

SM10

From Hassle to Harmony: Unifying Code Quality Experience in R and SAS

PHUSE EU CONNECT 2025
NOVEMBER 16-19

MICHAEL HEDEGAARD THOMSEN
MHZT@NOVONORDISK.COM



Disclaimer:
Views and opinions expressed are
those of the speaker and not
necessarily Novo Nordisk.

Introduction

Unit Test Framework for SAS using R

Implementing a test

Automated Tests using CI/CD

Conclusion

Agenda

Introduction

2015: No version control in Novo Nordisk

2017: PHUSE Edinburgh - build servers?

Can we establish a pipeline to

- Execute ADaM and TFL programs,
- Produce CSR,
- Do validation checks,
- Display results in a dashboard?
- [Execute unit tests – if we ever get them?]

Con:

“Major setup before everything runs smoothly”...



Background – standard programs

Standard programs

- Requires (manual) unit tests
- Version controlled
- Do not need review (when used in trials)

Two SAS template repositories in Git:

- `ADaM_programming`
- `TFL_programming`

Without unit tests:

- Manual code review
- Ad-hoc testing procedures

Git: Distributed version control system

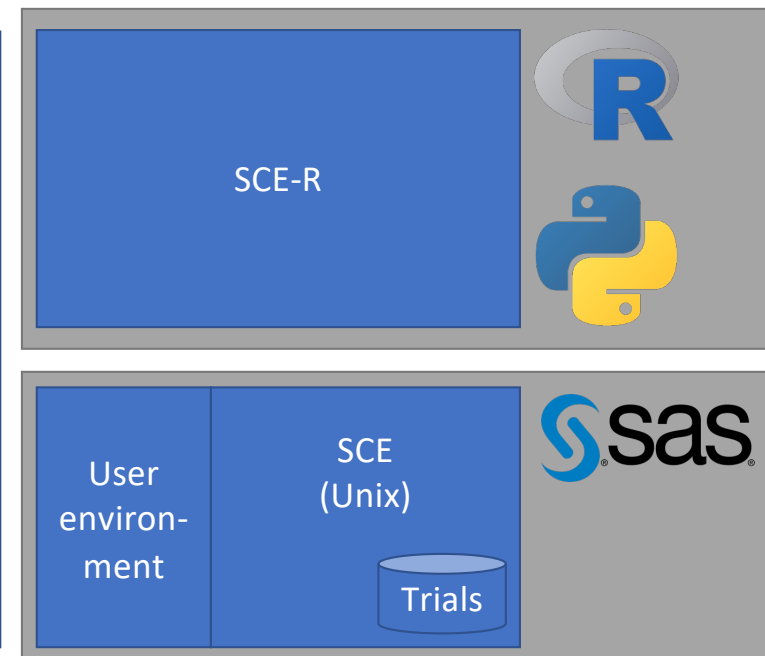
Standard programs as defined in SOP:

The term “standard program” covers a robust multi-use program which has been validated, tested and documented according to [SOP].

The goal - unit testing framework for SAS

Environments

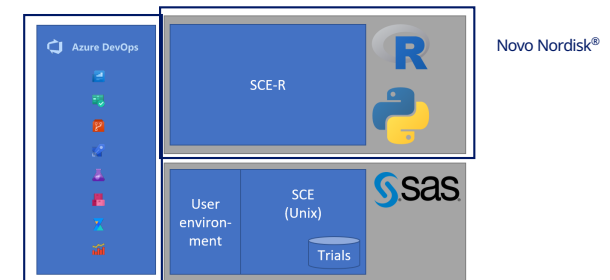
- Unix SCE
- SCE-R
- Azure DevOps
 - Git repositories
 - CI/CD Pipelines
 - Test Reporting



SCE: Statistical Computing Environment

CI/CD: Continuous Integration/Continuous Deployment

Building blocks



Use a general unit test framework to setup test cases

- `testthat` is a well known and documented framework in R
- Use package `NNremote` to execute code from SCE-R on Unix SCE (SAS)
- Use `SASunittest` package that bridges `NNremote` and `testthat`
- CI/CD via Azure DevOps pipelines
 - Run test cases
 - Test report available
 - Release after manual code review



