Teaching reproducibility: motivation, direct instruction and practice

Joel Ostblom *Tiffany A. Timbers*

Introduction

The data science definition that we embrace in the master of data science program at UBC and in the undergrad data science courses that we're developing there is the study and development of reproducible and audible processes to obtain insight from data. When you go into the data science classroom students are usually very excited about learning data science but what they're most excited about is the second part of the definition, the insight from data part.

Often they are not even aware about the reproducible and audible processes part and they see that more as a pain/inconvenience. So you have this barrier when you're teaching the reproducibility aspects of data science. This probably arises because they likely do not even know what reproducibility is, and even if they do know about it, it is not the thing that is obviously/directly providing insight and so they're not excited about it.

Then we have this third challenge, which is that the tools that we use for reproducibility are not necessarily smooth and easy to learn. They usually have a pretty steep learning curve. Over our five years of teaching these things at UBC we've found some key things that we've experienced at least for teaching reproducibility successfully:

1. placing extra emphasis on motivation
2. direct instruction
3. lots of practice

In this paper, we will discuss why we believe each of these are important, give some high-level examples of how we do these, and then we give one detailed example of how we do each in our courses.

Placing extra emphasis on motivation

Why do we need extra motivation when teaching reproducibility, compared to some other data science topics, such as machine learning? We think this is because students do not have intrinsic excitement or motivation on the topic of reproducibility, they have little prior knowledge on this topic, and reproducibility concepts and in particular tools are challenging to learn.

One example, the version control software that's the most commonly used one for reproducibility, Git, is notorious for being difficult to learn (Figure 1). Furthermore, there are many anecdotes that that most people don't actually learn it deeply and they just get by trying a variety command until they find some things that work. Which can lead to users getting themselves into a lot of trouble. Sometimes this trouble is so difficult to get out of, that some professional data scientists and data science educators, for example Jenny Bryan, recommed the practice of "burning it all down" and starting from scratch - which really defeats many of the purposes of version control.

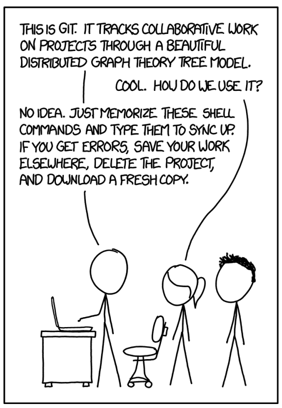

Fig. 1: Infamous xkcd comic that highlights the difficulty of learning and using the version control software Git.

Another example is R Markdown, which is an implemenation of literate code documents (cite Knuth here) that are useful for generating reproducible reports. Many aspects of R Markdown are quite user friendly, however rendering the source R Markdown document to PDF depends on LaTeX, and when users make formatting errors that impact LaTeX's job in the rendering you can get some very cryptic error messages that are not very clear to learners about where the error is coming from and what should be done next to fix the problem.

```
# Add a figure here that illustrates an example of an R Markdown doc being
# rendered to PDF that has an error, and then the cryptic error message.
# This could simply be a screen shot from RStudio with the R Markdown doc
# open in the editor and the R Markdown tab showing the error
```

Yet another example of a reproducibility tool that is challenging to learn is Docker. Docker is a containerization tool useful for creating and sharing computational environments. ADD SENTENCE HERE WHY COMPUTATIONAL ENV's ARE IMPT FOR REPRODUCIBILITY. This tool is an especially challenging one to teach and motivate students to learn because it's so different from writing code to for analysis. This is because it takes a long time to install things and it takes a long time to automate the process of installing things and making a shareable compute environment. It is also not an exciting part of a data analysis, things already work on the students own computer. So it is difficult to convince students why they should put effort into learning and using this.

```
# Need to figure out a way to illustrate this...
```

So what do we do to motivate learning reproducibility concepts and tools in our classrooms? We have found the following three strategies helpful:

1. Telling stories from the trenches
2. Letting them fail (in a controlled manner)
3. Study cases of failures with real world consequences

Telling stories from the trenches

One thing that we've used is telling stories from the trenches. Those of us who teach these courses usually have had some experience with doing research in their Ph.D.'s or Postdoctoral studies, or are still currently doing research, and through these are lived experiences of learning reproducibility tools and applying it to our research we have made mistakes, our collaborators have made mistakes and we can share these with our students. In the Master of Data Science program, a lot of the students have work experience that may have touched on data before and so they also have stories from the trenches. In a classroom with such students, you can carryout think pair share exercises around these stories and get the students to talk about their stories as well as hear your stories.

