Lesson 5 - The dark art of debugging code

MLE Software Online Course

Click here to view presentation online

Christopher Cahill
Associate Director
14 December 2023

Troubleshooting code as a skill that can be taught, learned, and mastered

Outline

- The origins of the bug 🐛 🐛 🐛

Outline

- The origins of the bug 🐛 🐛 🐛
- A general workflow for debugging code

Outline

- The origins of the bug 🐛 🐛 🐛
- A general workflow for debugging code
- Specifically we want to think about how we might:

Outline

- The origins of the bug 🐛 🐛 🐛
- A general workflow for debugging code
- Specifically we want to think about how we might:
  - Write fewer bugs
  - Reproduce bugs
  - Locate bugs
  - Fix bugs

Outline

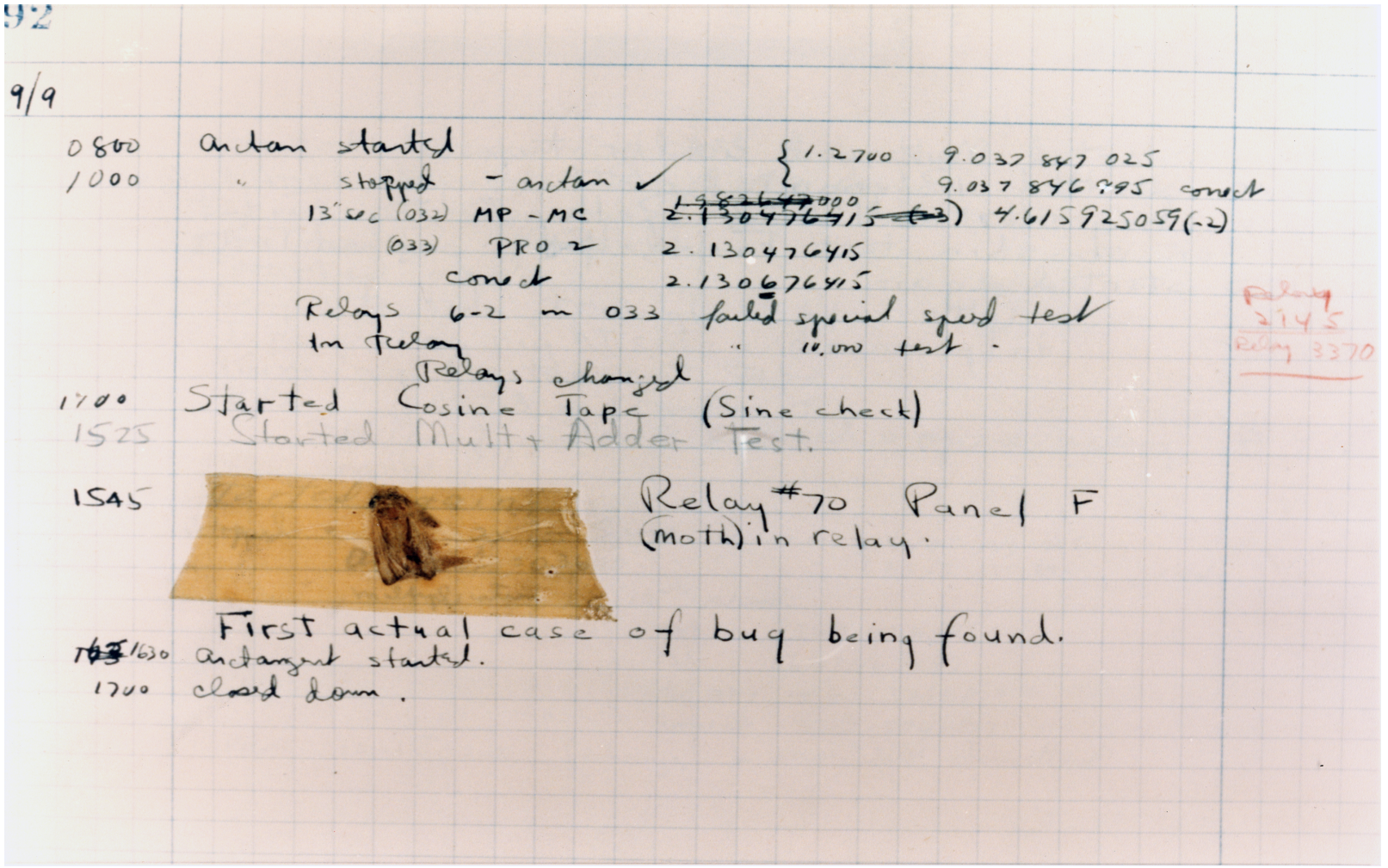
- The origins of the bug 🐛 🐛 🐛
- A general workflow for debugging code
- Specifically we want to think about how we might:
  - Write fewer bugs
  - Reproduce bugs
  - Locate bugs
  - Fix bugs
- Introduce a few useful tools that help with this workflow
  - `styler` library and `browser()`

