2014 ■ vol. 10 ■ no. 2

DOI: 10.20982/tqmp.10.2.p080

GRD: An SPSS extension command for generating random data

Bradley Harding [a], Denis Cousineau [a]

[a] School of Psychology, University of Ottawa

Abstract ■ To master statistics and data analysis tools, it is necessary to understand a number of concepts, many of which are quite abstract. For example, sampling from a theoretical distribution can help individuals explore and understand randomness. Sampling can also be used to build exercises aimed to help students master statistics. Here, we present GRD (Generator of Random Data), an extension command for SPSS (version 17 and above). With GRD, it is possible to get random data from a given distribution. In its simplest use, GRD will return a set of simulated data from a normal distribution. With subcommands to GRD, it is possible to get data from multiple groups, over multiple repeated measures, and with desired effect sizes. Group sizes can be equal or unequal. With further subcommands, it is possible to sample from any theoretical population, (not simply the normal distribution), introduce non-homogeneous variances, fix or randomize subject effects, etc. Finally, GRD's generated data are in a format ready to be analyzed.

Keywords ■ Statistics Teaching; Data Simulations; SPSS; Sampling;

BHARD024@uottawa.ca

Introduction

Statistics are often perceived as complex and can be a source of anxiety for many students. However, they are omnipresent in today's classroom and students must develop statistical literacy (Rumsey, 2002) in order to understand concepts such as calculating and comparing simple averages. Despite the fact that statistics are ubiquitous, it can be a difficult field to learn and master.

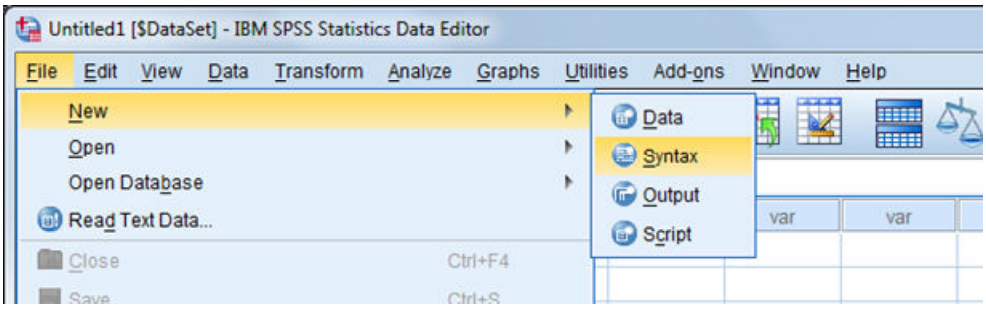
One reason for this difficulty is that the notion of randomness is not quite intuitive. Often, students will confuse events for which there are only two outcomes (e. g. smokers vs. non-smokers) with equiprobable events. Likewise, students sometimes come up with a notion of randomness akin to the uniform distribution, which clashes with the concept of the normal distribution. Additionally, the concept of sampling is frequently misunderstood. For example, students will often say that the mean of a sample of IQ's ought to be 100 because the population mean is 100. Without understanding concepts such as these, it is hard to understand theoretical distributions and statistical inferences.

To develop a notion of randomness, the best approach (one that fosters active learning) is physical exploration. As recommended in the GAISE College report (American Statistical Association et al., 2005, recommendation 4), die rolling, card shuffling and the Galton machine (marbles descending a plane on which nails are disposed in rows) are all excellent tools to grasp the ideas and theories behind uniform and binomial distributions. However, rapidly, these apparatuses become slow to operate and are only able to generate samples that are rather small in size (up to a few hundred). To reach larger sample sizes, these physical explorations should be supplemented with computer simulations. Computer simulations have the benefit of being rapid, with extensive sample sizes are attainable within a few seconds.

The second recommendation of the GAISE College report (American Statistical Association et al., 2005) stipulates that statistical tests performed in class should preferably use real datasets. These datasets are more likely to elicit conceptual and critical thinking, involve the students and lead to further interrogation and exploration on the part of the student. However, there comes a point where students must show proficiency by rapidly selecting and applying the correct statistical test for a given situation. At this stage, the existence of the elaborate context surrounding each example data may waste time, and the minute details of the data (outliers, skew, etc.) might become distracting to a number of students.

Finally, the existence of interaction effects found in two-way analysis of variance (ANOVA) might be an abstract concept to grasp at first (with either one or two main effects, a super-additive or under-additive interaction, or even a lack of effect only when both

Figure 1 ■ Opening a Syntax window



factors are present, creating a cross-pattern interaction). It may be useful to present a single research context (e. g., the impact of a drug and exercises on IQ, in a balanced 2 × 2 design) and simply adjust the dataset. As recommended by the GAISE College report, simulations can be useful when abstract concepts such as interactions need to be illustrated.

On a more practical note, simulated datasets are also convenient because it is possible to pack the desired effects in a single dataset (whether correlations are wanted or not, main effects or not, covariables or not, etc.). These desired effects might be difficult to achieve with a single real dataset. In addition, once the real dataset is used, students may have more difficulty practicing statistics as there is no novelty to their data and their analyses may feel redundant. This may lead to students not practicing enough or limiting themselves to their immediate comprehension by not providing as many alternative datasets as is required. Finally, working with actual datasets may be difficult due to the fact that it may take too much time to find some that exhibit the characteristics in need of being practiced.

Here, we will describe an extension command that we have created for *SPSS*: GRD, which stands for *Generator of Random Data*. The purpose of this extension is to create fictitious datasets following specifications given by the user. The generator can simulate, for example, a two-group design ready to be analyzed with a t-test within *SPSS*. It can also simulate data for multi-group designs (between-subject designs), repeated measures (within-subject designs) as well as mixed designs involving within-subject and between-subject factors. GRD is ideal to create a dataset to give your students an extra assignment and/or to test a research model. It is also useful from the student point of view, as GRD can be used to generate data, resulting in a clearer conceptualization of sampling and sampling error.

The remainder of this guide will help you install and use GRD. Firstly there will be two sections containing the basics: getting acquainted with the syntax mode in *SPSS* and explaining GRD with its subcommands. Follows a small section with various details related to GRD. We then conclude with possible use of GRD.

Working with Syntax

GRD is an extension command for *SPSS* that works only in the Syntax mode of *SPSS*. To access a syntax window, go to the menu "*File/New/Syntax*". The screen capture of Figure 1 demonstrates the menus and depicts how to access a syntax window. It is to note that the *SPSS* version used for the screenshots is version 21 for PC.

The window that will appear, shown in Figure 2, allows the user to write commands via text instead of going through one or several menus, as is typically done. One advantage of using text-based commands is the fact that there is a clearer view of the analyses that have been completed. It is also easier to locate mistakes made (e. g. generating datasets with an incorrect effect). Finally, it is possible to save all progress made. Saving a Syntax window is ideal for reproducing the analyses later or sharing commands with other *SPSS* users.