Letting them fail (in a controlled manner)

Another example that I learned from Jenny Bryan's teaching of STAT 545 at UBC is to ask them to run someone else's analysis and let them fail. We have taken this approach, but tweaked it so that they experience this in a more limited, or controlled manner. Below we present this example in more detail, but in short though, we provide students an analysis that is not reproducible, and thus will likely fail on someone else's machine, and then ask them to run it, and if they cannot run it, fix it so they can run it. Then we provide them the same analysis that has been made reproducible. Activities like this, in a relatively short time with minimally painful experiences clearly illustrate to students the benefit of reproducibility, which helps provide motivation to endure the steep learning curves of reproducibility concepts and tools.

The authors of this work have experienced a lot of failure in graduate school and during their postdoctoral research in regards to reproducibility. They admit these failures slowed down down their research. This has provided them much motivation to teach and use reproducibility concepts and tools. However, most undergraduates, and new graduate students have not experienced this (and we question whether they should to its full extent in the next section). So if we can set up these scenarios where they feel a little bit of this pain but only for a short period of time that can be very useful for creating motivation.

Study cases of failures with real world consequences

The third way to create motivation is something that the authors are currently trying to build into their courses and yet have limited experience test driving it, but think the idea has strong merit and so wish to present it here. This is the pedagogy of using case studies of failures that have had real world consequences.

WE SHOULD FLESH THIS OUT HERE AND FIND 3-5 CASE STUDIES TO CITE THAT WE PLAN TO USE NEXT YEAR.

Example lesson of letting them fail (in a controlled manner)

An example we have used in the classroom is letting students fail in a controlled manner. The classroom that this has been used in is DSCI 522 - Data Science Workflows that is 2/3 of the way into a professional Master's of Data Science program. This course is a project-based course where we teach them all the concepts and skills needed to do a reproducible data analysis. Near the end of the course one of the things that we get them to do is to make their compute environments for their analysis reproducible via containerization of the compute environment using a tool called Docker.

Docker is not an easy thing to teach or learn. There is a lot of overhead you're teaching that we need to teach them to

write docker files you know writing how

you install things in linux and half of

the students are windows users so like

there is a lot of you know

barrier and stickiness to teaching this

subject so you really need to motivate

it

so one way that i do this in a

demonstration is that we give students a

data analysis

project pipeline on github and we asked

them

go to go to that github repository read

the instructions

and try and replicate the analysis our
students already have some git skills at
this point in the course
so they're able to do the git clone
they're reading the readme trying to
follow along
uh with the instructions to run it um in
in the first instance
uh we've given them an analysis and
we've like intentionally put in a lot of
packages that we know they don't have
installed
like we've got off into
into crayon or into pipey eye and like
found some very bizarre packages we
don't even necessarily use them but you
just tuck them in there
um and if you want to be really you know
uh
sneaky you tuck them in there like in
the middle of the script so they're not
even at the top of the scripts
and you ask them to to work on getting
that analysis to run
and it takes some time and eventually
they figure it out but it's
it's frustrating and then you give them
the same analysis
in a different github repository um but
you give that it has a
it has a docker solution so there's a

docker image that exists on docker hub
um the readme gives clear instructions
on how to run it and replicate the
analysis
and they're able to do this and they're
able to replicate the analysis in a
couple minutes
so this is an example of how you can a
let them fail in a controlled manner
and then at the same time within the
same learning
time period give them a solution and
motivate them to that solution and then
they're in a good mind frame that
even when learning soccer beca is hard
and challenging
they have the motivation to learn it
because that they know that their
analysis is going to be more useful for
other people
afterwards

Direct instruction

okay so the second thing that i said was
important is direct instruction
so why is direct instruction important
well from um
i think for those of us who've been
using reproducibility
tools in our research what i'm going to
say here is probably

not new to you but reproducibility
is not something that most people or
students figure out through exploration
and
inquiry um based learning or if you do
it's not an efficient
way of doing it um there's a lot of
you know we're using a lot of borrowed
tools from software engineering that are
being repurposed
for um for for science and
reproducibility
and so a lot of the uh you know getting
up and getting started has a lot of
assumed knowledge behind it
and um there's not a lot of like clear
and easy on roads to
these things and because um
yeah i think and there's also
how would i say this uh just because
it's still fairly new i would say
that there's there's not um
a lot of culture around it that's like
that's very common uh to like show
people where these obvious armor apps
are
so i and then there's the challenge of
the tools that i talked about in the
previous example so i think again
having some direct instruction is is
important and i really love this excerpt

