Report Writing for Data Science in R

Roger D. Peng

Report Writing for Data Science in R

Roger D. Peng

This book is for sale at http://leanpub.com/reportwriting

This version was published on 2019-04-17

Also By Roger D. Peng

R Programming for Data Science

The Art of Data Science

Exploratory Data Analysis with R

Executive Data Science

Advanced Statistical Computing

The Data Science Salon

Conversations On Data Science

Mastering Software Development in R

Essays on Data Analysis

Contents

1. What is Reproducible Reporting?

Watch a video of this chapter.[1]

This chapter will be about reproducible reporting, and I want to take the opportunity to cover some basic concepts and ideas that are related to reproducible reporting, just in case you haven't heard about it or don't know what it is.

Before we get to *reproducibility*, we need to cover a little background with respect to how science works (even if you're not a scientist, this is important). The basic idea is that in science, *replication* is the most important element of verifying and validating findings. So if you claim that X causes Y, or that Vitamin C improves disease, or that something causes a problem, what happens is that other scientists that are independent of you will try to investigate that same question and see if they come up with a similar result. If lots of different people come up with the same result and replicate the original finding, then we tend to think that the original finding was probably true and that this is a real relationship or real finding.

The ultimate standard in strengthening scientific evidence is replication. The goal is to have independent people to do independent things with different data, different methods, and different laboratories and see if you get the same result. There's a sense that if a relationship in nature is truly there, then it should be robust to having different people discover

[1] https://www.youtube.com/watch?v=4rBX6r5emgQ

it in different ways. Replication is particularly important in areas where findings can have big policy impacts or can influence regulatory types of decisions.

What's Wrong with Replication?

What's wrong with replication? There's really nothing wrong with it. This is what science has been doing for a long time, through hundreds of years. And there's nothing wrong with it today. But the problem is that it's becoming more and more challenging to do replication or to replicate other studies. Part of the reason is because studies are getting bigger and bigger.

In order to do big studies you need a lot of money and so, well, there's a lot of money involved! If you want to do ten versions of the same study, you need ten times as much money and there's not as much money around as there used to be. Sometimes it's difficult to replicate a study because if the original study took 20 years to do, it's difficult to wait around another 20 years for replication. Some studies are just plain unique, such as studying the impact of a massive earthquake in a very specific location and time. If you're looking at a unique situation in time or a unique population, you can't readily replicate that situation.

There are a lot of good reasons why you can't replicate a study. If you can't replicate a study, is the alternative just to do nothing, just let that study stand by itself? The idea behind a reproducible reporting is to create a kind of minimum standard or a middle ground where we won't be replicating a study, but maybe we can do something in between. The basic problem is that you have the gold standard, which is replication, and then you have the worst standard which is doing nothing. What can we do that's

in between the gold standard and diong nothing? That is where reproducibility comes in. That's how we can kind of bridge the gap between replication and nothing.

In non-research settings, often full replication isn't even the point. Often the goal is to preserve something to the point where anybody in an organization can repeat what you did (for example, after you leave the organization). In this case, reproducibility is key to maintaining the history of a project and making sure that every step along the way is clear.

Reproducibility to the Rescue

Why do we need this kind of middle ground? I haven't clearly defined reproducibility yet, but the basic idea is that you need to make the **data** available for the original study and the **computational methods** available so that other people can look at your data and run the kind of analysis that you've run, and come to the same findings that you found.

What reproducible reporting is about is a *validation of the data analysis*. Because you're not collecting independent data using independent methods, it's a little bit more difficult to validate the scientific question itself. But if you can take someone's data and reproduce their findings, then you can, in some sense, validate the data analysis. This involves having the data and the code because more likely than not, the analysis will have been done on the computer using some sort of programming language, like R. So you can take their code and their data and reproduce the findings that they come up with. Then you can at least have confidence that the analysis was done appropriately and that the correct methods were used.

Recently, there's been a lot of discussion of reproducibility in the media and in the scientific literature. The journal *Science* had a special issue on reproducibility and data replication. Other journals of updated policies on publication to encourage reproducibility. In 2012, a feature on the TV show 60 minutes looked at a major incident at Duke University where many results involving a promising cancer test were found to be not reproducible. This led to a number of studies and clinical trials having to be stopped, followed by an investigation which is still ongoing.

