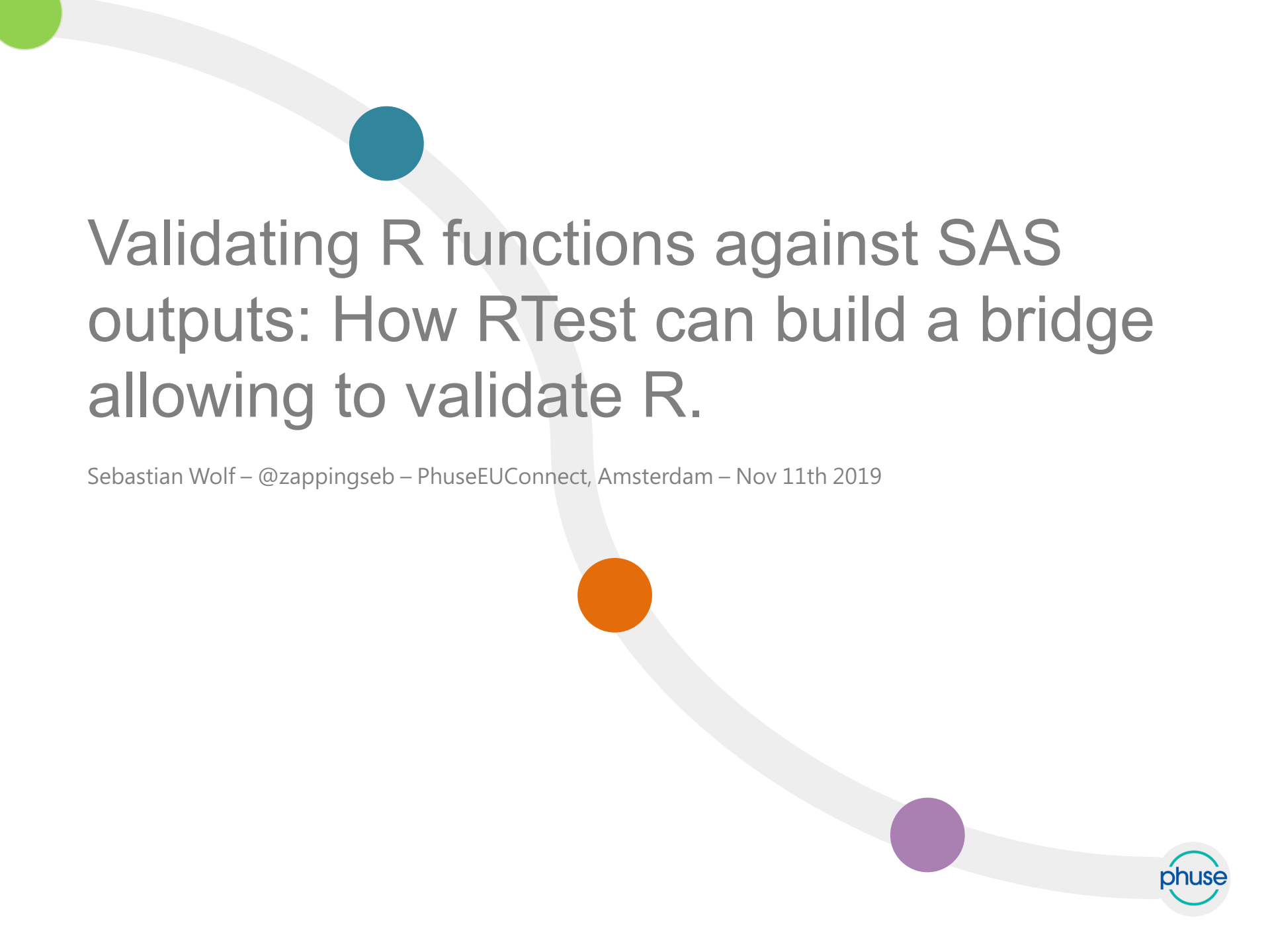