NNremote

Core components for utilising R at Novo Nordisk

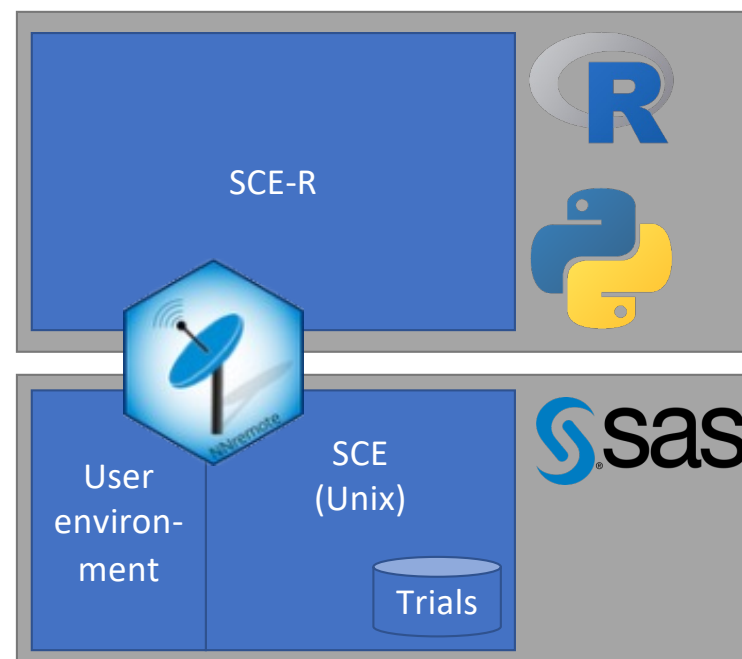
Primary responsibility:

- Setup a connection to the clinical data

Experimental `SAS ()` command

- Enable execution of SAS code from R by use of `ssh` commands,
- Read log files and `sas7bdat` datasets and convert into R readable format.

Novo Nordisk®



NNremote



SAS () method needed extensions:

- Ability to execute pre- and post- code to setup e.g. test data.
- Prepare SAS code with
 - Tagging of the SAS log
 - Logic to check for macro variable creation, fileref and libname assignments
- More unit tests to improve code quality.

```
/* TRACK INITIAL STATE */  
/* HELPER MACROS */  
/* PRECODE_START */  
/* PRECODE_END */  
/* CODE_START */  
* Macro definition;  
* Macro call;  
/* CODE_END */  
/* POSTCODE_START */  
/* POSTCODE_END */  
/* EXPORT CHANGES */
```



Reference

Expectations

Testthat style expectations for working with returned SAS objects

`expect_missing_values()`

Test that a SAS output table(s) contains any missing values

`expect_no_missing_values()`

Test that a SAS output tables do not contain any missing values

`expect_sasError()`

Test that a SAS output object contains a specific error

`expect_sasErrorFree()`

Test that a SAS output object contains no ERROR messages

`expect_sasNote()`

Test that a SAS output object contains a specific note

`expect_sasNoteFree()`

Test that a SAS output object contains no NOTE messages

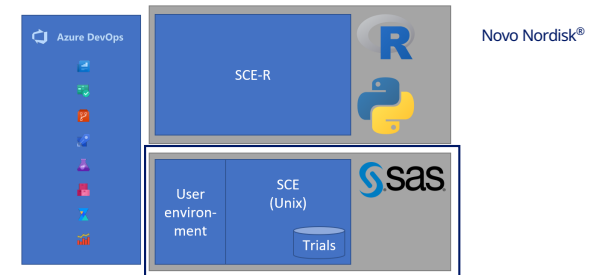
`expect_sasWarning()`

Test that a SAS output object contains a specific warning

`expect_sasWarningFree()`

Test that a SAS output object contains no WARNING messages

Implementing a unit test adamimpdtearly



Description : Date Imputation Rule - **EARLY_DT**

First day in month, first month in year,
missing if year missing.

```
%macro adamimpdtearly(di_datain = 
                        ,di_dataout = 
                        ,di_dtmin = 
                        ,di_dtout = 
                        );
```

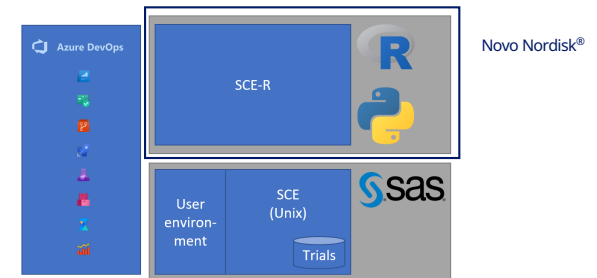
Diagram illustrating the macro parameters:

- Input dataset (points to `di_datain`)
- Output dataset (points to `di_dataout`)
- Variable to impute (points to `di_dtmin`)
- Variable name of imp. date (points to `di_dtout`)

[SAS CODE]

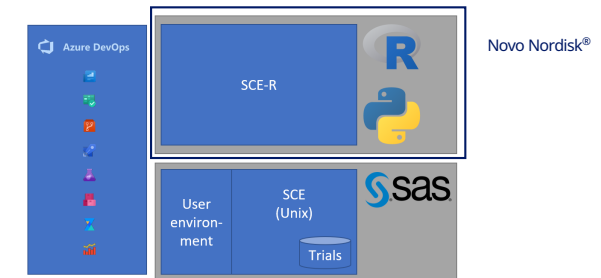
```
%mend adamimpdtearly;
```

Implementing a unit test



```
test_that("adamimpdtearly - full date, no imputation", {  
+ # SETUP -----  
+ # ACT -- Run SAS code -----  
+ # EXPECT -----  
})
```

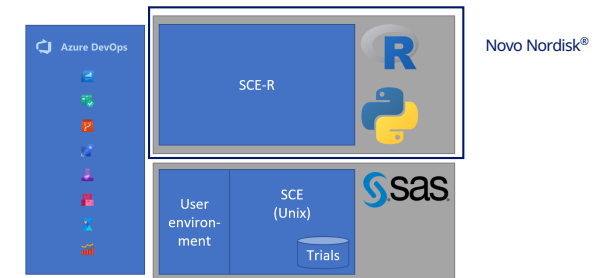
Implementing a unit test



```
test_that("adamimpdtearly - full date, no imputation", {
  # SETUP -----
  testdata <- tibble::tribble(
    ~USUBJID, ~AESTDTC,
    "01", "2025-11-17T17:10:00")
  adamimpdtearly <- SASunittest::r_function_from_sas_macro("adamimpdtearly")
  # Prepare SAS code
  macro_code <- adamimpdtearly(
    di_datain = "rin.testdata",
    di_dataout = "imputed_data",
    di_dtmin = "AESTDTC",
    di_dtout = "AESTDTC"
  )
+ # ACT -- Run SAS code -----
+ # EXPECT -----
})
```

```
macro_code:
%macro adamimpdtearly(      * Macro call;
  di_datain =              %adamimpdtearly(
    ,di_dataout =          di_datain=rin.testdata
    ,di_dtmin =            ,di_dataout=imputed_data
    ,di_dtout = );         ,di_dtmin=AESTDTC
/* macro code removed */  ,di_dtout=AESTDTC);
%mend adamimpdtearly;
```

Implementing a unit test

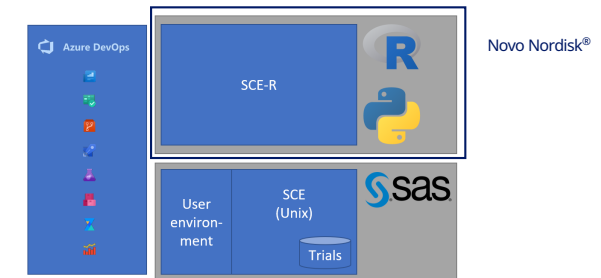


```
test_that("adamimpdtearly - full date, no imputation", {
+ # SETUP -----
  # ACT -- Run SAS code -----
  result <- NNremote::SAS(main_code = macro_code,
                          data_input = list("testdata" = testdata)) |>
    SASunittest::handle_SAS_exceptions()
+ # EXPECT -----
})
```

result:

- list of lists
- tables
- macro_variables
- libnames
- external_files
- sas_log
- sas_code
- paths
- system