Finally, the Institute of Medicine, in response to a lot of recent events involving reproducibility of scientific studies, issued a report saying that best practices should be done to promote and encourage reproducibility, particularly in what's called 'omics based research, such as genomics, proteomics, other similar areas involving high-throughput biological measurements. This was a very important report. Of the many recommendations that the IOM made, the key ones were that

- Data and metadata need to be made available;
- Computer code should be fully specified, so that people can examine it to see what was done;
- All the steps of the computational analysis, including any preprocessing of data, should be fully described so that people can study it and reproduce it.

From "X" to "Computational X"

What is driving this need for a "reproducibility middle ground" between replication and doing nothing? For starters, there are a lot of new technologies on the scene and in many different fields of study including, biology, chemistry

and environmental science. These technologies allow us to collect data at a much higher throughput so we end up with these very complex and very high dimensional data sets. These datasets can be collected almost instantaneously compared to even just ten years ago—the technology has allowed us to create huge data sets at essentially the touch of a button. Furthermore, we the computing power to take existing (already huge) databases and merge them into even bigger and bigger databases. Finally, the massive increase in computing power has allowed us to implement more sophisticated and complex analysis routines.

The analyses themselves, the models that we fit and the algorithms that we run, are much much more complicated than they used to be. Having a basic understanding of these algorithms is difficult, even for a sophisticated person, and it's almost impossible to describe these algorithms with words alone. Understanding what someone did in a data analysis now requires looking at *code* and scrutinizing the computer programs that people used.

The bottom line with all these different trends is that for every field "X", there is now "Computational X". There's computational biology, computational astronomy—whatever it is you want, there is a computational version of it.

Air Pollution and Health: A Perfect Storm

One example of an area were reproducibility is important comes from research that I've conducted in the area of air pollution and health. Air pollution and health is a big field and it involves a confluence of features that emphasize the need for reproducibility.

The first feature is that we're estimating very small, but very important, public health effects in the presence of a numerous much stronger signals. You can think about air pollution as something that's perhaps harmful, but even if it were harmful there are likely many other things that are going to be more harmful that you have to worry about. Pollution is going to be at the very top of the list of things that are going to harm you. In other words, there's an inherently weak signal there.

Second, the results of a lot of air pollution research inform substantial policy decisions. Many federal air pollution regulations in the United States are based on scientific research in this area and these regulations can affect a lot of stakeholders in government and industry.

Finally, we use a lot of complex statistical methods to do these studies and these statistical methods are subsequently subjected to intense scrutiny. The combination of an inherently weak signal, substantial policy impacts, and complex statistical methods almost require that the research that we do be reproducible.

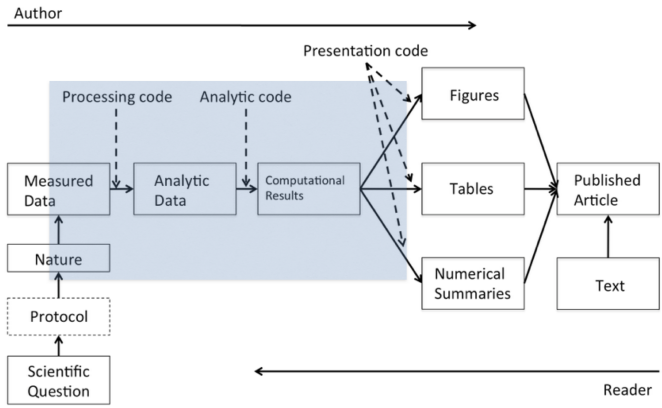
Summary

- Replication, whereby scientific questions are examined and verified independently by different scientists, is the gold standard for scientific validity.
- Replication can be difficult and often there are no resources to independently replicate a study.
- Reproducibility, whereby data and code are re-analyzed by independent scientists to obtain the same results of the original investigator, is a reasonable minimum standard when replication is not possible.

2. The Data Science Pipeline

Watch a video of this chapter.[1]

The basic issue is when you read a description of a data analysis, such as in an article or a technical report, for the most part, what you get is the report and nothing else. Of course, everyone knows that behind the scenes there's *a lot* that went into this report and that's what I call the **data science pipeline**.



The Data Science Pipeline

In this pipeline, there are two "actors": the *author* of the report/article and the *reader*. On the left side, the author is going from left to right along this pipeline. The reader is going from right to left. If you're the reader you read the

[1] https://www.youtube.com/watch?v=GXSRP--d3Q4

article and you want to know more about what happened: Where is the data? What was used here? The basic idea behind reproducibility is to focus on the elements in the blue blox: the analytic data and the computational results. With reproducibility the goal is to allow the author of a report and the reader of that report to "meet in the middle".

