Gentest: Automatic Test Generation for Data Science

Stochastic Solutions

Toronto Workshop on Reproducibility, 24 February 2022

Nicholas J. Radcliffe
Stochastic Solutions Limited
& Department of Mathematics, University of Edinburgh
& Smart Data Foundry

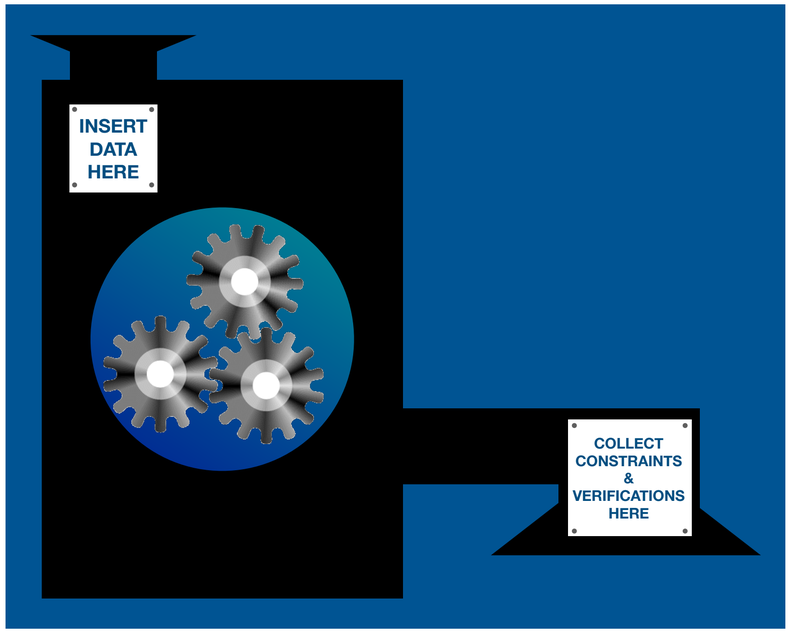
Gentest writes tests, so you don't have to™