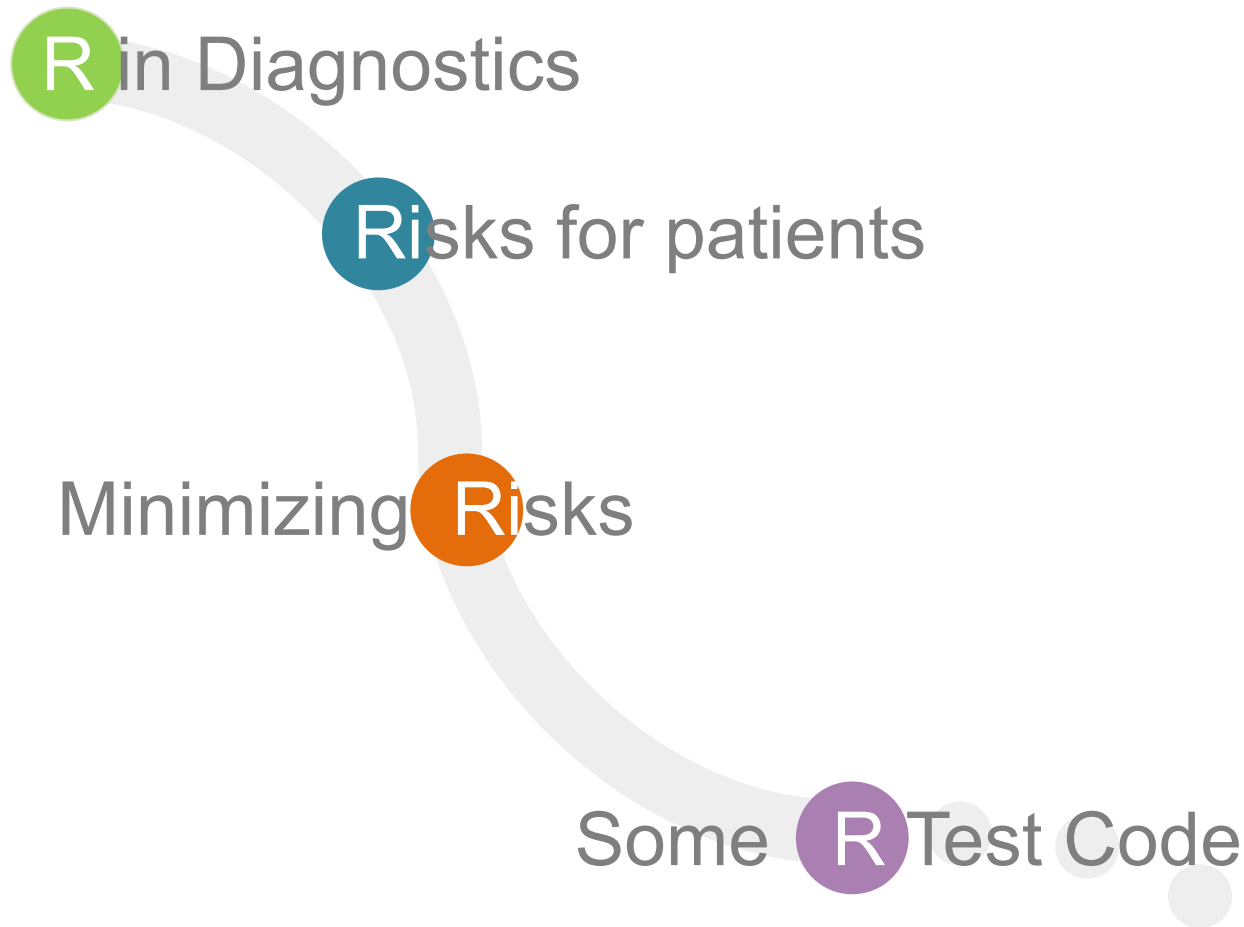


