The Basics of Data Management

By: Tara Reynolds & Christopher Schatschnieder

Florida State University

Planning ahead allows you to implement proper data management. You will get high-quality data that is organized, labeled, and accurate. This will save you time, resources and many headaches! It will also allow your data to be more easily shared, understood and reused by others. Good data management will result in wider dissemination of your work and more citations. Your data must be managed, though, throughout the life cycle of your research, from prior to the conduct of the study to publication.

We will go through 4 crucial steps that will help you get there: Planning Your Database, Creating a Data Entry Program and The Data Entry Process, Writing Cleanup Code, and Creating a Codebook.

PLANNING

Planning The Data Collection

Let's begin with your actual data instruments. You will want to develop the packets you will give to your participants. Packets are a collection of assessments/questionnaires that will be administered to your participants. These can be grouped together into packets.

We recommend creating a cover sheet to add on the front of your packet. On this, you will record any sensitive information such as participant name, date of birth, ethnicity and gender, along with the participant's unique Participant ID. The information recorded here is information that will usually not change throughout the course of the study. We also encourage people to create Packet ID's. Packet ID's are unique ID's associated with each packet. These packet ID's can be preprinted on each PAGE of the packet. We tend to use the "mail-merge" feature that exists in most word processing software. What is the purpose of a Packet ID? The Packet ID allows the researcher to minimize the number of places a participants name is linked to their assessments, and preprinting the Packet ID on every page allows for the multiple assessments to be linked together.

It is also important to "roster" your study before going out for data collection. Rostering means knowing which participants will be participating in your study. This implies having signed consent forms (if required by your study's IRB) and rosters usually obtained from the school.

This method allows your data entry team, upon receiving the packet, to tear off the first page of the packet and record that the participant's information has been received, while at the same time immediately separating any identifying information of the participant from their scored data. One, designated person would enter this identifying information into a Demographics Database. You should have one, master, demographic table/file. This is the key that will link between the Participant ID to their data. This process of recording the cover sheet into the Demographics Database prevents the people doing your regular data entry from accessing the names of people in the study, thereby limiting the number of people who have access to the names of your participants. All remaining pages of the

packet will contain labels with the participant's unique Participant ID, allowing you to enter the full packet's data at a later point in time in a lab where formal data entry occurs. You could also opt to record a Packet ID as well and use that on the additional pages instead. The benefits to this process is that participation is easier to track, a score sheet will never get separated from the student it belongs to, and it minimizes the number of times the students' names appears on forms, which is a requirement for some funding agencies. Remember, you should always leave the second page of the packet blank so that when you rip off the page and store it separately, you will not lose any test data that was collected!

To use this method, you must know ahead of time which students are in the study. To pick up the few students who might fall outside of that known list, assessors in the field could have a stack of assessments with a pre-printed Packet ID on them, and use these for any student who has not yet been assigned an id, allowing you to move forward in the study should you come across this scenario. You will want to consider any process that might stop your research in the field, and come up with a solution for this ahead of time.

Another idea is to record summary data on the cover sheet of the packet. The benefit of this is that you do, at least, maintain some amount of data if the packet is lost. The downside is that it introduces an additional place for errors to occur.

You will need to decide how to handle missing data. Missing data could mean a variety of situations, such as the participant did not know the answer, they refused to answer, the question was accidentally skipped, they were unable to answer, you cannot read the assessor's handwriting, there was an error made by the interviewer and not the participant, they answered outside of the range of allowable responses, the question was not applicable, they were supposed to skip the question and it actually should be blank, etc. Do you need to accurately record in your database why the data was missing? Some programs allow for different system-missing values.

You will need to develop an ID-creation system. The ID variable is the most critical variable in your dataset. Many follow the idea that the ID should not embed identifying information. Identifying information includes personal data as well as geographical, treatment/non treatment, classroom/school ID. Simply make other variables that represent that information. The ID variable should always formatted as a character variable, even if you are only using numbers in the ID. The simplest way is to start at 1 (or 100, or 1000 depending on how may participants you are planning on having!) and count up. But the key is to never change your numbering system, and never have a way for any participant to have two different IDs!

Calculated Fields: If an assessment requires a calculation of a score, this should be done by the assessor, in the field, after they've assessed the child. They will write this total score (a human calculated score) in the assessment's score box and also on the cover sheet of the packet. This forces people This cover sheet is also an additional protection for if your database crashes, you still have these paper forms to fall back on.