Implementing a unit test



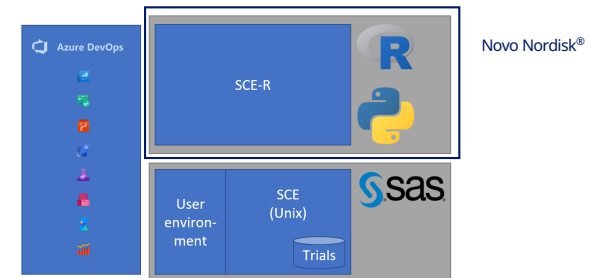
```
test_that("adamimpdtearly - full date, no imputation", {
+ # SETUP -----
+ # ACT -- Run SAS code -----
  # EXPECT -----
  imputed_data <- result$tables$imputed_data

  # Check imputed date
  expect_equal(as.character(imputed_data[["AESTDT"]]), "2025-11-17")

  # Check format of imputed date
  expect_equal(attr(imputed_data[["AESTDT"]], "format"), "DATE9.")

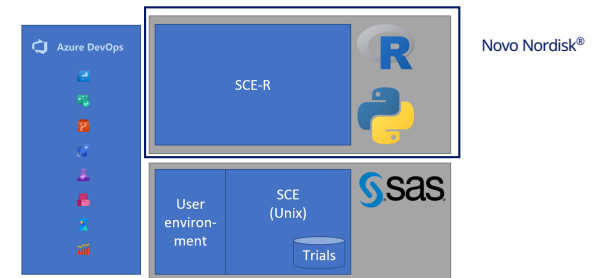
  # Check imputation flag
  expect_equal(as.character(imputed_data[["AESTDTF"]]), "")
})
```

Implementing more unit tests



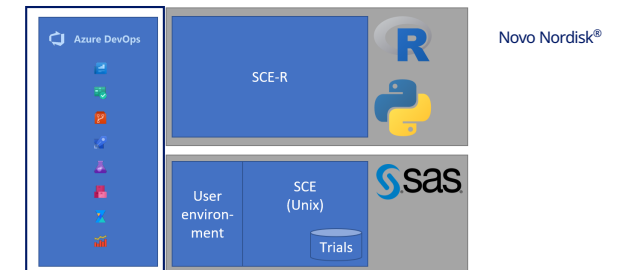
```
test_that("adamimpdtearly - full date, no imputation", {})  
test_that("adamimpdtearly - missing day", {})  
test_that("adamimpdtearly - missing month", {})  
test_that("adamimpdtearly - missing year", {})  
test_that("adamimpdtearly - missing month and day", {})  
test_that("adamimpdtearly - incorrect input date format", {})  
test_that("adamimpdtearly - multiple records", {})
```

Do Test

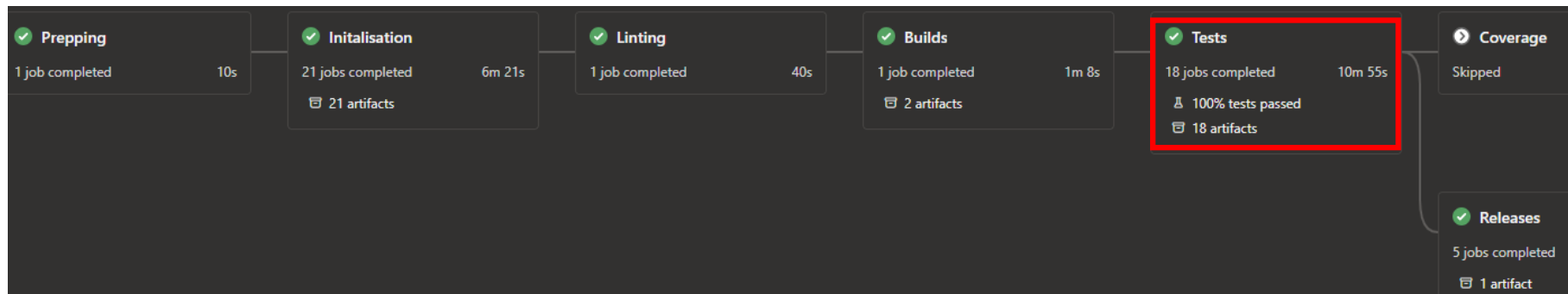


```
Console Terminal x Background Jobs x Workbench Jobs x
R 4.4.1
> devtools::test_active_file()
[ FAIL 0 | WARN 0 | SKIP 3 | PASS 12 ]
> devtools::test()
i Testing adamprogramming
✓ | F W S OK | Context
✓ |      1  0 | adamimpbaseavg
✓ |      7  | adamimpbaselvdp [31.6s]
✓ |     3 12 | adamimpdtearly [69.3s]
✓ |     2 18 | adamimpdteofdd [170.7s]
✓ |     6  7 | adamimpdtlate [18.9s]
✓ |     4 21 | adamimpdtvisitdate [182.0s]
✓ |     4 25 | adamimptmearly [87.2s]
```


