Smart Programming
Mark Friedman 2025-08-20

Moderation in All Things

If it takes a genius to understand it, you wrote it wrong.

Do not get me wrong. I think you should do the best you can and exercise your talents to the fullest. I am also sure that your code must make sense to you, and to those who may need to understand it in the future.

Macro level

We will start by looking at the forest and not the trees.

Someone hands you a 3,000 line script that works and tells you it takes four hours to complete and has some warnings/errors. They also remind you that the report formatting is still changing all the time. You thank them for the opportunity. You then spend two days trying to figure out what it does and why. You find out that it has six discrete processes that following sequentially and at the end a report is generated.

There are some ideas about how long a piece of code should be. Sometimes it must be long, but if possible, it should be short enough to navigate with a few page-up/page-downs or otherwise be collapsible by an editor into readable portions.

In the macro example, you might end up with a set of programs, the last of which producing a report. You structure it so that the steps that take the longest are completed before Step06 and Step07, so that as the report keeps evolving, you do not have to necessarily run the earlier parts if you have everything needed ready by Step06.

Step01-ReadData.RMD
Step02-ReCode.RMD
Step03-Merge.RMD
Step04-OutlierCorrection.RMD
Step05-Normalize.RMD
Step06-ReportPrep.RMD
Step07-ReportGen.RMD

Smart Programming
Mark Friedman 2025-08-20

Micro Level

Now we will look at the code itself: the trees, not the forest.

Here is a simple example of line of code that works. But for the person who must maintain it a year from now (or even the person who wrote it after a feverish night) may find it difficult to easily pick it apart.

Visually dense

Here is a fun example from Washington University.

https://courses.cs.washington.edu/courses/cse142/97su/josh/obfuscate.html

This code is for a c-based tic-tac-toe program.

```
                a(X){/*/X=-             a(X){/*/X=-
                -1;F;X=-                -1;F;X=-
                -1;F;}/*/               -1;F;}/*/
char*z[]={"char*z[]={","a(X){/*/X=-","-1;F;X=-","-1;F;}/*/","9999999999  :-| ",
"int q,i,j,k,X,O=0,H;S(x)int*x;{X+=X;O+=O;*x+1?*x+2||X++:O++;*x=1;}L(n){for(*",
"z[i=1]=n+97;i<4;i++)M(256),s(i),M(128),s(i),M(64),N;X*=8;O*=8;}s(R){char*r=z",
"[R];for(q&&Q;*r;)P(*r++);q&&(Q,P(44));}M(m){P(9);i-2||P(X&m?88:O&m?48:32);P(",
"9);}y(A){for(j=8;j;)~A&w[--j]||(q=0);}e(W,Z){for(i-=i*q;i<9&&q;)y(W|(1<<i++&",
"~Z));}R(){for(k=J[*J-48]-40;k;)e(w[k--],X|O);}main(u,v)char**v;{a(q=1);b(1);",
"c(1);*J=--u?O?*J:*v[1]:53;X|=u<<57-*v[u];y(X);K=40+q;q?e(O,X),q&&(K='|'),e(X",
",O),R(),O|=1<<--i:J[*J-48+(X=O=0)]--;L(q=0);for(s(i=0);q=i<12;)s(i++),i>4&&N",
";s(q=12);P(48);P('}');P(59);N;q=0;L(1);for(i=5;i<13;)s(i++),N;L(2);}",0};
                b(X){/*/X=-             b(X){/*/X=-
                -1;F;X=-                -1;F;X=-
                -1;F;}/*/               -1;F;}/*/
int q,i,j,k,X,O=0,H;S(x)int*x;{X+=X;O+=O;*x+1?*x+2||X++:O++;*x=1;}L(n){for(*
z[i=1]=n+97;i<4;i++)M(256),s(i),M(128),s(i),M(64),N;X*=8;O*=8;}s(R){char*r=z
[R];for(q&&Q;*r;)P(*r++);q&&(Q,P(44));}M(m){P(9);i-2||P(X&m?88:O&m?48:32);P(
9);}y(A){for(j=8;j;)~A&w[--j]||(q=0);}e(W,Z){for(i-=i*q;i<9&&q;)y(W|(1<<i++&
~Z));}R(){for(k=J[*J-48]-40;k;)e(w[k--],X|O);}main(u,v)char**v;{a(q=1);b(1);
c(1);*J=--u?O?*J:*v[1]:53;X|=u<<57-*v[u];y(X);K=40+q;q?e(O,X),q&&(K='|'),e(X
,O),R(),O|=1<<--i:J[*J-48+(X=O=0)]--;L(q=0);for(s(i=0);q=i<12;)s(i++),i>4&&N
;s(q=12);P(48);P('}');P(59);N;q=0;L(1);for(i=5;i<13;)s(i++),N;L(2);}
                c(X){/*/X=-             c(X){/*/X=-
                -1;F;X=-                -1;F;X=-
                -1;F;}/*/               -1;F;}/*/
```

It was probably written to be readable and tested, then formatted to look like a tic-tac-toe board, with letters for variable names, but you get the idea.

