The dark art of debugging code

Christopher Cahill
Associate Director
19 December 2024

Quantitative Fisheries Center
MICHIGAN STATE UNIVERSITY

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

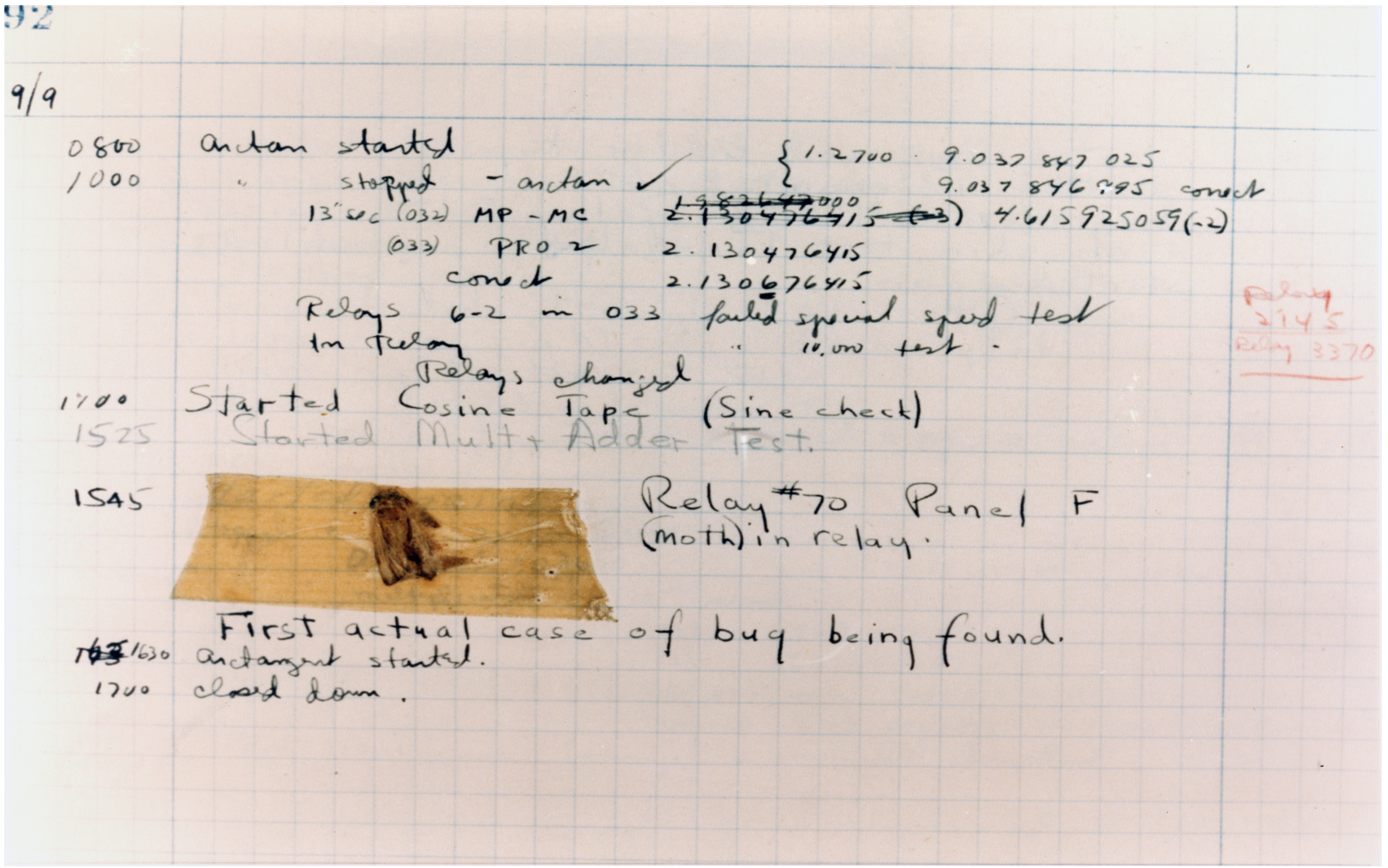
- The origins of the bug 🐛 🐛 🐛
- A general workflow for debugging code
- Specifically we want to think about how we might:
  - Write fewer bugs
  - Reproduce bugs
  - Locate bugs
  - Fix bugs
- Introduce a few useful tools that help with this workflow
  - `styler` library and `browser()`

