Nick Eubank CSDI, Vanderbilt University

January 23, 2018

- 1. Introduce principles of *Defensive Programming*
- 2. Learn four specific "best practices"

- 1. Introduce principles of *Defensive Programming*
- 2. Learn four specific "best practices"
  - · Don't transcribe, export

- 1. Introduce principles of Defensive Programming
- 2. Learn four specific "best practices"
  - · Don't transcribe, export
  - · Use tests

- 1. Introduce principles of Defensive Programming
- 2. Learn four specific "best practices"
  - · Don't transcribe, export
  - · Use tests
  - · Don't duplicate information

- 1. Introduce principles of Defensive Programming
- 2. Learn four specific "best practices"
  - · Don't transcribe, export
  - · Use tests
  - · Don't duplicate information
  - · Use good style

Philosophy of writing code

Philosophy of writing code motivated by the simple supposition:

Philosophy of writing code motivated by the simple supposition:

People are bad at writing code

Philosophy of writing code motivated by the simple supposition:

People are bad at writing code

If we want to avoid errors,

Philosophy of writing code motivated by the simple supposition:

### People are bad at writing code

If we want to avoid errors, not enough to "just be careful."

⇒ Need strategies take take our fallibility into account

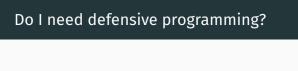
Set of best practices designed to:

Set of best practices designed to:

1. Minimize opportunities for errors to enter code

Set of best practices designed to:

- 1. Minimize opportunities for errors to enter code
- 2. Maximize the probability that *when* we commit errors, we catch them quickly



YES.

"To Err is Human"

#### "To Err is Human"

- · Among professional programmers, average error rate is 10
  - 50 bugs per 1,000 lines of delivered code Steve McConnell, 1993

"Bugs" ⇒ syntax errors

QJPS Replication Review: Before publication, test whether replication packages run and generate results in the paper.

QJPS Replication Review: Before publication, test whether replication packages run and generate results in the paper.

From 2012 - 2016

QJPS Replication Review: Before publication, test whether replication packages run and generate results in the paper.

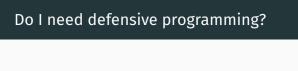
From 2012 - 2016

4 packages passed without modifications

QJPS Replication Review: Before publication, test whether replication packages run and generate results in the paper.

#### From 2012 - 2016

- · 4 packages passed without modifications
- 58% of packages generated results that were different from those in the paper.



- 1. If a correlation exists between sophistication of analysis and likelihood of errors, it is if anything positive.
  - · Senior, junior, fancy, basic: all had problems!

- 1. If a correlation exists between sophistication of analysis and likelihood of errors, it is if anything positive.
  - · Senior, junior, fancy, basic: all had problems!
- 2. Even if you trust yourself, do you trust your coauthors?

- 1. If a correlation exists between sophistication of analysis and likelihood of errors, it is if anything positive.
  - · Senior, junior, fancy, basic: all had problems!
- 2. Even if you trust yourself, do you trust your coauthors?
  - (Do you trust the version of you that wrote that code at 3am?))

- 1. If a correlation exists between sophistication of analysis and likelihood of errors, it is if anything positive.
  - · Senior, junior, fancy, basic: all had problems!
- 2. Even if you trust yourself, do you trust your coauthors?
  - (Do you trust the version of you that wrote that code at 3am?))
- 3. Do you trust the people who made the dataset you're using?

- 1. If a correlation exists between sophistication of analysis and likelihood of errors, it is if anything positive.
  - · Senior, junior, fancy, basic: all had problems!
- 2. Even if you trust yourself, do you trust your coauthors?
  - (Do you trust the version of you that wrote that code at 3am?))
- 3. Do you trust the people who made the dataset you're using?
  - If you estimate the share of a population that's female, and someone left a 7 in the female variable, if you don't catch it, that means your answer is wrong.

### Four Skills

Write tests

Don't duplicate information

Don't transcribe, export

Use good style

## Four Skills

#### Write tests

Don't duplicate information

Don't transcribe, export

Use good style

## If you use R:

· Go to: www.nickeubank.com/data-integrity-tests-r

## If you use R:

• Go to: www.nickeubank.com/data-integrity-tests-r

## If you use R:

• Go to: www.nickeubank.com/data-integrity-tests-r

If you don't use R and use Stata (or Python) instead...

1. Good for you for using a good language!

### If you use R:

• Go to: www.nickeubank.com/data-integrity-tests-r

- 1. Good for you for using a good language!
- 2. Same concepts, different syntax:

## If you use R:

• Go to: www.nickeubank.com/data-integrity-tests-r

- 1. Good for you for using a good language!
- 2. Same concepts, different syntax:
  - Stata: assert STATEMENT (i.e. assert 1 == 0)

#### If you use R:

• Go to: www.nickeubank.com/data-integrity-tests-r

- 1. Good for you for using a good language!
- 2. Same concepts, different syntax:
  - Stata: assert STATEMENT (i.e. assert 1 == 0)
  - Python: assert STATEMENT (i.e. assert 1 == 0)

### Write tests

#### If you use R:

· Go to:

www.nickeubank.com/data-integrity-tests-r

If you don't use R and use Stata (or Python) instead...

- 1. Good for you for using a good language!
- 2. Same concepts, different syntax:
  - Stata: assert STATEMENT (i.e. assert 1 == 0)
  - Python: assert STATEMENT (i.e. assert 1 == 0)

#### Stata tutorial:

www.nickeubank.com/data-integrity-tests-stata

Is age always positive?