stochasticsolutions.com/pdf/toronto2022-tdda-gentest.pdf

TEST-DRIVEN DATA ANALYSIS

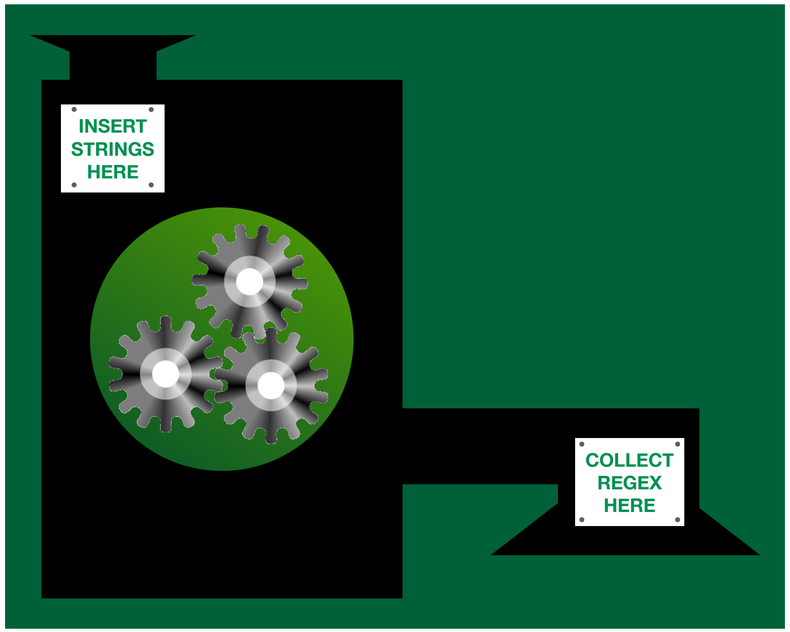
discover • verify • detect

VERIFYING DATA
WITH
CONSTRAINTS



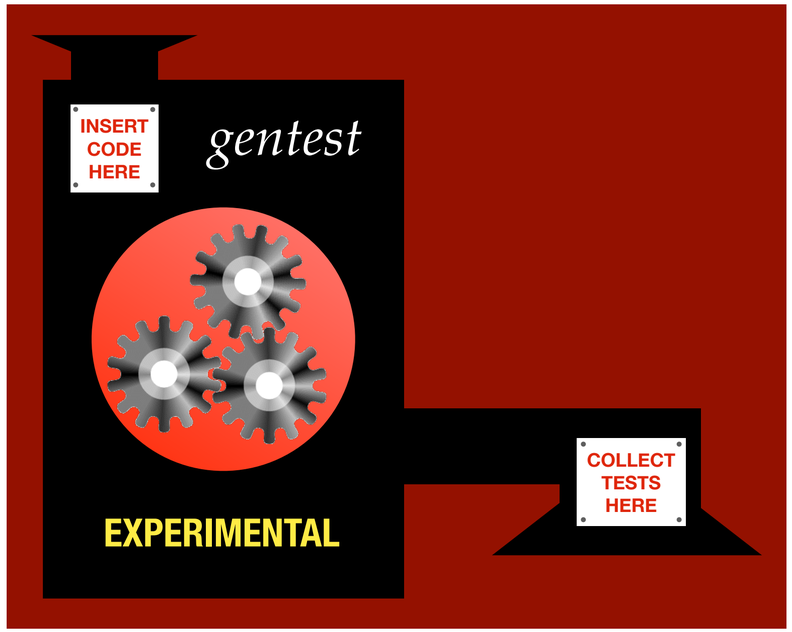
rexpy

GENERATING
REGULAR EXPRESSIONS
FROM EXAMPLES



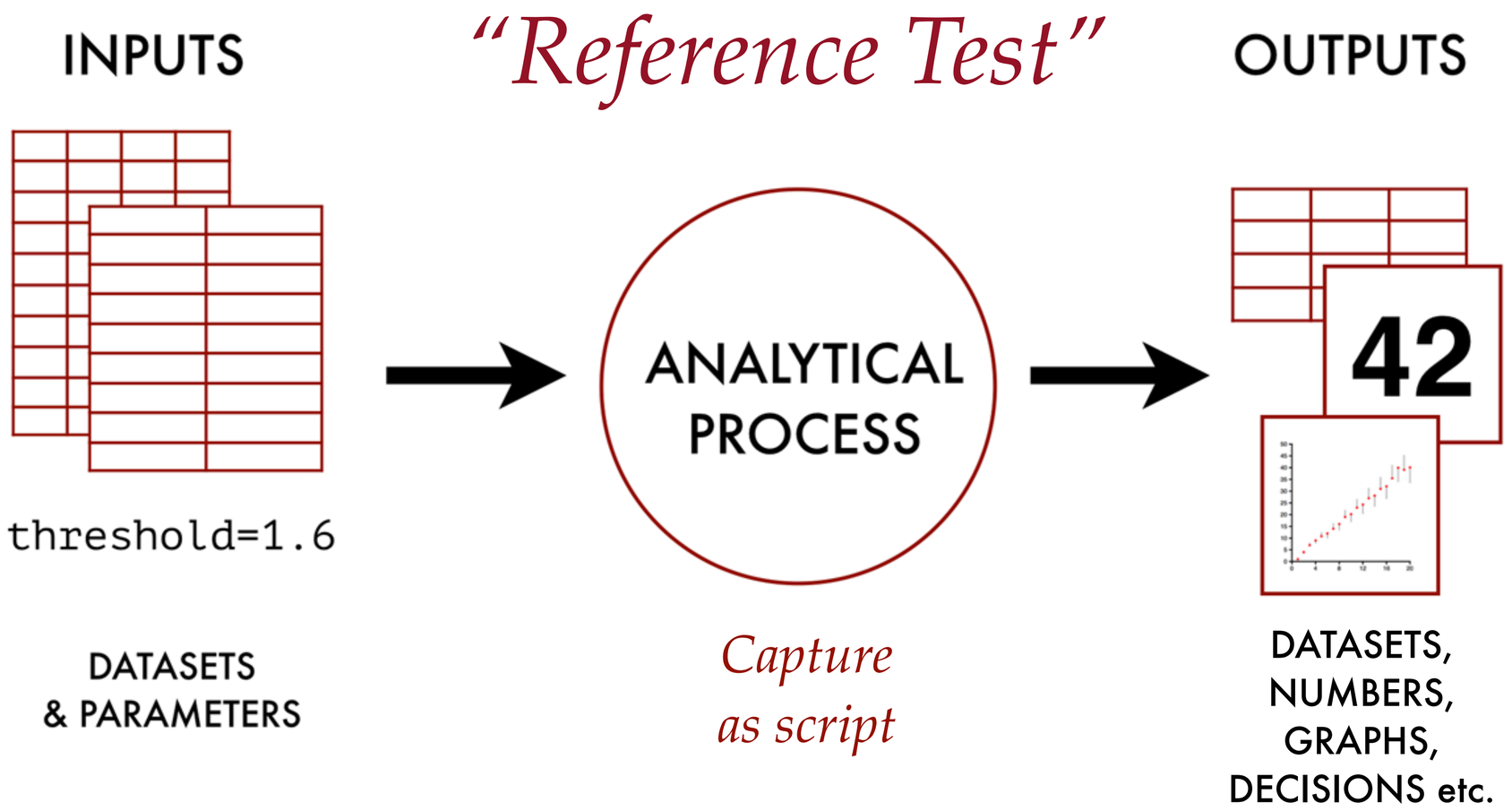
reference tests • gentest

TESTING
DATA
PROCESSES



Gentest writes tests, so you don't have to™

ANALYTICAL PROCESS



Develop a verification procedure (`diff`) *and periodically rerun:*
do the same inputs (still) produce ~~the same~~ equivalent outputs?

KINDS OF TESTS

Unit Tests — *test small units, a function etc.* **assert add(0, 0) = 0; add(2, 2) = 4**

System Tests — *test whole systems.* **assert R works**

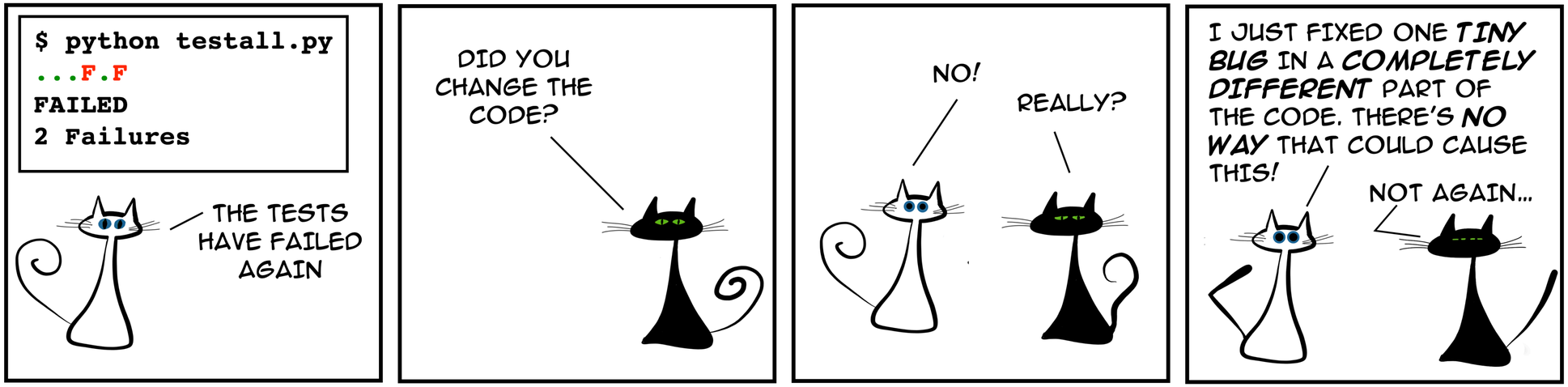
Integration Tests — *test system interoperability.* **assert R4.1.2 with Postgres 14.1**

Regression Tests — *test that what works before, still works/works the same way*
— especially things that were broken but have been fixed
assert add(2, 2) still returns 4

Reference Tests — *are mostly regression tests, with a hint of system and integration testing thrown in.*

You can also think of TDDA's constraint-based data verification as a kind of "unit tests for data" — but that's for another day

How would code spontaneously break?



etc. (Extreme Testing Cats)

tdda.info/1

E1 *Python or R changed*

E3 *OS Upgrade*

E2 *A database was upgraded*

C1-19 *The code **did** change (bug fix, upgrade etc.)*

T1 *The date changed*

S1 *You didn't fix the seed*

T1 *Disk corruption*

E25 *Password change*

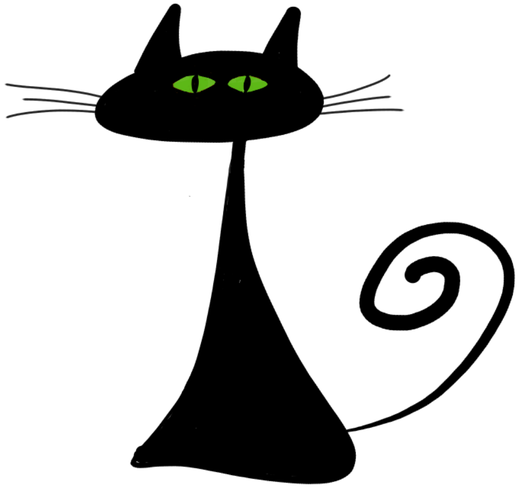
…

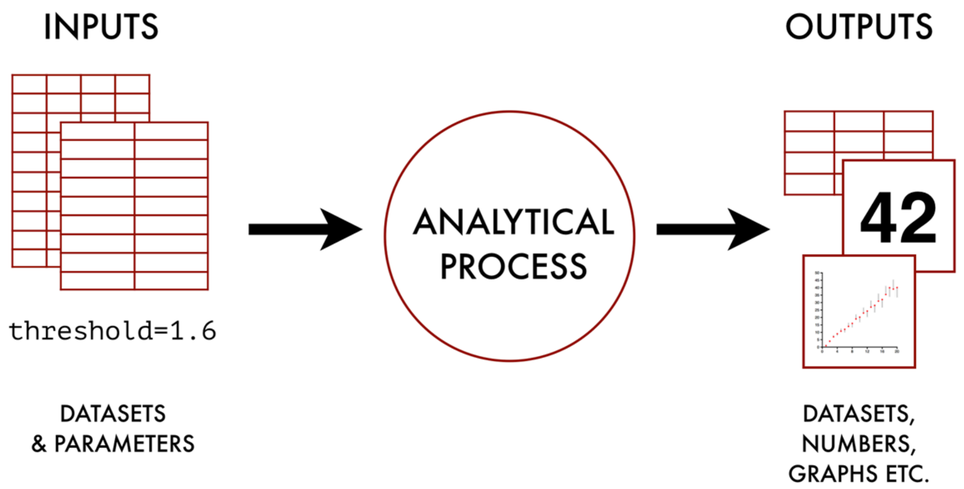
http://www.tdda.info/why-code-rusts — **77 ways and counting**

