

Automated Validation of Clinical Trial Analysis and Reporting Deliverables Using testthat

US Connect 2021

Amin Shirazi¹, Merck & Co., Inc., Kenilworth, NJ, USA

Jane Liao, Merck & Co., Inc., Kenilworth, NJ, USA

Suhas R. Sanjee, Merck & Co., Inc., Kenilworth, NJ, USA

June, 2021

¹Author working under contract with ClinChoice, Fort Washington, PA, USA

Overview

Overview

- Introduction
 - Problem Statement
 - Analysis deliverables
 - testthat package
- Validation workflow
 - Unit tests
 - Regression testing
- Solution
- Examples
 - Table and listings
 - Figures
 - Data frames
- Summary and conclusion

Introduction

Introduction

Problem statement

- Validation plays a critical role in clinical trial analysis and reporting (A&R)
 - Ensure clear and accurate representation of clinical trial data
 - Meet regulatory requirements, industry/company standards
- Some challenges include:
 - Testing is tedious, laborious and cumbersome
 - Evolving requirements and new scenarios add complexity
 - Difficult to streamline testing to automate and promote repeatability

Introduction

Analysis deliverables

- Clinical trial analysis and reporting (A&R) deliverables include tables, listings, and figures (TLFs)
 - Datasets: analysis datasets (ADaM) are created to support TLFs
 - Tables: contain characters that require special handling
 - Listings: large size, ICH listing may have hundreds of pages
 - Figures: expected output of figures hard to describe in machine readable format

Introduction

testthat package

- testthat: unit testing for R packages (Wickham 2011)
 - Facilitates structured testing of R packages
 - Provides functions to set up testing environment
 - Provides an efficient way to run test cases in a clean environment
 - Reports whether each test passes or fails
- Latest features
 - Snapshot testing [[testthat 3.0.0 Webinar, Hadley Wickham, August 2020](#)]
 - Allows testing of complex data, specially useful for testing figures
 - Allows to record expected output in external files, rather than in the code
 - Report all problems (i.e. errors, failures, warnings and skips)

Introduction

- Many existing articles emphasize on the utility of testthat package
- This work focuses on the validation of clinical trials A&R deliverables

Related work:

- *Organize, test, document, and share your code*, Wickham (2015)
- *testthat: Get Started with Testing*, Wickham (2011)
- *R unit test framework*, Burger et al. (2007)

Validation workflow

Validation workflow

Unit tests

- The goal of a unit test is to validate that each unit of program performs as expected (Burger et al. 2007)
- Provide the ability to evaluate each feature of a function one at a time
- `testthat` package provides functions to describe expected output using code so that it can be compared with actual output. This ensures:
 - Fewer bugs
 - Robust programs
 - Code does not change unexpectedly

Validation workflow

Unit tests

- Are not always feasible since it is hard to record the expected behavior with code

Some challenges include:

- Text output may include special characters like quotes and newlines that require special handling in a string
- For large outputs, it is hard to define the reference output since it results in a large test file that is difficult to navigate
- Binary formats like plots, images, and RTF files are very difficult to describe in code and are not human readable

Validation workflow

To create flawless and robust programs in R, the validation workflow includes:

1. Update the existing programs and/or test cases
2. Run test cases
3. Review test log for failures and warnings
4. Repeat #1, #2, and #3 until:
 - There are no failures and warnings
 - All tests pass

```
== Results ===== [ FAIL 0 | WARN 0 | SKIP 0 | PASS 9 ]
```