Preparing CI/CD pipelines



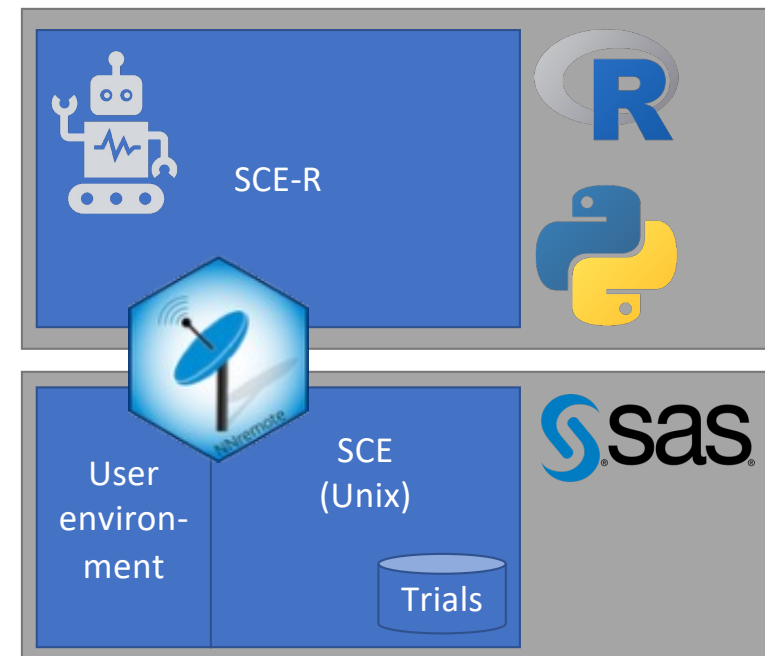
Re-use existing pipeline flow



Service account

SAS unit tests only executed interactively

- Setup service account to handle interaction between SCE-R and Unix SCE



NNremote - configuration



Looser coupling to Unix SCE

- Create variable group in ADO to store common metadata
- Remove hardcoded connection details
- Read configuration

Library > [icon] -execution

Variable group | Save | Clone | Security | Pipeline permissions | Approvals and checks | Help

Description

Variables to enable access to UNIX SCE [redacted] server in pipelines.
Also stores NNremote connection configuration to the SAS environment

☐ Link secrets from an Azure key vault as variables ⓘ

Variables

Name ↑	Value
NNremote_DataMountPoint	/home/[redacted]
NNremote_HomedirsPath	/home
NNremote_HomeMountPoint	/home/[redacted]
NNremote_RemoteServer	[redacted].novo.dk
NNremote_RemoteUser	[redacted]

NNremote – missing debug information



When things go wrong...

- Introduce *DebugLevel*
- Make configurable
- Put debug information in the log

```
560  ==== Testcase: adamimpdtearly - full date, no imputation ====
561  Exposing to data inputs to SAS via /home/██████████/.NNremote/_sassession8db118acf3b/rin
562  [1] "Executing the command: ssh -q -o BatchMode=yes -o ConnectTimeout=5 ██████████.novo.dk echo TRUE 2>&1"
563  [1] "Time spend in execution: 0.537  seconds"
564  [1] "Executing the command: scp -r ~/.temp_save_sas/RtmpaN4EBq ██████████.novo.dk:~ 2> /dev/null"
565  [1] "Time spend in execution: 0.682  seconds"
566  [1] "Preparing the command:"
567  [1] "cd /██████████/RtmpaN4EBq;/usr/local/pms/bin/sasall file8db4acbe07d.sas"
568  [1] "Executing the command: ssh ██████████.novo.dk 'cd /██████████/RtmpaN4EBq;/usr/local/pms/bin/sa
```