DIVISION OF LABOUR

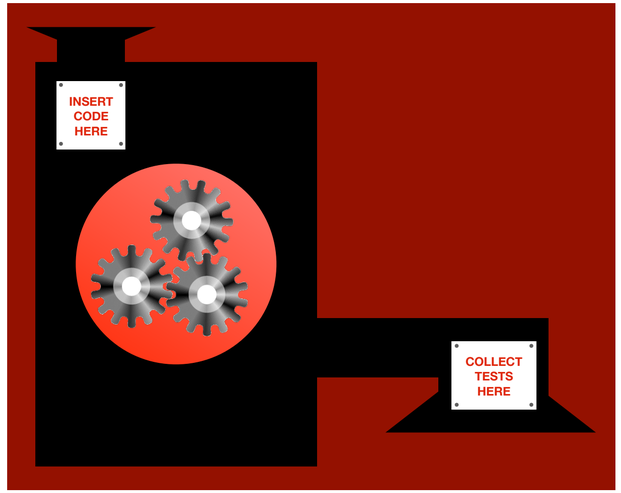
Reference Test

Data Scientist





Gentest



Assert that it works

Capture process as script

Record inputs

Gentest writes tests,
so you don't have to™

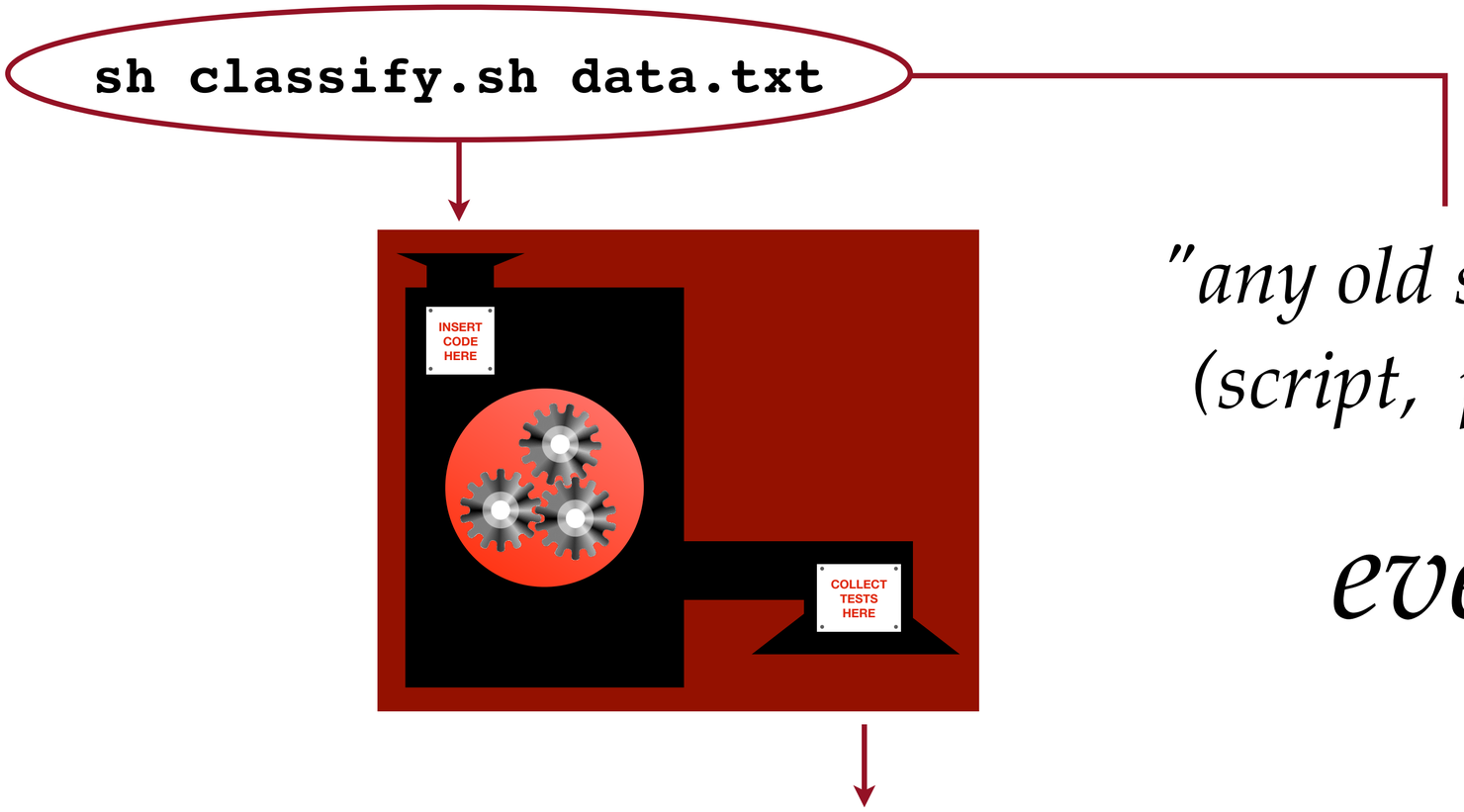
Record "reference" outputs

Develop the diff procedure

*Turn into **executable** test*

GENTEST

`tdda gentest "sh classify.sh data.txt"`



"any old shell command (script, program, ...)"

even R!

test script: **`test_sh_classify_sh_data_txt.py`**

reference outputs: **`ref/sh_classify_sh_data_txt`**

PYTHON ❤️ R ❤️ PYTHON

The Rtists and the Pythons should be friends
The Rtists and the Pythons should be friends
One of 'em uses PyPI
The other one uses CRAN, oh my
But that's no reason why they can't be friends

Data science folk should stick together
Data science folk should all be pals
Rtists work with some Python data
Pythons work with an Rtist's vals

