The importance of testing R code

Nora M. Villanueva

Galicia Research Center in Advanced Telecommunications

V Xornadas de usuarios de R Galicia

Santiago de Compostela, Spain. 25 October 2018

gradiant
Connectivity · Intelligence · Security
for your business

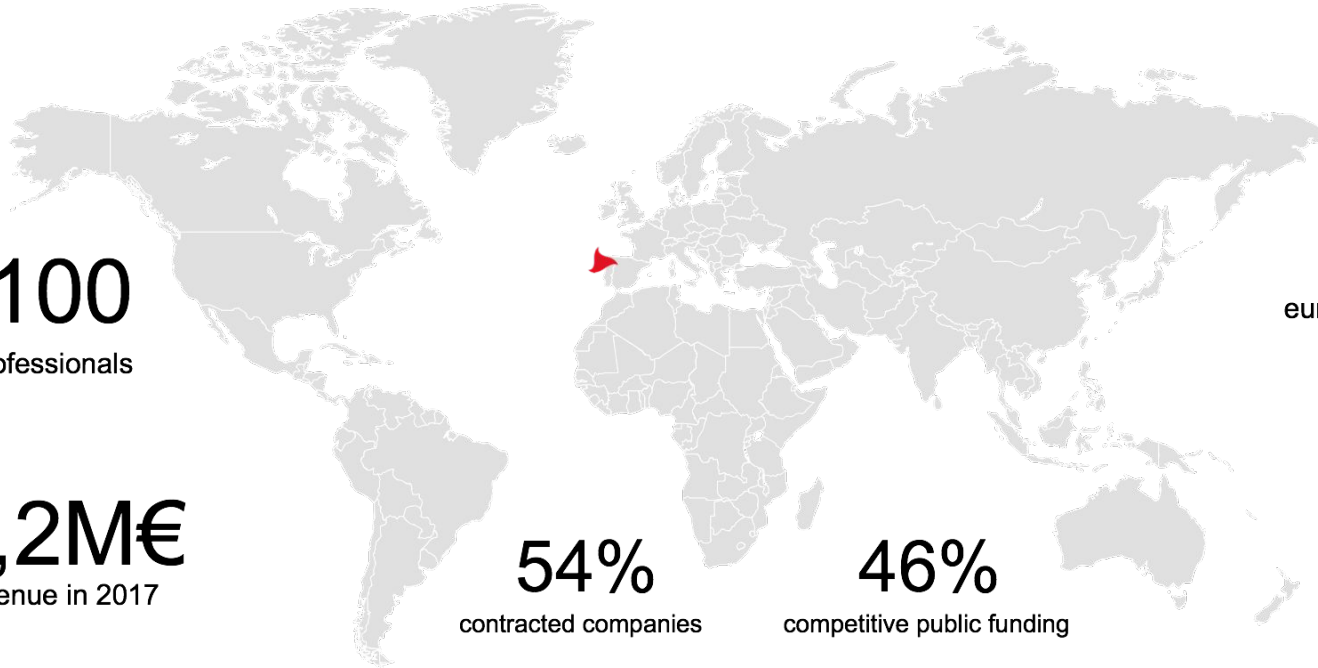
[+34] 986 120 430 | gradiant@gradiant.org | www.gradiant.org

Outline

1. Gradiant
2. Software development
3. Software testing
    - ➢ Unit tests
4. References

Gradiant, ICT technology centre in Spain

Since 2008, focused on technological development and knowledge transfer to industry