Assuming data from two groups of participants are present in the data editor, it is possible to instruct *SPSS* to perform a t-test using the command seen in Figure 3. When the command is finished, it is possible to execute the command by putting the cursor anywhere within the window and using the menu "*Run: Selection*" (or the keyboard shortcut Ctrl-R), or by clicking the green triangle resembling a "*Play*" button located on the syntax window's toolbar.

Figure 2 ■ An empty Syntax window

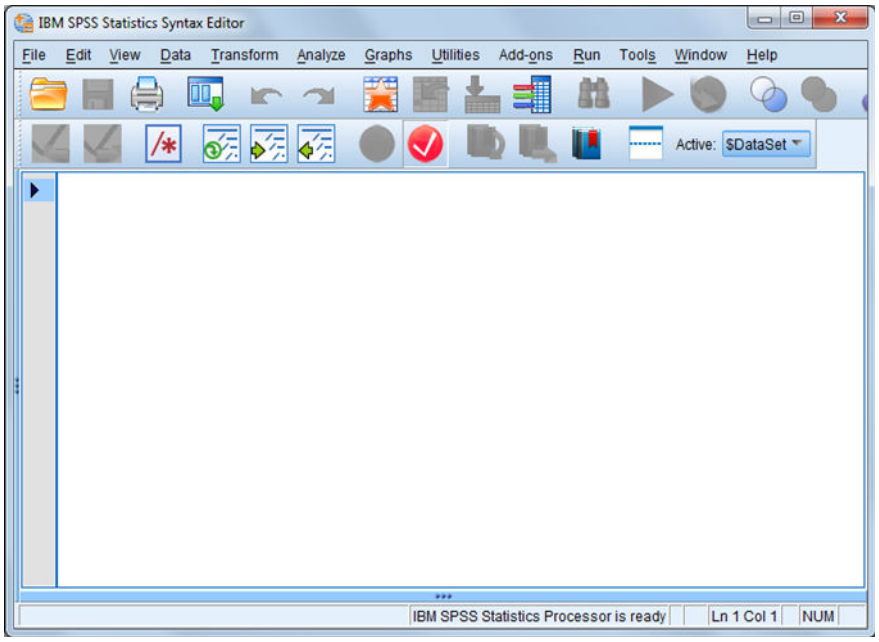
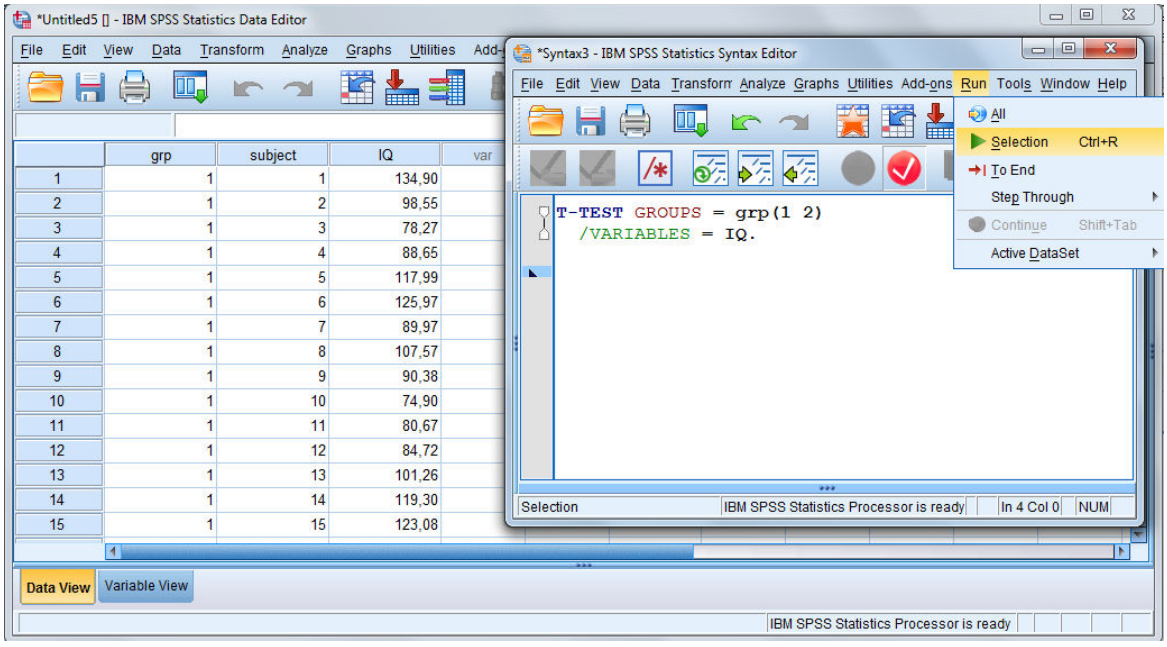


Figure 3 ■ One example of a Syntax that performs a t-test



When working with syntax, *SPSS* uses highlighted colors. When a command is recognized by *SPSS*, the name of the command automatically turns to red (here, the command is "*T-TEST*"). When the command is completed, the highlight color will automatically change to blue. If the command remains written in bold and red, it signifies that *SPSS* does not recognize the command or that there is a mistake in the command itself. To remedy this issue, review the script to ensure it is complete and properly written. Be sure to verify that the command ends with a period (".") and that all subcommands appear in green (e.g. "*VARIABLE*"). If this is not the case, ensure that the forward slash ("/") precedes the subcommand and that the subcommand is

Figure 4 ■ The dialog window that is used to perform an independent-samples t-test. By clicking the "Paste" button instead of the "OK" button, the t-test command will be pasted in the first available Syntax window.

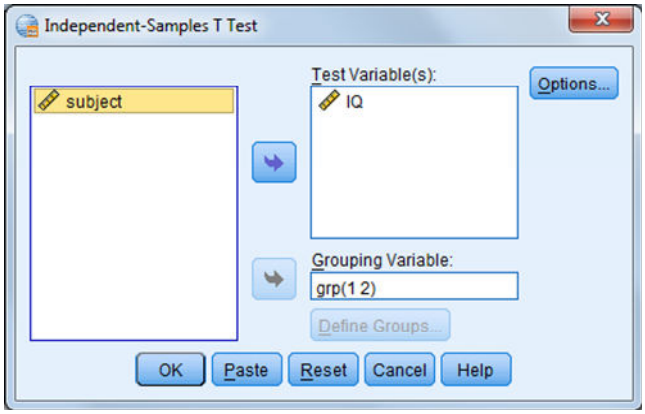
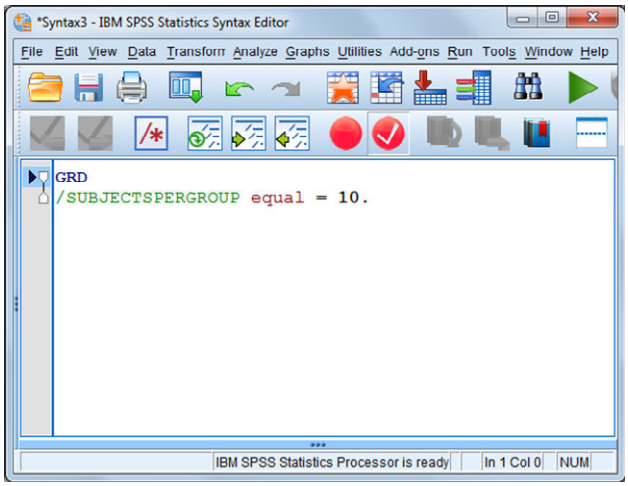


Figure 5 ■ The first GRD command. Note the blue highlight for the command name "*GRD*", the green highlight for the subcommand "*/SUBJECTPERGROUP*" and the red highlight for the option "*equal*".



spelled properly. Finally, all options should appear in light red (e.g. "*GROUPS*"). If they are not light red, review the script and confirm that the option is spelled properly. Because *SPSS* Syntax is not case sensitive, the cases used in our examples are but suggestions. We chose to use capitals for commands and subcommands, and lower case for options and variable names.