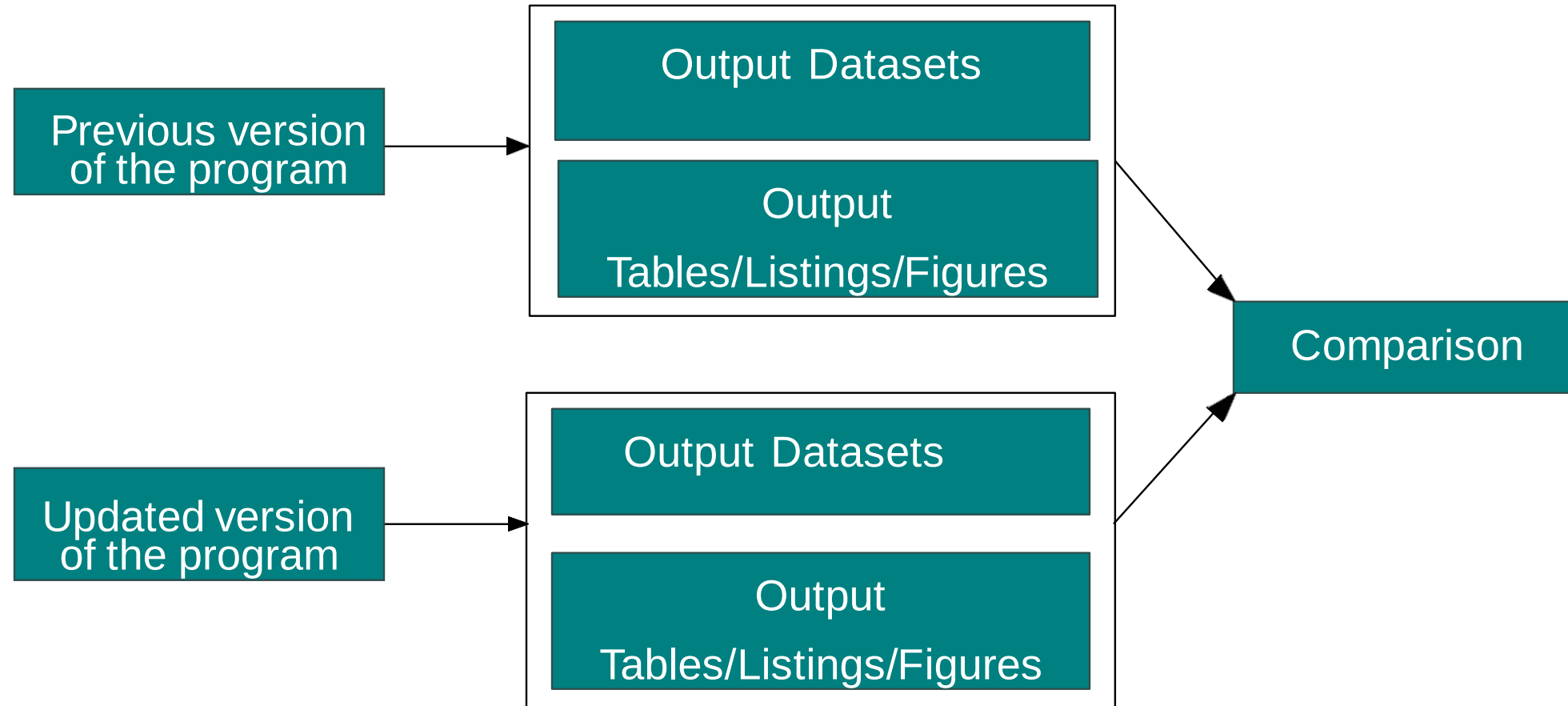
Validation workflow

Regression testing

- Type of testing done when a program is updated to ensure:
 - Changes are implemented correctly
 - No new bugs are introduced in the process
- Can be used in testing updates to dependent R packages, platform changes, updates in functionality

Validation workflow

Regression testing



Solution

- `testthat` can be used to automate validation of the clinical trials A&R deliverables
- Snapshot feature helps to automate regression testing
 - The output is automatically stored for the comparison with the output from previous version
 - Eliminate the need to write separate codes to record the expected behavior in the program
- While unit tests provide a way to validate one feature at a time, validation of the output as a whole is important
- `testthat` offers **snapshot test** as a powerful tool to validate outputs

Examples: : Tables and listings

Examples: Tables and listings

- Tables and listings are common deliverables in clinical trial studies
- `testthat` package can be used to make the validation process more efficient and accurate
- While unit tests provide a way to validate one feature at a time, validation of the table as a whole is important

Example: Summary of adverse events

	Placebo	Xanomiline Low Dose	Xanomiline High Dose
Participants in population	86	84	84
Participants with at least one adverse events	69	77	79
Participants with drug-related ^a adverse events	26	44	48
Participants with serious adverse events	0	1	2
Who died	2	1	0
^a Determined by the investigator to be related to the drug.			

