvariancesnotassumed_Lower	dv_gp Equal variances not assumed t-test for Equality of Means 95% Confidence Interval of the Difference Lower
dv_gp_Equalvariancesnotassumed_Upper	dv_gp Equal variances not assumed t-test for Equality of Means 95% Confidence Interval of the Difference Upper

We need to sort the cases by `boot_id` (`sort cases by boot_id`) such that cases in each pair of bootstrap samples (one for Group A and one for Group B) are grouped together to a bootstrap sample with 100 cases. The resulting file has 5000 bootstrap samples, each with 100 cases, 50 cases from Group A and 50 cases from Group B.

Step 2 is similar to those in previous examples, except that `T-TEST` is used to test the difference in sample means:

Because the sample sizes are constant ($n_1 = n_2 = 50$), we only need the sample *t*. In the `OMS` command, the `commands` option is `"T-Test"`, the `subtypes` option is `"Independent Samples Test"`. To instruct SPSS to have all values stored in one row, we set `dimnames` to `"Dependent Variables" "Assumptions" "Statistics"`. The `t-test` command is just the usual `t-test` command for one single sample.

```
split file by boot_id.
oms /destination viewer = no.
oms /select tables
    /if commands = ["T-Test"]
        subtypes = ["Independent Samples Test"]
    /destination format = sav
                outfile = "H:/temp/independent_samples_test.sav"
    /columns dimnames = ["Dependent Variables"
                         "Assumptions" "Statistics"].
   t-test groups = gp(1 2)
    /variables = dv_gp.
   omsend.
   split file off.
```

Table 6 The variable labels in the reliability results data file

Name	Label
Command_	
Subtype_	
Label_	
Var1_	
CronbachsAlpha	Cronbach's Alpha
NofItems	N of Items

After running this block, the output can be found in the destination ("`H:/temp/independent_samples_test.sav`" in this example). The sample *t*s are stored in `dv_gp_Euqalvariancesassumed_t` (Table 5). Unlike the previous examples, we need to compute bootstrap estimates of Cohen's *d* and Hedges's *g* ourselves using the formulas above.[13] This can be easily done because we already know the sample sizes for the two groups, and 5000 bootstrap *t* statistics are stored as a variable in the data file:

```
compute cohend = dv_gp_Equalvariancesassumed_t *
                 sqrt(1 / 50 + 1 / 50).
compute hedgesg = cohend * (1 - 3 / (4 * (50 + 50) - 9)).
execute.
```

Using the `EXAMINE` command, the 95% bootstrap percentile confidence interval is .212 to .989,[14] suggesting that the effect size estimate of .588, though significant, has a moderately wide interval that spans the range from small effect to large effect as labeled by Cohen (1988).

This example illustrates how DIY bootstrapping can be used to form confidence intervals for statistics that are not readily available in SPSS output tables but need to be computed from the results. This technique can be extended to other statistics, such as omega-square for ANOVA (Hays, 1988).

Cronbach's alpha

The last example we selected is Cronbach's alpha. This statistic is reported in many behavioral studies that used psychological measurements. However, unlike other statistics usually presented along with it, few studies reported the confidence intervals for Cronbach's alphas. Moreover, nonnormality is not uncommon for item-level data, making existing closed-form solutions inappropriate. In this example, the data file is `scale_items.sav`. It has 100 cases and 10 variables (`item1` to `item10`) that are used to form a 10-item scale. The factor scores used to generate that data were drawn from a χ^2 distribution, to simulate situations in which the attribute being measured is naturally skewed in the population. The Cronbach's alpha for this 10-item scale is .736.

After generating 5000 bootstrap samples, these commands will be used to compute the Cronbach's alpha 5000 times:

```
split file by boot_id.
oms /destination viewer = no.
oms /select tables
    /if commands = ["Reliability"]
        subtypes = ["Reliability Statistics"]
    /destination format = sav
                 outfile = "H:/temp/reliability_statistics.sav"
    /columns dimnames = ["Statistics"].
reliability
  /variables item1 to item10
  /scale('All items') all
  /model = alpha.
omsend.
split file off.
```

The command to be used is `RELIABILITY`. Therefore, the `commands` option in the `OMS` command is set to `"Reliability"`. The table in which Cronbach's alpha is reported is `"Reliability Statistics"` and so the `subtypes` option is set to this value. This table only has one row and so the `dimnames` is just `"Statistics"`.