— To the tune of *The Farmer and the Cowman,*
Hammerstein & Rogers, *Oklahoma!*

So here we go ...
to show how serious
I am, I'm going to
do a live demo of Gentest

using ... R

EXAMPLE 1:
TEXT TO SCREEN

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

```
# Set up R files for running scripts
# working directory.

# Load data into R
site.species <- read.delim("site.species.txt")
site.species.or <- read.delim("site.species.or.txt")
env.data <- read.delim("env.data.txt")
env.data.or <- read.delim("env.data.or.txt")
# refids <- read.delim("refids.or.txt")

# Merge biological and environmental data
dfmerge <- merge(site.species, env.data, by = "SITE.ID")
dfmerge.or <- merge(site.species.or, env.data.or, by = "SITE.ID")

# Set taxa.names
taxa.names <- c("ACENTRELLA", "DIPHETOR", "AMELETUS")
```

Common set-up code:

← `0-set-variables.R`

```
# Script to compute WA tolerance values

source("0-set-variables.R")

WA <- rep(NA, times = length(taxa.names))
             # Define a WA to be vector of
             # length the same as the
             # number of taxa of interest

for (i in 1:length(taxa.names)) {
  WA[i] <- sum(dfmerge[,taxa.names[i]]*dfmerge$temp)/
   sum(dfmerge[,taxa.names[i]])
}
names(WA) <- taxa.names
print(WA)
```

Common set-up code:

← `1-compute-weighted-average-tolerance-values.R`

EXAMPLE 1: TEXT TO SCREEN (CODE TO TEST)

Data & Code from
United States Environmental Protection Agency

https://www.epa.gov/caddis-vol4/download-r-scripts-and-sample-data

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 1: TEXT TO SCREEN: GENTEST OUTPUT

```
$ tdda gentest 'Rscript 1-compute-weighted-average-tolerance-values.R' one
```

Run Gentest

```
Running command 'Rscript 1-compute-weighted-average-tolerance-values.R' to generate output (run 1 of 2).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/ref/
Rscript_1_compute_weighted_average_tolerance_values_R/STDOUT.
Saved (empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/ref/
Rscript_1_compute_weighted_average_tolerance_values_R/STDERR.
```

Run 1;
capture output

```
Running command 'Rscript 1-compute-weighted-average-tolerance-values.R' to generate output (run 2 of 2).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/ref/
Rscript_1_compute_weighted_average_tolerance_values_R/2/STDOUT.
Saved (empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/ref/
Rscript_1_compute_weighted_average_tolerance_values_R/2/STDERR.
```

Run 2;
capture output

```
Test script written as /Users/njr/tmp/gentest_examples/r-examples/
test_Rscript_1_compute_weighted_average_tolerance_values_R.py
Command execution took: 0.27s
```

Write test script

```
SUMMARY:

Directory to run in:         /Users/njr/tmp/gentest_examples/r-examples
Shell command:               Rscript 1-compute-weighted-average-tolerance-values.R
Test script generated:       test_Rscript_1_compute_weighted_average_tolerance_values_R.py
Reference files: (none)
Check stdout:                yes (was 2 lines)
Check stderr:                yes (was empty)
Expected exit code:          0
Clobbering permitted:        yes
Number of times script ran: 2
```

Summary of
what Gentest did

```
$ python test_Rscript_1_compute_weighted_average_tolerance_values_R.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.268s

OK
```

Run tests:
4 passes

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

```
# -*- coding: utf-8 -*-
"""
test_Rscript_1_compute_weighted_average_tolerance_values_R.py: Automatically generated test code from tdda gentest.

Generation command:

tdda gentest 'Rscript 1-compute-weighted-average-tolerance-values.R' 'test_Rscript_1_compute_weighted_average_tolerance_values_R.py' '.'
"""

import os
import sys
import tempfile

from tdda.referencetest import ReferenceTestCase
from tdda.referencetest.gentest import exec_command

class TestRSCRIPT_1_COMPUTE_WEIGHTED_AVERAGE_TOLERANCE_VALUE(ReferenceTestCase):
    command = 'Rscript 1-compute-weighted-average-tolerance-values.R'
    cwd = os.path.abspath(os.path.dirname(__file__))
    refdir = os.path.join(cwd, 'ref', 'Rscript_1_compute_weighted_average_tolerance_values_R')

   @classmethod
    def setUpClass(cls):
        (cls.output, cls.error, cls.exception, cls.exit_code, cls.duration) = exec_command(cls.command, cls.cwd)

    def test_no_exception(self):
        self.assertIsNone(self.exception)

    def test_exit_code(self):
        self.assertEqual(self.exit_code, 0)

    def test_stdout(self):
        self.assertStringCorrect(self.output,
                                 os.path.join(self.refdir, 'STDOUT'))

    def test_stderr(self):
        self.assertStringCorrect(self.error,
                                 os.path.join(self.refdir, 'STDERR'))

if __name__ == '__main__':
    ReferenceTestCase.main()
```

EXAMPLE 1:
TEXT TO SCREEN
TEST CODE PRODUCED

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 2:
PDF GENERATION

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

```
source("0-set-variables.R")
```

1-compute-weighted-average-tolerance-values.R

```
CP <- rep(NA, times = length(taxa.names))
# Define a storage vector for the cumulative percentile
dftemp <- dfmerge[order(dfmerge$temp), ]
                                                    # Sort sites by the value
                                                    # of the environmental
                                                    # variable
cutoff <- 0.75                                      # Select a cutoff percentile
pdf('plots2.pdf')
par(mfrow = c(1,3), pty = "s")
                                                    # Specify three plots per page
for (i in 1:length(taxa.names)) {
     csum <- cumsum(dftemp[, taxa.names[i]])/
     sum(dftemp[,taxa.names[i]])
                                                    # Compute cumulative sum
                                                    # of abundances
     plot(dftemp$temp, csum, type = "l", xlab = "Temperature",
          ylab = "Proportion of total", main = taxa.names[i])
                                                    # Make plots like Figure 5
     ic <- 1
     while (csum[ic] < 0.75) ic <- ic + 1
                                                    # Search for point at which
                                                    # cumulative sum is 0.75
     CP[i] <- dftemp$temp[ic]
                                                    # Save the temperature that
                                                    # corresponds to this
                                                    # percentile.
     }
names(CP) <- taxa.names
print(CP)
dev.off()
```

EXAMPLE 2: GENERATE PDF GRAPH

EXAMPLE 2: PDF GENERATION

```
$ tdda gentest
```

```
Enter shell command to be tested: Rscript 2-compute-cumulative-percentiles.R
Enter name for test script [test_Rscript_2_compute_cumulative_percentiles_R]: two
Check all files written under $(pwd)?: [y]:
Check all files written under (gentest's) $TMPDIR?: [y]:
Enter other files/directories to be checked, one per line, then a blank line:

Check stdout?: [y]:
Check stderr?: [y]:
Exit code should be zero?: [y]:
Clobber (overwrite) previous outputs (if they exist)?: [y]:
Number of times to run script?: [2]: 4
```

```
Running command 'Rscript 2-compute-cumulative-percentiles.R' to generate output (run 1 of 4).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/ref/two/STDOUT.
Saved (empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/ref/two/STDERR.
Copied $(pwd)/plots2.pdf to $(pwd)/ref/two/plots2.pdf
```

```
Running command 'Rscript 2-compute-cumulative-percentiles.R' to generate output (run 2 of 4).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/ref/two/2/STDOUT.
Saved (empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/ref/two/2/STDERR.
Copied $(pwd)/plots2.pdf to $(pwd)/ref/two/2/plots2.pdf
```

```
Running command 'Rscript 2-compute-cumulative-percentiles.R' to generate output (run 3 of 4).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/ref/two/3/STDOUT.
Saved (empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/ref/two/3/STDERR.
Copied $(pwd)/plots2.pdf to $(pwd)/ref/two/3/plots2.pdf
```

```
Running command 'Rscript 2-compute-cumulative-percentiles.R' to generate output (run 4 of 4).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/ref/two/4/STDOUT.
Saved (empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/ref/two/4/STDERR.
Copied $(pwd)/plots2.pdf to $(pwd)/ref/two/4/plots2.pdf
```

