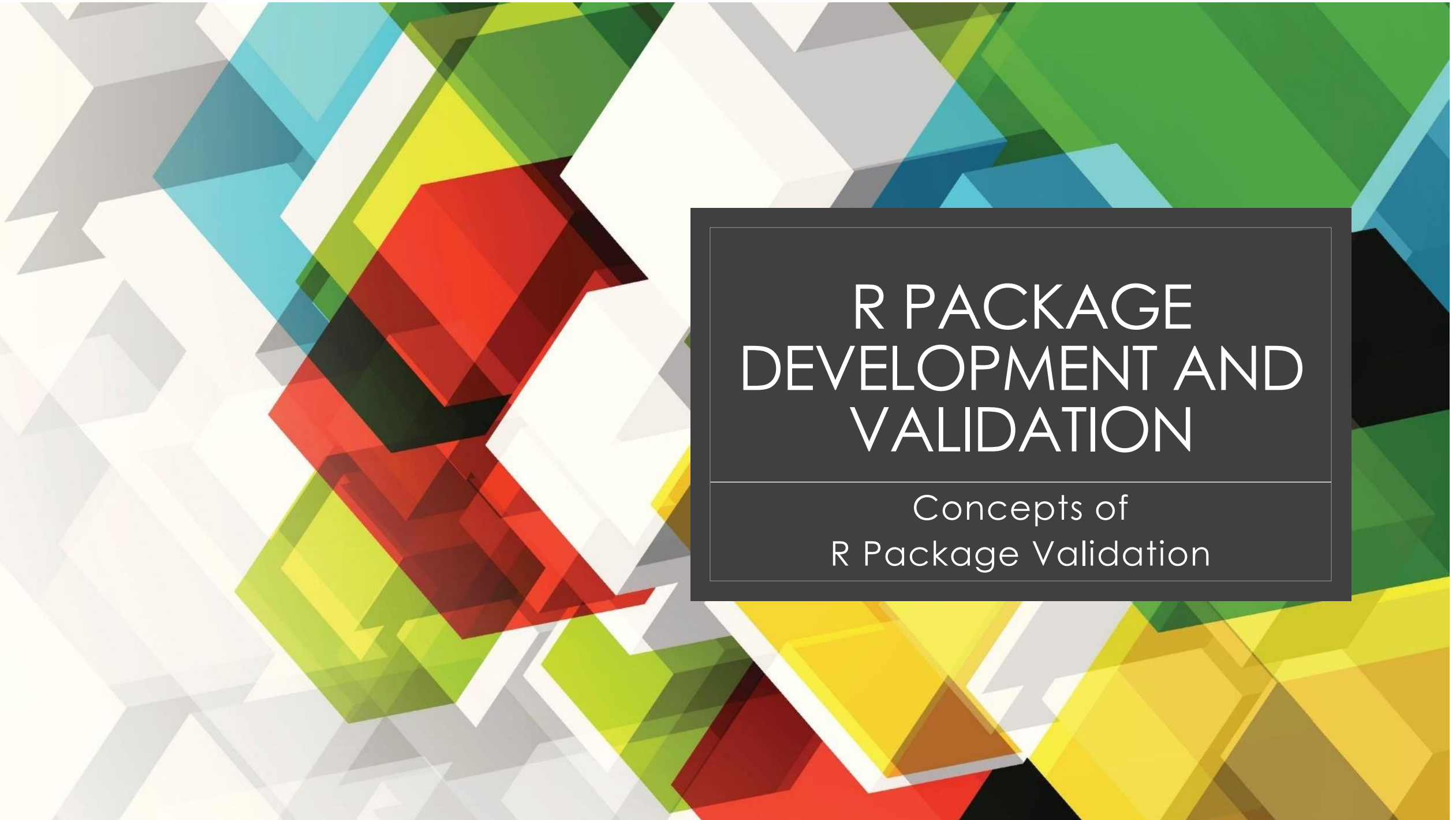

R PACKAGE DEVELOPMENT AND VALIDATION
Concepts of
R Package Validation

Welcome

- Validation
- R Packages
- R Package Validation Framework
- Q&A

VALIDATION

General Principles of Software Validation; Final Guidance for Industry and FDA Staff

FDA considers software validation to be "confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled."

