



`inch` gives you hints where to improve your docs. One Inch at a time.

Take a look at the project page with screenshots (live and in full color).

What can it do?

`inch` is a little bit like Code Climate, but for your inline code documentation (and not a webservice).

It is a command-line utility that suggests places in your codebase where documentation can be improved.

If there are no inline-docs yet, `inch` can tell you where to start.

Installation

Add this line to your application's Gemfile:

```
1 gem 'inch', require: false
```

And then execute:

```
1 $ bundle
```

Or install it yourself as:

```
1 $ gem install inch
```

Usage

To run Inch, simply type

```
1 $ inch
```

Given a `lib` directory with the following code inside:

```
1 class Foo
2   # A complicated method
3   def complicated(o, i, *args, &block)
4     # ... snip ...
5   end
6
7   # An example of a method that takes a parameter (+param1+)
```

```
8 # and does nothing.
9 #
10 # Returns nil
11 def nothing(param1)
12 end
13
14 def filename
15   "#{self.class}_#{id}.foo"
16 end
17 end
```

Inch will suggest that the docs could be improved:

```
1 # Properly documented, could be improved:|
2
3 B ↑   Foo#complicated
4
5 # Undocumented:|
6
7 U ↑   Foo|
8 U ↗   Foo#filename
9
10 You might want to look at these files:|
11
12 lib/foo.rb
13
14 Grade distribution (undocumented, C, B, A): L
15
16 Only considering priority objects: ↗→ (use `--help` for options).
```

Configuration

By default, Inch looks into `{app, lib}/**/*.rb` for Ruby source files. You can customize this by either passing the desired files to the executable:

```
1 $ inch suggest plugins/**/*.rb
```

or by creating a file named `.inch.yml` in your project directory:

```
1 files:
2   # define files included in the analysis (defaults to ["{app,lib
3     }/**/*.rb"])
4   included:
5     - plugins/**/*.rb
6   # define files excluded from the analysis (defaults to [])
7   excluded:
8     # you can use file paths
9     - plugins/vendor/sparkr/sparkr.rb
```

```
9      # or globs
10     - plugins/vendor/**/*.rb
11     # or regular expressions
12     - !ruby/regexp /vendor/
```

As you would expect, `included` sets an array of included files (or globs) and `excluded` sets an array of files, globs or regexes of files to exclude from the evaluation.

Philosophy

Inch was created to help people document their code, therefore it may be more important to look at **what it does not** do than at what it does.

- It does not aim for “fully documented” or “100% documentation coverage”.
- It does not tell you to document all your code (neither does it tell you not to).
- It does not impose rules on how your documentation should look like.
- It does not require that, e.g. “every method’s documentation should be a single line under 80 characters not ending in a period” or that “every class and module should provide a code example of their usage”.

Inch takes a more relaxed approach towards documentation measurement and tries to show you places where your codebase *could* use more documentation.

The Grade System

Inch assigns grades to each class, module, constant or method in a codebase, based on how complete the docs are.

The grades are:

- **A** - Seems really good
- **B** - Properly documented, but could be improved
- **C** - Needs work
- **U** - Undocumented

Using this system has some advantages compared to plain coverage scores:

- You can get an **A** even if you “only” get 90 out of 100 possible points.
- Getting a **B** is basically good enough.
- Undocumented objects are assigned a special grade, instead of scoring 0%.

The last point might be the most important one: If objects are undocumented, there is nothing to evaluate. Therefore you can not simply give them a bad rating, because they might be left undocumented intentionally.

Priorities ↑ ↓

Every class, module, constant and method in a codebase is assigned a priority which reflects how important Inch thinks it is to be documented.

This process follows some reasonable rules, like

- it is more important to document public methods than private ones
- it is more important to document methods with many parameters than methods without parameters
- it is not important to document objects marked as `:nodoc:`

Priorities are displayed as arrows. Arrows pointing north mark high priority objects, arrows pointing south mark low priority objects.

No overall scores or grades

Inch does not give you a grade for your whole codebase.