+100
professionals

12
european projects

5,2M€
revenue in 2017

54%
contracted companies

46%
competitive public funding

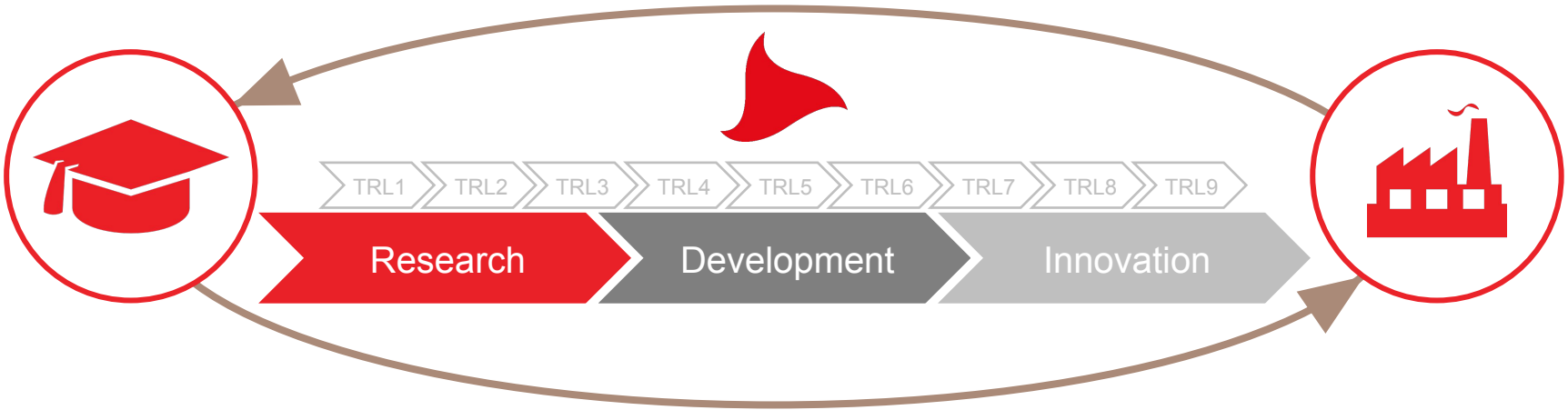
www.gradiant.org


gradiant

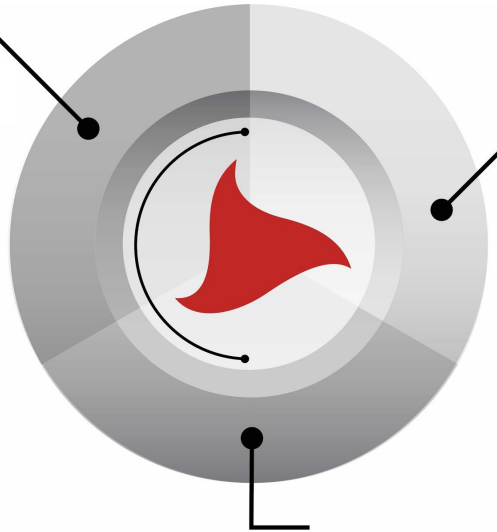
Our model

Focused on industry goals

Same language, flexibility, competitiveness, profitability



Focus on...



Connectivity

- Communication Subsystems
- IoT (Internet of Things)
- Integrated and Onboard Systems
- Networks
- Wireless Communications

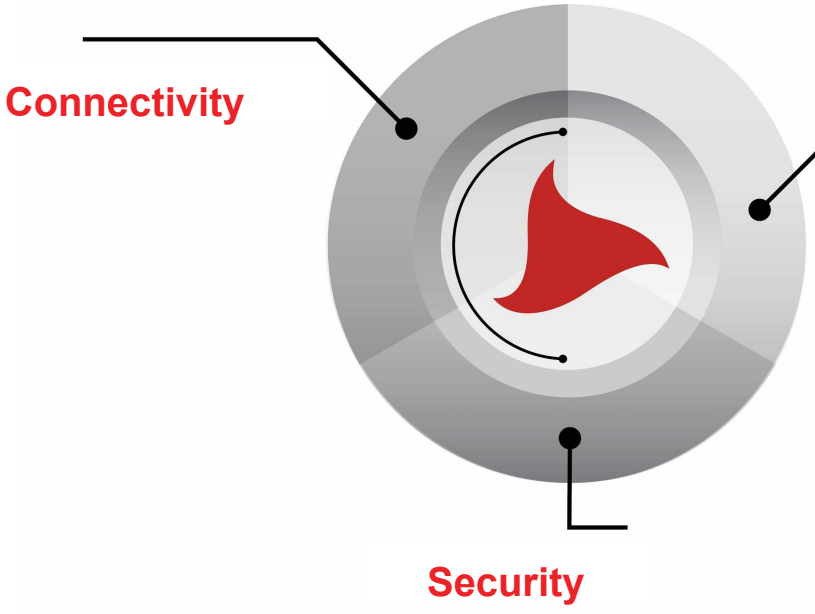
Intelligence

- Data Analytics and Big Data
- Intelligent video Analysis
- Learning Analytics and Adaptive Learning
- Bioinformatics