Elements of Reproducibility

What do we need for reproducibility? There's a variety of ways to talk about this, but one basic definition that we've come up with is that there are four things that are required to make results reproducible:

1. **Analytic data**. The data that were used for the analysis that was presented should be available for others to access. This is different from the *raw data* because very often in a data analysis the raw data are not all used for the analysis, but rather some subset is used. It may be interesting to see the raw data but impractical to actually have it. Analytic data is key to examining the data analysis.
2. **Analytic code**. The analytic code is the code that was applied to the analytic data to produce the key results. This may be preprocessing code, regression modeling code, or really any other code used to produce the results from the analytic data.
3. **Documentation**. Documentation of that code and the data is very important.
4. **Distribution**. Finally, there needs to be some standard means of distribution, so all this data in the code is easily accessible.

Authors and Readers

It is important to realize that there are multiple players when you talk about reproducibility–there are different types of parties that have different types of interests. There are authors who produce research and they want to make their research reproducible. There are also readers of research and they want to reproduce that work. Everyone needs tools to make their lives easier.

One current challenge is that authors of research have to undergo considerable effort to make their results available to a wide audience. Publishing data and code today is not necessarily a trivial task. Although there are a number of resources available now, that were not available even five years ago, it's still a bit of a challenge to get things out on the web (or at least distributed widely). Resources like GitHub[2] and RPubs[3] and various data repositories have made a big difference, but there is still a ways to go with respect to building up the public reproducibility infrastructure.

Furthermore, even when data and code are available, readers often have to download the data, download the code, and then they have to piece everything together, usually by hand. It's not always an easy task to put the data and code together. Also, readers may not have the same computational resources that the original authors did. If the original authors used an enormous computing cluster, for example, to do their analysis, the readers may not have that same enormous computing cluster at their disposal. It may be difficult for readers to reproduce the same results.

Generally the toolbox for doing reproducible research is small, although it's definitely growing. In practice, authors

[2] https://github.com
[3] http://rpubs.com

often just throw things up on the web. There are journals and supplementary materials, but they are famously disorganized. There are only a few central databases that authors can take advantage of to post their data and make it available. So if you're working in a field that has a central database that everyone uses, that's great. If you're not, then you have to assemble your own resources.

Summary

- The process of conducting and disseminating research can be depicted as a "data science pipeline"
- Readers and consumers of data science research are typically not privy to the details of the data science pipeline
- One view of reproducibility is that it gives research consumers partial access to the raw pipeline elements.

3. Literate Statistical Programming

Watch a video of this chapter.[1]

One basic idea to make writing reproducible reports easier is what's known as *literate statistical programing* (or sometimes called literate statistical practice[2]). This comes from the idea of literate programming[3] in the area of writing computer programs.

The idea is to think of a report or a publication as a stream of text and code. The text is readable by people and the code is readable by computers. The analysis is described in a series of text and code chunks. Each kind of code chunk will do something like load some data or compute some results. Each text chunk will relay something in a human readable language. There might also be presentation code that formats tables and figures and there's article text that explains what's going on around all this code. This stream of text and code is a literate statistical program or a literate statistical analysis.

Weaving and Tangling

Literate programs by themselves are a bit difficult to work with, but they can be processed in two important ways. Lit-

[1] https://www.youtube.com/watch?v=bwQWhZQmDuc
[2] http://www.r-project.org/conferences/DSC-2001/Proceedings/Rossini.pdf
[3] https://en.wikipedia.org/wiki/Literate_programming

erate programs can be **weaved** to produce human readable documents like PDFs or HTML web pages, and they can **tangled** to produce machine-readable "documents", or in other words, machine readable code. The basic idea behind literate programming in order to generate the different kinds of output you might need, you only need a *single source document*—you can weave and tangle to get the rist. In order to use a system like this you need a documentational language, that's human readable, and you need a programming language that's machine readable (or can be compiled/interpreted into something that's machine readable).

Sweave

One of the original literate programming systems in R that was designed to do this was called Sweave. Sweave uses a documentation program called LaTeX and a programming language, which obviously is R. It was originally developed by Fritz Leisch, who is a core member of R, and the code base is still maintained by R Core. The Sweave system comes with a any installation of R.