You will want to create a process for tracking everything that needs to happen, and a way to make all of these data collection steps happen properly: where do the received forms go, who enters it into the Demographics Database, what happens to the cover sheet, is it kept safe in a locked drawer, where does the packet go when it needs to be data entered, where does it go after data entry, who maintains the list of who is in the study/if they have been assessed, what should you do if you receive data on a

student that did not give consent. A project manager should know where the assessment packets are at all time, keeping a paper trail that you can use to check off data as it comes and goes.

As you move through this process of data management planning in the beginning stages of your study, you should keep a document that records the decisions you make, who made them, and why. This will help you keep stay consistent with your processes, and it will help you remember why you made particular decisions. What you chose to include will be specific to your study, but some examples might include: the above information we discussed, notes on training new staff, specific ways you might start the communication process with your participants, how you will record missing data, school requests that differ from school to school or are based on small specifics of classroom rules, and exactly when and how a follow-up phone call might be needed.

Planning How The Data Will Be Stored In Datasets

The data you collect from forms needs to be transferred into an electronic database. Before you setup a data entry program to do this, let's plan out what needs to be stored in your datasets, and how.

All datasets contain variables. You will write code, using these variables, to run your data analyses. You will want to create a variable naming standard, and you should be consistent with this standard throughout your project! There are many ways to do this. Some use the "One-up numbers" system, naming variables from V*1* through V*n* until all variables are named. Although this is consistent and clean, it provides no indication of what data is held in the variable and leaves more room for errors during data cleanup and analyses. Some use the "Question number" for naming variables, for example, Q1, Q2, Q2a, Q2b, etc, connecting the variable to the original questionnaire. This presents the same problems as the One-up method and gets confusing when one Question contains many parts/responses. We have also seen some questionnaires that contain two Question #3's! So it can get sketchy. We tend to use a naming system of 7 characters: the first two characters contain letters that represent the assessment name; the next three represent the subtest (if applicable) then the next two spaces represent what type of score it is (RS for raw score, SS for standard score etc…). This system was developed in the ancient times when analytic programs require variable names to be 8 characters or less, but we have found that keeping the variable names short and predictable makes data analysis much easier. If you are creating new variable names like this from scratch, run it by a few people first. Make sure that the abbreviation system makes sense to them as well. Another more systematic approach adds prefixes and suffixes onto roots that represent a general category. For example, your root might be MED representing medical history. MOMED might represent the mother's medical history and FAMED might represent the father's. Suffixes often indicate the wave or timepoint of data in longitudinal studies. Choosing a prefix, root, suffix system requires a lot of planning before setting up your database to establish a list of standard two- or three-letter abbreviations. And keep the naming consistent in your longitudinal projects!

Some further tips on creating variables:

- All variable names must be unique!
- The length of the variable must follow the requirements of the software package you will use for data entry and data analyses.

- Most programs do not allow you to begin variable names with a digit. Begin with a letter.
- Should not end with a period
- Should not contain "blanks"
- Should not contain special characters (!, ?, _)
- Many programs do not allow the length to exceed 64 characters, and we recommend only using 8 when possible, and not exceeding 12.
- Avoid using keywords that are used during statistical analyses like ALL, AND, BY, EQ, GE, GT, LE, LT, NE, NOT, OR, TO, WITH.
- In addition to naming them, variables have two other main properties: length and type (char, var, date, etc.) Be consistent with the length and type of variables across datasets. Otherwise merging datasets may become a nightmare!
- For item level data, always put the number of the item LAST. This is helpful in data analyses if you are running a sum on 300 variables, you can just say item1-item300 rather than typing out all 300 variable names.

Once you have your naming system in place, you will want to name and identify all of the variables you will collect. A helpful way to do this is to take a blank assessment packet, highlight every field you want to capture in data entry, and then write the desired variable names on it, along with their length and type.

Many data entry and statistical programs allow you to add a text description to the variable name that describes the content of that variable. This is called a Variable Label. For example, it might contain the item or question number of your data collection instrument, the word being assessed in a vocabulary test, or if it is a calculation field, the way that this number was calculated. The length of a label might be limited, so plan ahead and stay consistent with your labeling across the project. When participant responses to some items might contain the same answers (i.e. Strongly Agree, Slight agree…) keep these consistent across the project. This provides cleaner coding when you get to the data anlyst stage.