Security

- Biometric Systems
- Multimedia Security
- Cloud Security
- Privacy by Design
- Privacy Protection Systems

Focus on...



eLearning@Gradiant:

- Team consists of 13 members
- International projection
- Working with several publishers
- Learning Analytics & Talent Management
- Adaptive Learning & Adaptive Instruction
- Gamification & Intelligent Tutoring

netex

UMI Gradiant with Netex

smartED product

netex

gradiant
Connectivity · Intelligence · Security
for your business

Software Development

(+34) 986 120 430 | www.gradiant.org

Software development

- **Academy contributes with new findings to society.**

- **Company develops a product**


Time


Bugs


Money


Multidisciplinary Team working together

Software development. Good practices

- **Pair Programming**
- **Refactoring**
- **Code Reviews**
- **Continuous integration**
- **Test Driven Development**

Extreme Programming

Bugs, money, time

gradiant
Connectivity · Intelligence · Security
for your business

Software Testing

(+34) 986 120 430 | www.gradiant.org

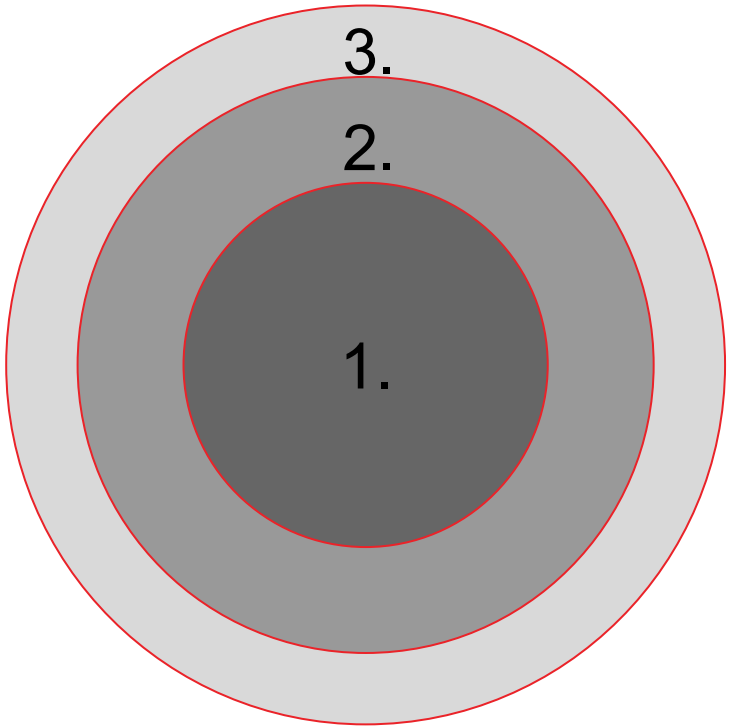
Software Testing

Objectives

- Attempt to execute a program or app with the intent of finding software bugs
- Provide objective, independent information about the quality of software and risk of its failure to users.