“Why?” you might ask. Look at the example below:

```
1 Grade distribution (undocumented, C, B, A): ████
```

In this example there is a part of code that is still undocumented, but the vast majority of code is rated A or B.

This tells you three things:

- There is a significant amount of documentation present.
- The present documentation seems good.
- There are still undocumented methods.

Inch does not really tell you what to do from here. It suggests objects and files that could be improved to get a better rating, but that is all. This way, it is perfectly reasonable to leave parts of your codebase undocumented.

Instead of reporting

```
1 coverage: 67.1% 46 out of 140 checks failed
```

and leaving you with a bad feeling, Inch tells you there are still undocumented objects without judging.

This provides a lot more insight than an overall grade could, because an overall grade for the above example would either be an **A** (if the evaluation ignores undocumented objects) or a weak **C** (if the evaluation includes them).

The grade distribution does a much better job of painting the bigger picture.

Features

Inch is built to parse YARD, RDoc and TomDoc style documentation comments, but works reasonably well with unstructured comments.

These inline-docs below all score an **A** despite being written in different styles:

```
1 # Detects the size of the blob.
2 #
3 # @example
4 #   blob_size(filename, blob) # => some value
5 #
6 # @param filename [String] the filename
7 # @param blob [String] the blob data
8 # @param mode [String, nil] optional String mode
9 # @return [Fixnum,nil]
10 def blob_size(filename, blob, mode = nil)
```

```
1 # Detects the size of the blob.
2 #
3 #   blob_size(filename, blob) # => some value
4 #
5 # Params:
6 # +filename+:: String filename
7 # +blob+:: String blob data
8 # +mode+:: Optional String mode (defaults to nil)
9 def blob_size(filename, blob, mode = nil)
```

```
1 # Public: Detects the size of the blob.
2 #
3 # filename - String filename
4 # blob - String blob data
5 # mode - Optional String mode (defaults to nil)
6 #
7 # Examples
8 #
9 #   blob_size(filename, blob)
10 #   # => some value
11 #
```

```
12 # Returns Fixnum or nil.
13 def blob_size(filename, blob, mode = nil)
```

But you don't have to adhere to any specific syntax. This gets an A as well:

```
1 # Returns the size of a +blob+ for a given +filename+.
2 #
3 #   blob_size(filename, blob)
4 #   # => some value
5 #
6 def blob_size(filename, blob, mode = nil)
```

Inch let's you write your documentation the way you want.

Subcommands

It comes with four sub-commands: `suggest`, `stats`, `show`, and `list`

inch suggest

Suggests places where a codebase suffers a lack of documentation. The command line exit status will be above zero when suggestions are found.

```
1 $ inch suggest
2
3 # Properly documented, could be improved:|
4
5 B ↑   Inch::CLI::Command::BaseList#prepare_list|
6 B ↑   Inch::CodeObject::Ruby::MethodParameterObject#initialize|
7 B ↗   Inch::CLI::Command::Stats#run|
8 B ↗   Inch::CLI::CommandParser#run
9
10 # Not properly documented:|
11
12 C ↑   Inch::CodeObject::NodocHelper#implicit_nodoc_comment?|
13 C ↑   Inch::CLI::Command::Output::Suggest#initialize|
14 C ↑   Inch::Rake::Suggest#initialize
15
16 # Undocumented:|
17
18 U ↑   Inch::Evaluation::ConstantObject#evaluate|
19 U ↑   Inch::Evaluation::MethodObject#evaluate|
20 U ↑   Inch::SourceParser#find_object
21
22 You might want to look at these files:|
```

```

23
24 lib/inch/code_object/proxy/base.rb|
25 lib/inch/code_object/proxy/method_object.rb|
26 lib/inch/evaluation/role/object.rb
27
28 Grade distribution (undocumented, C, B, A): ███
29
30 Only considering priority objects: ↗↘→ (use `--help` for options).

