Main takeaways:

Coding etiquette makes things easier for everyone! (Collaborators and future self. Nobody wants migraines from staring at poor code.)

What is version control?

- Helps keep track of work and helps easily explore the changes you have made, be it data, coding scripts etc
- Example of version control software is Git. Online platform like GitHub stores files, meaning you have online back up of your work, beneficial for you and collaborators.

Making code easy to reproduce:

- **Packages** should be loaded at the top of the script, so it's easy to see which ones the example needs.

easiest way to include **data** in a question is to use `dput()` to generate the R code to recreate it. For example, to recreate the `mtcars` dataset in R, I'd perform the following steps:

1. Run `dput(mtcars)` in R
2. Copy the output
3. In my reproducible script, type `mtcars <-` then paste.

- Make code easy to read (Etiquette! below)

Etiquette Naming stuff:

- Variable names should be nouns. e.g. abundance richness
- Function names should be verbs. e.g. calc.sp.richness
- Use an underscore to separate words within a script file. e.g. LPI_analysis_Apr_2017.R
- The preferred form for object/variable names is **all lower case letters** and words separated with underscores e.g. (object_name$variable_name).
- For functions, all lower case letters and words separated by dots e.g. (function.name).
- calculate_avg_clicks , calculateAvgClicks *# Bad. The convention is that functions are defined using **dots**, not underscores.*
- Don't place a space before left parentheses, except in a function call.

```
E. g

# Good

if (debug) do(x)

plot(x, y)
```

```
# Bad

if(debug)do(x)

plot (x, y)
```

- **Do not place spaces around code in parentheses or square brackets (unless there's a comma, in which case see above).**

```
# Good

if (debug) do(x)

diamonds[5, ]



# Bad

if ( debug ) do(x)  # No spaces around debug

x[1,]   # Needs a space after the comma

x[1 ,]  # Space goes after comma not before
```

- An opening curly brace should never go on its own line and should always be followed by a new line. **A closing curly brace should always go on its own line, unless it's followed by "*else*". Always indent the code inside curly braces.**
- When using pipes from dplyr package, **keep the pipe operator %>% at the end of the line** and continue your pipe on a new line.

  -pipes are a way to chain 2 functions together

- When using ggplot2, **keep the + at the end of the line** and continue adding on layers on a new line.

A note on BIG DATA!:

- If your data is big, consider if your big data problem might actually be a small data problem in disguise. While the complete data might be big, the data needed to ans a specific question is small.
- Another possibility is that big data problem is actually a large number of small data problems. For example, you might want to fit a model to each person in your dataset. That would be trivial if you had just 10 or 100 people, but instead you have a million. Fortunately each problem is independent of the others (a setup that is sometimes called embarrassingly parallel), so you just need a system (like Hadoop or Spark) that allows you to send different datasets to different computers for processing. Once you've figured out how to answer the question for a single subset using the tools described in this book, you learn new tools like sparklyr, rhipe, and ddr to solve it for the full dataset.

Rectangular data?

- I.e collections of values that are each associated with a variable and an observation. There are lots of datasets that do not naturally fit in this paradigm, including images, sounds, trees, and text.
  Hypothesis generation is hypothesis confirmation: Tricky because…
  1. Need a precise math model in order to generate falsifiable predictions. Requires stats sophistication
  2. Can only use an observation once to confirm a hypothesis. As soon as you observe, back to exploratory analysis. To do hyp confirmation you need to "pregister" (i.e write out in advance) your analysis plan, NOT deviate from it when you have seen the data.
     What is hypothesis confirmation?
     - modelling as a tool for hypothesis confirmation vs visualisation as a tool for hypothesis generation. **But...** that's a **false dichotomy**: models are often used for exploration, and with a little care you can use visualisation for confirmation. The **key difference is how often do you look at each observation**: **if you look only once, it's confirmation; if you look more than once, it's exploration.**

Terms revision:

- Functions are in a code font and followed by parentheses, like `sum()`, or `mean()`
- Other R objects (like data or function arguments) are in a code font, without parentheses, like `flights` or `x`
- to make it clear what package an object comes from, we'll use the package name followed by two colons, like `dplyr::mutate()`, or
- `nycflights13::flights`. This is also valid R code.

Objected-oriented programming: programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).
A feature of objects is that an object's own procedures can access and often modify the data fields of itself (objects have a notion of this or self). In OOP, computer programs are designed by making them out of objects that interact with one another

- include: Java, C++, C#, Python, R, PHP, Visual Basic.NET, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB, and Smalltalk