Levels of Testing

1. Unit Tests
2. Integration Tests
3. System Tests

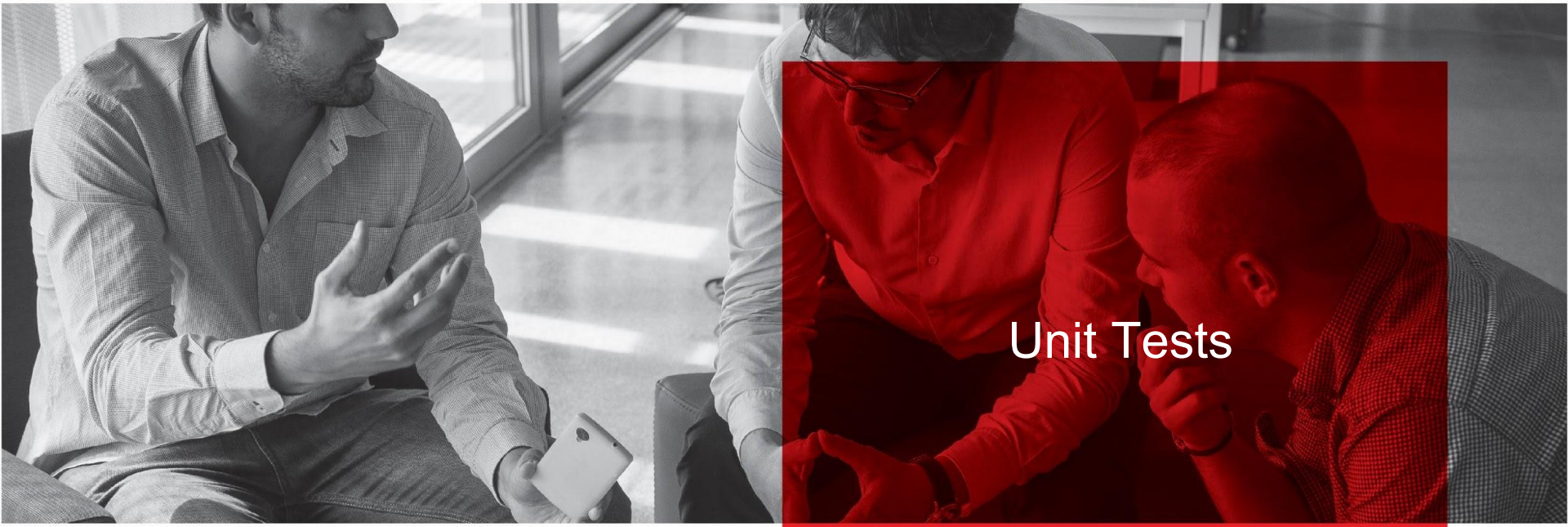


When should one do software testing?

1. Test Driven Development (TDD)

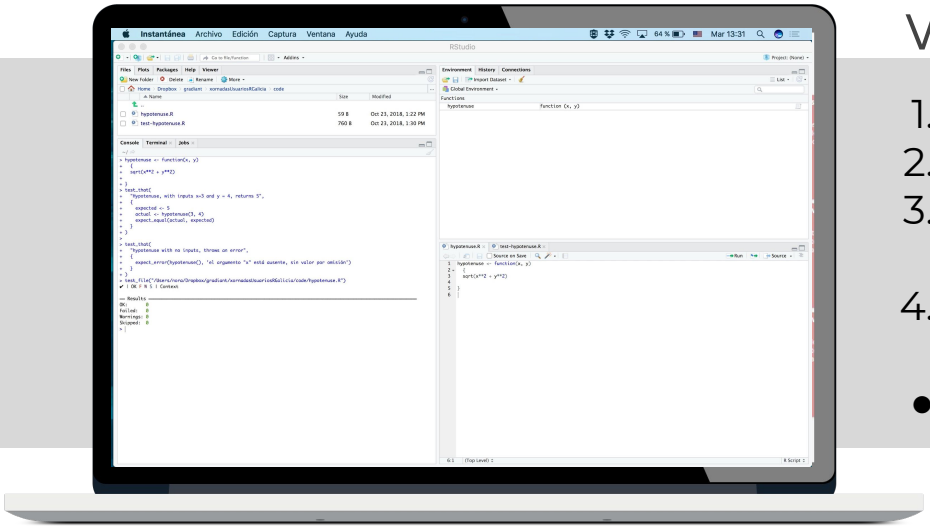
2. Test After Development (TAD)