When translating text to data, It is important to really consider your questions and how it will be recorded from a data standpoint. For example, "Check here if you child does not have siblings." Typically, people looking at data consider a "checked" box is stored as a "Yes" response or a "1" in data. So if your variable name is "siblings" and the value is "1" many people will assume 'yes, this child has siblings' (which is not the case). You might want to change the question to say "Check if your child has siblings." Then a checked value creates siblings =1 which means it is 'true' and they do have siblings. If it makes sense on the form to "check" for *not* having a specific value or the papers have already been designed, then maybe consider naming your variable "nosiblings" (then the check mark can still=1 and be true).

When creating variables and values:

- Usually 1 = Yes or TRUE or correct or present
- Usually 0 = No or FALSE or absent
- Instead of using Sex or Gender as a variable, use Male (or Female). Then make the 1's represent that variable as being present. So if you used Male then make 1 represent males.
- With multiple response items, your assessment should follow a numbering systems that makes sense, because in the end, you will want your scale to match the data so there is less confusion. For example, if you give them a multiple choice and their options are numbered 1-5, you will want to

store that as 1-5 in your dataset as well. This will allow someone looking at the paper form and also looking at a dataset variable value to understand what is stored there. So, if you want an answer like "none" to be a "0" in your dataset, it should be 0 on the paper also (as people using your data will likely refer to the paper assessment also). Otherwise, you have a paper score of a "2" circled, yet you are storing it as a "1" in your database.

- Try to stay consistent with scoring your scaled and multiple choice options. Do you like starting scales at 0 or 1?
- When coding grouped responses (similar from question to question) keep the values that represent the response the same. Ambiguity will cause coding difficulties and problems with the interpretation of the data later on.
- Avoid entering text responses where possible. Try to make them select a predetermined response. Do not let them type in a response as you will get a wide variety of answers for something that should be scored exactly the same. For example, grade: they could type in 1, 1st, First, Elementary, or even have a typo. Make them select or circle "1st Grade", and then store that value as "1", not the text.

Based on your study, you may want to have separate datasets for different types of data collection. For example, you might consider maintaining a separate table for teachers/schools/parents/or other "upper level" units.

CREATE THE CODEBOOK

Once you have decided on your variable names, you will want to create the codebook.

The codebook at a minimum should contain the names of all the variables collected in your study and notation that describes that that variable represents. For variables that use codes to describe categories, those codes should come with a description of what that code represents. The person performing the data analyses will need this codebook to analyze the data, It also allows others, who are not familiar with your project, to completely understand your data, cite it, and reuse it responsibly well into the future.

Again, at minimum, a Codebook should contain:

- Variable names
- Variable labels
- Values of the data stored there (which again, when applicable, should be consistent across the project).

For the variable "label", we recommend using the exact wording of the question asked for that variable.

Other information you might want to include in the Codebook, per variable, are:

- Metadata that was implied during data entry, but is not shown in the dataset. This might include the fact that it was collected in one specific state; which wave it was; or that one dataset included teacher's responses, and another included a school psychologist's response, yet those values were not stored on the dataset.
- Skipping patterns (the reason certain answers were skipped by the respondent, i.e. "If your response is No, then go to Question #10.")

- Per-project-unique codes or missing data codes and exactly what they meanA description of a variable that was calculated during or after data entry, including the code used to get that number.

Some chose to use XML or searchable PDF, to stay in compliance with the Data Documentation Initiative (DDI), an international metadata standard for data documentation and exchange. Tools that can help create a codebook can be found here: https://ddialliance.org/training/getting-started-new-content/create-a-codebook Producing a high-quality codebook and keeping it updated as projects progress takes a lot of time but is a critical step in ensuring that your information is shared effectively both within your team and with other researchers. It allows all members of your team to explain results and to share data in open science.