General Principles of Software Validation; Final Guidance for Industry and FDA Staff

FDA considers software validation to be "**confirmation by examination and provision of objective evidence** that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled."

General Principles of Software Validation; Final Guidance for Industry and FDA Staff

FDA considers software validation to be "confirmation by examination and provision of objective evidence that **software specifications conform to user needs and intended uses**, and that the particular requirements implemented through software can be consistently fulfilled."

General Principles of Software Validation; Final Guidance for Industry and FDA Staff

FDA considers software validation to be "confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the **particular requirements implemented through software can be consistently fulfilled**."

Create **documentation** that the users needs are **satisfied by the requirements**, and **record results from testing** that that each requirement has been met by the software

Create **documentation** that the stakeholder needs are **satisfied by the requirements**, and **record results from testing** that that each requirement has been met by the software

Why bother validating

- Requirements by government or industry authorities
  - FDA (Pharma)
- Understanding of the process and goals of stakeholders
- Trust in your thoroughly vetted software returning correct results
- Speed in processing because details have been abstracted
- Reduced cost of maintenance

Risk-Based Valdation

- Not everything needs to be validated
- Risk based – which should be validated
  - Word processing software
  - Software processing and developing response calls of assay data

R PACKAGES

R package and validation

- Benefits of R packages
  - Shared code
  - Centralize management
  - Documentation for sharing knowledge

- Working in a regulated industry means code could be used as part of a project where validation measures need to apply

R PACKAGE VALIDATION FRAMEWORK

Validation can be a high bar

- Fill out a form for specifications, planned uses, environments
- Write Code and record function authorship in some excel file
- Get another form to document test cases, testing environment, …
- Maybe the last form to show that testing plan is comprehensive
- Manually evaluate tests and screenshot results
- Review documentation, combine into final report and signoffs for release

R Package Validation Framework

Requirements

Record the expectations, goals, and risks of project

R Package Development

Implementation of requirements into R package

Test Cases

Describes how code meets specifications

Test Code

Implementation of the test cases in code

Validation Report

Combines all prior content into report documenting proof that code meets specifications

Validation Report

2 Validation Environment

Environment Detail	Environment Record
R Version	R version 4.0.2 (2020-06-22)
Dependency Versions	testthat (v2.3.2)
	knitr (v1.29)
	rmarkdown (v2.3)

3 Authorship

3.1 Specifications

Specification Name	Last updated by	Last updated date
Specifications Presentation Success	Ellis Hughes	2030-01-01

3.2 Functions

Function Name	Last updated by	Last updated date
joke	Joe King	2030-04-01

3.3 Test case

Test Case Name	Last updated by	Last updated date
Presentation Success Test Cases 001	Ellis Hughes	2030-01-01

3.4 Test code

Test Code Name	Last updated by	Last updated date
T001		
T1.1	Jess Terr	2030-01-01

1. Specification 1: Tell a Joke

		Specification 001		
		S1.1	S1.2	S1.3
Presentation Success Test Cases 001	T1.1	x	x	x

- *Specifications*
  - 1.1 Presentation must explain validation procedure.
  - 1.2 Presentation will be between 15-20 minutes long
  - 1.3 Be entertaining, sucess measures by causing at least 3 people to laugh.
  - 1.4 (Optional) Fame and Glamour and start branded accessories chain.
- *Test Cases*
  - Setup: Create Presentation
  - T 1.1 Test that specifications 1.1, 1.2, and 1.3 are met by practicing presentation on unsuspecting co-workers
    - Present to a captive audience of coworkers and later your significant other
    - T 1.1.1 Test that the presentation was informative by asking what your audience learned.
    - T 1.1.2 Time the presentation and make sure it was between 15 and 20 minutes.
    - T 1.1.3 Test that you were entertaining by counting the amount of chuckles (>3)
      - T 1.1.3.1 Alternatively, eye-rolls from your significant other (>4)
- *Test Results*