This dataset is much easier to understand than those in the previous examples (Table 6). The bootstrap sample estimates of Cronbach's alpha are stored in the variable `CronbachsAlpha`. The percentile confidence interval is .634 to .804. Padilla et al. (2012) recommended using the NT confidence interval for Cronbach's alpha. The bootstrap estimate of the standard error is the standard deviation of `CronbachsAlpha`, which is .044. Therefore, the 95% NT confidence interval of the Cronbach's alpha is $(.736 - 1.96 \times .045)$ to $(.736 + 1.96 \times .045)$, or .650 to .822. Although its point estimate passed the usual cutoff of .70, there is not sufficient evidence to conclude that the population Cronbach's alpha is higher than .70 (although also not sufficient evidence to conclude that it is lower than .70).

Simplifying the syntax files by using ad hoc macros

To make a syntax file doing DIY bootstrapping more compact and readable, researchers can write simple ad hoc macros for their analyses. Examples can be found in the `macros` folder on OSF page. For example, the syntax commands for Step 2 in the example for standardized regression

[13] SPSS 27 and later can report Hedges's *g* directly. We did not have access to this version when we prepared the first draft of this manuscript. We kept this example because there may be users who have no access to SPSS 27, and this example can also illustrate how to compute other desired statistics from the OMS output, such as other newly proposed effect size measures not yet supported in SPSS.

[14] Note that the signs have been reversed as we did for the *t* statistics, for the same reason mentioned before.

coefficients and adjusted R^2 can be simplified to these four commands using macros:

```
bregrsq out = "H:/temp/regression_model_summary.sav".
bregcoef out = "H:/temp/regression_coefficients.sav".
regression
 /dependent dv
 /method = enter iv1 iv2.
bomsend.
```

The first two commands, `bregrsq` and `bregcoef`, are simple macros that accept one argument, the destination file. It is easy to write this kind of macros. For example, the SPSS syntax commands to define `bregrsq` are as follow:

```
define bregrsq(out = !tokens(1))
split file by boot_id.
oms /destination viewer = no.
oms /select tables
 /if commands = ["Regression"] subtypes = ["Model Summary"]
 /destination format = sav outfile = !out
 /columns dimnames = ["Model" "Statistics"].
!enddefine.
```

The first command, `define`, sets the name of the macro (`bregrsq`) and the argument it accepts (`out = !tokens(1)`), which is one argument named `out`, with one element (`!tokens(1)`). The definition of this macro ends at `!enddefine`. The lines between these two commands are the same commands used in previous examples, except that the name of the output file after `outfile` is replaced by `!out`. When `bregrsq` is run, it will replace `!out` by its value (`"H:/temp/regression_model_summary.sav"` in the example), and then run the modified syntax commands. Macros for the `OMS` commands for other procedures and tables can be written similarly because usually the destination file is the only variable that may change across analysis. The last macro, `bomsend`, only has two commands:

```
define bomsend()
omsend.
split file off.
!enddefine.
```

It has no argument. Its purpose is to wrap up a block of the analysis. Researchers can use our templates in the file `diybootstat_macros.sps` at the macros folder of the OSF page to write macros for frequently used blocks of commands for doing bootstrapping for other statistics.

Final remarks on using bootstrapping

Although nonparametric bootstrapping is a useful technique, there are several issues that researchers need to pay attention to. First, nonparametric bootstrapping, and bootstrapping in general, may yield suboptimal confidence intervals in some situations for some statistics. Therefore, the appropriateness of bootstrapping for a particular statistic should not be taken for granted but should be based on empirical evidence in previous statistical studies. For example, although nonparametric percentile bootstrap confidence interval performed well for indirect effect (Cheung, 2009; MacKinnon et al., 2004) and is the default in PROCESS (Hayes, 2018), it has been found to have suboptimal coverage probabilities when applied to the difference between two independent R-square when both population R-squares are zero (Chan, 2008). Although DIY bootstrapping can be used for virtually any statistics reported in SPSS, or statistics derived from them, researchers still need to consult simulation studies to decide whether bootstrap confidence intervals are likely to be appropriate for their target statistics. Researchers cannot assume that using bootstrapping can automatically make any statistics robust. There are situations in which, if robustness is a concern, the solution is to combine a robust estimator with bootstrapping, rather than to use bootstrapping on a non-robust estimator (Rousselet et al., 2021).