There are many limitations to the original Sweave system. One of the limitations is that it is focused primarily on LaTeX, which is not a documentation language that many people are familiar with. Therefore, it can be difficult to learn this type of markup language if you're not already in a field that uses it regularly. Sweave also lacks a lot of features that people find useful like caching, and multiple plots per page and mixing programming languages.

knitr

One of the alternative that has come up in recent times is something called `knitr`. The `knitr` package for R takes a lot of these ideas of literate programming and updates and improves upon them. `knitr` still uses R as its programming language, but it allows you to mix other programming languages in. You can also use a variety of documentation languages now, such as LaTeX, markdown and HTML. `knitr` was developed by Yihui Xie while he was a graduate student at Iowa State and it has become a very popular package for writing literate statistical programs.

Summary

- Literate statistical programming tools can make it easier to write up reproducible documents containing data analyses.
- Sweave was one of the first literate statistical programming tools, which weaved together a statistical language (R) with a markup language (LaTeX).
- knitr is a package that builds on the work of Sweave and provides much more powerful functionality, including the ability to write in Markdown and create a variety of output formats.

4. Reproducibility Check List

Reproducibility can be more or less easy to achieve depending on the context, the scientific area, the complexity of a data analysis, and a variety of other factors. However, over time, I've developed a few rules of thumb that I think are useful for at least encouraging reproducibility, if not guaranteeing it. In this chapter, I put together a simple "check list" of ideas that I've developed in my experience doing data analysis and computational research.

Start With Good Science

Good science, generally speaking, or a good *question*, is the key to any worthwhile investigation. The general rule of "garbage in, garbage out" applies here. If you do not start with a meaningful question, then no amount of data analysis or statistical machinery will be able to make the results interesting to you. If the question and the results are not interesting to you or your colleagues, there will be relatively little motivation to make the results reproducible. This is a problem.

Having a coherent, focused question simplifies many problems and will make it easier to determine whether you are on the right track or if an error has occurred. Vague and broadly defined questions can fit many different scenarios

and are more likely to encourage sloppiness and unclear thinking.

Related to working on a problem that interests you is working with good collaborators. Collaborators that you work well with will reinforce good practices and will encourage you to do the best work. Ultimately, if you are uncomfortable with the people you are working with, or more seriously, if you do not completely *trust* the people you are working with, then there will be breakdowns in communication and things will get lost. If you don't feel comfortable (politely) challenging a colleague's work when needed, then bad work will make it through, which can lead to non-reproducible results. Working with the right people is an important, but often unmentioned, aspect of making work reproducible.

Don't Do Things By Hand

If this chapter could be boiled down to one rule, it might be "Don't do things by hand". What do I mean by that? Here are a few examples that are common, but are bad practice:

- Editing spreadsheets of data to "clean it up". Often this is doen to remove outliers, do quality assurance or quality control checks (QA/QC), or validating individual data entries
- Editing tables or figures (e.g. rounding, formatting) to make then look better
- Downloading data from a web site using a web browser
- Moving data around your computer
- Splitting or reformatting data files

Often, the motiviation for doing all of the above things is that "We're just going to do this once." The thinking is that if the activity is only going to be done once, it doesn't need to be automated (i.e. programmed into a computer).

But programming a procedure into a computer is not necessarily about automation. It is also about *documentation*. The problem with things that are done by hand, is that things done by hand need to be precisely documented (this is harder than it sounds). Often, it can very difficult to communicate to someone what was done after the fact. It can be easy to miss a step that "isn't important" when in fact it is.

Don't Point And Click

Pointing and clicking is obviously related to doing things by hand. Most modern operating systems have a windowing interface that allow you to click on menus that can lead to automated built-in routines. Many data processing and statistical analysis packages have graphical user interfaces (GUIs) that simplify the use of the program, but the actions you take with a GUI can be difficult for others to reproduce because there's not necessarily a log of what was clicked.

Some GUIs for statistical analysis packages produce a log file or script which includes equivalent commands for reproducing the behavior of the GUI, but this is by no means the standard. In general, be careful with data analysis software that is highly *interactive*. There is often a trade-off between the ease of use of a software package and the tendency to lead to non-reproducible results.

Of course, this doesn't mean that all interactive software. Some software has to be interactive, like text editors or word processors, and that's fine. It's just when the software

must be used to *conduct data analysis*, you must be careful not to be seduced by the ease-of-use of an interactive interface.

