Automate the Testing of JSL Using Hamcrest

JMP Discovery Summit Europe 2019 (2019-EU-45MP-061)

Justin Chilton, JMP Senior Associate Test Engineer, SAS

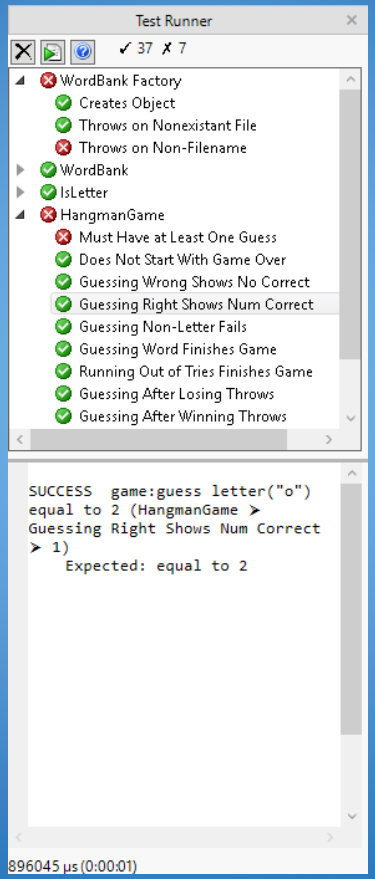
Evan McCorkle, JMP Software Developer, SAS

DEMO

Red to Green

jmp® Statistical Discovery.™ From SAS.

Automated Testing



What did we just do?

- Determine if our script is performing as expected
- Realized it wasn't working correctly
- Fixed the bug
- Verified that it was fixed

What is Hamcrest?

- Funny name
- Unit Testing Framework
  - 3rd-Generation
  - Declarative
  - Readable
- Self-contained
- Extension of existing JSL Testing Framework
- Used to test JMP itself
- Available now

How did we do it?

- Using a new JSL library called JSL-Hamcrest
- By writing tests
  - Built from assertions about the scripts behavior
- And running them
  - In an automated way
  - With clear reporting of success and failure

```
ut test("HangmanGame", "Guessing Non-Letter Fails", Expr(
  game = new HangmanGame("food", 5);
  ut assert that(Expr(game:guess letter("@")),
                 ut throws("Invalid Guess!"));
));

// Success!
```

History

What is Hamcrest?

- Funny name
- Unit Testing Framework
  - 3rd-Generation
  - Declarative
  - Readable
- Self-contained
- Extension of existing JSL Testing Framework
- Used to test JMP itself
- Available now

Details

- Originally developed for Java and jUnit (~2012)
- Focused on creating easy-to-read declarative assertions
- Ported to many other languages
  - C++, C#, Objective-C, Python, PHP, JavaScript, Erlang, R, Rust, Swift
  - And now JSL
- JSL-Hamcrest also comes with a Unit Testing framework similar to jUnit with
  - Test Cases for consistency
  - Named tests for human-readable expectations
  - Customizable reporting

History

1st Generation

```
assert(x == y)
```

2nd Generation

```
assert_equal(x, y)
```

3rd Generation

```
assert_that(x, equal_to(y))
```

1st Generation

```
assert(x == y)
```

- Not very informative on failure
- Intent of test writer not always clear
- Imperative programming

```
FAILURE assert(x == y)
```

History

1st Generation

```
assert(x == y)
```

2nd Generation

```
assert_equal(x, y)
```

3rd Generation

```
assert_that(x, equal_to(y))
```

2nd Generation

```
assert_equal(x, y)
assert_not_equal(x, y)
assert_contained_in(x1, x)
```

- Informative failures
- Intent is clear
- Declarative, not imperative
- Explosion of assertions for different situations
  - However, often never extended, meaning four or five assertions are abused for all tests, leading back to uninformative failures and unclear intent

```
FAILURE assert_contained_in(x1, x)
    5 not contained in {1, 2, 4, 6}
```

History

1st Generation

```
assert(x == y)
```

2nd Generation

```
assert_equal(x, y)
```

3rd Generation

```
assert_that(x, equal_to(y))
```

3rd Generation

```
assert_that(x, equal_to(y))
assert_that(x, not(equal_to(y)))
assert_that(x1, contained_in(x))
```

- Informative failures
- Clear intent
- Declarative, not imperative
- Reusable/Composable assertions

```
FAILURE assert_that(x1,contained_in(x))
    5 not contained in {1, 2, 4, 6}
```