from roger peng's blog post
from a couple of years ago that he wrote
on the theory of data analysis
and he writes here that there is no need
for a new data analyst to learn
about reproducibility from experience we
don't need to lead a junior data
analysis
down a months long winding path of
non-reproducible analysis
until they are finally bitten by the
non-reproducibility
bug and therefore learn their lesson we
can just tell them
hey in the past we found it useful to
make our data analysis reproducible
here's a workflow guide for you to use
in your own analysis
within that one statement we can
compress with over 20 years of
experience
we i think owe it to our students to
directly instruct them with like
the best practices that you know that
the reproducibility community has
arrived on
to date and then show them how to use
these tools explicitly
so how do we use direct exam instruction
in the classroom
um so we do a lot of live demos so in

the programming classes we do
you know live coding to show how to use
r and python um but then when we're
talking about other tools like docker
uh our markdown or jupyter for doing
reproducible reports
using version control with git and
github we do a lot of live demos with
those two
tools as well and in doing that it makes
it i think
uh obvious to this
more obvious to the students of like how
to use these things number one
and number two um you make mistakes
and that humanizes the experience of
of working with these tools that are
somewhat challenging because students
make mistakes too and they see that the
experts are also making mistakes
and then you're able to usually make
those mistakes usually make your
mistakes where things are a little bit
more difficult or are a little bit more
sticky
and it gives you more time to spend on
that area of the topic
and explain why you made the mistake and
where the misconceptions come in and
and fix them the other thing that we use
are guided worksheets and tutorials so

there's a lot us giving the live demos
which is useful but you can't you know
for
uh all the time be up in front of people
and it's good for
people to um work through and
actively engage with material out
themselves and get a little bit of uh
practice but not you know enough totally
like free-for-all do your own thing
whatever you think is best but in a
guided way so um we have a lot of those
in the program
um but one thing i want to say is be
careful when you uh
just there are some dragons when you
when you teach this stuff
so um because we're teaching things that
involve graphical user interfaces
because we teach things that are coming
from
software engineering it's a very
fast-moving field and which means that
every time i teach this stuff
i need to go through it before i send it
out to the students because something
has changed and something or something
has broken
and i have to come up with a work around
so a story from this past term
is that github which is

you know the largest code hosting
repository in the world
uh decided to i think very rightfully so
changed the name of their default branch
they switched it from master to main
but that caused lots of things to break
that caused all of our notes to have to
um be rewritten um and it's still like
a half solved problem because we have a
whole bunch of resources that are still
sitting on master branch that we haven't
been able to or will not be able to
uh quickly change over so
um it is really important to have this
direct instruction
but these are things that you're going
to have to kind of do new in the
classroom every year make a new demo
every year
curate your gut worksheets and your
tutorials every year because otherwise
they're going to quickly
fall out of usefulness an example of
direct instruction
um for teaching version control so we
teach version control in our very first
year introduction to data science
uh course um and so we do this in
uh kind of like a three-pronged approach
for directed instruction
so we give them a textbook reading that

they're

able to to use this is something we have

to update every year because the

graphical user interface that we choose

to use

changes we don't teach the command line

for this in the first year because it's

a bit too

overwhelming i think for the students um

then we do a live demonstration where

they

they watch us use the github website

they watch us use the get gui they watch

us move files and

add and commit and push and pull and

then finally they work through a guided

worksheet

that asks them to do the same thing that

we just did

and then ask them questions along the

way uh to test that they

like really understand like what is

committing what is adding what is

pulling what is pushing where is the

work going

and if folks are interested i have uh

some links embedded in this talk

that will take you to some of the

examples or resources that i'm talking

about here

okay and so the final thing that i said

um

Lot's of practice

is lots of practice so why do we need

lots of practice

for reproducibility for learning

reproducibility workflows and

tools well there are really two

fundamental ways that we commit things

to long-term memory

one is one trial learning and that

usually requires some sort of emotional

impact so that's like sometimes it's

traumatic events and sometimes it's

really good positive emotional events

that you had like a really

great birthday or your wedding or

something like that you don't need those

things to be repeated multiple times so

that you remember them

but that's not most of the things in

school most of the things in school

we learn about are through this

repetitive space training

um and so uh the the you know the best

way to commit

uh something to long-term memory that's

not really emotional

is to revisit it and repeat it multiple

times and have breaks between

those things and so that lets you commit

it to memory
however i think you want to go even a
step further with reproducibility
because when we teach reproducibility
workflows and skills as instructors
we actually want students to do more
than just learn about these
things we actually want them to use them
and put them into practice
um in the classroom outside of the
classroom in their work after the
classroom
and so we actually want to change their
habits or behaviors and it's it's quite
i think
important to realize that okay it's not
just understanding an algorithm
it's it's understanding the concepts
behind something like version control
understanding the concepts behind
something like a shippable and shareable
compute environment
and then knowing how to use those things
and then
once you leave the classroom wanting and
being able to use those things without
like
saying no that's too hard or too tricky
you want them to just
do it out of habit because that's what
they usually do