Data Entry

Creating The Data Entry Program

Decide which software you will use for data entry. If you have a quick and simple study, you might want to pick a user friendly database package like SPSS, EXCEL etc. If your project is large, you may want to create a relational database or use a pre-existing program that allows for data entry such as , RedCap, SAS, FileMaker, Microsoft Access, MYSQL, or Oracle. These programs will let you define variable characteristics, create "data entry screens", incorporate data cleaning allowing you minimize errors in real time, do sum-to-total checks, incorporate range restrictions, allow you to link multiple datasets with a linking variable, and lock data at the record level. All methods have their respective pros and cons. Using spreadsheets should be avoided whenever possible. It becomes difficult for the people performing data entry to keep track of what column they are in, and also one "bad sort" can ruin an entire data set. Spreadsheets should be reserved for very small studies where if one had to re-enter all the data it would not be devastating.

When you create the data entry screens, you will want to make them look as similar to the packets as possible. This will not only save time for data entry personnel but will help them stay focused on what they are entering, rather than focusing on where they are in the program and feeling lost.

The flow of the forms on your data entry screens should follow the flow of the paper packet in the data entry person's hand. When they have finished entering one assessment, and turn the page, your "next" button on the data entry screen should advance them to the next packet they are looking at. Keep it all consistently flowing. This helps them save time and stay focused.

Do everything you can to help prevent data entry errors. Define range limits on your data where possible (for example, the score must be a value between 0 and 30; the birthday and/or assessment date must be between certain dates). Do not allow a required field to have a null value. Do not allow entry into certain fields if the criteria of the participant should not allow them to answer specific values.

Check that all of your variables follow a consistent naming system and make sense. You should define each variable as numeric or character, based on what value it should contain, selecting numeric when possible. Repeatedly check during your data design stages to ensure you have kept these consistent across your project: variables naming, variable labeling, what coded values represent, value lists.

Calculated fields: The data entry team will be entering item level and sum score information (calculated by a human, in the field). You will probably want to create a calculated field using a formula, and compare your calculation to the assessor's calculation. This code calculation happens AFTER data entry, on the back end. (You do not want it calculated on the data entry person's screen, as they should not be looking at calculations and comparing them during this time. They should just be doing data entry.) If the two calculated scores do not match, then send it back to the data entry team to find the problem

Ensure that you have some kind of data entry quality checks. We recommend all data get entered twice. That is, have a primary and a secondary dataset that are supposed to contain the same information. Then these two datasets should be compared for discrepancies and any data that do not match should be flagged and investigated. Discrepancies can be reconciled either in a third dataset or through hard coding (syntax). This brings us to a very important principle.

CHANGES TO DATA ALWAYS FLOW FORWARD.

What does that mean? That means that the primary and secondary datasets used for reconciliation should never be changed. Reconciliation should either occur in a third dataset (for example, make a copy of the primary dataset and then make the necessary changes to it and save it as a third, reconciled datasets. We often call these transactional datasets.) or using syntax to read in the primary dataset and make the changes in that fashion. Why do we recommend this? Because if something goes wrong or mistakes are made in the data cleaning process you will always have the original datasets to fall back on.

Longitudinal databases can take on many forms. We recommend creating different files/sheets/tables for different waves of data. Use the univariate format – which means use the same variable names for the data collected over multiple time points. Don't add numbers to variables to designate "wave" or end them with a F or S to represent fall and spring. Those can be added later programmatically, and the univariate layout makes data cleaning easier. In the "wave" datasets, only enter what you collected at that wave (plus ID and date of collection). Don't add anything else. Maintain a separate file with demographic information for each subject (this is the only place that a subject name can appear). All other datasets should only use IDs

After creating the program, you should sit with the data entry team and watch them enter a few forms. You will find there might be misunderstandings about the data, or that there is something you can tweak on the screen to make the process easier for them. Test it with them before going "live". After the enter a few forms, you will also want to check the data against the participant's paper form to make sure all data was translated correctly; and then, run the data by the PI and the Data Analyst to make sure it is what they were expecting.

You will want to make sure that your department/university has a file server. This allows multiple computers to access the same data files, and it is easy to set a process that backs up your data. And most importantly, MAKE SURE YOUR DATA IS BEING BACKED UP THROUGHOUT THE ENTIRE PROCESS!

The Data Entry Process

You'll need to set up data entry stations or a lab where people can simply sit down and enter data. The data should stay at a station and should never follow a person if they need to walk away or go ask someone a question.

The project manager must maintain some kind of master list that tells her/him what is expected at the end of each wave in terms of which participants were tested. This can be created from consent forms or school rosters and can be easily maintained in EXCEL. At all times, this person should have a master list of all students and know who is in the study. This master list might be from the Demographic Database and contain contact information on the family.