A light gray curved line starts from the top left and arcs towards the bottom right. Along this line are four colored circles: a green one at the top left, a teal one, an orange one, and a purple one. The text is positioned to the right of the teal and orange circles.

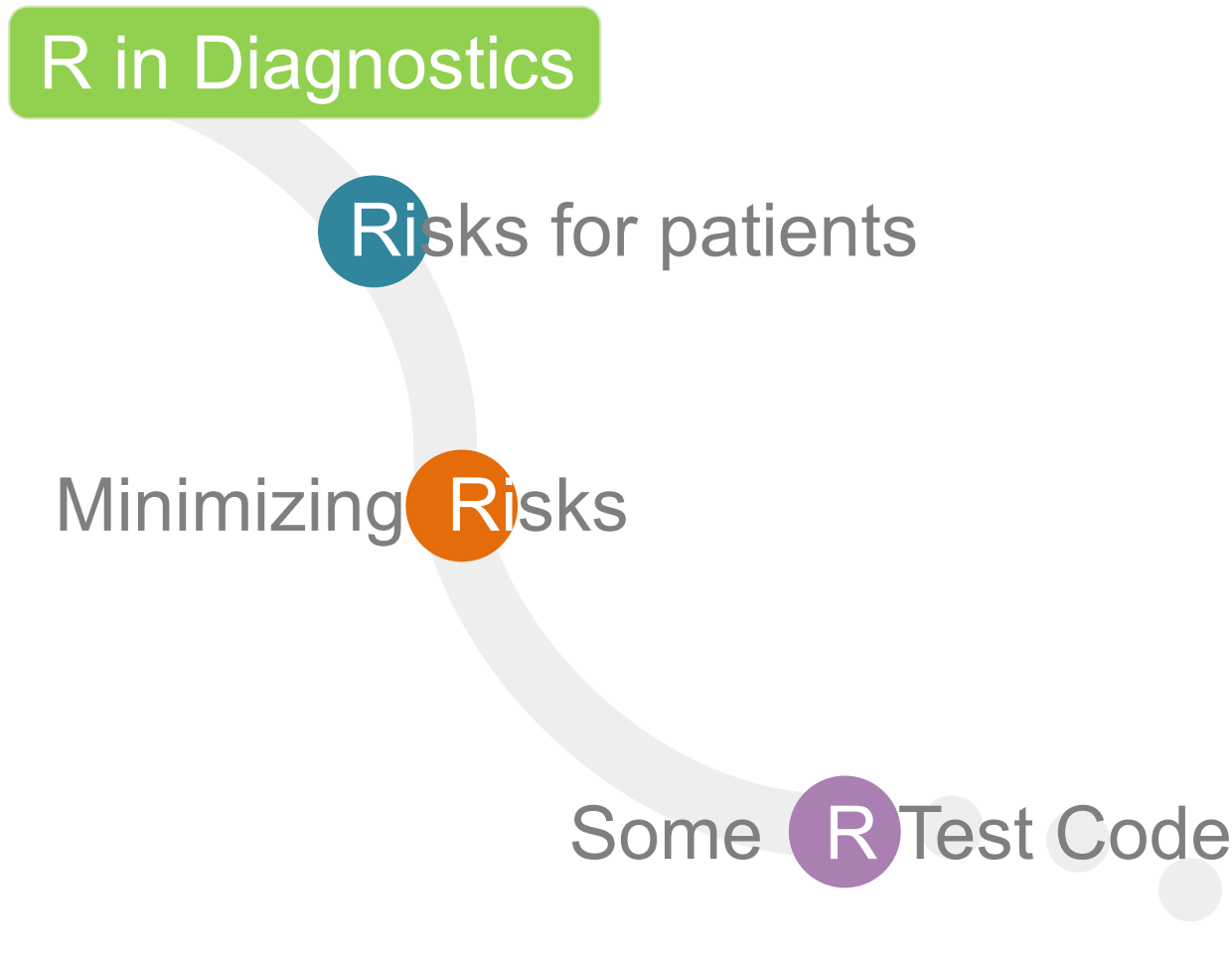
Validating R functions against SAS outputs: How RTest can build a bridge allowing to validate R.