Run Gentest Wizard (no parameters)

Wizard dialogue
Accept most defaults, but change:
— base name to `two`
— number of runs to `4`

Deliberately slow down (see later for why)

Run 1; capture output

Run 2; capture output

Run 3; capture output

Run 4; capture output

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 2: PDF GENERATION

```
Test script written as /Users/njr/tmp/gentest_examples/r-examples/test_two.py
Command execution took: 0.36s

SUMMARY:

Directory to run in:         /Users/njr/tmp/gentest_examples/r-examples
Shell command:               Rscript 2-compute-cumulative-percentiles.R
Test script generated:       test_two.py
Reference files:
    $(pwd)/plots2.pdf
Check stdout:                yes (was 4 lines)
Check stderr:                yes (was empty)
Expected exit code:          0
Clobbering permitted:        yes
Number of times script ran: 4
```

Summary of what Gentest did

```
$ python test_two.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.351s

OK
```

Run tests: 4 passes

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 2: PDF GENERATION TEST CODE PRODUCED

```
# -*- coding: utf-8 -*-
"""
test_two.py: Automatically generated test code from tdda gentest.

Generation command:

tdda gentest 'Rscript 2-compute-cumulative-percentiles.R' 'test_two.py' '.'
"""
# <imports omitted>
class Test(ReferenceTestCase):
    command = 'Rscript 2-compute-cumulative-percentiles.R'
    cwd = os.path.abspath(os.path.dirname(__file__))
    refdir = os.path.join(cwd, 'ref', 'two')

    generated_files = [
        os.path.join(cwd, 'plots2.pdf')
    ]
    @classmethod
    def setUpClass(cls):
        for path in cls.generated_files:
            if os.path.exists(path):
                os.unlink(path)
        (cls.output, cls.error, cls.exception, cls.exit_code, cls.duration) = exec_command(cls.command, cls.cwd)

    def test_no_exception(self):
        self.assertIsNone(self.exception)

    def test_exit_code(self):
        self.assertEqual(self.exit_code, 0)

    def test_stdout(self):
        self.assertStringCorrect(self.output,
                                 os.path.join(self.refdir, 'STDOUT'))
    def test_stderr(self):
        self.assertStringCorrect(self.error,
                                 os.path.join(self.refdir, 'STDERR'))
    def test_plots2_pdf(self):
        patterns = [r'^/[A-Z][a-z]+[A-Z][a-z]{3} \(D:[0-9]{14}\)$']
        self.assertTextFileCorrect(os.path.join(self.cwd, 'plots2.pdf'), os.path.join(self.refdir, 'plots2.pdf'),
                                   ignore_patterns=patterns, encoding='iso-8859-1')

if __name__ == '__main__':
    ReferenceTestCase.main()
```

Here, Gentest has noticed that there is some run-to-run variation, and has used a test that ignores parts of the PDF that are change.

In this case, it used TDDA's rexpy functionality to generate a regular expression that characterises the lines to ignore.

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 2: PDF GENERATION

Why did we run 4 times in this case?

- Gentest is *not* trying to ensure that the results are identical every time, but rather that they are consistent
  - Very often, output is different each time. As a trivial example, if the output includes the time the code was run, that will be different each time. Ideally, Gentest will notice this and write a test that ignores normal, unimportant variation.
- In this case, the output actually does include a timestamp, to the second, in the PDF, but if you only run it twice, the timestamp might be identical each time. Here's an example of the output when that happens

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 2: GENERATE CODE IF THINGS RUN TOO FAST

```
# -*- coding: utf-8 -*-
"""
test_two.py: Automatically generated test code from tdda gentest.

Generation command:

tdda gentest 'Rscript 2-compute-cumulative-percentiles.R' 'test_two.py' '.'
"""
# <imports omitted>
class Test(ReferenceTestCase):
    command = 'Rscript 2-compute-cumulative-percentiles.R'
    cwd = os.path.abspath(os.path.dirname(__file__))
    refdir = os.path.join(cwd, 'ref', 'two')

    generated_files = [
        os.path.join(cwd, 'plots2.pdf')
    ]
    @classmethod
    def setUpClass(cls):
        for path in cls.generated_files:
            if os.path.exists(path):
                os.unlink(path)
        (cls.output, cls.error, cls.exception, cls.exit_code, cls.duration) = exec_command(cls.command, cls.cwd)

    def test_no_exception(self):
        self.assertIsNone(self.exception)

    def test_exit_code(self):
        self.assertEqual(self.exit_code, 0)

    def test_stdout(self):
        self.assertStringCorrect(self.output,
                                 os.path.join(self.refdir, 'STDOUT'))
    def test_stderr(self):
        self.assertStringCorrect(self.error,
                                 os.path.join(self.refdir, 'STDERR'))
    def test_plots2_pdf(self):
        self.assertTextFileCorrect(os.path.join(self.cwd, 'plots2.pdf'), os.path.join(self.refdir, 'plots2.pdf'),
                                   encoding='iso-8859-1')


if __name__ == '__main__':
    ReferenceTestCase.main()
```

In this case, because Gentest saw no variation,
it writes a simpler test that will, in practice fail

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

If we run this version, the `plots2.pdf` *test fails, because the new PDF differs from the test version saved.*

