

What Software Engineers can share with Data Scientists: ... with Automatic Tests

Andrea Melloncelli

andrea.melloncelli@quantide.com

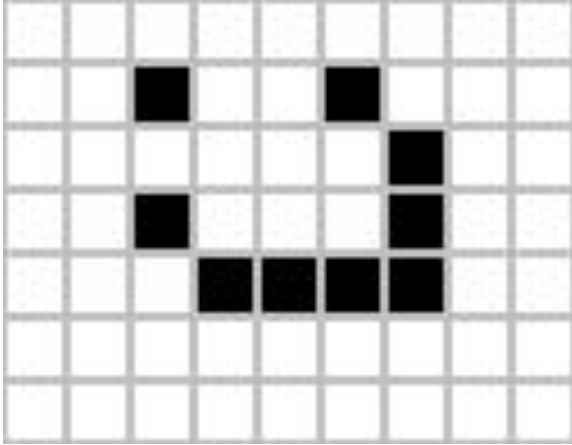
sponsored by



Outline

1. Conway's Game of Life
 2. Why do tests
 - 2.1. Validation
 - 2.2. Working documentation
 - 2.3. Readable code
 3. Testing Strategies
 - 3.1. Test Driven Development (TDD)
 - 3.2. Test After Development (TAD)
 4. Testing tools
 - 4.1. Testthat package
 - 4.2. Shinytest package
- Summary

1. Conway's Game of Life



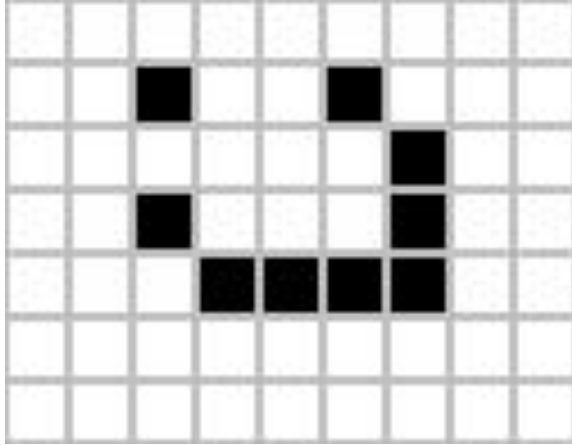
Legend:



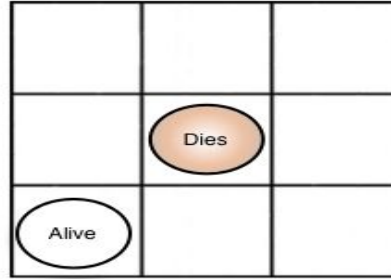
Dead

Alive

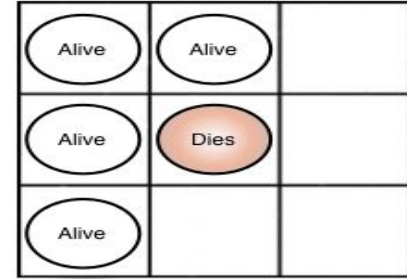
1. Conway's Game of Life



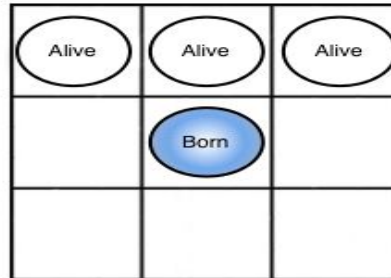
1. Any live cell with fewer than two live neighbors dies, as if by isolation.



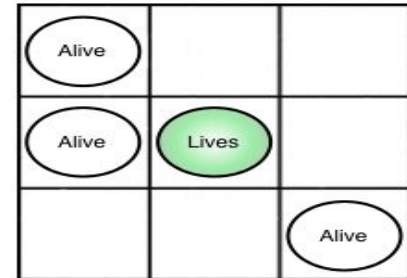
2. Any live cell with more than three live neighbors dies, as if by overcrowding.



3. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.



4. Any live cell with two or three live neighbors lives on to the next generation.



Legend:



Dead

Alive

Outline

1. Conway's Game of Life
 2. Why do tests
 - 2.1. Validation
 - 2.2. Working documentation
 - 2.3. Readable code
 3. Testing Strategies
 - 3.1. Test Driven Development (TDD)
 - 3.2. Test After Development (TAD)
 4. Testing tools
 - 4.1. Testthat package
 - 4.2. Shinytest package
- Summary

2. Why do tests?

1. Validation
2. Working documentation
3. Readable and reusable code

2. Why do tests?

2.1. Validation

```
context("evolution of a single cell")
```

```
...
```

Context: *Evolve*

2. Why do tests?

2.1. Validation

```
context("evolution of a single cell")
```

```
...
```

```
test_that(desc = "rule1",...)
```

```
test_that(desc = "rule2",...)
```

```
test_that(desc = "rule3",...)
```

```
test_that(desc = "rule4",...)
```

Context: *Evolve*

1. **Test 1: rule 1**
 - 1.1. ...
2. Test 2: rule 2
 - 2.1. ...
3. Test 3: rule 3
 - 3.1.
4. Tests ...