History

1st Generation

```
assert(x == y)
```

2nd Generation

```
assert_equal(x, y)
```

3rd Generation

```
assert_that(x, equal_to(y))
```

Generational Progress

- Each generation building on the previous
- Hamcrest assertions
  - Can be written relatively naturally
  - Can be read relatively natural
  - Provide context on failure
    - You know **that** the test failed
    - And you know **how** it failed
    - Often, you needn't even read the test to know what you broke
  - Describe what you want to test, not how to extract that information
  - Can be mixed together to form entirely new assertions

Using Hamcrest

Examples

anything
all of
not
equal to
contains
subset of
starts with
missing
close to
title is
ignoring whitespace
throws
...

Matchers – The Key to Hamcrest

```
assert_that("hamcrest",
            is(anagram_of("matchers"))
```

- Matchers do two things:
  - Describe their expectations
  - Determine if those expectations are satisfied by a given value

Examples

anything
all of
not
equal to
contains
subset of
starts with
missing
close to
title is
ignoring whitespace
throws
…

Matchers – The Key to Hamcrest

```
assert_that("ham sandwich",
            is(anagram_of("matchers"))
```

- Matchers do two things:
  - Describe their expectations
  - Determine if those expectations are satisfied by a given value

Examples

anything
all of
not
equal to
contains
subset of
starts with
missing
close to
title is
ignoring whitespace
throws
…

Matchers – The Key to Hamcrest

```
assert_that("Copenhagen",
            city_within("Denmark"))
```

- Matchers do two things:
  - Describe their expectations
  - Determine if those expectations are satisfied by a given value

Examples

anything
all of
not
equal to
contains
subset of
starts with
missing
close to
title is
ignoring whitespace
throws
...

Matchers – The Key to Hamcrest

```
assert_that("London",
            city_within("Denmark"))
```

- Matchers do two things:
  - Describe their expectations
  - Determine if those expectations are satisfied by a given value

Examples

anything
all of
not
equal to
contains
subset of
starts with
missing
close to
title is
ignoring whitespace
throws
...

Matchers – The Key to Hamcrest

```
assert_that("London",
            not(city_within("Denmark")))
```

- Matchers do two things:
  - Describe their expectations
  - Determine if those expectations are satisfied by a given value

Categories

- Direct
  - Leaf
  - Actually Tests
  - equal, less than, has key, starts with
- Indirect
  - Non-Leaf
  - Delegates Testing
  - not, contains, as char, ignoring whitespace

Matchers – The Key to Hamcrest

```
assert_that(x, is_not(equal_to(y)));
```

- `assert_that(value, matcher)`
  - Harness. Is the matcher satisfied by the value?
- `is_not(matcher) → Matcher`
  - Negates any matcher inside of it (Indirect)
- `equal_to(value) → Matcher`
  - Only satisfied by values equal to value (Direct)
  - JSL is unique in that x is an expression

DEMO

Writing Tests

jmp® Statistical Discovery.™ From SAS.

Example

Why do I want Test Cases?

- Group your tests
- Name your tests
- Isolate them from each other
  - Delete symbols
  - Close Windows
- Increase consistency
- Reduce duplication
  - Setup
  - Teardown

```
case = ut test case("GroupA")
  <<Setup(Expr(…));

ut test(case, "Test", Expr(
  ut assert that(…)
));

ut test("GroupB", "Test", Expr(
  ut assert that(…)
));
```

DEMO

Writing Test Cases

jmp® Statistical Discovery.™ From SAS.

Extending JSL-Hamcrest

Possibilities

Log File

Log Window

GUI

Custom Report

Send Emails

…

Reporting Success and Failure

- Reporter object
  - Receives success and failure messages
  - Can be customized to your liking
- JSL-Hamcrest has several builtin Reporters
  - Write failures to the Log Window
  - Write success/failures to a Test Runner GUI
- But you can write your own Reporter to ensure that JSL-Hamcrest presents results exactly how you want them
- Used internally when testing JMP itself

Interface

<< Matches

Success/Failure

How

<< Describe

Expectation

Factory Function

ut my matcher

Writing your own Matchers

- One of the best things about Hamcrest is the ability to write new Matchers
  - Domain specific
  - Meaningful to you
- Write what you are testing, not how
  - Write it in your own words
  - Add any relevant information to failures
- If you do it correctly, you can mix your new Matcher with existing ones
  - Flexible and expressive
  - Multiply your work
- JSL-Hamcrest has a suite of builtin Matchers