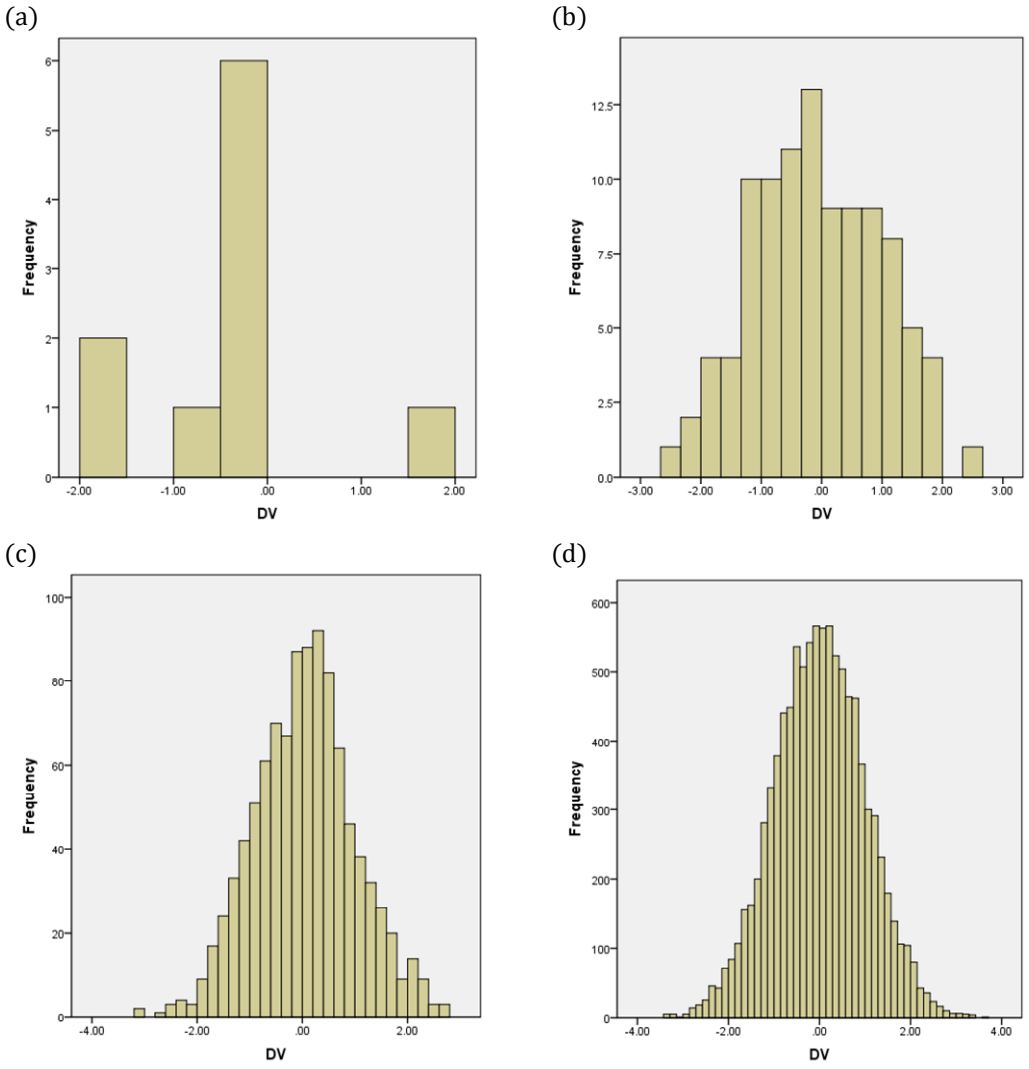
It is possible to learn how to write a command using Syntax by utilizing the "*Paste*" button available on all dialog windows of *SPSS*. The user must go to the window of the analysis that he/she wants to complete, fill the fields as one normally would and press "*Paste*" instead of "*OK*". This will copy the command into a Syntax window so that it is possible to view its subcommands and options. Figure 4 shows how the syntax from Figure 3 was obtained. Every time SPSS is used with the dialog windows, the Syntax commands are written in background. The command is then run when the "OK" button is pressed.

If you are not well versed in coding with *SPSS*, this guide will go through the general concepts and ideas. You can learn more about coding from a number of websites and books (e.g. UCLA: Academic Technology Services, Statistical Consulting Group, 2012; Einspruch, 2004; Cousineau, 2009).

Basic specifications to GRD

At this point, GRD should be installed. Please see Appendix A for instructions on the installation of GRD. GRD must be installed only once per computer. All the

Figure 6 ■ Increasing group size and its effect on the distribution's shape with the following options for /SUBJECTPERGROUP: a) 10, b) 100, c) 1000, d) 10 000



possible options to GRD described here are summarized in Appendix B.

Creating Data

The minimum specification to run GRD is the subcommand that indicates the number of subjects per condition. By executing the command that follows (see below, also shown in Figure 5 inside the Syntax editor), GRD will generate a single group of 10 subjects, each having a single score in a column named "*dv*" which stands for "*dependant variable*". The keyword "*equal*" is used to indicate that all the groups have the same number of subjects (in this example, there is only one group created but as the command becomes more complicated, this subcommand will become more elaborate). By default, the data is generated with a mean of 0, a standard deviation of 1, and it follows a normal distribution. To execute the command, press Ctrl+R or click the large green triangle resembling a "*play*" button[1].

```
GRD
/SUBJECTSPERGROUP equal = 10.
```

By executing the GRAPH command in the very same Syntax window as follows, it is possible to view the distribution of the data created above

[1] When there are multiple commands, you can run all of them at once by first selecting them all before executing.

Figure 7 ■ The effect of a linear range of 20 on between subject factors generated by GRD with a mean of 100 and a standard deviation of 15. Because there are only two groups, a range of 20 implies a raw effect size of 20 points of IQ (and a Cohen's *d* of 1.33).