um so an aside uh just a little
if people are interested a book that's
really recently made me think about more
how we can tangibly do this is called
atomic habits by james clear
he's done a really good job of like
bringing the science of
habit building and behavior change uh
into an accessible book
and um i think that when we think about
getting students practice and changing
their behavior with
reproducible skills and workflows
there's a lot of really
interesting insight from behavior change
and psychology and habit building that
we can that we can borrow
so um now i'll talk more practically
about so
at least right now what are we doing in
the classroom to embed
lots of practice so
what we do is when we do our live demos
we don't just have
us do it then we pause and say okay
students your turn
do what i just did and so they saw
it and then they actually have to type
it into the keyboard or click it their
mouse around
the graphical user interface so they

they practice it that
way then we have lots of low stakes
assessments
with small or short problems so um
we've moved into a lot of flipped
classroom
um in in at ubc or at least in the data
science so our introduction to data
science course is a primarily flipped
classroom so um we have uh
literate code documents that have uh
automated tests in them
that the students are answering all
kinds of questions about the data
science content
and then they're very short little
pieces that are well guided but they get
immediate feedback um and these things
aren't worth very much and they do a lot
of them so they do two of them a week in
the data science course
in the master of data science um program
we've also started implementing this in
some of our classrooms
and the students really like this
practice and it helps them really
prepare for things like larger
assessments like quizzes and
and their their lab homework but it
gives them lots of practice
and then the other thing that we do is

the learning technologies and platforms
that we use
are built and use authentic data science
reproducibility tools and so i'll give
an example of that now
so in almost all of the master beta
science courses so i'm talking about 20
courses here
21 credit courses
so 21 month long courses we use
version control particularly github as
our course management system
so the homework instructions and
assignments are distributed to the
students as github repositories
and the only way that they can submit
their homework is by putting their
homework in that github repository
so they have to go through the cloning
procedure or at least be able to somehow
download this from the
github uh website and then they have to
be able to hopefully through
things like pushing and committing send
their work back to github but they would
at least have to interface with the
github
uh website to do this um
to try and uh incentivize
um the actual actually using the get
machinery to interact with github

we also put part of the marks of each
of these assignments as to mechanics and
so um
we need to see for example like three
commits associated with every single
assignment because
um we think you know we're trying to
build these good habits and practices
around like there's reasons why you use
version control not just to submit your
homework but
to active as a backup or in case you
want to go back in time and change
things
so by the end of this program um the
students have version controlled their
work in over 80 different repositories
um so they have a lot of so they're very
practiced and very used to it
and they're basically you know you want
them to be able to do it in their sleep
um almost and so that when they leave
the program and they go to work
somewhere else it's just
natural it's just one of their habits at
this point that if they're going to work
on a project it's going to go under
version control
so we do this using tools and here i've
listened there's many tools now which is
pretty cool we're not the only program

doing this

at all there's many tools now for using

github

as a classroom learning management

system um and so i've listed a couple of

them here

folks are interested so

Conclusions

We think over the past five years from teaching in the Master of Data Science and our Introduction to Data Science course that key things for teaching reproducibility in the data science classroom are providing extra emphasis on motivation, providing direct instruction so it's not a mystery of how you get started and what you need to do, as well as lots and lots of practice so that we can not only teach them the material and the concepts but so that this actually changes their practices and their workflows and they will use it after leaving your classroom.