2. Why do tests?

2.1. Validation

```
context("evolution of a single cell")
...
test_that(desc =
  paste("Any dead cell",
        "with exactly three live neighbours",
        "becomes a live cell,",
        "as if by reproduction."),
  ...

```

Context: *Evolve*

1. Test 1: rule 1
 - 1.1. ...
2. Test 2: rule 2
 - 2.1. ...
3. Test 3: rule 3
 - 3.1.
4. Tests ...

2. Why do tests?

2.1. Validation

```
context("evolution of a single cell")
...
test_that(desc =
  paste("Any dead cell",
        "with exactly three live neighbours",
        "becomes a live cell,",
        "as if by reproduction."),
  code = {
    state <- dead

    evolved_state <- evolve(state, neigh = 3)

    expect_equal(evolved_state, alive)
  })
...

```

Context: *Evolve*

1. Test 1: rule 1
 - 1.1. ...
2. Test 2: rule 2
 - 2.1. ...
3. Test 3: rule 3
 - 3.1. Setup
 - 3.2. Function run
 - 3.3. Validation
4. Tests ...

2. Why do tests?

sponsored by



2.2. Working documentation

A test file provides:

1. Information about the feature (context)
2. Some working examples of how that feature is implemented (`test_that`)

2. Why do tests?

sponsored by



2.3. Readable code

Refactoring: improving the code without adding further functionalities.

```
if (wday(now) > 2 &&
    wday(now) < 6 &&
    hour(now) > 8 &&
    hour(now) < 17 )
{
    cat("I'm working.")
} else {
    cat("I'm out of the
office.")
}
```

2. Why do tests?

sponsored by




2.3. Readable code

Refactoring: improving the code without adding further functionalities.

```
if (wday(now) > 2 &&
    wday(now) < 6 &&
    hour(now) > 8 &&
    hour(now) < 17 )
{
  cat("I'm working.")
} else {
  cat("I'm out of the
office.")
}
```

```
is_working_time <- function(time) {
  wday(time) > 2 &&
  wday(time) < 6 &&
  hour(time) > 8 &&
  hour(time) < 17
}
```



```
if (is_working_time(now))
{
  cat("I'm working.")
} else {
  cat("I'm out of office.")
}
```

Outline

1. Conway's Game of Life
 2. Why do tests
 - 2.1. Validation
 - 2.2. Working documentation
 - 2.3. Readable code
 - 3. Testing Strategies**
 - 3.1. Test Driven Development (TDD)**
 - 3.2. Test After Development (TAD)**
 4. Testing tools
 - 4.1. Testthat package
 - 4.2. Shinytest package
- Summary