Gentest offers a `diff` *command to see this.*

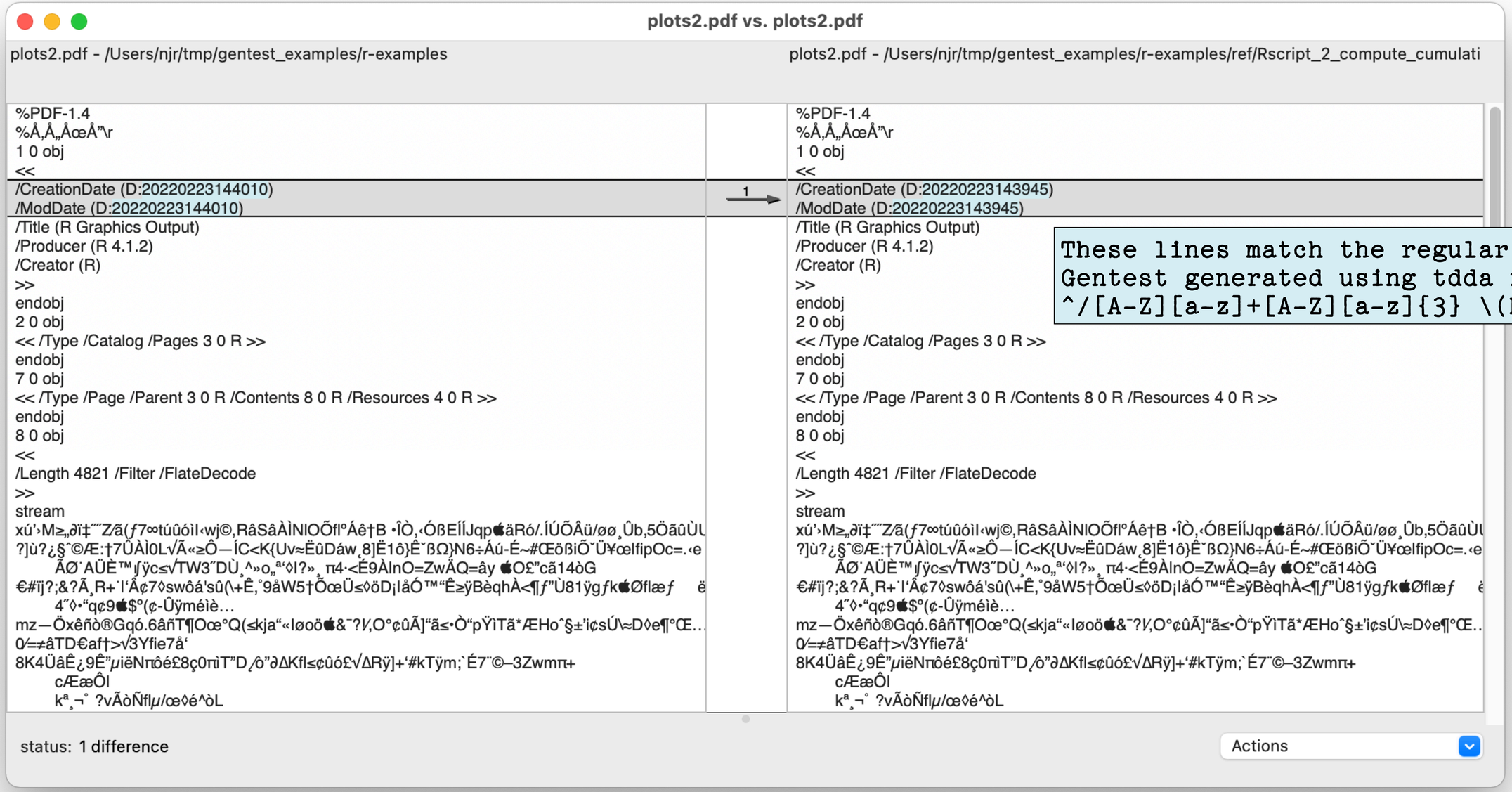
```
$ python test_two.py
..2 lines are different, starting at line 5
Compare with:
    diff /Users/njr/tmp/gentest_examples/r-examples/plots2.pdf /Users/njr/tmp/gentest_examples/r-examples/ref/
Rscript_2_compute_cumulative_percentiles_R/plots2.pdf

F..
======================================================================
FAIL: test_plots2_pdf (__main__.TestRSCRIPT_2_COMPUTE_CUMULATIVE_PERCENTILE)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_two.py", line 54, in test_plots2_pdf
    encoding='iso-8859-1')
AssertionError: False is not true : 2 lines are different, starting at line 5
Compare with:
    diff /Users/njr/tmp/gentest_examples/r-examples/plots2.pdf /Users/njr/tmp/gentest_examples/r-examples/ref/
Rscript_2_compute_cumulative_percentiles_R/plots2.pdf


----------------------------------------------------------------------
Ran 5 tests in 0.344s

FAILED (failures=1)
```

EXAMPLE 2: TEST FAILURE

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

```
diff /Users/njr/tmp/gentest_examples/r-examples/plots2.pdf\
     /Users/njr/tmp/gentest_examples/r-examples/ref/\
      Rscript_2_compute_cumulative_percentiles_R/plots2.pdf
```

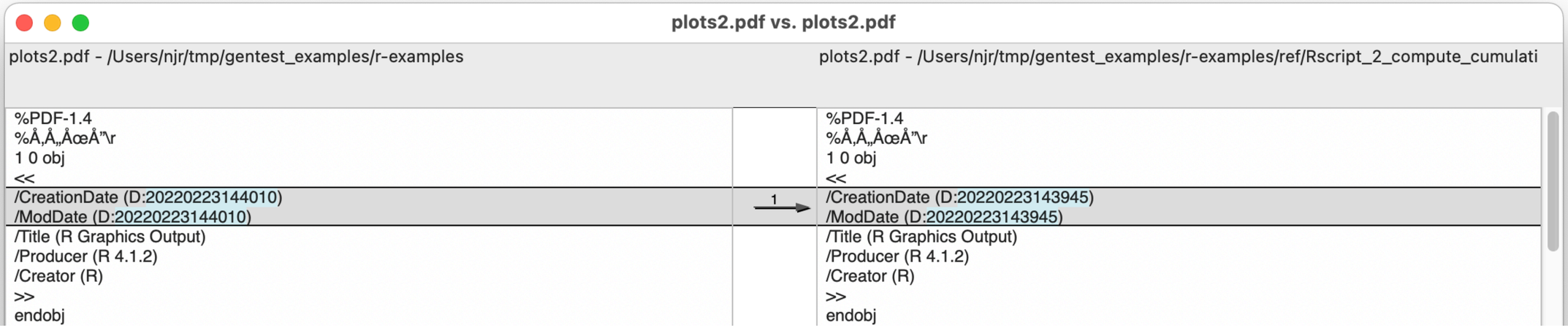
EXAMPLE 2: A VISUAL DIFF OF THE PDF



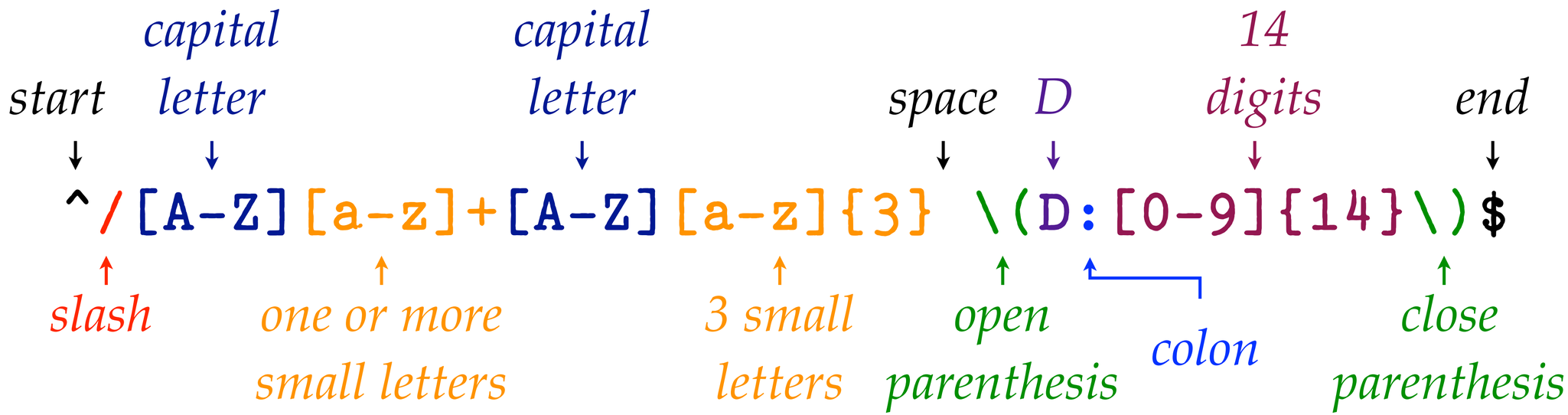
REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

```
diff /Users/njr/tmp/gentest_examples/r-examples/plots2.pdf\
     /Users/njr/tmp/gentest_examples/r-examples/ref/\
      Rscript_2_compute_cumulative_percentiles_R/plots2.pdf
```