Examples: Tables and listings

- Summary of adverse events after updates in the program:
 - Update in the attributes of table
 - Update in the content of table

	Placebo	Xanomiline Low Dose	Xanomiline High Dose
Participants in population	86	84	84
Participant with at least one adverse events	69	77	79
Participants with drug-related ^a adverse events	26	44	48
Participants with serious adverse events	0	1	2
Who died	2	1	0
^a Determined by the investigator to be related to the drug.			

- Validate the changes using testthat:
 - Use `expect_snapshot_output` to validate the output

```
testthat::test_that("rtf table is rendered correctly", { local_edition(3)
testthat::expect_snapshot_output(rtf())
})
```

Examples: Tables and listings

- Unit test fails when the output changes due to updates made to the program
- Test log provides a summary of the number of passed/failed unit tests

```
== Results ==  
Duration: 1.1 s  
  
[ FAIL 1 | WARN 0 | SKIP 0 | PASS 0 ]
```

- `testthat` also provides a test log with details of the changes between the previous and current outputs

Examples: Tables and listings

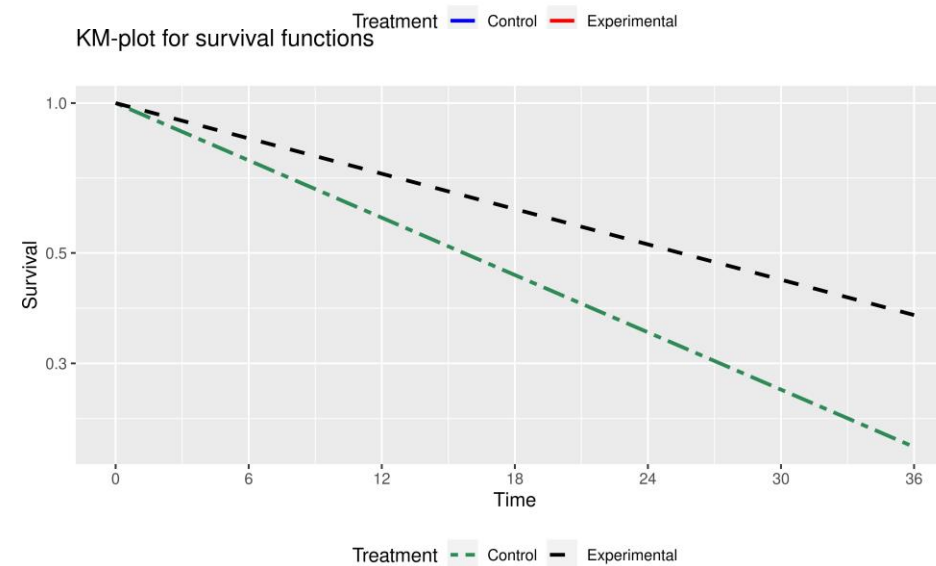
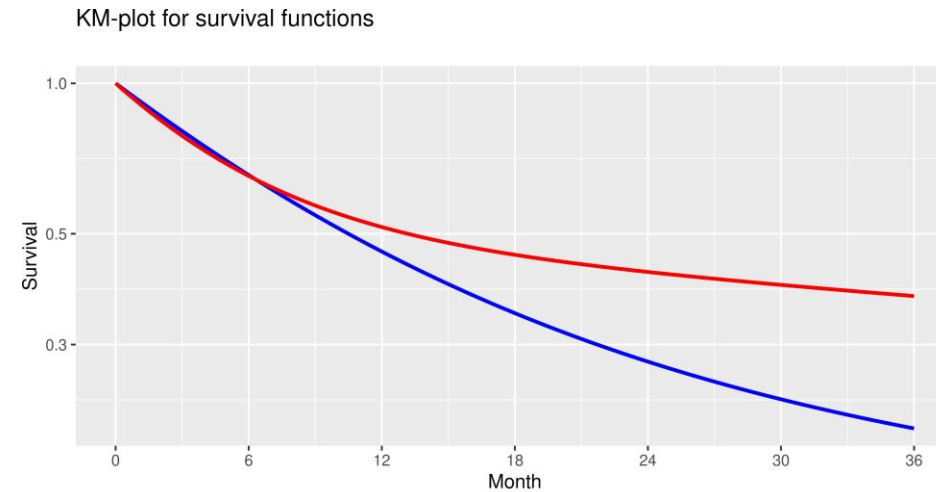
```
@@ -45,11 +45,11 @@
45 45 \trowd\trgaph108\trleft0\trqc
46 46 \clbrdr1\brdrs\brdrw15\clbrdrt\brdrw15\clbrdrb\brdrw15\cellx3000
47 47 \clbrdr1\brdrs\brdrw15\clbrdrt\brdrw15\clbrdrb\brdrw15\cellx5000
48 48 \clbrdr1\brdrs\brdrw15\clbrdrt\brdrw15\clbrdrb\brdrw15\cellx7000
49 49 \clbrdr1\brdrs\brdrw15\clbrdrt\brdrw15\clbrdrb\brdrw15\cellx9000
50 - \pard\intbl\sb15\sa15\ql{\f0\fs18 Participants with at least one adverse events}\cell