This will pass (do nothing):

```
age = c(42, 20, 31, 18)
# Make sure age is positive:
stopifnot( age > 0 )
```

But if, for example, "missing" was coded as -99, this would throw an error:

```
age = c(42, 20, 31, -99)
stopifnot( age > 0 )
```

For vectors, **stopifnot** checks if ALL values are TRUE.

This will fail:

```
# Are all values True?
v = c(1, 2, 3)
stopifnot( v == 2 )
```

This will pass:

```
stopifnot( v > 0 )
```

Are at least SOME values non-zero?

```
v = c(0, 0, 1, 0)
stopifnot( any(v != 0) )
```

Can combine with functions:

```
stopifnot( length(VECTOR) == 100 )
```

Setup:

Setup:

1. Download project file

### Setup:

- 1. Download project file
- 2. Open my\_analysis.R, set the working directory to the DefensiveProgramming\_Part1 folder, and load data.

#### Setup:

- 1. Download project file
- 2. Open my\_analysis.R, set the working directory to the DefensiveProgramming\_Part1 folder, and load data.
- 3. Run the file.

#### Exercises:

### Setup:

- 1. Download project file
- 2. Open my\_analysis.R, set the working directory to the DefensiveProgramming\_Part1 folder, and load data.
- 3. Run the file.

#### Exercises:

1. The average household size in this data SHOULD be about 4.1 people. What's wrong?

#### Setup:

- 1. Download project file
- 2. Open my\_analysis.R, set the working directory to the DefensiveProgramming\_Part1 folder, and load data.
- 3. Run the file.

#### Exercises:

- The average household size in this data SHOULD be about 4.1 people. What's wrong?
- 2. Write a test to catch the problem.

#### Setup:

- 1. Download project file
- 2. Open my\_analysis.R, set the working directory to the DefensiveProgramming\_Part1 folder, and load data.
- 3. Run the file.

#### Exercises:

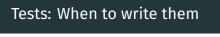
- 1. The average household size in this data SHOULD be about 4.1 people. What's wrong?
- 2. Write a test to catch the problem.
- 3. What tests do you think we should have around the merge here? Add some tests.

Extra credit: ELF is also wrong. WITHOUT focusing on what in the code might be wrong, write some tests for general

Tests: Exercise 1 (Part 2!)

Your co-author just updated your data!

- Change your working directory to DefensiveProgramming\_Part2, update file names to \_v2, and run it!
- 2. Did you find a problem with the merge?



• After merges No where are problems with data made more clear then in a merge. ALWAYS add tests after a merge!

- After merges No where are problems with data made more clear then in a merge. ALWAYS add tests after a merge!
- After complicated manipulations If you had to think about it, you should test to do it.

- After merges No where are problems with data made more clear then in a merge. ALWAYS add tests after a merge!
- After complicated manipulations If you had to think about it, you should test to do it.
- Before dropping observations

- After merges No where are problems with data made more clear then in a merge. ALWAYS add tests after a merge!
- After complicated manipulations If you had to think about it, you should test to do it.
- · Before dropping observations

Adriane Rule: Most of use check things interactively to make sure we did it right. A good rule of thumb is that when you catch yourself checking something interactively, stop and write it as a test.

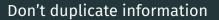
## Four Skills

Write tests

Don't duplicate information

Don't transcribe, export

Use good style



• If information is represented in many places, when you make changes you have to find all those places.

- If information is represented in many places, when you make changes you have to find all those places.
- If information is represented once, and everything else points back to that representation, one change will always change everything.

Duplicated information:

```
df$var1 <- gsub("armadillo", "Mr. Armadillo", df$va
df$var2 <- gsub("armadillo", "Mr. Armadillo", df$va
...
[Other manipulations]</pre>
```

df\$var7 <- gsub("armadillo", "Mr. Armadillo", df\$va</pre>

One representation:

```
pre change name <- "armadillo"</pre>
replacement name <- "Mr. Armadillo"
df$var1 <- gsub(pre change name,
                 replacement name, df$var1)
df$var2 <- gsub(pre change name,</pre>
                 replacement name, df$var2)
df$var7 <- gsub(pre change name,</pre>
                 replacement name. df$var7)
```

One representation:

```
pre change name <- "armadillo"</pre>
replacement name <- "Dr. Armadillo"
df$var1 <- gsub(pre change name,
                 replacement name, df$var1)
df$var2 <- gsub(pre change name,</pre>
                 replacement name, df$var2)
df$var7 <- gsub(pre change name,</pre>
                 replacement name. df$var7)
```

## Four Skills

Write tests

Don't duplicate information

Don't transcribe, export

Use good style

### Don't transcribe results!

Number one reason papers don't match real results at QJPS.

#### R:

- · stargazer
- Tutorial: http:
  - //jakeruss.com/cheatsheets/stargazer.html
- Custom: http://stanford.edu/~ejdemyr/ r-tutorials/tables-in-r/

#### Stata:

 Summary: http://www.nickeubank.com/ exporting-results-stata-latex/

Use for numbers in your text as well!

#### Don't transcribe results: Exercise 2

Oh man, our tests found all these problems. Ugh, why did we put all those old results in by hand?! Now we have to copy them again. Or... we could make the automatically updating!

- 1) Open our latex analysis file (my\_writeup.tex).
- 2) Export the regression table at the end of my\_analysis.R using stargazer. The syntax is:

3) Import it into your latex document using the
input{} command.

## Don't transcribe: Exercise 2 (Part 2)

Now, in the text, we say that households have an average size of 8, but we know that's wrong. Can you export the average size of the household from R and import it into LaTeX? Hint: Here's how you write a number to a file as text:

```
x = 1/3
x_as_string = format(x, digits=2)
write(x_as_string, "my_file.tex")
```

## Four Skills

Write tests

Don't duplicate information

Don't transcribe, export

Use good style