Q & A's from talk

(leaving here for now in case there is anything inspirational to add to the paper)

so i'm happy to take questions now here

um

or uh you can tweet to me on twitter and

i'm happy to answer there and again i've

posted the link for the slides

thanks very much tiffany amy did you

want to share the question period or

shall i um i

can i'm just gonna check the thread

i don't see anything right now

anyone have anything to start off with

i have some if there's none um otherwise

maybe mina has

comments i'd be i'd love just to have

mina and tiffany just like
tell us everything you know between the
two of you
was there something that you started
doing that you i mean obviously
it's a evolution right the these these
all of these
programs are just in the evolution stage
uh was this uh
something that you started doing that
you've really moved away from
yeah i would say the pandemics even
placed a greater emphasis on this
so when we started teaching the master
of data science program
um we had a small cohort and we were in
person which allowed us to provide a lot
of support
but as you scale these things um
having that intimate close support is
more difficult and so in the very first
year of the program
um in the mbs program we have like very
we we have this philosophy that they
should be able to
be somewhat experts of running stuff on
their laptop we do teach them some cloud
tools but
you should be able to install your
software stack and and be able to set up
your path and these sorts of things

and so um we have a pretty i'd say like
intensive list of like 20 things that
they need to install in the first week
they need to use git to submit their
homework in the first week
and it's a bit overwhelming and it's a
lot and so
um what we've kind of moved we've kind
of like eased off on that
and and moved um to we get them there
but we take long
we take longer now to doing that so uh
we've set up this year a jupiter hub to
have them work in a cloud-based setting
for the first week or two
and then after the first week or two is
when we transition them to their own
laptop
so that we give time for the no like the
the
expectation that setting up everybody's
system
um is going to encounter some bugs and
take some time and that's going to be
tricky
we've also for the first assignment we
no longer ask them in the first
assignment to
submit to github um that's assignment
two so that we have to give ourselves
like a week or two

to to get them up to speed for getting
github
um and so i think that uh that sort of
thing
has um has definitely changed and been
inspired with
so first i started working in the master
data science program and then i started
teaching undergraduates
and teaching undergrad graduates has
made me have to reframe things and think
about things differently
um and think about like how do i remove
barriers so that
people you know maybe people who who are
for whatever reason more sensitive to
not feeling like data sciences for
them i don't want them to drop my class
because they couldn't install something
um so i think yeah that's something
that's changed a lot
let's quickly get to meena's question
where she needs to go um
a question about have you seen any
changes in computing experiences of
students applying to your
ms program um are more students coming
in with familiarity with these tools or
not yet
yeah yeah that's a great question i do a
little survey

every year about like in in the first
class
i'm like what tools have you used before
and usually about half of the students
have used r
maybe three quarters of the students
have used python almost all of them have
seen jupiter
almost none of them still have seen get
in github um so it's really quite
amazing
that um
computer science programming
prerequisite uh so they have
it's they don't have to have had our pro
or python before
um but yeah it's still interesting that
even
though i'll comment in a second that we
are seeing people with more
technical or data science skills coming
to our program
it's still the reproducibility
experience with reproducibility related
tools are
are aside from jupiter like um not
as present as one might expect i am
seeing more and more
uh people having like in data science
applying to our program which is
something kind of new and interesting

for us to think about because our
program was really designed
not for somebody who is like a data
science undergraduate like somebody who
had
an undergraduate in a different
discipline and wants to then apply
you know data science to their
discipline so we're still thinking about
like how
how we're going to handle the change of
like there's going to be more and more
undergraduates coming in with this
expertise
yeah it's a super interesting problem um
john's asking
um what to do with docker and windows
um is there something special about
documents yeah
so it uh it can work
um but everything with windows is a
little bit more challenging
uh so what my strategy is is i have
um i i'm a mac person uh but i have a
i also have a separate pc laptop where i
have linux and windows installed
so that before i teach every course i go
through and make sure that
i know how to install things on windows
and
what instructions to provide to students

there's still always surprises
um one thing we do this on quite a large
scale largest scale with the master
students but i'd say like
we're dealing with 50 or 60 windows
different windows laptops every
every year and so to make our lives
easier
um we've been really tightly restricting
which version of windows
that they have because uh then it's
easier to know so we say you have to
have windows 10
you have to have this build um and it
can't be windows home um basically it
has to be enterprise pro
or or education and by doing that that
has reduced some problems but every year
something new comes up like
i can just tell you this week i'm
teaching uh python packaging with poetry
and git bash doesn't work with poetry
anymore this year it worked last year
but it doesn't work anymore there's a
game
that have issue open it's not resolved
so now we're using anaconda prompt on
the windows machines
we have a solution um but it's
it's just it's it's one of those there
there will be dragons in this field

yeah it's yeah keeping changing things
it's just so much work right
you think you're done and at least 20
years ago right the folks they write out
their theory equations and that was it
they were done for the next 20 years
we've got to update ourselves every six
months
oh did you have any other closing
comments or thoughts that you wanted to
say