Test	Results	Pass/Fail
T1.1.1	As expected	Pass
T1.1.2	As expected	Pass
T1.1.3	As expected	Pass

> The framework turns validation into a transparent, easy-to-follow process that not only that simplifies validation but improves the quality of both the package and validation.
>
> - Me

PHUSE Working Group

- White Paper
  - Distillation of framework
  - Collaboration
- R Package
  - Implementation of framework
  - Inspired by {devtools} and {usethis}

```
remotes::install_github("phuse-org/valtools")
```

Requirements

Record the expectations, goals, and risks of project

R Package Development

Implementation of requirements into R package

Test Cases

Describes how code meets specifications

Test Code

Implementation of the test cases in code

Validation Report

Combines all prior content into report documenting proof that code meets specifications

Requirements

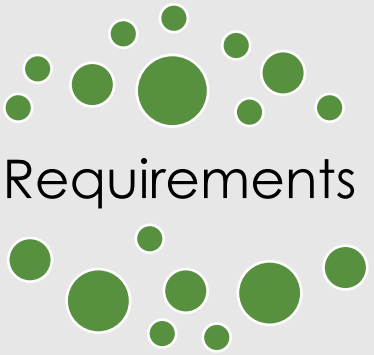
- *Goals and Purpose*
- Risk assessment
  - Probability of defects
  - Impact when there are defects
- Provides all context required or points to references
- Stakeholder approval of each requirement

Documenting Requirements

- Human and machine readable
- Capture editor and edit date
- Record and save within the validation folder
  - Working_dir/validation/requirements

```
#' @title Requirements For Presentation Success
#' @editor Ellis Hughes
#' @editDate 2025/01/01

+ 1.1 Presentation must explain validation procedure.
+ 1.2 Presentation will be between 20-25 minutes long
+ 1.3 Be entertaining, sucess measures by causing at least 3 people to laugh.
+ 1.4 (Optional) Fame and Glamour and start branded accessories chain.
```



Requirements

Record the expectations, goals, and risks of project

R Package Development

Implementation of requirements into R package

Test Cases

Describes how code meets specifications

Test Code

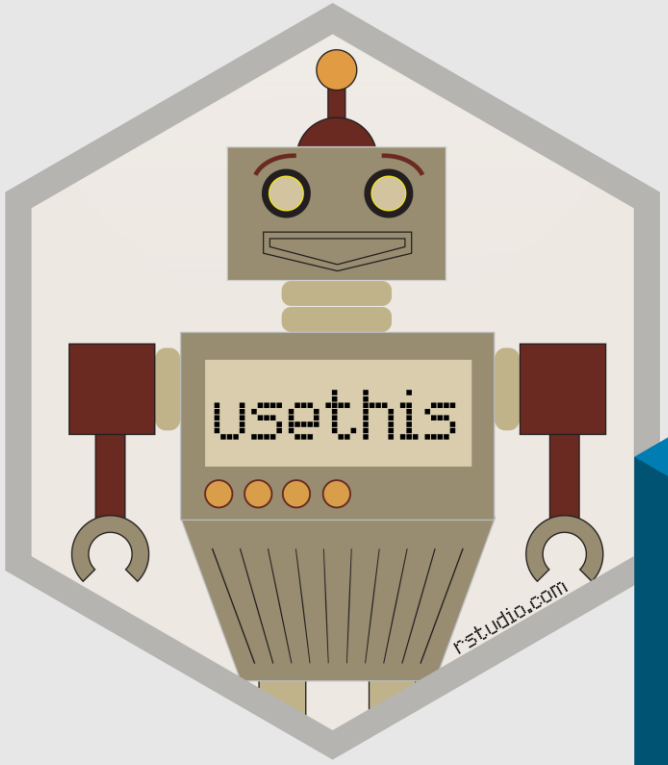
Implementation of the test cases in code