3. Testing Strategies

1. Test Driven Development (TDD)



2. Test After Development (TAD)

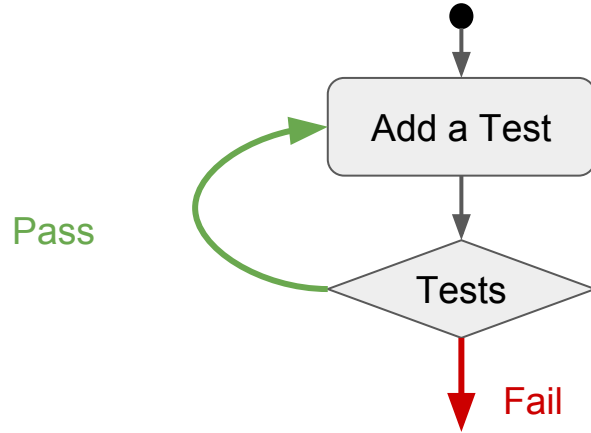


3. Testing Strategies

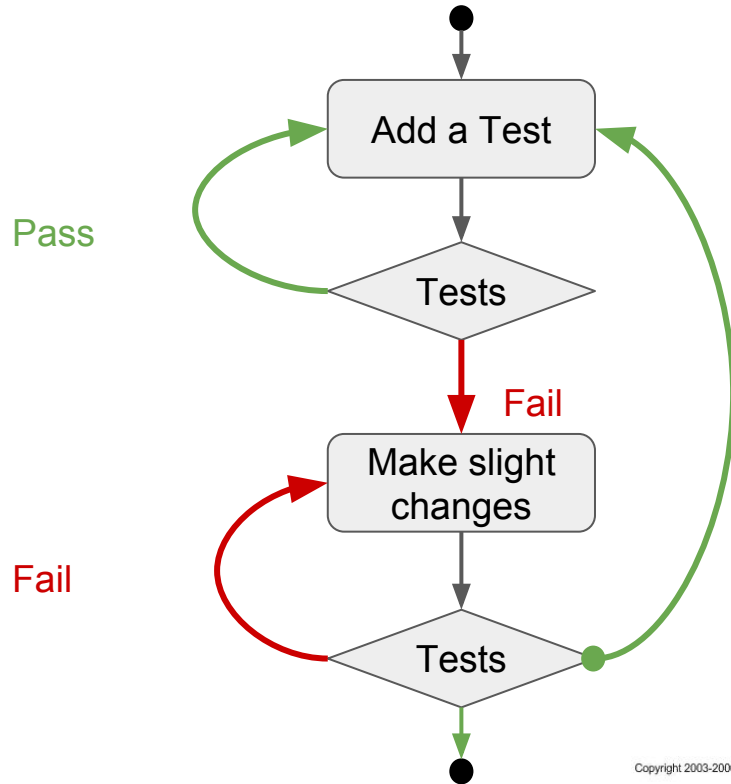
sponsored by



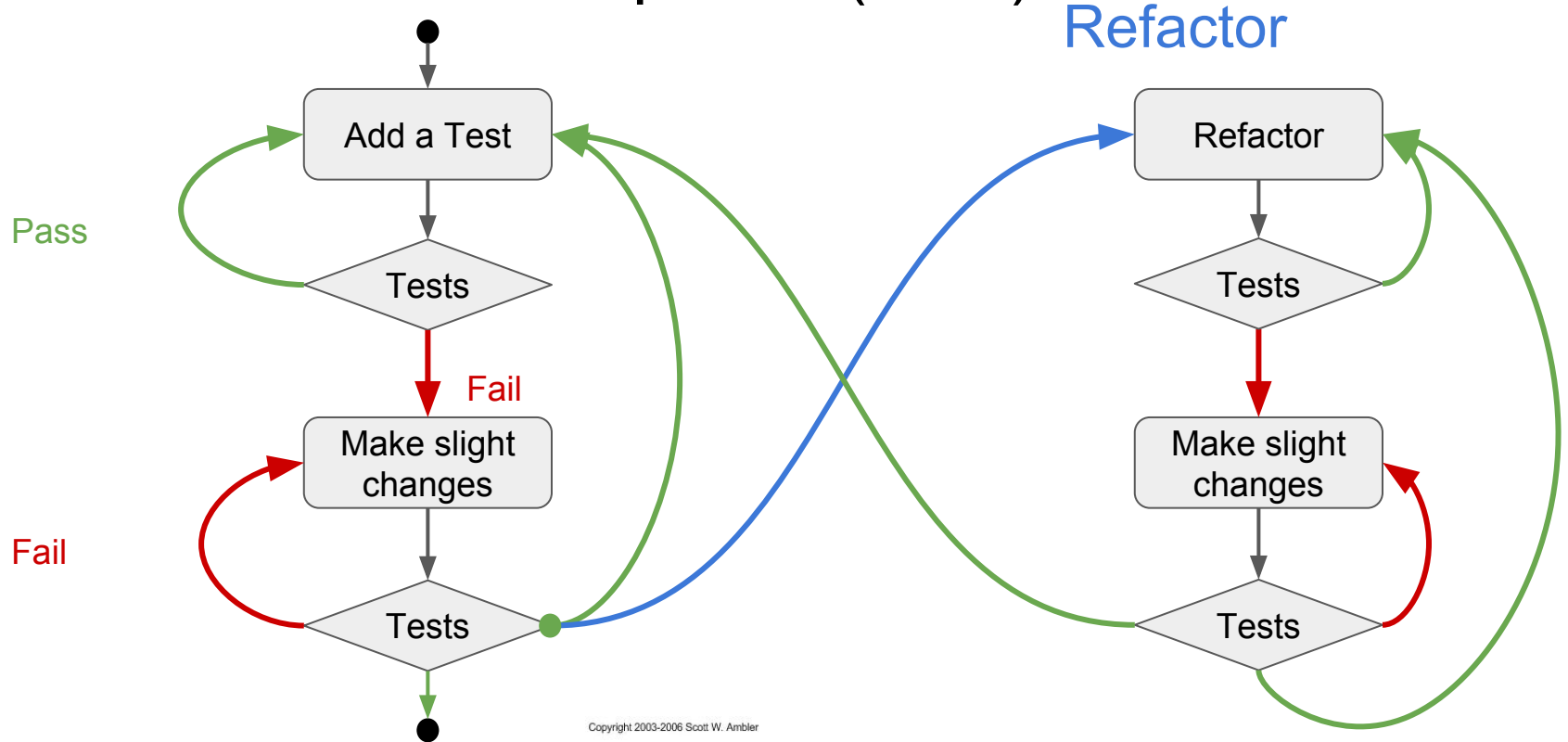
3.1. Test Driven Development (TDD)



3.1. Test Driven Development (TDD)



3.1. Test Driven Development (TDD)



3.2. Test After Development (TAD)

Test After Development (TAD)

Implementation then Tests



1. Old way of operate, I only need to add tests to my implementation
2. It is always available
3. Useful when the result can't be predicted (models, ...)