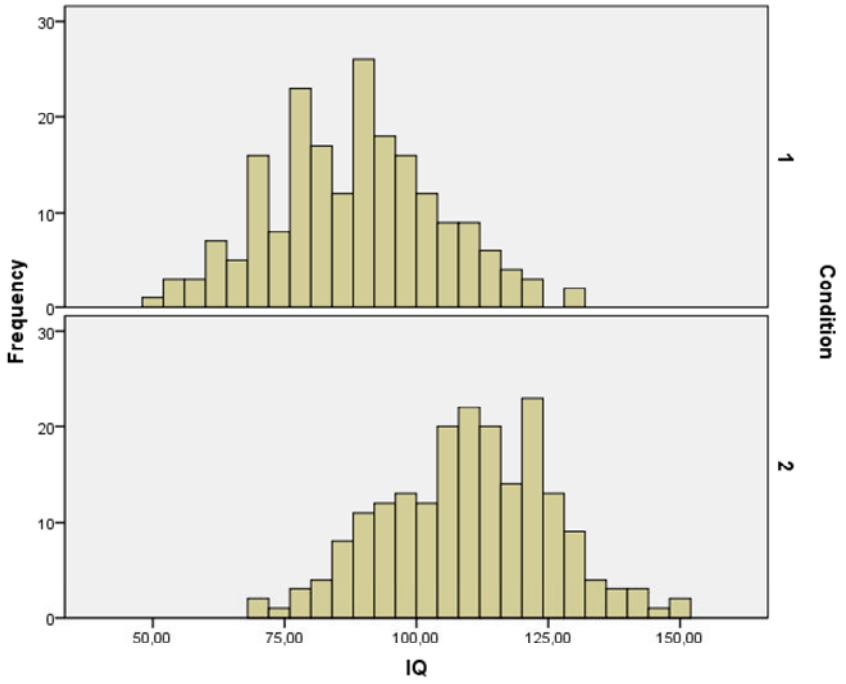
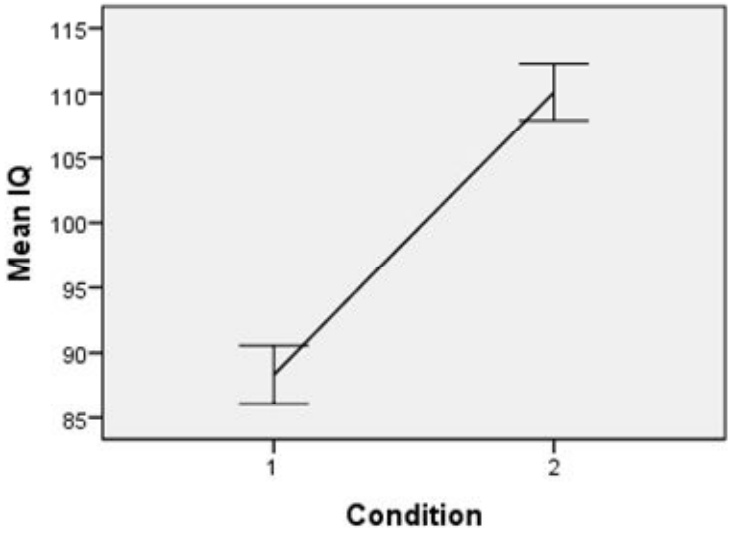


Figure 8 ■ Means as a function of gender along with error bars illustrating the 95% confidence intervals. As is shown, the means seem different, despite a large amount of overlap in the distributions, seen in Figure 7.




```
GRAPH
/HISTOGRAM dv.
```

When increasing the group's size, GRD creates a larger sample size. Because the underlying population distribution is normal, the sample's histogram should look more and more symmetrical as the sample size increases. With larger sample sizes, the normality of the data becomes evident. Figure 6 shows four plots with sample sizes of 10, 100, 1000 and 10 000 respectively.

To change the overall mean and standard deviation of the distribution, the /OVERALL subcommand may be

added to GRD. By executing the following command, the underlying distribution will have a mean of 100 and a standard deviation of 15. These settings can be used for generating scores simulating IQ scores. These options only change the distribution's location and scale. The distribution shape is not affected and therefore remains a normal distribution.

```
GRD
/SUBJECTSPERGROUP equal = 10
/OVERALL mean = 100 stddev = 15.
```

As mentioned previously, the dependent variable created by GRD is named "*dv*" by default. The user has the option to change the name of the dependent variable. To do so, write the subcommand "*/RENAME dv =*" followed by the variable name chosen. When the variable is renamed, any analysis done on the variable must use the new name. In other words, if the variable's new name is "*IQ*" then all analyses on the "*dv*" must utilize IQ as the focal point of the analyses. The following, for example, will create 200 scores mimicking IQ:

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/OVERALL mean = 100 stddev = 15.
```

Generating groups of data

GRD can also create multiple groups and repeated measures. To simulate data in a repeated measure design, use the subcommand /WSFACTORS. This manipulation creates new columns of data for each simulated subject. These columns contain the repeated measures. By writing the following command, GRD will create 3 columns for each of the subjects to simulate measurements at three different ages.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/OVERALL mean = 100 stddev = 15
/WSFACTORS Age (3).
```

It is possible to have a between-subject factor that divides the subjects into two or more groups. By writing the /BSFACTORS sub-command, the subjects are divided into groups of equal size. Groups will have the same number of subjects as specified in the /SUBJECTSPERGROUP subcommand. This GRD command will create two groups of 200 cases for a total of 400 cases.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/OVERALL mean = 100 stddev = 15
/BSFACTORS Gender (2).
```

It is possible to have more than one factor. By adding a second factor to either within-subject or between-subject factors, GRD will generate data accordingly. It is also feasible to combine /WSFACTORS and /BSFACTORS in the same GRD command for more complex designs. However, it is not possible to have more than four factors in total, combining both /BSFACTORS and /WSFACTORS subcommands.

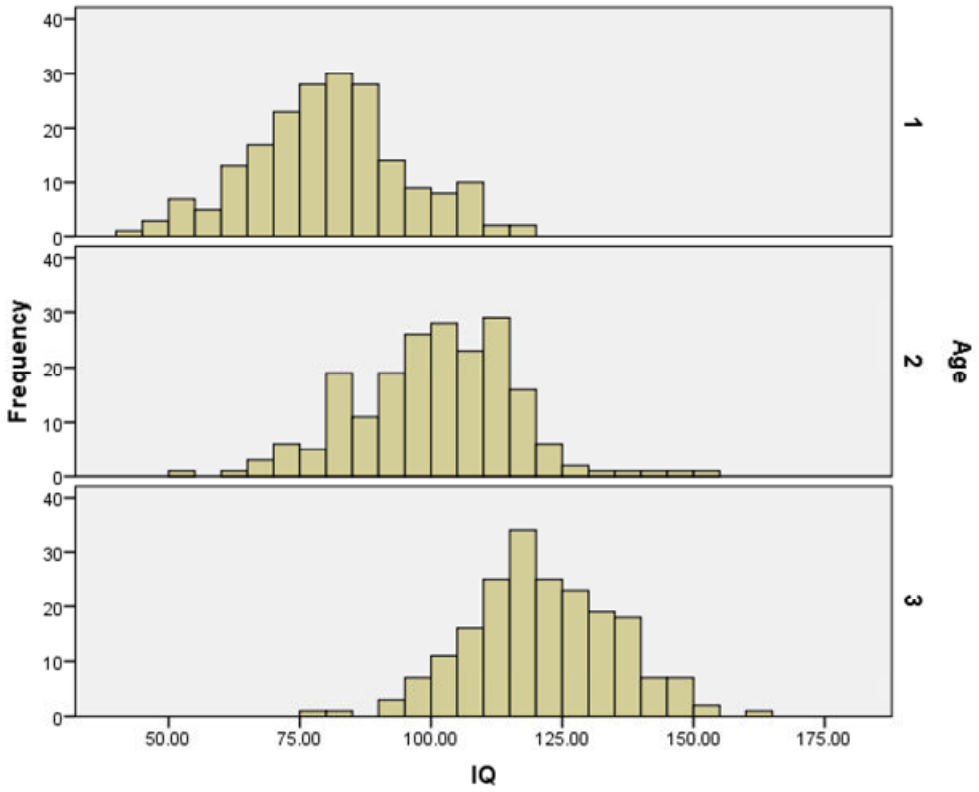
The following lines generate data in a 2 × 2 × (3) design with four groups (2 × 2) and three repeated measures (3 columns). In the following command, there is an ellipsis ("…") before and after the command. This signifies that it is the same as the command specified above. The only thing to add is the "*/*BSFACTORS*…*" part of the code.