gradiant
Connectivity · Intelligence · Security
for your business



Unit Tests

Using Unit Tests. Tools and examples



Workflow

1. Write a function
2. Load it
3. Experiment with it in the console to see if it works
4. Repeat

- Recommend testthat package

Using Unit Tests. testthat package

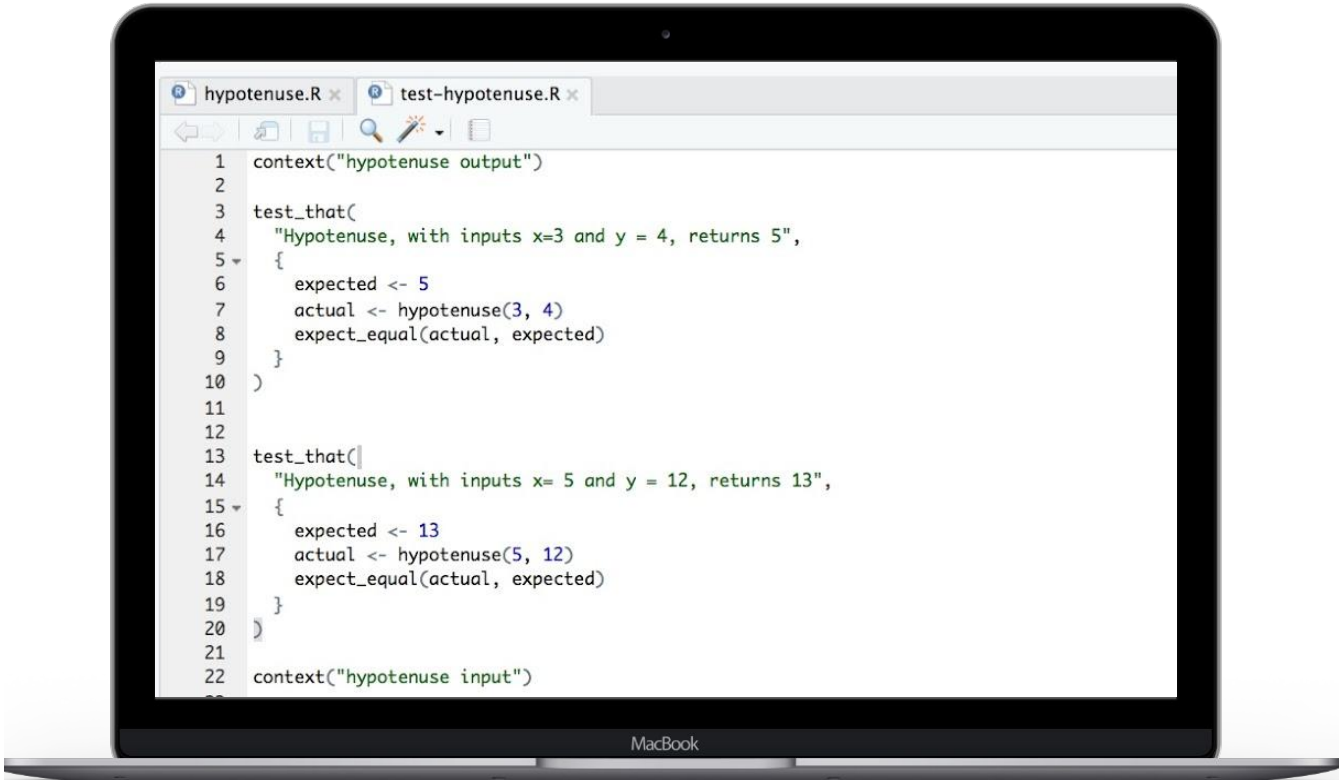
```
context("hypotenuse")

test_that(
  "Hypotenuse, with inputs x=3 and y = 4, returns 5",
  {
    expected <- 5
    actual <- hypotenuse(3, 4)
    expect_equal(actual, expected)
  }
)
```

```
hypotenuse <- function(x, y)
  {
  sqrt(x**2 + y**2)
}
```

Tests are organised hierarchically: **expectations** are grouped into **tests** which are organised in **files**

Ideally, **tests** should be **written once and run many times**.