Second, as argued by Zou (2007), bootstrap confidence intervals should not be unconditionally used in place of analytic solutions. If the assumptions for an analytic solution are tenable, it can have better performance than nonparametric bootstrapping, even in small samples. Efron (1988) remarked that "bootstrap methods are intended to supplement rather than replace parametric analysis, particularly when parametric methods cannot be used because of model uncertainties or theoretical intractability" (p. 296). Therefore, using bootstrap confidence statistics for a statistic should be an informed choice based on previous findings, rather than a default choice mechanically adopted. For example, even though an analytic solution for the confidence of Cohen's *d* is available (Cumming & Finch, 2001), a researcher may decide to use bootstrapping because substantial nonnormality in the population is suspected, and previous studies found that a particular variant of bootstrap confidence interval performed satisfactorily in situations similar to theirs.

Third, it must be stressed that the performance of bootstrapping is affected by sampling error, similar to parametric methods. If the sample size is too small, its performance can be adversely affected (Rousselet et al., 2021). Therefore, when using bootstrapping for a statistic, previous studies should be consulted to judge the probable performance of this approach in the situation at hand.

Last, Rousselet et al. (2021) recommended researchers to report "the full bootstrap distribution (and the code) as it contains much more information than the confidence interval" (p. 9), to which we agree. This is why we emphasized storing the bootstrap estimates and examining their distributions using `EXAMINE`, histograms, and normal Q-Q plots in our examples. The SPSS syntax files, and the bootstrap samples, if feasible, should be shared to make the results reproducible. Even if the graphs cannot be included in the main text due to space limits, they should be made available as supplementary materials such that readers can examine them.

Limitations and future development

The current version of the extension command can only do nonparametric bootstrapping. It has not yet implemented parametric bootstrapping. It does not yet do a more complicated sampling scheme as the SPSS built-in `BOOTSTRAP` command does. However, our intention is not to develop a versatile tool. Instead, we illustrated how existing SPSS commands can already be used to do basic bootstrapping for many statistics, including those not readily available but can be computed from reported results (e.g., Hedges's *g* in SPSS 26 or earlier versions). We hope our illustration can help researchers prepare syntax commands for other statistics and scenarios they encounter. We highlight some possible future directions below.

Inspection of the Python code of `GENERATE NONPAR BOOTSAMPLES` will show that most of the steps are actually implemented in SPSS syntax commands. It is the random resampling step that is conducted in Python native functions. This step can also be implemented in native SPSS commands (see Nichols, 1996, for an example). We used Python to show the possibility to use functions in Python packages to implement other sampling schemes. Interested readers can examine the source code, adapt it for other kinds of resampling schemes, and share with other researchers.

Though not the main focus of this manuscript, our examples above also illustrate the potential to write macro commands that make use of `OMS` redirection to compute statistics based on reported results (e.g., Hedges's *g*). Researchers may develop and share macros for commonly reported statistics, such as omega-square in ANOVA (Hays, 1988) and composite reliability (Raykov, 1997), using an approach similar to that in our examples. Other researchers not familiar with `OMS` commands can then conduct Step 2 using these macros, making DIY bootstrapping even simpler than the ad hoc macros approach we illustrated above.

Conclusion

Some researchers may find Step 2, doing the data analysis on each bootstrap sample, complicated. If researchers usually use the graphical user interface and the dialog boxes to do analysis, it may take some time to learn writing the syntax commands, especially the `OMS` commands for the main analysis, computing statistics not available in existing procedures, and the ad hoc macro commands. They may even think that it is easier to do this task in other computing environments, such as R. We are not advocating the use of any particular platforms or packages. Even though we ourselves use R more in our work, we hope more methods are available in more platforms, such that researchers can spend their efforts on doing analysis on their preferred environments, instead of learning a new environment just to use a particular method.