Teach a Computer

The opposite of doing things by hand is teaching a computer to do something. Computers need very precise instructions to accomplish a task so there's no room for ambiguity. This is a Good Thing if your goal is to make your procedures and processes reproducible. If something needs to be done as part of your analysis or investigation, try to teach your computer to do it, *even if you only need to do it once*. In the end, teaching a computer to do something almost guarantees reproducibilty.

Example: Downloading data

Downloadling datasets is something data scientists are constantly doing. But if you're using a web browser to download data, you're probably not downloading data in a reproducible way. Suppose you wanted to obtain a dataset to analyze from the UCI Machine Learning Repository. One way to do that by hand would be to

1. Go to the UCI Machine Learning Repository at http://archive.ics.uci.
2. Download the Bike Sharing Dataset[1] by clicking on the link to the Data Folder, then clicking on the link to the zip file of dataset, and choosing "Save Linked File As…" and then saving it to a folder on your computer

But this involves doing things by hand! Normally, the interactive nature of the web browser is a great feature, but

[1] http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

not when you need to download a dataset that is important to your analysis

Another way to accomplish this task is to teach your computer to do the same thing using R:

```
> download.file("http://archive.ics.uci.edu/ml/machine-le\
arning-databases/00275/Bike-Sharing-Dataset.zip",
+     "ProjectData/Bike-Sharing-Dataset.zip")
```

Notice here that

- The full URL to the dataset file is specified (no clicking through a series of links)
- The name of the file saved to your local computer is specified ("Bike-Sharing-Dataset.zip")
- The directory in which the file was saved is specified ("ProjectData")
- The code can always be executed in R (as long as link is available)

Now that you've taught a computer to do this task, it is far more reproducible than writing down a series of instructions directing someone to use a web browser. In fact, the R code is a far more compact representation of this task.

Use Some Version Control

Version control systems is not something we've explicitly covered in this book so far, so I won't go into great detail here. Briefly, version control systems are software systems designed to help you keep track of changes to a set of code files in a given project. They are primarily designed for

software projects where code files are typically reasonably small text files, but they can also be applied to data analysis projects. Examples of popular version control systems these days are git[2], subversion[3] (svn), and mercurial[4] (hg).

If there's one reason for using a version control system to track the changes to your data analysis project, it is that the version control system can help to **slow things down**. In many instances with data analyses, it's tempting to zoom ahead and start plowing into the data to find something interesting. This excitement is good, of course, but not at the expense of keeping track of what's happening. Version control systems can be helpful for reminding you that changes need to be tracked and notes need to be taken, if only to remind *yourself* of what happened a little be later (much less for communicating to team members).

Version control systems have many benefits, such as being able to track snapshots of a project and to mark/tag major milestones. They also allow for simple collaboration across networks (internal or external) and for publishing your work. With complementary web sites like GitHub[5], BitBucket[6], and SourceForge[7], it is now straightforward to publish your projects so that anyone can view your work. Most of these sites have some free tier that allows you to host your projects without any cost to you.

[2] http://git-scm.com
[3] http://subversion.apache.org
[4] https://mercurial.selenic.com
[5] https://github.com
[6] https://bitbucket.org
[7] http://sourceforge.net

Keep Track of Your Software Environment

If you work on a complex project involving many tools and datasets, the software and computing environment can play a critical role in determining whether your analysis is reproducible. In the extreme case, if your analysis depends on some custom proprietary software or hardware that only you possess, then obviously no one else will be able to reproduce your analysis. However, there are many cases short of that extreme one where the software and hardware environment in which a data analysis was conducted can be important for reproducibility.

Here are a few things that you should keep in mind as you keep track of your environment.

- **Computer architecture**: What kind of CPU does your computer use? Intel, AMD, ARM, etc.? And are you using graphical processing units (GPUs)?
- **Operating system**: Are you using Windows, Mac OS, Linux / Unix, something else? The more obscure your operating system, the more difficult it might be to reproduce your work unless you do things in a cross-platform manner.
- **Software toolchain**: This includes things like compilers, interpreters, the command shell, programming languages (C, Perl, Python, etc.), database backends, and any data analysis software.
- **Supporting software and infrastructure**: Software libraries, R packages, software dependencies
- **External dependencies**: Your data analysis is likely to depend on things outside of your computer, like web sites, data repositories, remote databases, and software repositories.

- **Version numbers**: Ideally, you should keep track of the version numbers for everything you use, if possible. This is particularly important for software and libraries because often certain versions of software do not work with other versions of software, so a mismatch in version numbers may prevent another person from reproducible your work. Communicating the appropriate version numbers to others can improve the chances of them reproducing what you've done.