```
…
/BSFACTORS Gender (2) Room (2)
/WSFACTORS Age (3)
…
```

Varying the effects

When it is not specified that there are differences between groups, GRD creates random data using the mean and the standard deviation specified with the /OVERALL subcommand. This means that if there are multiple factors, these factors are identical to each other in terms of data distribution. By writing the /EFFECTS command, it is possible to vary the distribution for the /BSFACTORS and /WSFACTORS. There are 3 different ways to specify an effect on the datasets. The first is a *linear range,* which changes the mean between the levels of the factors named. The overall mean remains the same; however, the range covered by the /EFFECTS subcommand corresponds to the value of the dispersion between the group's means above and below the overall mean. This change in range is also known as the raw effect size (Cohen, 1992). For example, suppose that we examine IQ for gifted children as well as children who suffered from famine in their early childhood. Although the overall mean is 100, we can expect a difference of 10 points below and above that overall mean. This result in a range of 20:

Figure 9 ■ The effect of a linear slope of 20 on between subject factors generated by GRD with a mean of 100 and a standard deviation of 15. With a slope of 20, the raw effect size between two successive groups is 20 points of IQ (and Cohen's $f = 1.33$).



one group has a theoretical mean of 90 and the other, of 110. The GRAPH command will plot the distribution of the two groups, as seen in Figure 7.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/BSFACTORS Condition (2)
/OVERALL mean = 100 stddev = 15
/EFFECTS Condition (linear range = 20).

GRAPH
/HISTOGRAM=IQ
/PANEL ROWVAR = Condition.
```

As seen in Figure 7, there is a lot of overlap between the groups' scores, but the groups have different means. To illustrate the means, you can plot

```
GRAPH
/LINE = Mean (IQ) by Condition
/INTERVAL CI (95.0).
```

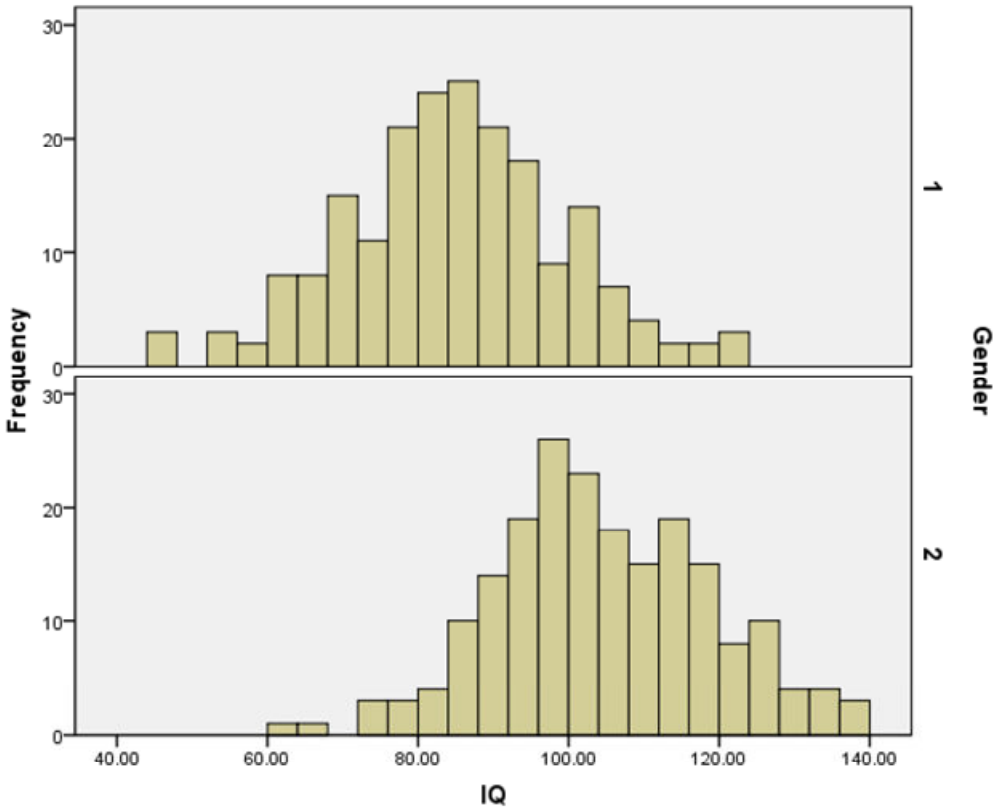
This command only works for one or two between-subject factors. It also shows the 95% interval of confidence error bars. Figure 8 shows the resulting mean plot.

To formally assess the importance of the difference between groups, here is a T-TEST command that conducts a t-test that should result in showing that there is in fact a significant difference:

```
T-TEST GROUPS = Condition (1 2)
/VARIABLES = IQ.
```

Here, the generated dataset had a significant difference (t (398) = -13.7, $p < .001$). Based on this simulated data, the difference between the two groups of children is outstanding. Of course, the data being different every time GRD is run, the results will not always be identical and sometimes may not show significance when replicated. These generated datasets are cases of type II errors. Whereas type 1 errors should occur on average once every twenty runs if the criterion chosen is $\alpha = 0.05$, type-II errors depend on the effect size. Here, because the effect size is very important ($d = 1.33$), type-II error should be quite exceptional.

Figure 10 ■ The effect of a custom manipulation with the first group having a mean 15 points less than the overall mean and the second group having a mean 4 points above the overall mean of 100 on between subject factors generated by GRD.