EXAMPLE 2: A VISUAL DIFF OF THE PDF



These lines match the regular expression Gentest generated using tdda rexpy:



REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 3: RMARKDOWN

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 3: RMARKDOWN

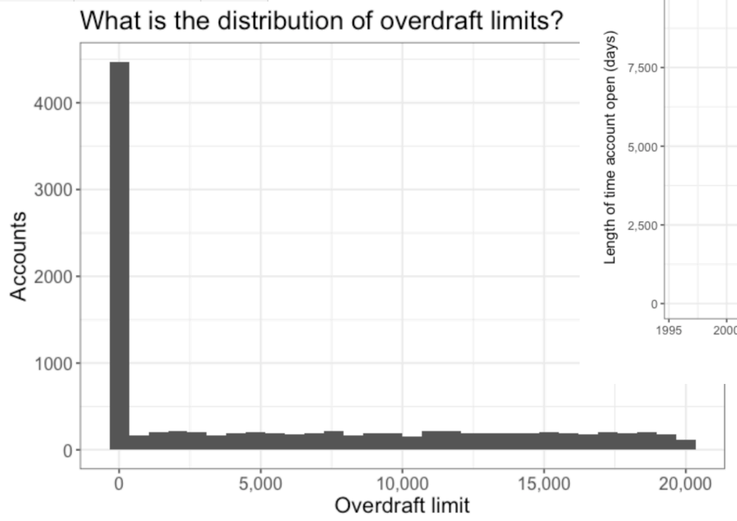
2. Produce a cross-table showing the number of account holders broken down by title and account_type (i.e. a count of the number of people with each combination of possible account types and titles)

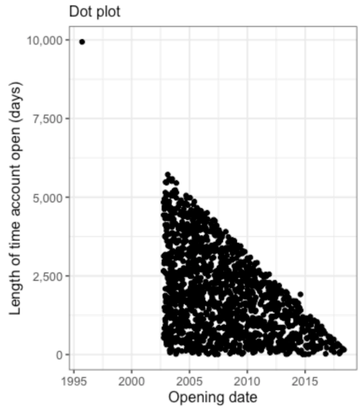
Duplication: The email address grory2@yahoo.com appears twice, but the surnames are different. I have assumed this is an artefact of the data synthesizing, or a parent/child, the entries both remain.

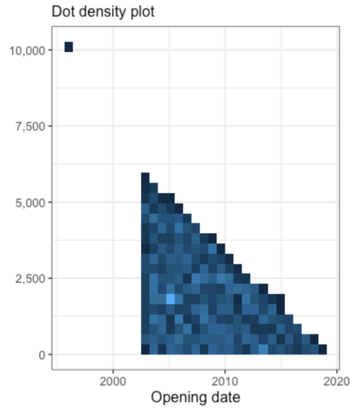
```
df_acc %>%
  count(title, account_type) %>%
  pivot_wider(names_from = account_type, values_from = n, values_fill = 0) %>%
  kable(format.args = list(big.mark = ","))
```

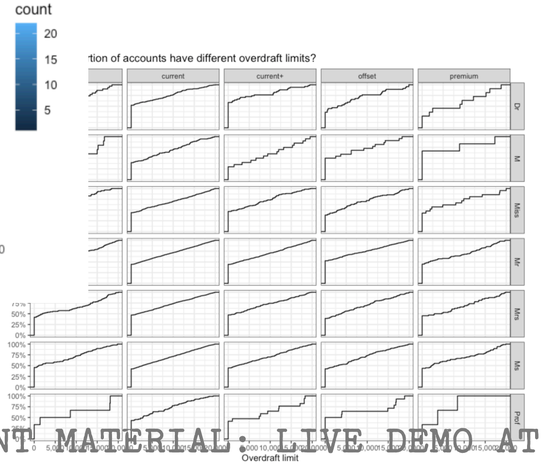
title	basic	current	current+	offset	premium	NA
Dr	31	259	51	47	11	0
M	10	149	20	22	6	1
Miss	58	508	120			
Mr	237	2,903	606			
Mrs	110	1,216	252			
Ms	136	1,560	300			
Prof	6	93	17			

title	n
Mr	4,348
Ms	2,310
Mrs	1,802









The final example is a full RMarkdown code written by Mike R Spencer (@MikeRSpencer) for an interview exercise.

The essentially unmodified code was run through Gentest. It produces 2 markdown files, 3 HTML files and 4 PNG images.*

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 3: RMARKDOWN WIZARD

```
$ tdda gentest
Enter shell command to be tested: Rscript generate_webpages.R
Enter name for test script [test_Rscript_generate_webpages_R]: mikermd
Check all files written under $(pwd)?: [y]:
Check all files written under (gentest's) $TMPDIR?: [y]:
Enter other files/directories to be checked, one per line, then a blank line:

Check stdout?: [y]:
Check stderr?: [y]:
Exit code should be zero?: [y]:
Clobber (overwrite) previous outputs (if they exist)?: [y]:
Number of times to run script?: [2]:
```

Run Gentest wizard

```
Running command 'Rscript generate_webpages.R' to generate output (run 1 of 2).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/mike/ref/mikermd/
STDOUT.
Saved (non-empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/mike/ref/mikermd/
STDERR.
Copied $(pwd)/gofcoe_exercise_1.html to $(pwd)/ref/mikermd/gofcoe_exercise_1.html
Copied $(pwd)/gofcoe_exercise_1.md to $(pwd)/ref/mikermd/gofcoe_exercise_1.md
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5 hist-1.png to $(pwd)/ref/mikermd/E1Q5
hist-1.png
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5-1.png to $(pwd)/ref/mikermd/E1Q5-1.png
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5a-1.png to $(pwd)/ref/mikermd/E1Q5a-1.png
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q6-1.png to $(pwd)/ref/mikermd/E1Q6-1.png
Copied $(pwd)/gofcoe_exercise_2.html to $(pwd)/ref/mikermd/gofcoe_exercise_2.html
Copied $(pwd)/gofcoe_exercise_2.md to $(pwd)/ref/mikermd/gofcoe_exercise_2.md
```