Outline

1. Conway's Game of Life
 2. Why do tests
 - 2.1. Validation
 - 2.2. Working documentation
 - 2.3. Readable code
 3. Testing Strategies
 - 3.1. Test Driven Development (TDD)
 - 3.2. Test After Development (TAD)
 4. Testing tools
 - 4.1. Testthat package
 - 4.2. Shinytest package
- Summary

4. Testing tools

1. Testthat package (TDD + TAD)
2. Shinytest package (TAD)

4. Testing tools

4.1. Testthat Package

- Complete set Testing tools
- Developed by RStudio
- Compatible with different Testing Strategies (TAD and TDD)



4.1. Testthat Package

```
context("evolution of a single cell")
...
test_that(desc =
  paste("Any dead cell",
        "with exactly three live neighbours",
        "becomes a live cell,",
        "as if by reproduction."),
  code = {
    state <- dead

    evolved_state <- evolve(state, neigh = 3)

    expect_equal(evolved_state, alive)
  })
...
```

Context: *Evolve*

1. Test 1: rule 1
 - 1.1. ...
2. Test 2: rule 2
 - 2.1. ...
3. Test 3: rule 3
 - 3.1. Setup
 - 3.2. Function run
 - 3.3. Validation
4. Tests ...

4. Testing tools

4.2. Shinytest

The strategy is TAD (Test After Development), therefore:

1. Having a working Shiny application
2. Record a test using the application as the final user
3. Run all tests sequentially
4. And if something is wrong....



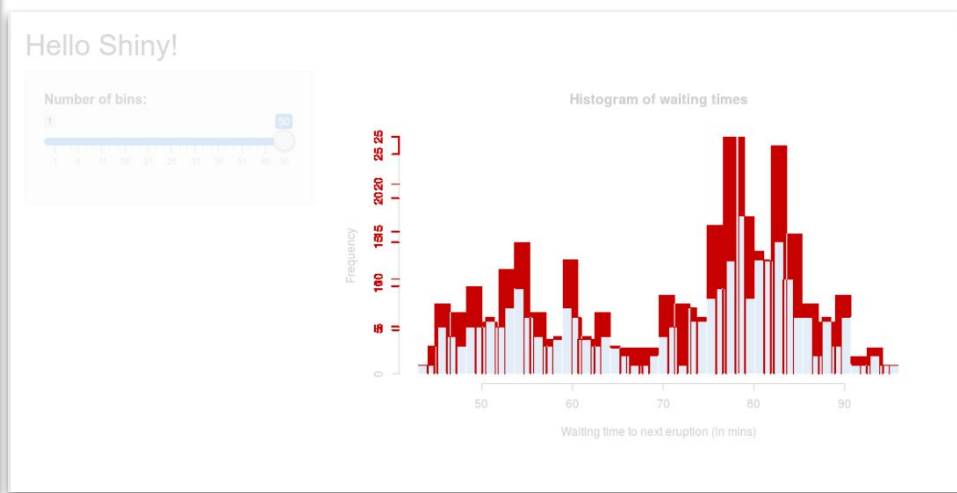
4. Testing tools

4.2. Shinytest

4. If something is wrong... Get notified graphically!

```
002.json MODIFIED
@@ -3,18 +3,18 @@
 3 3     "bins": 50
 4 4     },
 5 5     "output": {
 6 6       "distPlot": {
 7 7 -     "src": "[image data sha1: 1b162d14db213289390dccad2d7bcff52708947b]",
 8 8 +     "src": "[image data sha1: 970975e03c6e9915f55e28f95dcd48740c80486]",
 9 9     "width": 631,
10 10    "height": 400,
11 11    "coordmap": [
12 12      {
13 13        "domain": {
14 14          "left": 40.88,
15 15          "right": 98.12,
16 16 -          "bottom": -1,
17 17 +          "top": 26,
18 18 +          "bottom": -1.08,
19 19 +          "top": 28.08
20 20        },
21 21      },
22 22    ],
23 23    "range": {
24 24      "left": 59.04,
25 25      "right": 600.76,
```

You see the differences between the recorded run and the current coloured out.



sponsored by



Summary

Why are tests so important in our work?

Questions?

sponsored by