50 + \pard\intbl\sb15\sa15\ql{\f0\fs18 Participant with at least one adverse events}\cell
51 51 \pard\intbl\sb15\sa15\qj\tqdec\tx1000{\f0\fs18 69}\cell
52 52 \pard\intbl\sb15\sa15\qj\tqdec\tx1000{\f0\fs18 77}\cell
53 53 \pard\intbl\sb15\sa15\qj\tqdec\tx1000{\f0\fs18 79}\cell
54 54 \intbl\row\pard
55 55 \trowd\trgaph108\trleft0\trqc
@@ -82,11 +82,11 @@
82 82 \pard\intbl\sb15\sa15\qj\tqdec\tx1000{\f0\fs18 1}\cell
83 83 \pard\intbl\sb15\sa15\qj\tqdec\tx1000{\f0\fs18 0}\cell
84 84 \intbl\row\pard
85 85 \trowd\trgaph108\trleft0\trqc
86 86 \clbrdr1\brdrs\brdrw15\clbrdrt\brdrw15\clbrdrb\brdrw15\clbrdrb\brdrw15\cellx9000
87 - \pard\intbl\sb15\sa15\ql{\f0\fs18 {\super a} Determined by the investigator to be related to the drug.}\cell
87 + \pard\intbl\sb15\sa15\qc{\f0\fs18 {\super a} Determined by the investigator to be related to the drug.}\cell
88 88 \intbl\row\pard
89 89
90 90 {\pard\par}
91 91
92 92 }
```

- User can decide whether to accept or reject the changes

Examples: Figures

Examples: Figures

- KM plots are widely used in clinical trials to:
 - Estimate the survival functions
 - Compare survival functions over time
 - Assess the relationship of survival functions to survival time



Examples: Figures

- Common features figure programs compared during regression testing are:
 - Axes labels
 - Curve colors
 - Line types
- `expect_snapshot_file()` was used to validate the changes in the plots

```
testthat::test_that("survival curves are correctly plotted", { local_edition(3)  
testthat::expect_snapshot_file(ggsave_png(KM_plot()), "survival_curves.png")  
})
```