When a packet arrives from the field, it should be immediately checked by someone for accuracy (missing data, miscalculations, missing labels). Checking this right away can catch administration errors that can greatly impact your data. Checking the protocols at the front end will reduce significant cleaning time at the back end.

If the packet looks good, immediately log it into the Demographic Database. The database should only exist in one place. It will contain the student ID, packet ID, and any other sensitive information. The benefit to doing this is that safe keeping of your data. One, trusted person looks at it, as you will want as few eyes as possible seeing any sensitive, identifying information. The process will also help your team keep track of which students still need to be tested at the schools (because you have already rostered; and you know who needs to be tested) or if perhaps, a form has been misplaced. Many students miss kids/a classroom, and this will help you get more of your data, on time.

There should be an easy system in place, where each person, without effort or documentation, knows what needs to be entered. You might want to have file cabinets clearly labeled, for example, "Needs double entry." The data entry team should never have to figure out which packets have been entered, and which haven't, by pulling up a spreadsheet. This also will prevent simple errors like a packet being entered twice.

Data entry should be as thoughtless as possible. No math should be done in the middle of data entry. Scores should have been calculated in the field, but if not, perhaps have a place in your filing cabinet system for "needs scores", before "Needs to be entered."

People should only have access to their data entry program module. If your data entry program has other studies in it, they should not be able to see or access those. Once logged in to their study, they should only have data access to the functions that are available only to their role in the project (first entry, secondary, reviewer, etc.) and no one else's roles. Make sure your system, as defined by your research grant, is in place and being followed.

The PI and Data Analyst should develop Data Entry "Rules" for ambiguous cases. The data entry team should be well trained on this, and have a written, easy Rule Book at each data entry station that they can refer to while entering data. Cases might include what to do when: the form or question has no answer, there are blanks, the respondent didn't follow directions, or the respondent circled two options when only one was allowed. The key, as always, is to be consistent!

When the data entry team is completely finished with their entries, they will notify the Data Manager, who will extract the data from the data entry program.

Cleaning Data

You will now create and manage the working (aka transitional) and master datasets. As you clean your data, remember to always keep a trail of everything you do. Always work in a copy of the dataset, maintaining the original dataset in its raw state. Always back up your data and documentation as you work.

All data management code should be saved and annotated as you go. Explain what you are looking for, what kind of a check you are running, or why you are doing this (for example: 'Required by IRB to …. "). Anyone who looks at your code should be able to read your annotations at every level and understand what is happening and why. They should not have to read your programming code to figure out what you were doing or what your intentions were.

> Example: Your assessment has a rule that the assessor records the highest item number the child answered. So, in your annotation, you might note: /*This grabs the highest item number the assessor said the child answered (for example, Question #85) and makes sure no questions were scored beyond that (if child answered questions higher than question 85, then you want to flag it) and that all previous questions before Q85 were indeed answered.*/
>
> Second example: Might need to remove certain rows before passing for analyses? If child was removed from study, didn't have enough data, moved to a new school, etc. but you need to explain why they were deleted from the dataset.

Not only will these annotations help the next person that might be looking at your data process, but these detailed notes will also help you when you revisit your data multiples times during the study. It prevents you from overlooking decisions made about the study, and it allows you to quickly run through all of these requirements if you need to run the file again for some reason in the future. (Example: 2 more participants were added to the data entry program after the project was "over", and all data must be re-extracted, and code must be re-run).

Now let's begin cleaning the data. Structure your data into the file layouts that work for your project. You may need to join tables or rearrange your data into long or wide datasets. Get the data files organized in the way that fits your research needs.

Make sure you are only dealing with the rows of data that have been double-entered and verified. Data entry programs often maintain the first entry, second entry, and finalized entry. Make sure your working dataset only contains the finalized record for that participant.

Any Identifying variables and the Study ID should be located at the front of the dataset.

You will begin checking the data, to make sure everything looks correct. Mistakes can still be present after double entry. You will want to write code to check for obvious mistakes. You will also want to, via code, check your dataset for project-specific rules. This process requires fully understanding the project, the protocols, the assessments, the data, the study requirements, and the intentions of the Data Analyst. You will have to define these code steps on your own, based on the specifics of your project.