Outline

- The origins of the bug 🐛 🐛 🐛
- A general workflow for debugging code
- Specifically we want to think about how we might:
  - Write fewer bugs
  - Reproduce bugs
  - Locate bugs
  - Fix bugs
- Introduce a few useful tools that help with this workflow
  - `styler` library and `browser()`
- Work through some of examples

The original sin (bug)

The first bug 🐛



Grace Hopper and the origins of bugs and debugging

Step 1: Write fewer bugs

Follow a consistent style

Why coding style matters

Link to the Tidyverse style guide

Follow a consistent style

- Good coding style is like correct punctuation: you can manage without it, butitsureMakestHingseaSiertoreAdWHYgodWHY

Why coding style matters

Link to the Tidyverse style guide

Follow a consistent style

- Good coding style is like correct punctuation: you can manage without it, butitsureMakestHingseaSiertoreAdWHYgodWHY
- The importance of adopting a consistent style cannot be stressed enough

Why coding style matters

Link to the Tidyverse style guide

Follow a consistent style

- Good coding style is like correct punctuation: you can manage without it, butitsureMakestHingseaSiertoreAdWHYgodWHY
- The importance of adopting a consistent style cannot be stressed enough
- QFC trying to follow the tidyverse style guide

Why coding style matters

Link to the Tidyverse style guide

Follow a consistent style

- Good coding style is like correct punctuation: you can manage without it, butitsureMakestHingseaSiertoreAdWHYgodWHY
- The importance of adopting a consistent style cannot be stressed enough
- QFC trying to follow the tidyverse style guide
  - Most things lower case and words separated by underscores

Why coding style matters

Link to the Tidyverse style guide

Follow a consistent style

- Good coding style is like correct punctuation: you can manage without it, butitsureMakestHingseaSiertoreAdWHYgodWHY
- The importance of adopting a consistent style cannot be stressed enough
- QFC trying to follow the tidyverse style guide
  - Most things lower case and words separated by underscores
  - Implemented via the Styler library

Why coding style matters

Link to the Tidyverse style guide

Follow a consistent style

- Good coding style is like correct punctuation: you can manage without it, butitsureMakestHingseaSiertoreAdWHYgodWHY
- The importance of adopting a consistent style cannot be stressed enough
- QFC trying to follow the tidyverse style guide
  - Most things lower case and words separated by underscores
  - Implemented via the Styler library
- Why care about style?

Why coding style matters

Link to the Tidyverse style guide

Follow a consistent style

- Good coding style is like correct punctuation: you can manage without it, butitsureMakestHingseaSiertoreAdWHYgodWHY
- The importance of adopting a consistent style cannot be stressed enough
- QFC trying to follow the tidyverse style guide
  - Most things lower case and words separated by underscores
  - Implemented via the Styler library
- Why care about style?
  - Consistent naming conventions, indendation, and spacing help us train our eyes to spot bugs

Why coding style matters

Link to the Tidyverse style guide

An example program that you get from a collaborator

```
f = function(pars) {
  getAll(data, pars);
  Linfmn = exp(logLinfmn); logLinfsd = exp(loglogLinfsd);
  Linfs = exp(logLinfs); K = exp(logK );
         Sig = exp(logSig);
  nponds = length(Linfs);
   nages = length(A);
  predL = matrix(0, nrow = nages, ncol = nponds);
      # fill one column (pond) at a time:
  for (i in 1:nponds) { predL[, i] = Linfs[i] * (1 - exp(-K * (A - t0)));}
  nll = -sum(dnorm(x = L, mean = predL, sd = Sig, log = TRUE));
   nprand = -sum(dnorm(x = logLinfs, mean = logLinfmn, sd = logLinfsd, log =
  jnll =  nll +  nprand;
  jnll;
}
```

An example program

- Make it pretty via `styler()`

```
f <- function(pars) {
  getAll(data, pars)
  Linfmn <- exp(logLinfmn)
  logLinfsd <- exp(loglogLinfsd)
  Linfs <- exp(logLinfs)
  K <- exp(logK)
  Sig <- exp(logSig)
  nponds <- length(Linfs)
  nages <- length(A)
  predL <- matrix(0, nrow = nages, ncol = nponds)
  # fill one column (pond) at a time:
  for (i in 1:nponds) {
    predL[, i] <- Linfs[i] * (1 - exp(-K * (A - t0)))
  }
  nll <- -sum(dnorm(x = L, mean = predL, sd = Sig, log = TRUE))
  nprand <- -sum(dnorm(x = logLinfs, mean = logLinfmn, sd = logLinfsd, log =
  jnll <- nll + nprand
  jnll
}
```