The second way to affect the generated data is by specifying a slope effect. The linear slope is similar to the linear range; however, for every group, the mean increases by the amount specified in the *slope* option. Hence, if there are three groups with different IQ tests and the distribution has a slope of 20, the generated dataset will have an effect range of 40.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/BSFACTORS Age (3)
/OVERALL mean = 100 stddev = 15
/EFFECTS Age (linear Slope = 20).

GRAPH
/HISTOGRAM=IQ
/PANEL ROWVAR = Age.
```

Figure 9 shows how the linear slope affects the distribution of the data created by running the GRD code above.

Finally, the third way to specify an effect is to use the *custom* specification. This method generates a custom mean for each factor group. It is useful when there are more than two groups. These means are all relative to the overall mean specified in the /OVERALL subcommand. With the following command, the first group should have a mean of 85 (100 - 15) and the second, a mean of 104 (100 + 4).

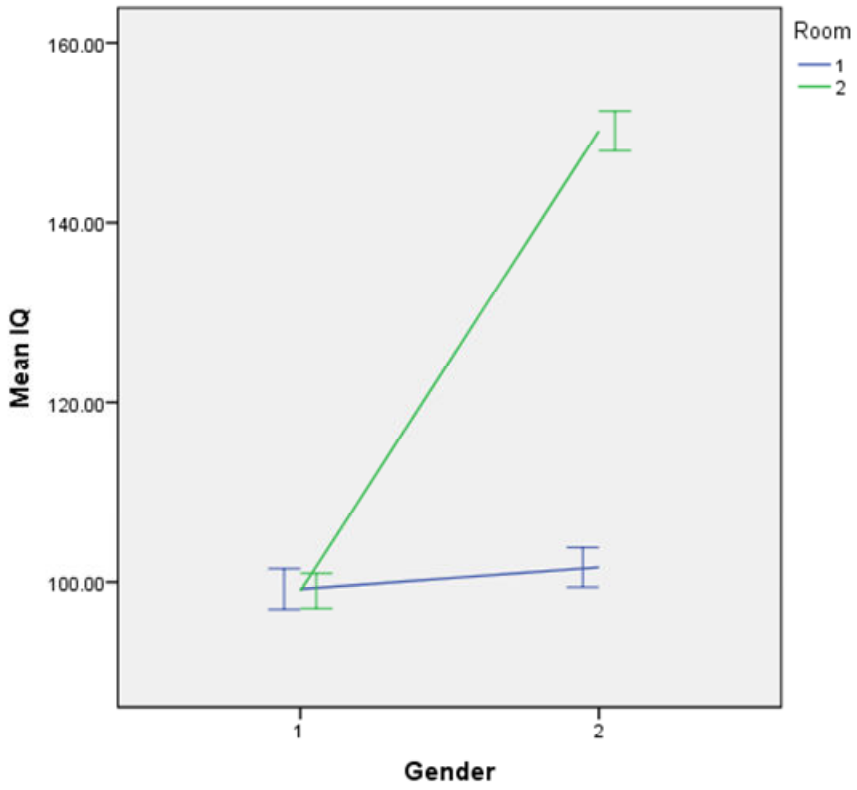
```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/BSFACTORS Gender (2)
/OVERALL mean = 100 stddev = 15
/EFFECTS Gender (custom -15 4).

GRAPH
/HISTOGRAM=IQ
/PANEL ROWVAR = Gender.
```

Figure 10 shows the resulting two distributions generated from GRD by running the GRAPH command.

Finally, if there is more than one factor, there is the possibility to specify multiple entries in the /EFFECTS

Figure 11 ■ Example of mean results in a 2 × 2 design involving two genders (horizontal axis) and two different rooms (color). The error bars represent the 95% confidence intervals of each interaction.



subcommand. For example, the following will have an effect for gender of 10, (which signifies means of 90 and 110 for both groups respectively) as well as a custom effect of -15 depending on which room the test took place.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/BSFACTORS Gender (2) Room (2)
/OVERALL mean = 100 stddev = 15
/EFFECTS     Room (custom -15 0) Gender
(linear slope = 10).
```

Joint effects can also be given using the notation: *factor* * *factor*. This is useful for interactions between factors. The "*" crosses the factors and shows the output effect of said interaction. For example, the following will consider four groups ($Gender_1$, $Room_1$; $Gender_1$, $Room_2$; $Gender_2$, $Room_1$; $Gender_2$, $Room_2$) each having a slope effect of 10. Hence, the groups should have means of 85; 95; 105; 115.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/BSFACTORS Gender (2) Room (2)
/OVERALL mean = 100 stddev = 15
/EFFECTS Gender*Room (linear slope = 10).
```

In addition, there is the possibility to generate interactions defined by using the *custom* option on joint effects. The following text illustrates these interactions. In this situation, imagine that the experiment took place in a room that enhanced a particular gender's IQ with non-conscious primes.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP equal = 200
/BSFACTORS Gender (2) Room (2)
/OVERALL mean = 100 stddev = 15
/EFFECTS Gender*Room (custom 0 0 0 50).
```

As seen in Figure 11, the interaction is important: one group is diverging from all the other groups. This

signifies that there is an interaction for this group. Interactions are a frequent aspect of statistics and their available use in GRD is a strongpoint of the command. The mean plot was obtained using:

```
GRAPH
/LINE = MEAN (IQ) BY Gender BY room
/INTERVAL CI (95.0).
```

The interaction is confirmed using an ANOVA (F (1, 792) = 580.1, $p < .0001$) with the command

```
GLM IQ BY gender room.
```

Unequal sample sizes

As mentioned above, the number of subjects per group is equal for each group when the option *equal* is used. It is possible to change this by using the *unequal* option of the /SUBJECTSPERGROUP subcommand. With this option, you must list the same number of group sizes as there are between-subject groups. Here is an example with 3 different groups of different sizes.

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP unequal = 32 33 35
/BSFACTORS Area_of_study (3).
```

Changing the population's distribution

As mentioned above, the default population distribution is the normal distribution. Each group has a mean given by the /OVERALL mean plus the treatment effects and with a standard deviation given by the /OVERALL "*stddev*" subcommand. This default specification can be given explicitly with the subcommand: "*/SCORES population =* " and using double-quote signs (" "):

```
…
/SCORES population = "RV.NORMAL(#effect,
#stddev)"
…
```

In the /SCORES subcommand, the code "*RV.NORMAL*" indicates the population's distribution, the code "*#effect*" indicates the effect for that condition (considering both the /OVERALL mean and the /EFFECTS specifications) and the code "#*stddev*" is the standard deviation specified in the /OVERALL subcommand.

The above codes can be changed. For example, the code "*#stddev*" can be replaced by a value of 5, as demonstrated below:

```
GRD
/RENAME dv = IQ
/SUBJECTSPERGROUP unequal = 200
/BSFACTORS Gender (2) Room (2)
/WSFACTORS Age_group (3)
/OVERALL mean = 100 stddev = 15
/SCORES population = "RV.NORMAL(#effect, 5)"
/EFFECTS Gender (linear range = 4).
```

This example is trivial but shows that the population scores can be given by any formula as long as it is a valid *SPSS* instruction. To have a standard deviation varying with the group number, there is the possibility to use the code "*#A*" to refer to the first factor's level. Hence, the following subcommand:

```
…
/OVERALL stddev = 15
/SCORES Population = "RV.Normal(#Effect,
#A*#stddev)"
…
```

This subcommand would return groups with standard deviations of 15 (group 1), 30 (group 2), etc., in violation of the homogeneity of variance assumption.

As another example, it is possible to have the standard deviation proportional to the effect:

```
…
/SCORES population = "RV.NORMAL(#effect,
#STDDEV* #Effect/10)"
…
```

This will also invalidate the homogeneity of variance assumption in ANOVA.

It is also possible to have no effect on the mean scores, only an effect on the groups' variance with the following:

```
…
/SCORES population = "RV.NORMAL(#GM,
#Effect/10)"
…
```

In this instruction, #GM is the name of the parameter representing the overall mean (the grand mean given by /OVERALL mean). All the possible codes are listed at the end of the section.

It is possible to change the shape of the distribution by changing the function used for generating random values ("*RV.___*"). The existing functions are listed in the help menu. To access this list, once in the IBM *SPSS* help

menu (by pressing F1 for help), follow "*Command syntax reference/universals/transformations and expressions/random variable and distribution/random variable functions*".

For an example using a distribution other than the normal (e. g., the lognormal distribution), use the following subcommand:

```
…
/SCORES population = "RV.LNORMAL(#effect,
#stddev)"
…
```

This will return data that are asymmetrical, violating the normality assumption mandatory in most parametric analyses.

The possible shortcuts understood within /SCORES are #A, #B, #C, #D, #STDDEV, #GM, #S (for the subject number) and #EFFECT (lower case and upper case don't matter). Basic operations can also be used: + (addition), - (subtraction), *(multiplication), / (division), ** (exponentiation), log (logarithm), sqrt (square root), etc.

For more, see the "compute" section in the "Command syntax reference" in the help menu of *SPSS*. For random value functions, see "Universals\Transformation expressions\Random variable and distribution functions".

Replication of the dataset

As it may be useful to use the same dataset multiple times, replication of the dataset is could show itself to be useful. The seed is a parameter of GRD that represents the randomized signature of the dataset. By overriding the seed and setting it at a specific number, we are able to create the same random dataset over and over again. To manually override the seed every time the GRD command is run, write the SET command and change the MTINDEX to any number. Here the seed is 314. To simply have a random dataset every time the GRD command is run, simply do not write the SET command.

```
SET MTINDEX=314.

GRD
…
```

Miscellaneous aspects of GRD

This section lists the various aspects of GRD to observe whilst coding.

- GRD can only generate one dataset at a time. By running the command again, it erases the random data previously created and generates a new set.
- To manually override the seed, set the MTINDEX to a fixed number as is shown in the Replication of the dataset section of the manuscript.
- Upper and lower case letters do not matter. We chose UPPER CASE letters for the GRD command and its subcommands (e.g. EFFECTS, BSFACTORS, etc.) and lower-case letters for the options (e.g. dv, mean, stddev, etc.) but this is arbitrary and any combinations of lower and upper case letters will work as well (e. g., GrD EffECtS, etc.).
- Each subcommand can be present only once in every GRD command.
- The order of subcommands is not important.
- There can be only one dependent variable; in repeated-measure designs, the dependent variables will be named dv.l, dv.2, etc.
- Factors listed in WSFACTORS and BSFACTORS cannot have the same name.
- There can only be a combined maximum of 4 factors when working with GRD (for example, 1 between-subject factor and 3 within-subject factors; alternatively, 2 between-subject factors and 2 within-subject factors)
- If you want to do subsequent manipulation of the data, you must first finish the GRD command with a period (".").
- If numbers used in /SCORE have decimals, use a period (".") , not a comma (",")
- The larger the sample (number of subjects per group) is, the more evident will the effects of the /EFFECTS subcommand be.

Conclusion

GRD is an extension command for *SPSS*. It creates datasets to simulate results from a variety of experimental designs, including between groups, within groups and mixed designs. Effects (linear or custom) can be defined and the underlying distribution can be changed. The data produced by GRD are ready to be analyzed.

For the teachers who already use *SPSS*, GRD is a good addition because it allows the students to have a single integrated environment where they can both generate and analyze data. If you don't use *SPSS*, it might be preferable to use web applets (e.g., Pruhs, 2013). However, this may not be as convenient due to the fact that most random number generators utilize

uniform distribution whereas most applications in statistics are based on normal distribution. With GRD, any theoretical distribution can be used which is one of its main advantages. If you are using *R*, *Matlab* or *Mathematica*, the use of GRD is not recommended, as these softwares have superior random number generators. Regardless of the software used in the classroom, the instructor should use a single teaching environment when it can fulfill all the teaching requirements. *SPSS* was lacking a convenient random dataset generator, and this is why we created GRD.

We used GRD in recent years in an introductory statistics class in order to cover the notion of randomness, the notion of probability from a frequentist point of view and the notion of sampling error. At the graduate level, we used GRD to explain the relation between the Bernoulli distribution, the binomial distribution, the normal distribution, the χ^2 distribution and the Fisher F distribution. To do so, participants must generate binary events (like the marble rebound in the second example of section 5), then aggregate the results (using sum) to get the binomial distribution. They also normalize those scores, square them and add many of them to get to the χ^2 distribution. Along the way, we can introduce the law of large number, the theory of errors (Gauss, 1809), and asymptotic theories (Cousineau, Goodman & Shiffrin, 2003). In these two possible applications, the students are using GRD individually.

We also personally use GRD to prepare examples, figures, datasets and exercises. In this case, the students are unaware of the origin of the data, although we always specify that the datasets are fictitious. When outliers are needed, we add them manually.

The underlying idea behind GRD is that the laws of distribution are the most important concepts to understand randomness, and randomness is the most crucial concept behind statistical inference. The more students are acquainted with such constructs, the more rapidly and profoundly they will understand an important aspect underlying statistical inference.

Authors' notes and acknowledgments

We would like to thank Laurence Morissette as well as Kaia Myers-Stewart for their help reviewing the text.

References

American Statistical Association with Aliaga, M., Cobb, G., Cuff, C., Garfield, J., Gould, R., Lock, R., Moore, T., Rossman, A., Stephenson, B., Utts, J., Velleman, P. & Witmer, J. (2005), *Guidelines for Assessment and Instruction in Statistics Education [GAISE] College Report,* American Statistical Association, Alexandria, VA.

Cohen, J. (1992), "A power primer," *Psychological Bulletin*, 112, 155-159.

Cousineau, D. (2009), *Panomara des statistiques pour psychologues* (1st Edition). Bruxelles, Belgium: Groupe de Boeck.

Cousineau, D., Goodman, V. & Shiffrin, R. M. (2002), "Extending statistis of extremes to distributions varying on position and scale, and implication for race models," *Journal of Mathematical Psychology*, 46, 431-454.

Einspruch, E. (2004), *Next Steps with SPSS*, Thousand Oaks, CA: Sage Publications, Inc.

Gauss, C.F. (1809/1864), Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum. Paris: A. Bertrand.

Pruhs, K. (2013), Random Numbers. From http://people.cs.pitt.edu/~kirk/cs1501/animations/Random.html (accessed May 20, 2013).

Rumsey, D. J. (2002), "Statistical literacy as a goal for introductory statistics courses," *Journal of Statistics Education*, 10(3).

Statistical Tests in SPSS. UCLA: Academic Technology Services, Statistical Consulting Group. From http://www.ats.ucla.edu/stat/spss/whatstat/whatstat.htm (accessed September 1, 2012).

Appendix A: Installation of *GRD*

GRD works with *SPSS* version 17 or later on any operating system supporting *SPSS*. To install this extension command, *SPSS* must already be installed prior to the use of GRD. In addition, an extra module called *Python Essentials* must also be installed. This module is available free from *SPSS*. The following steps describe how to install the *Python Essential* module and verify that it was installed correctly, and how to install GRD within *SPSS*.

Installing the Python Essential module

To install GRD, the *Python Essentials module* of a version matching with the *SPSS* version must be installed (for

instance, *SPSS* 19 must have the *Python Essentials 19* installed; in addition, if the 32-bit version of *SPSS* is installed, the 32-bit version of the *Python Essentials* module must be installed with it). With this module, extensions can be added to *SPSS* to expand its capabilities.

To install the module, first download it from

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/We70df3195ec8_4f95_9773_42e448fa9029/page/Downloads%20for%20IBM%C2%AE%20SPSS%C2%AE%20Statistics.

If *SPSS* 21 or above is installed, the *Python Essentials* module is directly provided with *SPSS*, however with version 21 *Python Essentials* module, even if it is provided, is not installed. It is necessary to install the module seperately. With *SPSS* 22, the *Python Essentials* module is directly installed with the installation of *SPSS*. Make sure to download the version appropriate for the computer's specifications (Windows 64-bit only if *SPSS* 64-bit is installed as well[2]). Follow the instructions in the Installation documents available on the above web site. Once *Python Essentials* is installed, restart *SPSS* to ensure the bundle's proper use.

Verifying that the Python Essential module is working

Step 1 above is the step most likely to fail as the correct installation of the module can be sensible. To help detect whether or not the *Python Essentials* module was installed properly, we provide here a short syntax. If it runs properly, the module was installed correctly. Within *SPSS*, open a Syntax window (using menu "*File/New/syntax*"), then type the following command:

```
BEGIN PROGRAM python.
print "installation of Python Essentials succeeded!"
END PROGRAM.
```

After the execution of this command, if the message: "Installation of Python Essentials succeeded!" appears in the output window without error messages, it means that the module was properly installed. If not, return to Step 1 above.

Administrative rights

If you have all the administrative rights to your computer, carry on to the next step. If unsure, either contact your systems administrator or go to control panel/user account and find your user name. If it says "Administrator", carry on to Step 4. If not or if unsure, follow these instructions.

As installation of *SPSS* varies from computer to computer, the following section is an outline of what an installation SHOULD look like. The instructions provided are guidelines and may not represent exactly the files and pathways utilized in the reader's configuration.

1 Find the *SPSS Extensions* folder. Usually it is located in "*Computer/Local Disk/Program files/IBM/Statistics/SPSS/19/*". As this may vary from computer to computer, you may search the main hard drive of the computer that *SPSS* is installed for a folder named "*Extensions*".

2 Right click on the folder named "*Extensions*" and click "*properties*"

3 Under the "*Security*" tab, locate "*groups and user names*" and click "*edit…*"

4 Locate your user name (the computer's login username) and click the small check box for "*full control*" in the "*Allow*" column. This allows for that user, if not already the administrator to possess administrative rights to this folder.

By following these steps, you have given yourself permission to modify the *Extensions* folder of *SPSS*. Hence, you can now add new applications and extensions. In the present case, the extension is GRD and once you completed Step 4, a subfolder of that name will appear here.

Installing GRD

■ Download the file "*GRD.spe*" that accompanies this article to a folder of your choice or the desktop

[2] For Windows users, if you are not certain which of the 32-bit or 64-bit version of SPSS was installed, you can read the JRE notice installed with SPSS in the folder *Program Files|IBM|SPSS|Statistics|19|JRE|notices.txt*.

■ In *SPSS*, go to the menu "*utilities*", submenu "*extension bundles*" and proceed to "*install extension*"
■ Locate where the "*GRD.spe*" file is and click "*OK*"
■ If installation is successful; you should get the message: "*congratulations on your installation of GRD*". If you receive the error message: "*cannot find directory…*", make sure Step 3 was completed.
■ Quit then Restart *SPSS*.

You may verify the installation, by going to "*Check installed extensions*" in the "*extension bundle*" menu.

Appendix B: Reference guide

The following section is the syntax diagram of GRD using the *SPSS* nomenclature. It shows all the subcommands available in GRD. The curly brackets ("{}") denote alternatives to each subcommand and the square brackets ("[]") denote the optional subcommands and optional options. The asterisks refer to notes following the syntax.

```
GRD
 /SUBJECTSPERGROUP {EQUAL = n               }
                   {UNEQUAL = n1 n2 n3 ... }
[ /RENAME DV = onename* ]
[ /BSFACTORS factor(nlevel) ... ] †
[ /WSFACTORS factor(nlevel) ... ] †
[ /OVERALL [MEAN = { 0**   } ] [STDDEV { 1**   } ]
           [       { value } ] [       { value } ]
[ /SCORES POPULATION={"RV.NORMAL(#effect, #stddev)"***}
                     {"some formula in quote " ****   }
[ /EFFECTS { factor ( { LINEAR RANGE = value   } ) }
           {          { LINEAR SLOPE = value   }   }
           {          { CUSTOM value value ... }   }
           {          { none                   }   }
           { factor ...                           }
           { factor * factor ...                  }
[ /PRINT [{DEBUG} {ES}]
```

* Default if omitted is DV
** Default of 0 for mean and 1 for standard deviation if omitted
*** Default if omitted
**** The formulas are any functions that can be processed in a COMPUTE command. It typically involves a random number generator (a function beginning with RV), and may use the following codes: #EFFECT for the effect size of that condition, #STDDEV for the standard deviation set in /OVERALL STDDEV, #GM for the overall mean set in /OVERALL MEAN, #S for the subject number in the group, #A for the condition number for the first factor (idem for #B, #C and #D). The /PRINT DEBUG subcommand is only used to debug the GRD extension command and can be ignored.
† No more than four factors can be listed in total

Citation

Harding, B., & Cousineau, D. (2014). GRD: An Spss extension command for generating random data. *The Quantitative Methods for Psychology*, *10* (2), 80-94.

Received: 23/11/13