Test log

```
Failure (test-KM_plot.R:10:3): x axes is labelled 'Month'  p$labels$x (`actual`) not identical to "Month" (`expected`).
```

```
`actual`: "Time"
```

```
`expected`: "Month"
```

```
Failure (test-KM_plot.R:20:3): line colors are correct
```

```
x$colour %>% unique() (`actual`) not identical to c("blue", "red") (`expected`).
```

```
`actual`: "seagreen" "black"
```

```
`expected`: "blue" "red"
```

```
Failure (test-KM_plot.R:25:3): linetypes are correct
```

```
x$linetype %>% unique() (`actual`) not identical to 1 (`expected`).
```

```
`actual` is a character vector ('twodash', 'dashed')
```

```
`expected` is a double vector (1)
```

```
Failure (test-KM_plot.R:40:3): KM_plot correctly plots the survival curves  Snapshot of `ggsave_png(KM_plot())` to 'KM_plot/survival_curves.png' has changed
```

```
Run `testthat::snapshot_review('KM_plot')` to review changes
```

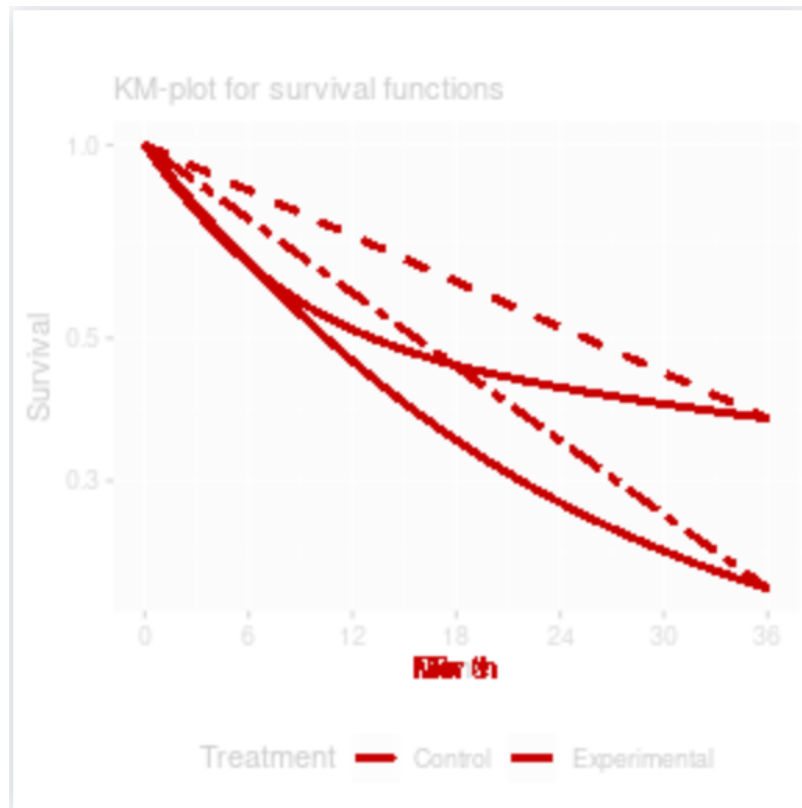
```
== Results ==
```

```
Duration: 0.8 s
```