Using Unit Tests. Tools



Using Unit Tests. testthat package

```
✔ | OK F W S | Context
✔ |  1       | hypotenuse output

══ Results ═══════════════════════════════════════════════════════════════════
OK:       1
Failed:   0
Warnings: 0
Skipped:  0
```


Using Unit Tests. Different expectations

- **expect_true, expect_false, expect_null:** for common return types
- **expect_length:** the length of a vector
- **expect_gt, expect_lt :** for numeric inequalities
- **expect_named:** for the names of variables
- **expect_type, expect_is**: for the type/class of variables
- **expect_error, expect_warning, expect_message:** for testing feedback

Using Unit Tests. Different expectations

```
context("hypotenuse")

test_that(
  "Hypotenuse, with inputs x=3 and y = 4, returns 5",
  {
    expected <- 5
    actual <- hypotenuse(3, 4)
    expect_equal(actual, expected)
  }
)

test_that(
  "hypotenuse with no inputs, throws an error",
  {
    expect_error(hypotenuse(), 'el argumento "x" está ausente, sin valor por omisión')
  }
)
```

```
hypotenuse <- function(x, y)
  {
  sqrt(x**2 + y**2)
}

```


Using Unit Tests. Organising tests using contexts

```
context("hypotenuse")

test_that(
  "Hypotenuse, with inputs x=3 and y = 4, returns 5",
  {
    expected <- 5
    actual <- hypotenuse(3, 4)
    expect_equal(actual, expected)
  }
)

test_that(
  "hypotenuse with no inputs, throws an error",
  {
    expect_error(hypotenuse(), 'el argumento "x" está ausente, sin valor por omisión')
  }
)
```

```
hypotenuse <- function(x, y)
  {
  sqrt(x**2 + y**2)
}
```


Using Unit Tests. Running test

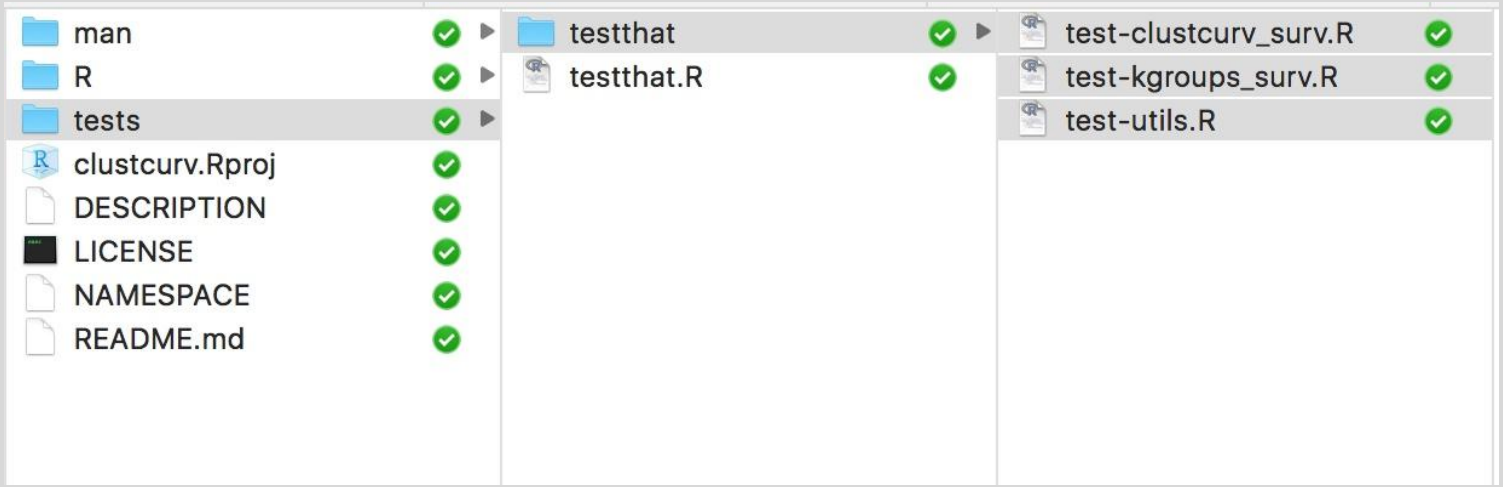
```
> test_dir("/Users/nora/gradiant/xornadasUsuariosRGalicia18/code")
```

```
 > test_file("test-hypotenuse.R")
```