Acknowledgments This study was supported by the Multi-Year Research Grant (MYRG2020-00242-FSS) from University of Macau, Macao SAR, China.

References

Algina, J. (1999). A comparison of methods for constructing confidence intervals for the squared multiple correlation coefficient. *Multivariate Behavioral Research*, *34*(4), 493–504. https://doi.org/10.1207/S15327906MBR3404_5

Algina, J., Keselman, H., & Penfield, R. D. (2006). Confidence interval coverage for Cohen's effect size statistic. *Educational and Psychological Measurement*, *66*(6), 945. https://doi.org/10.1177/0013164406288161

Algina, J., Keselman, H. J., & Penfield, R. D. (2010). Confidence intervals for squared semipartial correlation coefficients: The effect of nonnormality. *Educational and Psychological Measurement*, *70*(6), 926–940. https://doi.org/10.1177/0013164410379335

Appelbaum, M., Cooper, H., Kline, R. B., Mayo-Wilson, E., Nezu, A. M., & Rao, S. M. (2018). Journal article reporting standards for quantitative research in psychology: The APA Publications and Communications Board task force report. *American Psychologist, 73*(1), 3–25. https://doi.org/10.1037/amp0000191

Bishara, A. J., & Hittner, J. B. (2012). Testing the significance of a correlation with nonnormal data: Comparison of Pearson, Spearman, transformation, and resampling approaches. *Psychological Methods, 17*(3), 399–417. https://doi.org/10.1037/a0028087

Bishara, A. J., & Hittner, J. B. (2017). Confidence intervals for correlations when data are not normal. *Behavior Research Methods, 49*(1), 294–309. https://doi.org/10.3758/s13428-016-0702-8

Canty, A., & Ripley B. D. (2021). *boot: bootstrap R (S-Plus) functions*. R package version 1.3–28

Chan, W. (2009). Bootstrap standard error and confidence intervals for the difference between two squared multiple correlation coefficients. *Educational and Psychological Measurement*, *69*(4), 566–584. https://doi.org/10.1177/0013164408324466

Cheung, M. W.-L. (2009). Comparison of methods for constructing confidence intervals of standardized indirect effects. *Behavior Research Methods*, *41*(2), 425–438. https://doi.org/10.3758/BRM.41.2.425

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd Ed.). .

Craig, C. C. (1936). On the frequency function of *xy*. *The Annals of Mathematical Statistics, 7*(1), 1–15. https://doi.org/10.1214/aoms/1177732541

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement*, *61*(4), 532–574. https://doi.org/10.1177/0013164401614002

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*(1), 1–26. https://www.jstor.org/stable/2958830

Efron, B. (1987). Better bootstrap confidence intervals. *Journal of the American Statistical Association*, *83*(397), 171–185. https://doi.org/10.2307/2289152

Efron, B. (1988). Bootstrap confidence intervals: Good or bad? *Psychological Bulletin*, *104*(2), 293–296. https://doi.org/10.1037/0033-2909.104.2.293

Efron, B., & Hastie, T. (2016). *Computer age statistical inference: Algorithms, evidence, and data science*. Cambridge University Press. https://doi.org/10.1017/CBO9781316576533

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap*. Chapman & Hall/CRC.

Fox, J. (2016). *Applied regression analysis and generalized linear models* (3rd ed.). Sage.

Gayen, A. K. (1951). The frequency distribution of the product-moment correlation coefficient in random samples of any size drawn from non-normal universes. *Biometrika, 38*(1–2), 219–247. https://doi.org/10.1093/biomet/38.1-2.219

Greenland, S., Senn, S. J., Rothman, K. J., Carlin, J. B., Poole, C., Goodman, S. N., & Altman, D. G. (2016). Statistical tests, *P* values, confidence intervals, and power: A guide to misinterpretations. *European Journal of Epidemiology, 31*(4), 337–350. https://doi.org/10.1007/s10654-016-0149-3

Harding, B., & Cousineau, D. (2016). GSD: An SPSS extension command for sub-sampling and bootstrapping datasets. *The Quantitative Methods for Psychology*, *12*(2), 138–146. https://doi.org/10.20982/tqmp.12.2.p138

Hawkins, D. L. (1989). Using U statistics to derive the asymptotic distribution of Fisher's *z* statistic. *The American Statistician, 43*(4), 235. https://doi.org/10.2307/2685369

Hayes, A. F. (2018). *Introduction to mediation, moderation, and conditional process analysis: A regression-based approach* (2nd Ed.). The Guilford Press.