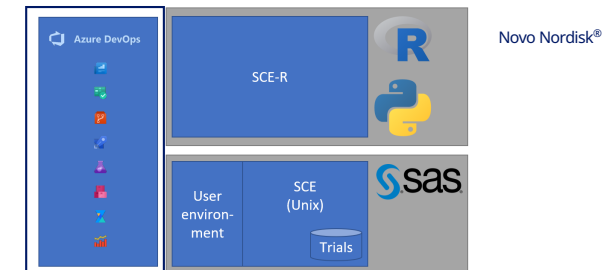
NNremote – non-deterministic failures

Network instability

- Introduce try-catch-retry
- Configure
 - Command timeout
 - Max attempts
 - Time between retries
- Improve robustness

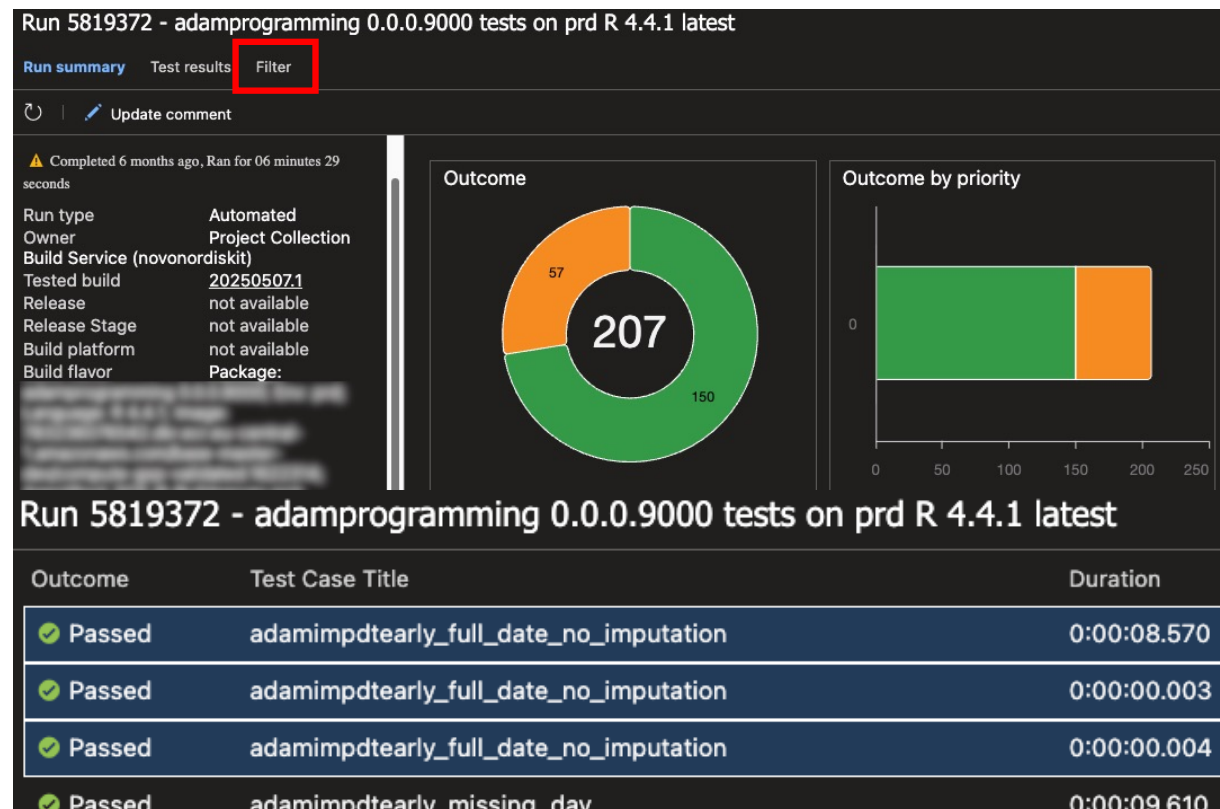
```
856 [1] "Preparing the command:"  
857 [1] "cd /RtmpaN4EBq;/usr/local/pms/bin/sasall file8db75e19c2.sas"  
858 [1] "Executing the command: ssh /RtmpaN4EBq;/usr/local/pms/b  
Connection reset by peer  
Attempt 1 failed
```