One important function in R that can be useful for documenting your R environment is the `sessionInfo()` function. This function displays various details about your R environment like the search path, which packages are loaded, the version number of R, the locale, and the operating system of your computer. For example, here's what it outputs for my environment.

```
> sessionInfo()


R version 3.2.2 RC (2015-08-08 r68921)
Platform: x86_64-apple-darwin14.4.0 (64-bit)
Running under: OS X 10.10.4 (Yosemite)

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_\
US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  bas\
e

other attached packages:
```

```
[1] knitr_1.10.5

loaded via a namespace (and not attached):
[1] magrittr_1.5  formatR_1.2   tools_3.2.2   stringi_0.5\
-5 stringr_1.0.0
[6] evaluate_0.7
```

Including a call to `sessionInfo()` at the end of each report written in R (perhaps with markdown or knitr) can be useful for communicating to the reader what type of environment is needed to reproduce the contents of the report (it may not be necessary but it's likely sufficient for simple analyses).

Don't Save Output

Saving output from various stages in a data analysis may seem like a responsible thing to do (what if the computer crashes), but it should be avoided if possible. The reason is that output files are often undocumented and the manner in which they were constructed can be difficult to reproduce. Better to save the inputs and code that were used to create a given piece of output rather than save the output itself. That way, if changes need to be made (or if output is lost), you can simply re-run the code with the appropriate input.

Outputs that you should avoid saving are things like tables, figures, summaries, and processed data. The one exception here is if it took a very long time to create that output. Then it might make sense to *temporarily* save some output for efficiency purposes. But in those cases, it's important to document carefully how the output was generated, perhaps via a version control system. Ultimately, if an output file

cannot be easily connected with the means by which it was created, then it is not reproducible.

Set Your Seed

This is a niche issue that may not be generally applicable, but is often the source of non-reproducible results in statistical output or simulations. Many sophisticated statistical routines these days depend on the generation of random numbers. Think Markov chain Monte Carlo, random forests, and bootstrapping. Any procedure that depends on randomness will not generate the exact same output if you run it twice (the very definition of non-reproducibility). However, on a computer, random numbers are not truly random, rather they are pseudo-random. Therefore, it is possible to exactly reconstruct a sequence of pseudo-random numbers if you have the initial *seed*.

In R you can use the `set.seed()` function to set the random number generator seed and to specify which random number generator to use (see `?set.seed` for details). Setting the seed allows for the stream of random numbers to be exactly reproducible at a later date. Whenever you generate random numbers for a non-trivial purpose, **always set the seed** at the beginning of your code.

Here's an example of some random numbers.

```
> rnorm(5)
```

```
[1]  2.22414093  0.09524444 -1.16593756  0.59730725  1.34\
369099
```

There is now no way for me to go back (via code) and re-generate those numbers because I didn't set the seed. The next time I call `rnorm()` it will generate different numbers.

```
> rnorm(5)
```

```
[1] -1.9432379  0.6078967  1.8811491 -1.0447159  0.3690495
```

However, if I set the seed first, I can always re-generate the same numbers if needed.

```
> set.seed(10)
> rnorm(5)
```

```
[1]  0.01874617 -0.18425254 -1.37133055 -0.59916772  0.29\
454513
```

And again.

```
> set.seed(10)
> rnorm(5)
```

```
[1]  0.01874617 -0.18425254 -1.37133055 -0.59916772  0.29\
454513
```

Think About the Entire Pipeline

Data analysis is a lengthy process, starting from obtaining data all the way to generating results and communicating

output. It is not just fitting a few prediction models or creating tables, figures, and reports. Typically, there will be raw data, processed or analytic data, analysis results, and then a final report. In addition to all that there will be code files that correspond to each of those transitions. They key thing to remember is that *how you got the end is just as important as the end itself*. The more of the *entire* data analysis pipeline you can make reproducible, the better for everyone (most importantly, yourself).

Summary

Here is the basic reproducibility check list:

- Are we doing good science?
- Was any part of this analysis done by hand?If so, are those parts *precisely* document? Does the documentation match reality?
- Have we taught a computer to do as much as possible (i.e. coded)?
- Are we using a version control system?
- Have we documented our software environment?
- Have we saved any output that we cannot reconstruct from original data + code?
- How far back in the analysis pipeline can we go before our results are no longer (automatically) reproducible?