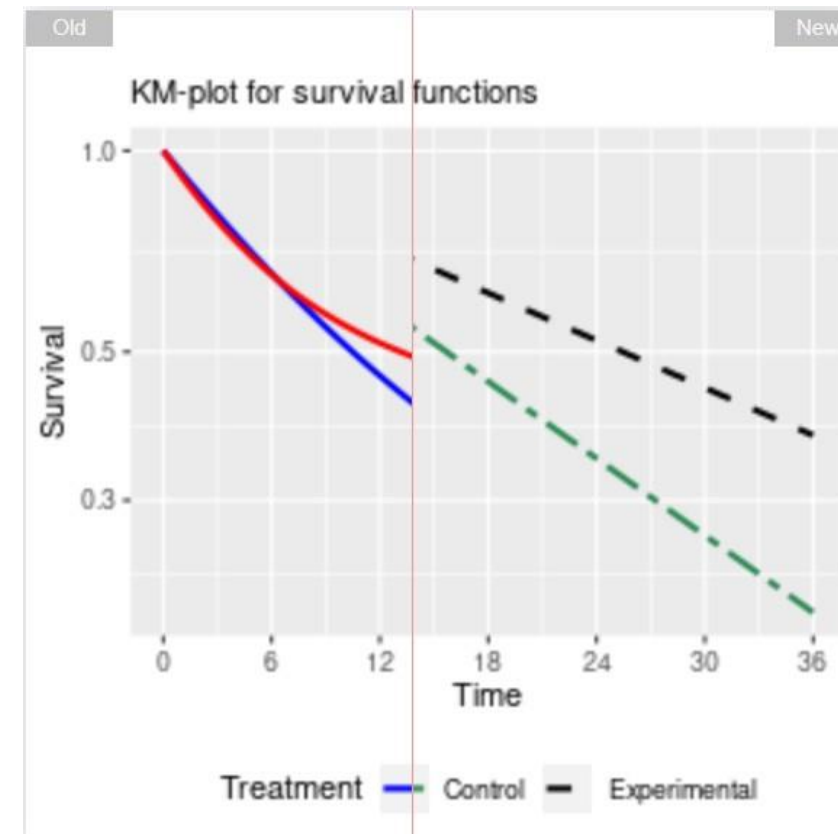
```
[ FAIL 4 | WARN 0 | SKIP 0 | PASS 2 ]
```

Snapshot tools to review the changes

- Overall differences



- Pixel by pixel comparison



Example: Data frames

Example: Data frames

- Frequently used in clinical analysis & reporting
- Analysis results stored as data frame are machine readable enabling automation of validation activities
- Example:

Var	Placebo	Xanomiline Low Dose	Xanomiline High Dose
Participants in population	86	84	84
Participant with at least one adverse events	69	77	79
Participants with drug-related adverse events	26	44	48
Participants with serious adverse events	0	1	2
Who died	2	1	0

Example: Data frames

- Automated tests can validate whether the data frames are identical after each update in the program
- Snapshot test stores the data frame and compares it to the updated results of the program
- Updated data frame:

Var	Placebo	Xanomiline Low Dose	Xanomiline High Dose
Participants in population	87	84	84
Participant with at least one adverse events	69	77	79
Participants with drug-related adverse events	26	40	48
Participants with serious adverse events	0	1	2
Who died	1	1	0

Example: Data frames

- Use snapshot test to compare the current and updated data frame
- The test fails if the data frame is updated
- Test log:

```
== Results ==  
Duration: 2.4 s  
[ FAIL 1 | WARN 1 | SKIP 0 | PASS 2 ]
```

Example: Data frames

- Tools to review the changes

@@		Var	Placebo	Xanomiline Low Dose	Xanomiline High Dose
→	1	Participants in population	86→87	84	84
	2	Participant with at least one adverse events	69	77	79
→	3	Participants with drug-related adverse events	26	44→40	48
	4	Participants with serious adverse events	0	1	2
→	5	Who died	2→1	1	0

Summary & Conclusion

Summary & Conclusion

Snapshot tests

- Are useful in clinical trial A&R when new data is available over time and the A&R programs need to be re-run
- Provide precise comparison between the current and updated A&R deliverables
- Record the changes in the output automatically
- Allow updates to be merged into the current results after user's review
- Allow user to decide whether to accept or reject the updates
- Snapshot feature of `testthat` package provides an efficient way to automate regression testing during development of R packages

Summary & Conclusion

- `testthat` provides options to compare expected vs. actual results for clinical trial deliverables
 - tables, data frames and graphical outputs
- `testthat` can be used to conduct structured testing to validate datasets and TLFs used in clinical trial A&R
- Regression testing can be done efficiently using snapshot feature since it facilitates comparison of output from previous run and current run
- The whole process is automated enabling validation of complex outputs in a streamlined way

References

- Burger, M., Juenemann, K., and Koenig, T. (2007), “RUnit: R unit test framework,” R.
- Wickham, H. (2011), “Testthat: Get started with testing,” *The R Journal*, Citeseer, 3, 5–10.
- Wickham, H. (2015), *R packages: Organize, test, document, and share your code*, "O'Reilly Media, Inc.".

Thank You