Using Unit Tests. Running test

```
✔ | OK F W S | Context
✔ |  3       | hypotenuse output
✔ |  3       | hypotenuse input
✔ |  2       | hypotenuse checking small values
✔ |  1       | hypotenuse comparison for identical and equal expectation

== Results ===================================================================
Duration: 0.1 s

OK:       9
Failed:   0
Warnings: 0
Skipped:  0
```


Using Unit Tests. Integrating test into a package

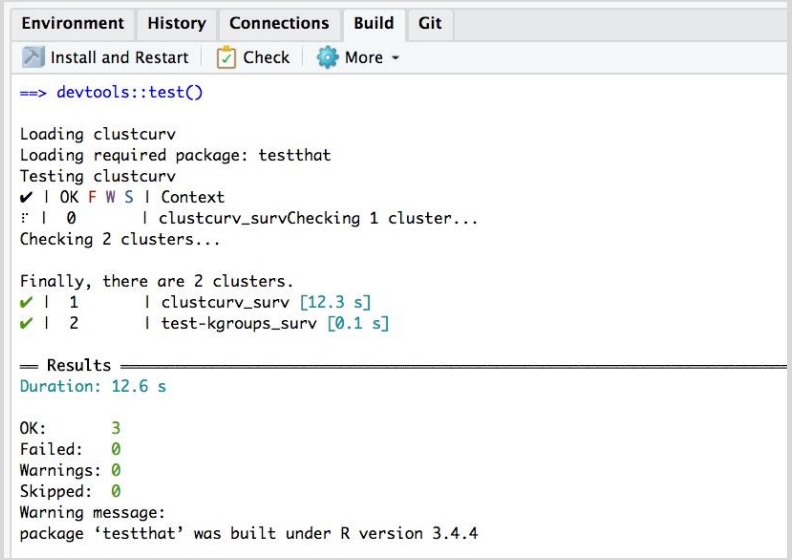


Using Unit Tests. Integrating test into a package

tests/testthat.R

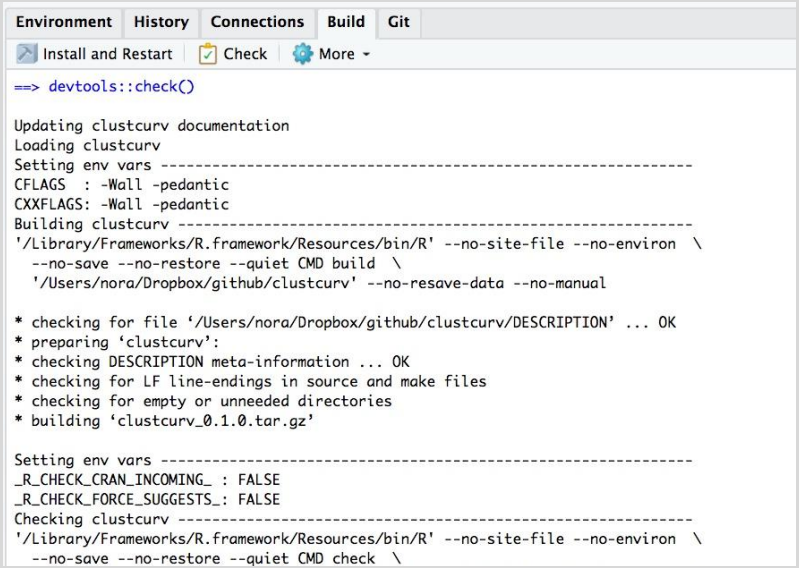
```
> library(testthat)
> library(clustcurv)
```