Examples of some general checks we run include:

- The numbers of participants you have match the number of participants given to you by the project manager
- Every ID that is in the table should have been assessed (ties back to master list), to make sure you have no unexpected students and/or to make sure no one is missing
- Eyeballing your data – surprisingly, this is quite helpful
- Is the variable in the correct format (char or numeric)?
- Sum-to-total checks
- Sum scores on the summary sheet match the sum scores on the score sheets
- Range checks
- Duplicate ID entries, duplicate form entries
- Basal and Ceiling rules
- Validity checks
- Run descriptives on key variables/calculated scores
- Run frequency codes just to get a fresh look at your data. You might find a value that just seems off
- Run Consistency Checks that are particular to your study and based on your extensive knowledge of the study.
- Double checking all calculated fields like "Total Scores", whether it was entered by a human or calculated in the data entry program
- Are the scores expected for that particular wave and timepoint present; and vice versa, do you have scores that were not expected for that participant/timepoint
- People have been assessed at every timepoint required of them by the protocol (pretest, post test, delayed post test, etc.)
- Do the means make sense across ages/grades/timepoints
- Do you need to find incomplete records?
- You are only looking at rows that have been double entered and verified.
- Inconsistent responses per participant
- Check project-specific protocols. There will be a lot of these. This might include things like "no one was given a PPVT assessment in the Spring." So, make sure there is no data for that. Check that all project-specific requirements have been met.
- Or, they marked that this assessment "was not administered" yet have data for it. Go back and have the data entry team research it.
- Ensure that the variable and their values match those in the codebook, you don't want to have a "5" response in your study that is not explained in your data manual.

Sidenote: We recommend running some of these code checks during the data entry process here and there. You never know what kind of issue might pop up, and it's better to fix it earlier in the process rather than later.

You should create a system for tracking issues your find within the data and their resolutions. When mistakes are found, often you will pull the protocols and conduct a "hard target" search. This means retrieving the original paper assessment to make the corrections. How you make the correction might depend on the data entry program you use. For example, if you are using a program that is currently pulling data from one screen to continue data entry on another screen (i.e. birthday is incorrect, and all future scores might be calculated incorrectly) (or some PI's use the data entry program to review data

and run their own pre-analyses stats), you might be able to have the data entry person correct the error on their end. Send these issues back to them.

If however, you feel like the error should be corrected via hard code within your working dataset, then you will begin the process of making a "transactional dataset." Always work from a copy, and keep extra copies of clean data in other locations. You should never rewrite the original files produced from data entry. Using code to correct the data ensures you can track all the changes made from entry to your final dataset. DATA FLOWS FORWARD, NOT BACKWARD. You should always be able to go back to the originally entered data and be able to see where values got changed and why they were changed.

Within your transactional dataset, you might also calculate or add in values like standard scores, percentile ranks, confidence bands, and normative scores. For assessments that convert raw scores to percentile scores, you might have had a person manually add a score during data entry using a table to match each student's criteria (age/grade/score) to the standard score and then enter that into the system during data entry. If not, however, you might have decided to instead use lookup tables to create those values. You create a separate dataset with all possible scores, and their corresponding percentile/standard scores. You then join each participant to this table to retrieve their score, based on the assessment criteria. It is more work on the front end, but it is more error free and works out nicely when dealing with larger amounts of data. Some assessment already have published norming tables out there that you can use.

All changes to data need to happen in one file. You should not have multiple versions of data files. If you need to add 800 flags, go ahead and do it, but do it all in ONE dataset.

Remember to document where your data is backed up and stored. Multiple versions of final datasets ensure their safety. Documentation should include how many copies you will keep and how you will synchronize them. It is a good practice to also include security issues such as passwords, encryption, power supply or disaster backup, and virus protections.

Once you have made all of your data management changes, you will have one master dataset that will be ready for analyses. This dataset will hold everything, and can be quite large.

Analysis datasets

Datasets that can be used for analysis can also be created. These datasets are typically a subset of the variables or subjects from the master dataset. These datasets should always be created from the master datasets, but the master datasets should never be changed. Once data analysis has been completed and the manuscript sent off for review, the analysis dataset needs to be "frozen". That is, no additional changes should be made to it, and the datasets should no longer be reconstructed from the original master datasets. This ensures that you will always be able to replicate the results of the analyses reported in any paper.