Hays, W. L. (1988). *Statistics* (4th ed.). Thomson Learning.

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics*, *6*(2), 107–128. https://doi.org/10.3102/10769986006002107

Hotelling, H. (1953). New light on the correlation coefficient and its transforms. *Journal of the Royal Statistical Society. Series B (Methodological), 15*(2), 193–232. http://www.jstor.org/stable/2983768

Jones, J. A., & Waller, N. G. (2013). Computing confidence intervals for standardized regression coefficients. *Psychological Methods*, *18*(4), 435–453. https://doi.org/10.1037/a0033269

MacKinnon, D. P., Lockwood, C. M., & Williams, J. (2004). Confidence limits for the indirect effect: Distribution of the product and resampling methods. *Multivariate Behavioral Research, 39*(1), 99–128. https://doi.org/10.1207/s15327906mbr3901_4

Mair, P., & Wilcox, R. (2020). Robust statistical methods in R using the WRS2 package. *Behavior Research Methods, 52*(2), 464–488. https://doi.org/10.3758/s13428-019-01246-w

Nichols, D. (1996). Here's some syntax to create bootstrapped samples, written by my colleague David Marso. Newsgroup: sci.stat.consult. Retrieved from https://groups.google.com/g/sci.stat.consult/c/AwLGLd1GpRQ/m/SvLdg6ukDnEJ. Accessed 21 June 2021.

Ohtani, K. (2000). Bootstrapping R^2 and adjusted R^2 in regression analysis. *Economic Modelling*, *17*(4), 473–483. https://doi.org/10.1016/S0264-9993(99)00034-6

Padilla, M. A., & Divers, J. (2015). A comparison of composite reliability estimators: Coefficient omega confidence intervals in the current literature. *Educational and Psychological Measurement*. https://doi.org/10.1177/0013164415593776

Padilla, M. A., Divers, J., & Newton, M. (2012). Coefficient alpha bootstrap confidence interval under nonnormality. *Applied Psychological Measurement*, *36*(5), 331–348. https://doi.org/10.1177/0146621612445470

Pek, J., & Flora, D. B. (2018). Reporting effect sizes in original psychological research: A discussion and tutorial. *Psychological Methods, 23*(2), 208–225. https://doi.org/10.1037/met0000126

R Core Team. (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing. https://www.R-project.org/

Raykov, T. (1997). Estimation of composite reliability for congeneric measures. *Applied Psychological Measurement*, *21*(2), 173–184. https://doi.org/10.1177/01466216970212006

Raynald, L. (n.d.) *Bootstrap and random numbers*. Raynald's SPSS tools. Retrieved from https://www.spsstools.net/en/syntax/syntax-index/bootstrap-and-random-numbers/. Accessed 23 Mar 2021.

Rousselet, G. A., Pernet, C. R., & Wilcox, R. R. (2021). The percentile bootstrap: A primer with step-by-step instructions in R. *Advances in Methods and Practices in Psychological Science*, *4*(1), 2515245920911881. https://doi.org/10.1177/2515245920911881

Yuan, K.-H., & Chan, W. (2011). Biases and standard errors of standardized regression coefficients. *Psychometrika*, *76*(4), 670–690. https://doi.org/10.1007/s11336-011-9224-6

Zou, G. Y. (2007). Toward using confidence intervals to compare correlations. *Psychological Methods*, *12*(4), 399–413. https://doi.org/10.1037/1082-989X.12.4.399

Open practice statement All the code and sample datasets are available in the Open Science Framework project page for this manuscript: https://osf.io/2twf5

Author Note This study was supported by the Multi-Year Research Grant (MYRG2020-00242-FSS) from University of Macau, Macao SAR, China. Correspondence concerning this article should be addressed to Shu Fai Cheung, Department of Psychology, Faculty of Social Sciences, University of Macau, Avenida da Universidade, Taipa, Macao SAR, China or by email (sfcheung@um.edu.mo).