Run 1;
capture output

```
Running command 'Rscript generate_webpages.R' to generate output (run 2 of 2).
Saved (non-empty) output to stdout to /Users/njr/tmp/gentest_examples/r-examples/mike/ref/mikermd/
2/STDOUT.
Saved (non-empty) output to stderr to /Users/njr/tmp/gentest_examples/r-examples/mike/ref/mikermd/
2/STDERR.
Copied $(pwd)/gofcoe_exercise_1.html to $(pwd)/ref/mikermd/2/gofcoe_exercise_1.html
Copied $(pwd)/gofcoe_exercise_1.md to $(pwd)/ref/mikermd/2/gofcoe_exercise_1.md
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5 hist-1.png to $(pwd)/ref/mikermd/2/E1Q5
hist-1.png
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5-1.png to $(pwd)/ref/mikermd/2/E1Q5-1.png
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5a-1.png to $(pwd)/ref/mikermd/2/E1Q5a-1.png
Copied $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q6-1.png to $(pwd)/ref/mikermd/2/E1Q6-1.png
Copied $(pwd)/gofcoe_exercise_2.html to $(pwd)/ref/mikermd/2/gofcoe_exercise_2.html
Copied $(pwd)/gofcoe_exercise_2.md to $(pwd)/ref/mikermd/2/gofcoe_exercise_2.md
```

Run 2;
capture output

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 3: RMARKDOWN

```
Test script written as /Users/njr/tmp/gentest_examples/r-examples/mike/test_mikermd.py
Command execution took: 10.73s

SUMMARY:

Directory to run in:         /Users/njr/tmp/gentest_examples/r-examples/mike
Shell command:               Rscript generate_webpages.R
Test script generated:       test_mikermd.py
Reference files:
    $(pwd)/gofcoe_exercise_1.html
    $(pwd)/gofcoe_exercise_1.md
    $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5 hist-1.png
    $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5-1.png
    $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q5a-1.png
    $(pwd)/gofcoe_exercise_1_files/figure-gfm/E1Q6-1.png
    $(pwd)/gofcoe_exercise_2.html
    $(pwd)/gofcoe_exercise_2.md
Check stdout:                yes (was 258 lines)
Check stderr:                yes (was 73 lines)
Expected exit code:          0
Clobbering permitted:        yes
Number of times script ran: 2
```

Gentest output summary

```
$ python test_mikermd.py
............
----------------------------------------------------------------------
Ran 12 tests in 10.090s

OK
```

Run 1;
capture output

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

EXAMPLE 3: RMARKDOWN

Yes, but is it actually testing anything?

title	n
Mr	4,348
Ms	2,310
Mrs	1,802

*Fragment of **reference** HTML output*

Change one person in the source data from Mr to Ms.
(Could equally change the code.)

Changes an image `E1Q5a_1_png`

Changes a Markdown file `gofcoe_exercise_1_md`

Changes an HTML file `gofcoe_exercise_1_html`

title	n
Mr	4,347
Ms	2,311
Mrs	1,802

*Corresponding fragment of **actual** HTML output*

```
$ python test_Rscript_generate_webpages_R.py
..First difference at byte offset 62684, actual length 139351, expected length 139307.
F..First difference at byte offset 13234, actual length 411361, expected length 411301.
F12 lines are different, starting at line 32
Compare with:
    diff /Users/njr/tmp/gentest_examples/r-examples/mike/gofcoe_exercise_1.md /Users/
njr/tmp/gentest_examples/r-examples/mike/ref/Rscript_generate_webpages_R/
gofcoe_exercise_1.md

F.....
======================================================================
FAIL: test_E1Q5a_1_png (__main__.TestRSCRIPT_GENERATE_WEBPAGE)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_Rscript_generate_webpages_R.py", line 76, in test_E1Q5a_1_png
    os.path.join(self.refdir, 'E1Q5a-1.png'))
AssertionError: False is not true : First difference at byte offset 62684, actual length
139351, expected length 139307.

======================================================================
FAIL: test_gofcoe_exercise_1_html (__main__.TestRSCRIPT_GENERATE_WEBPAGE)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_Rscript_generate_webpages_R.py", line 60, in test_gofcoe_exercise_1_html
    os.path.join(self.refdir, 'gofcoe_exercise_1.html'))
AssertionError: False is not true : First difference at byte offset 13234, actual length
411361, expected length 411301.

======================================================================
FAIL: test_gofcoe_exercise_1_md (__main__.TestRSCRIPT_GENERATE_WEBPAGE)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_Rscript_generate_webpages_R.py", line 64, in test_gofcoe_exercise_1_md
    os.path.join(self.refdir, 'gofcoe_exercise_1.md'))
AssertionError: False is not true : 12 lines are different, starting at line 32
Compare with:
    diff /Users/njr/tmp/gentest_examples/r-examples/mike/gofcoe_exercise_1.md /Users/
njr/tmp/gentest_examples/r-examples/mike/ref/Rscript_generate_webpages_R/
gofcoe_exercise_1.md

----------------------------------------------------------------------
Ran 12 tests in 9.214s

FAILED (failures=3)
```

REPLACEMENT MATERIAL: LIVE DEMO AT WORKSHOP

Reference tests don't really check
that the code or analyses are correct.

I describe Gentest as "minimally artificially intelligent"

References test check that what your
code did when you were happy with it previously,
it still does when you run the tests.

. . . which is ***really*** *useful*

WHAT HAVE WE SEEN?

- Automatic Test Creation
- STDOUT, STDERR, Spot generated files
- Exit codes, Error-free running
- Text Files (variable output; regular expressions, substrings etc.; different encodings)
- Image Files
- CSV Files
- Context Detection
- Arbitrary language (even R!)
- Context: host, date, user, directory etc.
- Wizard
- Regeneration recipe

WHAT'S MISSING?

- Doesn't cope well if file names change
- Doesn't yet do anything very clever with dataset output (CSV, DataFrame etc.)
- No database support yet
- Doesn't combine different tests yet
- All a bit brittle

"Coming real soon" (!)

- Notebooks? GUIs? Perhaps never (though maybe that's not all bad for production systems)
- Requires Python to run.
  - Could, in principle write some tests in R (**`testthat`**? **`tinytest`**?)
  - But handling variable output would be harder

Don't hold your breath

TDDA LIBRARY

Install from PyPI (recommended)

```
python -m pip install tdda
```

or from Github (source)

```
git clone https://github.com/tdda/tdda.git
python setup.py install
```

TDDA LIBRARY

- Runs on Python, Mac, Linux & Windows, under `unittest` and `pytest`
- MIT Licensed
- Documentation:
  - Sphinx source in `doc` subdirectory
  - Built copy at `http://tdda.readthedocs.io`
- Quick reference:

  `http://www.tdda.info/pdf/tdda-quickref.pdf`

stochasticsolutions.com • smartdatafoundry.com

http://tdda.info

https://github.com/tdda

Gentest writes tests,
so you don't have to™

njr@StochasticSolutions.com

https://linkedin.com/in/njradcliffe

#tdda*

* *tweet (DM) us your email address for invitation. Or email me.*

@tdda0 @njr0 @StochasticSolns @SmartDataFdry

Correct interpretation: Zero (Error of interpretation: Letter "Oh")