Validation Report

Combines all prior content into report documenting proof that code meets specifications

Package Development

- Follow GPP
- Documentation
  - Function manuals
  - Vignettes
- What we talked about in the earlier sessions



Function documentation

- Record *with* function using Roxygen
  - Easier to remember
  - Adds to documentation

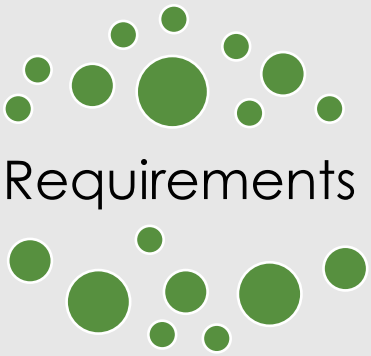
  @editor
  @editDate

```r
#' @title Deliver Jokes
#' @description
#'    Deliver jokes with punchlines. Wait 3 seconds for the punchline.
#' @param Setup Joke setup
#' @param Punchline Joke punchline
#' @examples
#' joke('To the person who stole my presentation -','I hope you do not Excel.')
#'
#' @editor Ellis Hughes
#' @editDate 2030/01/01

joke <- function(Setup, Punchline){
  print(Setup)
  Sys.sleep(3)
  print(Punchline)
}
```

```r
#' @title Deliver Jokes
#' @description
#'    Deliver jokes with punchlines. Wait 3 seconds for the punchline.
#' @param Setup Joke setup
#' @param Punchline Joke punchline
#' @param wait how long to wait to serve the punchline
#' @examples
#' joke('Why did the PowerPoint Presentation cross the road?','To get to the other slide.')
#'
#' @editor Joe King
#' @editDate 2030/04/01

joke <- function(Setup, Punchline, wait = 3){
  print(Setup)
  Sys.sleep(wait)
  print(Punchline)
}
```



Requirements

Record the expectations, goals, and risks of project

R Package Development

Implementation of requirements into R package

Test Cases

Describes how code meets specifications

Test Code

Implementation of the test cases in code

Validation Report

Combines all prior content into report documenting proof that code meets specifications

Test Cases

- Document how to prove package meets specifications
- Should represent realistic scenarios
- Every requirement must be satisfied by at least one test case
- A test case can satisfy multiple requirements

Requirement	Test Case
Requirement 1	Test Case 1, Test Case 2
Requirement 2	Test Case 2, Test Case 3
Requirement 3	Test Case 3

Documenting Test Cases

- Human and machine readable
- Capture editor, edit date, coverage
- Record and save within the validation folder
  - working_dir/validation/test_cases

```
#' @editor Ellis Hughes
#' @editDate 2030/01/01
#' @coverage:
#' T1.1: S1.1, S1.2, S1.3

+ _Test Cases_

+ Setup: Create Presentation

+ T 1.1 Test that requirements 1.1, 1.2, and 1.3 are met by practicing presentation on unsuspecting co-
workers

  - Present to a captive audience of coworkers and later your significant other
  + T 1.1.1 Test that the presentation was informative by asking what your audience learned.
  + T 1.1.2 Time the presentation and make sure it was between 20 and 25 minutes.
  + T 1.1.3 Test that you were entertaining by counting the amount of chuckles (>3)
      + T 1.1.3.1 Alternatively, eye-rolls from your significant other (>4)
```

Record the expectations, goals, and risks of project

Implementation of requirements into R package

Describes how code meets specifications

Implementation of the test cases in code

Combines all prior content into report documenting proof that code meets specifications