Publishing test results

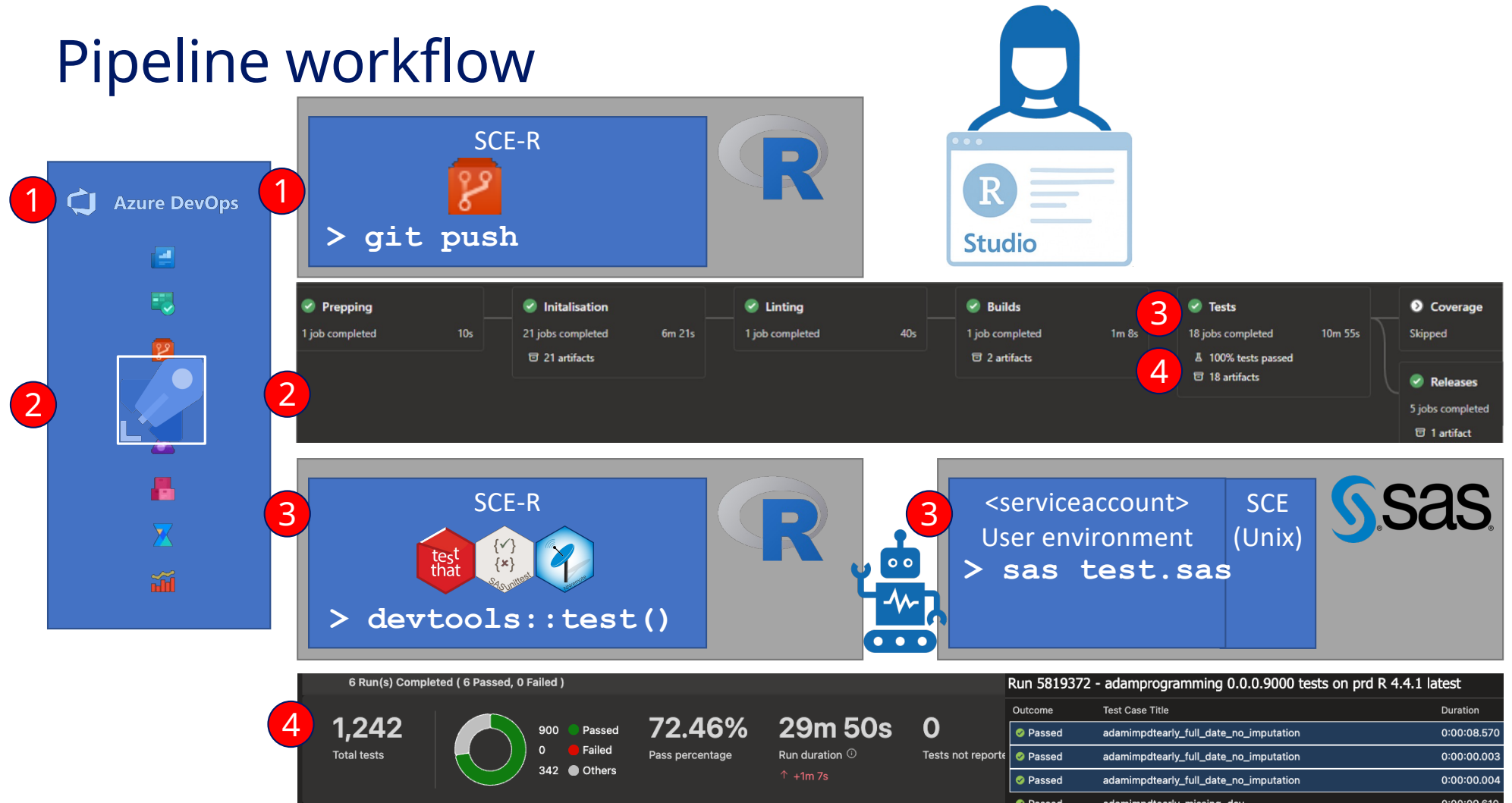


Pipelines are re-used from general R package pipeline setup

- Test result are stored in `junit` format
- Leverage ADO documentation facilities



Pipeline workflow



Conclusion

- Hassle of testing is minimised with automated tests
- Harmony with common test tool and infrastructure
- Empower SAS programmers to learn R
- Not a final solution
 - Lacks ways to publish different versions
 - Automatic signing of review

Learnings

- Debugging can be tough – logging is key
- Configuration introduces flexibility
- “Major setup before everything runs smoothly”...

Thank you

Questions or comments?

Acknowledgements:

Nicolai Skov Johnsen

Matthew Phelps

Michael Hedegaard Thomsen

mhzt@novonordisk.com