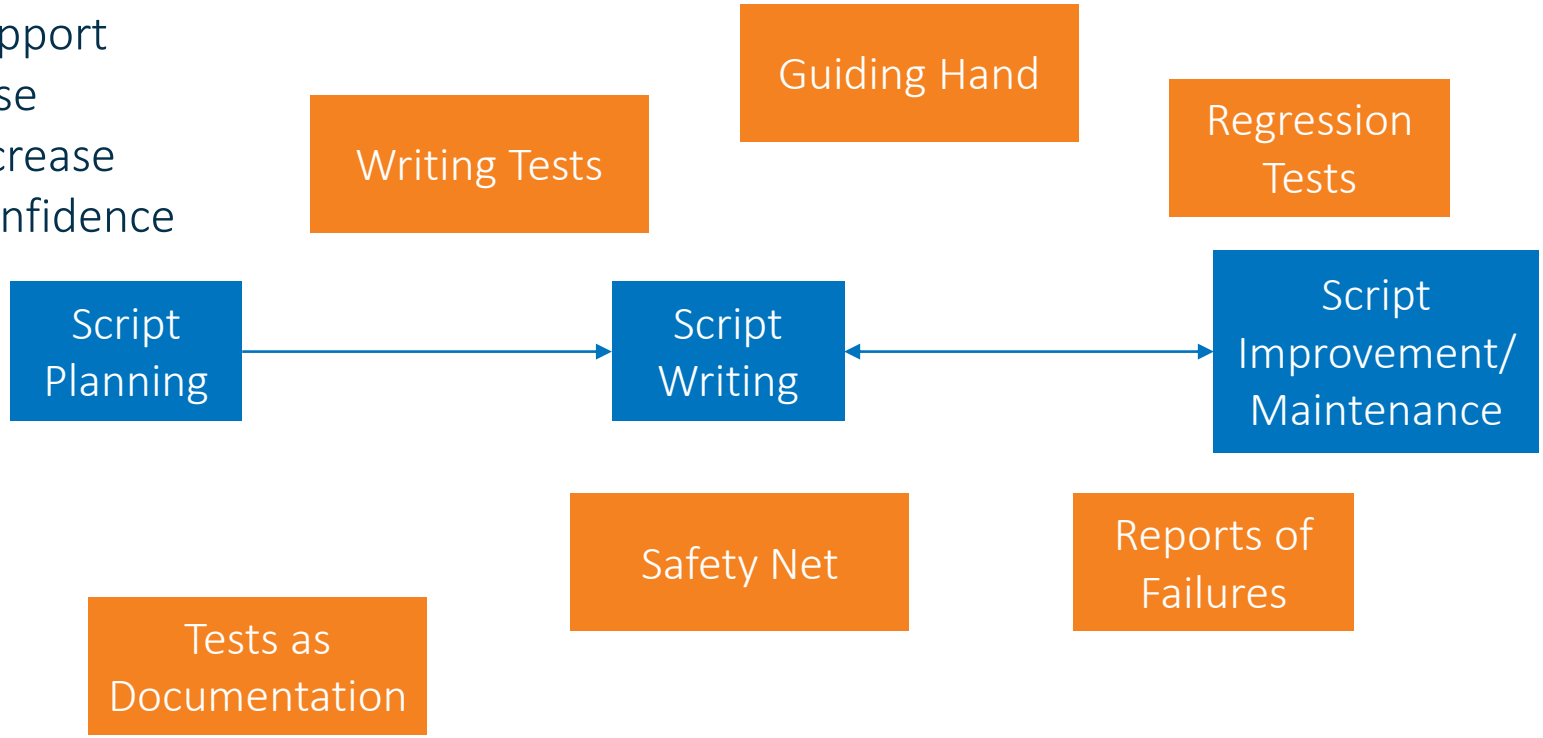
DEMO

Writing a Matcher

jmp® Statistical Discovery.™ From SAS.

How can JSL-Hamcrest help?

- Enable
- Support
- Ease
- Increase Confidence



v1.0

Available Now
for JMP 14.1+

Get it on the
JMP User Community

Tell us what you think

Release of JSL-Hamcrest 1.0 Includes

- Library with
  - Test Cases
  - Suite of Matchers
  - Several Reporters
  - Documentation
- Add-In providing
  - The above library
  - Test Runner GUI
  - One-click “Run Tests”

DEMO

Add-In

jmp® Statistical Discovery.™ From SAS.

Questions?

jmp.com