Test Code

- Implementation of test cases as code
- Uses {testthat} as a familiar testing framework
  - Self contained
  - Clear expectations

{testthat}

- Reporter Object
  - Captures each expectation result
  - Records test name

```
|  OK F W S | Context
|   3       | test_case_001 [3.2 s]

-- Results -------------------------------------
Duration: 3.2 s

OK:       3
```

Test	Results	Pass/Fail
T1.1.1	As expected	Pass
T1.2.1	As expected	Pass
T1.2.2	As expected	Pass

Writing Test Code

- Capture editor, edit date
- Record and save within the validation folder
  - working_dir/validation/test_code

```
#' @title T1.1
#' @editor Jess Terr
#' @editDate 2030/01/01

test_that('T1.1',{

  joke_result <- joke('What do you call a fake noodle?','An Impasta')
  expect_true(is_dad_joke(joke_result))
  expect_true(caused_laugher(joke_result))
  expect_true(embarrased_significant_other(joke_result))

})
```

Record the expectations, goals, and risks of project

Implementation of requirements into R package

Describes how code meets specifications

Implementation of the test cases in code

Combines all prior content into report documenting proof that code meets specifications

Validation Report

- Dynamically captures and records all elements generated across the framework
- Records testing environment, package version
  - Extensible to organizations SOPs
- Runs test code and captures results

Validation Report

2 Validation Environment

Environment Detail	Environment Record
R Version	R version 4.0.2 (2020-06-22)
Dependency Versions	testthat (v2.3.2)
	knitr (v1.29)
	rmarkdown (v2.3)

3 Authorship

3.1 Specifications

Specification Name	Last updated by	Last updated date
Specifications Presentation Success	Ellis Hughes	2030-01-01

3.2 Functions

Function Name	Last updated by	Last updated date
joke	Joe King	2030-04-01

3.3 Test case

Test Case Name	Last updated by	Last updated date
Presentation Success Test Cases 001	Ellis Hughes	2030-01-01

3.4 Test code

Test Code Name	Last updated by	Last updated date
T001		
T1.1	Jess Terr	2030-01-01

1. Specification 1: Tell a Joke

		Specification 001		
		S1.1	S1.2	S1.3
Presentation Success Test Cases 001	T1.1	x	x	x

- *Specifications*
  - 1.1 Presentation must explain validation procedure.
  - 1.2 Presentation will be between 15-20 minutes long
  - 1.3 Be entertaining, sucess measures by causing at least 3 people to laugh.
  - 1.4 (Optional) Fame and Glamour and start branded accessories chain.
- *Test Cases*
  - Setup: Create Presentation
  - T 1.1 Test that specifications 1.1, 1.2, and 1.3 are met by practicing presentation on unsuspecting co-workers
    - Present to a captive audience of coworkers and later your significant other
    - T 1.1.1 Test that the presentation was informative by asking what your audience learned.
    - T 1.1.2 Time the presentation and make sure it was between 15 and 20 minutes.
    - T 1.1.3 Test that you were entertaining by counting the amount of chuckles (>3)
      - T 1.1.3.1 Alternatively, eye-rolls from your significant other (>4)
- *Test Results*

Test	Results	Pass/Fail
T1.1.1	As expected	Pass
T1.1.2	As expected	Pass
T1.1.3	As expected	Pass

Record the expectations, goals, and risks of project

Implementation of requirements into R package

Describes how code meets specifications

Implementation of the test cases in code

Combines all prior content into report documenting proof that code meets specifications


PUSH BUTTON GET VALIDATED

Validating the Package

1. Execute validation report on release for package build using this framework
   - generating a tarball people to install from, generating validation report as part of the generation

2. Execute validation report on installation for package build using this framework
   - Install from source, running the validation report

3. Execute validation report on update of system environment on package build using framework
   - Running validation report on an already installed package

Validation ≠ 4ever

Q&A

Pose questions into slack or unmute your microphone!