Step 2: Reproduce the bug

Reproduce the bug

- Reproducing a bug allows us to better understand why the program went wrong

“Debugging is like being the detective in a crime movie where you are also the murderer.” - Filipe Fortes

Reproduce the bug

- Reproducing a bug allows us to better understand why the program went wrong
- Pay attention to versions of R and packages

"Debugging is like being the detective in a crime movie where you are also the murderer." - Filipe Fortes

Reproduce the bug

- Reproducing a bug allows us to better understand why the program went wrong
- Pay attention to versions of R and packages
- Pay attention to warnings, errors, and other messages

"Debugging is like being the detective in a crime movie where you are also the murderer." - Filipe Fortes

Reproduce the bug

- Reproducing a bug allows us to better understand why the program went wrong
- Pay attention to versions of R and packages
- Pay attention to warnings, errors, and other messages
- Emphasis here on pay attention

"Debugging is like being the detective in a crime movie where you are also the murderer." - Filipe Fortes

Reproduce the bug

- Reproducing a bug allows us to better understand why the program went wrong
- Pay attention to versions of R and packages
- Pay attention to warnings, errors, and other messages
- Emphasis here on pay attention
- ISOLATE THE BUG

"Debugging is like being the detective in a crime movie where you are also the murderer." - Filipe Fortes

Determining versions in R

```
1 packageVersion("RTMB")
```

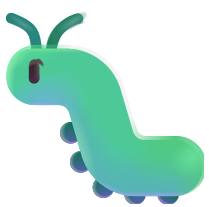
```
[1] '1.3'
```

Determining versions in R

```
1 sessionInfo()
```

```
R version 4.2.2 (2022-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22621)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.2.2  fastmap_1.1.1   cli_3.6.1       tools_4.2.2
 [5] htmltools_0.5.5 rstudioapi_0.14 yaml_2.3.7      rmarkdown_2.23
 [9] knitr_1.43      jsonlite_1.8.5  xfun_0.39       digest_0.6.31
[13] rlang_1.1.1     evaluate_0.21
```

Step 3: Locate the bug



Manual debugging

Wikipedia link to Shotgun Debugging

Manual debugging

- What most people are used to, links back to Jim's bit about understanding the math

Wikipedia link to Shotgun Debugging

Manual debugging

- What most people are used to, links back to Jim's bit about understanding the math
- There are smart and effective ways to do this

Wikipedia link to Shotgun Debugging

Manual debugging

- What most people are used to, links back to Jim's bit about understanding the math
- There are smart and effective ways to do this
- There are also stupid ways to do this

Wikipedia link to Shotgun Debugging

Formal debugging tools in R

- There are several, but I am going to teach you `browser()`

Useful article on debugging tools in RStudio

Formal debugging tools in R

- There are several, but I am going to teach you `browser()`
- `browser()` gives us a way to interrupt the execution of an expression and allow the inspection of the environment

Useful article on debugging tools in RStudio

Formal debugging tools in R

- There are several, but I am going to teach you `browser()`
- `browser()` gives us a way to interrupt the execution of an expression and allow the inspection of the environment
  - When I say environment here, think local environment inside a function

Useful article on debugging tools in RStudio

The downside of `browser()`

- Have to manually add it and then remove it from your code later on

Step 4: Fix the bug

Exorcise the bug (demon)

Image reference

My strategy for debugging code

1. Format the code in a sensible and consistent way
2. Reproduce and isolate the bug, noting software versions
   2b. Locate the bug using debugging tools
3. Explore the unexpected behavior
4. Conduct an exorcism to remove the demon
5. Make the code more robust for future users

Simulation as a critical debugging tool

- All of the bugs we have discussed so far are the ones that cause errors
- **These are not the only bugs**
  - The worst bugs allow code to run but unknowingly return incorrect answers
- Maximum likelihood methods have a theoretical property of *consistency*, which you can use to your advantage
- A word to the wise: most of the bugs Lisa Chong (QFC postdoc) is finding in Great Lakes assessment models are of this type 💀 💀 👻

Wikipedia link on the consistency property of the Maximum Likelihood Estimator

References

- Kasper Kristensen, Anders Nielsen, Casper W. Berg, Hans Skaug, Bradley M. Bell. 2016. TMB: Automatic Differentiation and Laplace Approximation. Journal of Statistical Software, 70(5), 1-21. doi:10.18637/jss.v070.i05
- Kristensen K. 2023. RTMB: R Bindings for TMB. R package version 1.0, https://github.com/kaskr/RTMB