```

inch stats

Shows you an overview of the codebase.

```

1 $ inch stats
2
3 Grade distribution: (undocumented, C, B, A)
4
5 Overall: ███ 439 objects
6
7 Grade distribution by priority:↗██
8
9 10 objects↗██
10
11 302 objects→███
12
13 73 objects↘██
14
15 54 objects↓██
16
17 0 objects
18
19 Priority distribution in grades: (low to high)↘↙↗↘↗
20
21
22 U: ██████████ 243 objects
23
24 C: ██████████ 73 objects
25
26 B: ██████████ 19 objects
27
28 A: ██████████ 104 objects
29
30
31 Try `--format json|yaml` for raw numbers.

```

inch show

Shows you details about what can be improved in a specific object.

```
1 $ inch show Inch::SourceParser#find_object
2
3 # Inch::SourceParser#find_object |
4
5 -> lib/inch/source_parser.rb:16 |
6 ----- |
7 Grade: C - Needs work |
8 ----- |
9 + Add a comment describing the method |
10 + Describe the parameter 'path' |
11 + Describe the return type of 'find_object' |
12 + Add a code example (optional) |
13 -----
```

inch list

Lists all objects in your codebase with their grades.

```
1 $ inch list
2
3 # Seems really good |
4
5 A ↑ Inch::CLI::Command::Output::Console#object |
6 A ↗ Inch::CodeObject::Proxy#depth |
7 A ↗ Inch::CLI::Command::Base#description |
8 A ↗ Inch::CodeObject::NodocHelper#nodoc? |
9 ... (omitting 75 objects)
10
11 # Proper documentation present |
12
13 B ↑ Inch::CLI::Command::Suggest#run |
14 B ↑ Inch::CodeObject::Ruby::MethodParameterObject#initialize |
15 B ↗ Inch::CLI::Command::Stats#run |
16 B ↗ Inch::CLI::CommandParser#run
17
18 # Needs work |
19
20 C ↑ Inch::CodeObject::NodocHelper#implicit_nodoc_comment? |
21 C ↑ Inch::CLI::Command::Output::Console#initialize |
22 C ↑ Inch::Evaluation::ConstantObject#evaluate |
23 C ↑ Inch::SourceParser#find_object |
```

```
24 ... (omitting 248 objects)
25
26 This output omitted 323 objects. Use `--all` to display all objects.
```

Rake task

Add this to your `Rakefile`:

```
1 require 'inch/rake'
2
3 Inch::Rake::Suggest.new
```

This creates a rake task named `inch`. Change the name by passing it to the constructor. Use the `args` config option to add any command-line arguments from `inch suggest --help`.

```
1 require 'inch/rake'
2
3 Inch::Rake::Suggest.new("doc:suggest") do |suggest|
4   suggest.args << "--private"
5 end
```

Limitations

How you document your code is up to you and Inch can't actually tell you how good your docs are.

It can't tell if your code examples work or if you described parameters correctly or if you have just added `# TODO: write docs` to each and every method.

It is just a tool, that you can use to find parts in your codebase that are lacking documentation.

How is this different from ...?

Documentation coverage

Documentation coverage checks (like they can be found in `cane` and `rubocop`) look at all code objects and determine if the found documentation meets a certain threshold/expectation.

Inch takes a different approach as it aims for “properly documented” rather than “100% coverage”.

Yardstick

Yardstick is a tool that verifies documentation coverage of Ruby code and is specifically designed for YARD-style documentation. It is a great tool to see where your docs could benefit from YARD's extra

features over RDoc, but, at the same time, it is very overwhelming when applied to a codebase that does not yet adhere to YARD's standards.

Inch takes a less YARD specific, more "relaxed" approach: It recognizes different forms of documentation (even in the same codebase) and assigns grades instead of coverage measurements. So you can get an "A" rating without employing every technique YARD has to offer.

Contributing

1. Fork it!
2. Create your feature branch (`git checkout -b my-new-feature`)
3. Commit your changes (`git commit -am 'Add some feature'`)
4. Push to the branch (`git push origin my-new-feature`)
5. Create new Pull Request

Author

René Föhring (@rrrene)

Credits

Inch would not exist without Loren Segal's YARD.

License

Inch is released under the MIT License. See the LICENSE file for further details.

For YARD's licensing, see YARD's README under <http://yardoc.org/>