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 = TRUE));
  jnll =  nll +  nprand;
  jnll;
}
```

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 = TRUE));
  jnll =  nll +  nprand;
  jnll;
}
```

- this is misery ☠️😭💀😱

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 = TRUE))
  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

An example of isolating a bug

```
dgmrf(1:10, mu=2:11, Q=Matrix:::Diagonal(10), log=TRUE)
#[1] -14.18939
dgmrf(1:10-2:11, Q=Matrix:::Diagonal(10), log=TRUE)
#[1] -14.18939
dgmrf(1:10, mu=2:11, Q=Matrix:::Diagonal(10)/2/2, log=TRUE)
#[1] -17.37086
dgmrf(1:10-2:11, Q=Matrix:::Diagonal(10)/2/2, log=TRUE)
#[1] -17.37086
dgmrf(1:10-2:11, Q=Matrix:::Diagonal(10), scale=2, log=TRUE)
#[1] -17.37086
dgmrf(1:10, mu=2:11, Q=Matrix:::Diagonal(10), scale=2, log=TRUE)
#[1] -96.74586
```

- The last result is wrong

https://github.com/kaskr/RTMB/issues/38

Determining versions in R

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

```
[1] '1.6'
```

Determining versions in R

```
1 sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: x86_64-pc-linux-gnu
Running under: Linux Mint 21.3

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so;  LAPACK version 3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

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

loaded via a namespace (and not attached):
 [1] compiler_4.4.2    fastmap_1.2.0     cli_3.6.3         tools_4.4.2
 [5] htmltools_0.5.8.1 yaml_2.3.8        rmarkdown_2.27    knitr_1.47
 [9] isoplite_1.8.8    xfun_0.45         digest_0.6.36     rlang_1.1.4
```

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

Three more RTMB specific bugs and tricks

Avoid struggles with JIT

- R's just-in-time compiler kicks in at unpredictable times
- There are a few conflicts with RTMB, because JIT does not preserve specific definitions made by RTMB
- Can usually be avoided with the following added to the top of your RTMB function

```
f <- function(par) {
    getAll(data, par, warn = FALSE)
    "[<-" <- ADoverload ("[<-") # hey Jim look here =)
    ... more code below...
```

- The forboding sign: phantom errors creeping about in the shadows

credit: Anders Nielsen DTU

Isolating RTMB bugs during optimization

- This is sometimes where happiness goes to die
- Remember that the optimization routine for hierarchical models involves an inner and an outer optimization
- We can inspect both the inner and outer optimization components of a model fit to find bugs in our implementation (coding bugs)

Outer optimization despair ☠️😭💀😱

```
obj = MakeADFun(f, par, random = c("a", "b"), silent = TRUE)
opt <- nlminb(obj$par, obj$fn, obj$gr)
sdr <- sdreport(obj)
sdr
```

```
sdreport(.) result
       Estimate Std. Error
mua    8.467355        NaN
sda    3.463551        NaN
mub   29.044188        NaN
sdb   10.541116        NaN
sdc    1.000000        NaN
sdeps 12.890654        NaN
Warning:
Hessian of fixed effects was not positive definite.
Maximum gradient component: 1.117405e-05
```

Inner optimization despair ☠️😭💀😱

```
obj = MakeADFun(f, par, random = c("a", "b"))
obj$fn()
```

```
iter: 1  value: 61833.54 mgc: 34813 ustep: 0.9999
iter: 2  value: 61833.54 mgc: 0.004086614 ustep: 0.99995
iter: 3  value: 61833.54 mgc: 4.813844e-08 ustep: 0.999975
iter: 4  Error in newton(par = c(a = 0, a = 0, a = 0, a = 0, a = 0, a = 0, a =
0,  :
  Newton failed to find minimum.
[1] NaN
```

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