Here is a simpler and more realistic example

```
if (a>b) {e<-c-a} else {e<-c-a}; if (d<b) {e<-(c-a)+(b-d)} else {e<-(b-d) + (c-a)}
print(e)
```

What do the variables a, b, c, d, and e mean? "If else" statements can be hard enough to read but several of them changed together one line is too much. In a few months, you will have to research what the variables meant and what this is calculating if someone asks.

To improve, even without adding comments, at least the logic can be read by using more descriptive variable names and visually breaking it into readable chunks (forgive me: the actual logic of what it calculates is completely made up).

```
if (ageDiagnosis> baseAge) {
   expectedTime <- cataractAge - ageDiagnosis
} else {
   expectedTime <- cataractAge - ageDiagnosis
}

if (deathAge < baseAge) {
   expectedTime <- (cataractAge - ageDiagnosis) + (baseAge - deathAge)
} else {
   expectedTime <- (baseAge - deathAge) + (cataractAge - ageDiagnosis)
}

Print(expectedTime)
```

Procedures as black boxes

Procedures should allow you to focus on the task at hand and leave things you do not need to know about hidden. But that means they shouldn't hide what you need to make clear.

Here is when procedures can get out of hand:

```
if (deathAge < baseAge)     {proc2calculateTime1}
if (ageDiagnosis > baseAge) {proc2CalculateTime2}
if (deathAge < ageDiagnosis){proc2CalculateTime3}
```

Unfortunately, there is too much hidden here. The procedure names do not tell us much and we do not see what they are using. We would never know they created a variable until we saw it referenced somewhere. I have seen complicated programs that were made even more complicated by having every calculation be called through a different procedure, even when the calculation was simple and could have been directly displayed to added clarity to what was happening.

Even a slight improvement can come from writing the procedure more carefully and having it explicitly showing what is being used as input.

Smart Programming
Mark Friedman 2025-08-20

```
expectedTime <- procCalculateExpectedAge(death   =deathAge,
                                         baseline=baseAge,
                                         diagAge =ageDiagnosis)
```

Packages

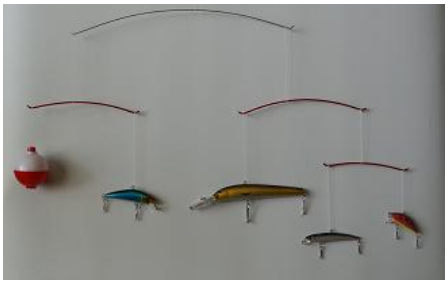
R has thousands of packages. It is true that sometimes you need something so obscure you have to find a new package among dozens of contenders to do what you need. But you should also try to develop expertise in a core set of packages and do your utmost to try and not let package-creep make your environment complicated. I do not have any concrete examples to demonstrate. I just know that when two people on the same project use two different packages to read in spreadsheets in two different programs then nobody benefits.

Life

I may not be the best spokesperson for moderation.

When Covid-19 happened, I was inside, and a lot of activities were restricted. I started making art-mobiles, bookmarks, and refrigerator magnets out of extra stamps I had from my collection. I then got some junk currency on eBay and made mobiles out of them. I then starting getting crystals, wooden-balls, dominos, cheap chess pieces, origami paper, art books, maps, fishing lures, anything that I could turn into mobiles. I made well over seven hundred mobiles over the course of a few years. I am down to my last thirty to give away (but still have material to make more).






So what did I learn? Maybe 700 was overkill. While my mind enjoyed the almost mindless process of laminating, tying knots, bending wire, and putting them togther, my wife said I should have made fewer but more diverse ones.

Stuff we collect in our lives are a good place to practice moderation. How many pieces of furniture do we really need? How many boxes of… How many drawers filled with… I've been

able to take a large amount of things to a person who makes his living selling ‘stuff’ on a table in the old city section of Beersheva. I’m glad to give it away, knowing what I give him is usable and hopefully someone will use it. I’m also glad to have less stuff.

It is a work in progress and I try to watch out for what I keep in the category of ‘if no one can understand it, you wrote it wrong.’ I label things, organize things, and try to make sure when I’m gone, those still here aren’t left wondering “what is this supposed to do? And why is there so much of it?”