Using Unit Tests. Integrating test into a package



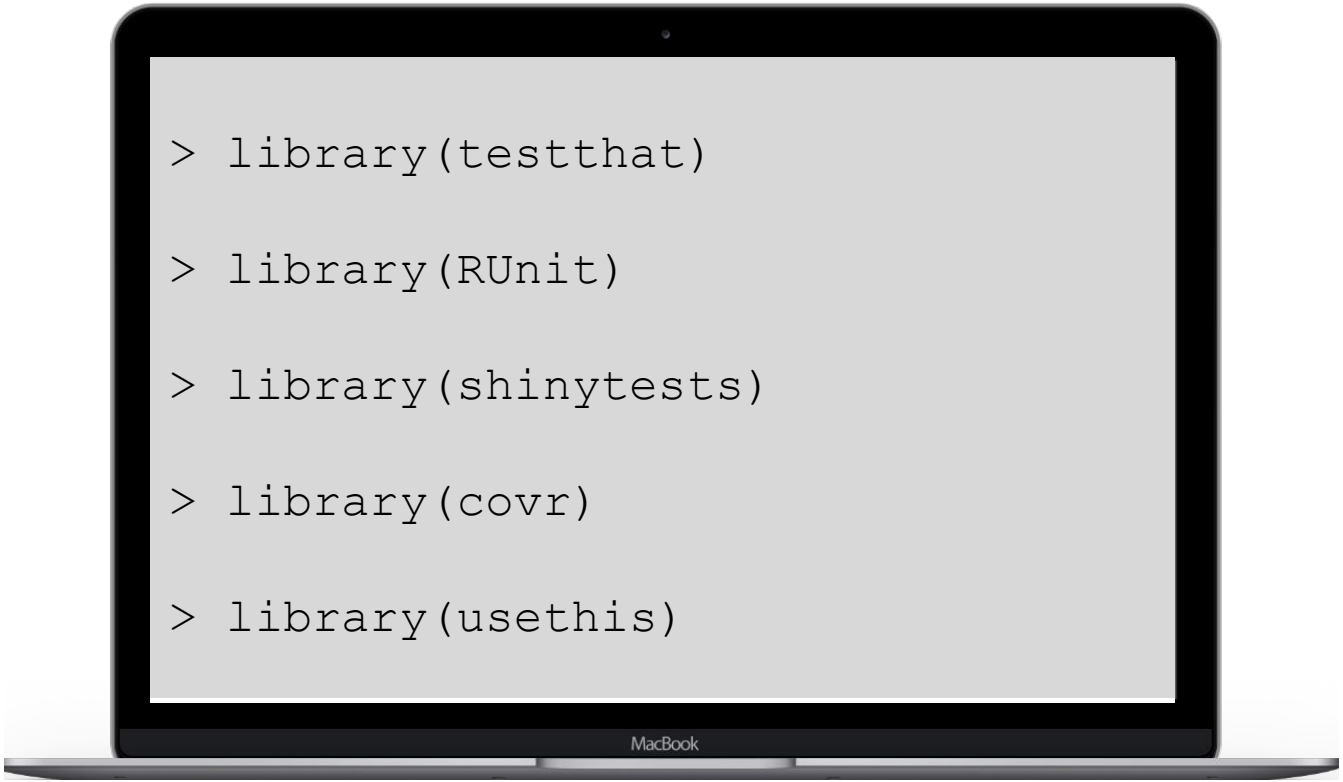
- Running all test into the package
- Continuous Integration ServicesTravisCI,Jenkins, ...

Using Unit Tests. Integrating test into a package



- Ensuring DESCRIPTION
- Dependencies declared in DESCRIPTION and NAMESPACE
- Functions are correctly described
- All necessary files are present
- Running all examples
- Running all the test

Using Unit Tests. Tools



References

- Cotton, R. (2016). assertive: Readable Check Functions to Ensure Code Integrity, 2016. R package version 0.3--0.
- Cotton, R. (2017). Testing R Code. A Chapman and Hall.
- Hester, J. (2015). covr: Test coverage for Packages. R package version 1.2.0.
- Wickham, H. (2011). testthat: Get Started with testing. *The R journal*, 3, 5--10.
- Wickham, H. (2015). R package. O'Reilly.
- Wickham, H. and Chang, W. (2016). devtools: Tools to make Developing R packages Easier. R package version 1.10.0.

Oriented to Industry requirements

Nora M. Villanueva nmvillanueva@gradiant.org