Sebastian Wolf – @zappingseb – PhuseEUConnect, Amsterdam – Nov 11th 2019

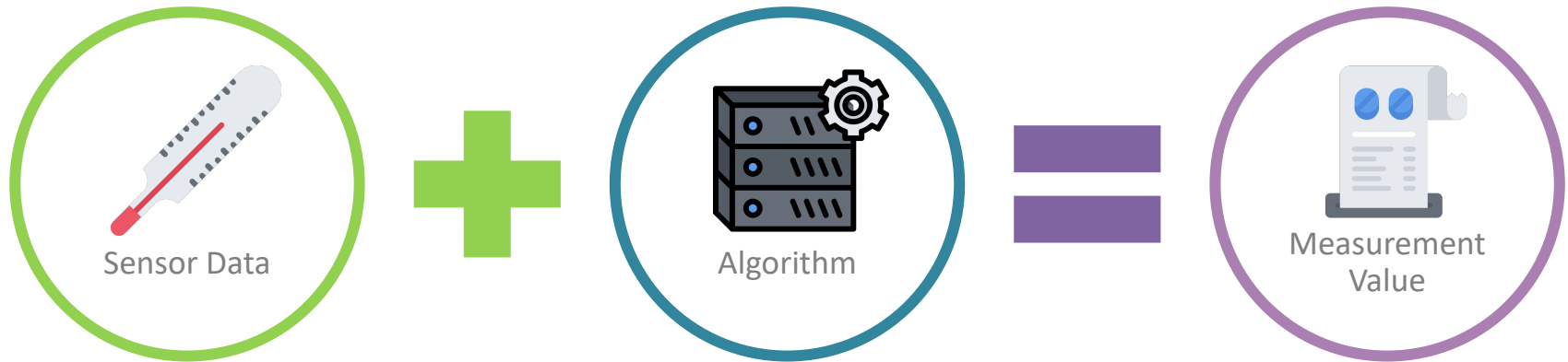
The story of RTest



The story of RTest



R in Diagnostics



Inserting a blood sample of a patient into a measurement **device**.

The blood sample and **chemical reagent** will produce a signal of some kind, mostly **optical**.

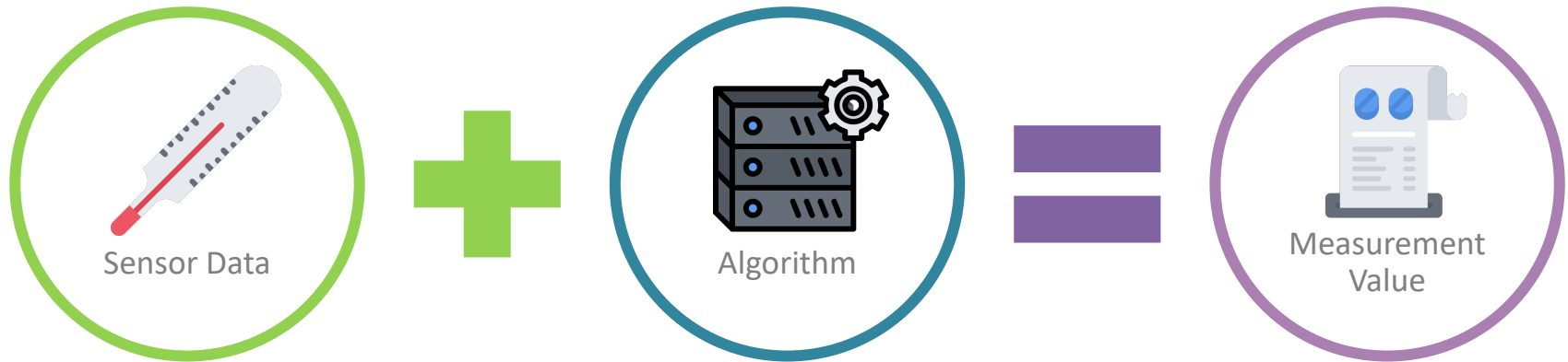
Algorithms transform optical signals into human interpretable values.

Algorithms are used to **standardize** signals for different reagents and devices.

The measurement will inform the doctor about the level of a certain compound in the blood.

Compounds can be vitamins, hormones, ...

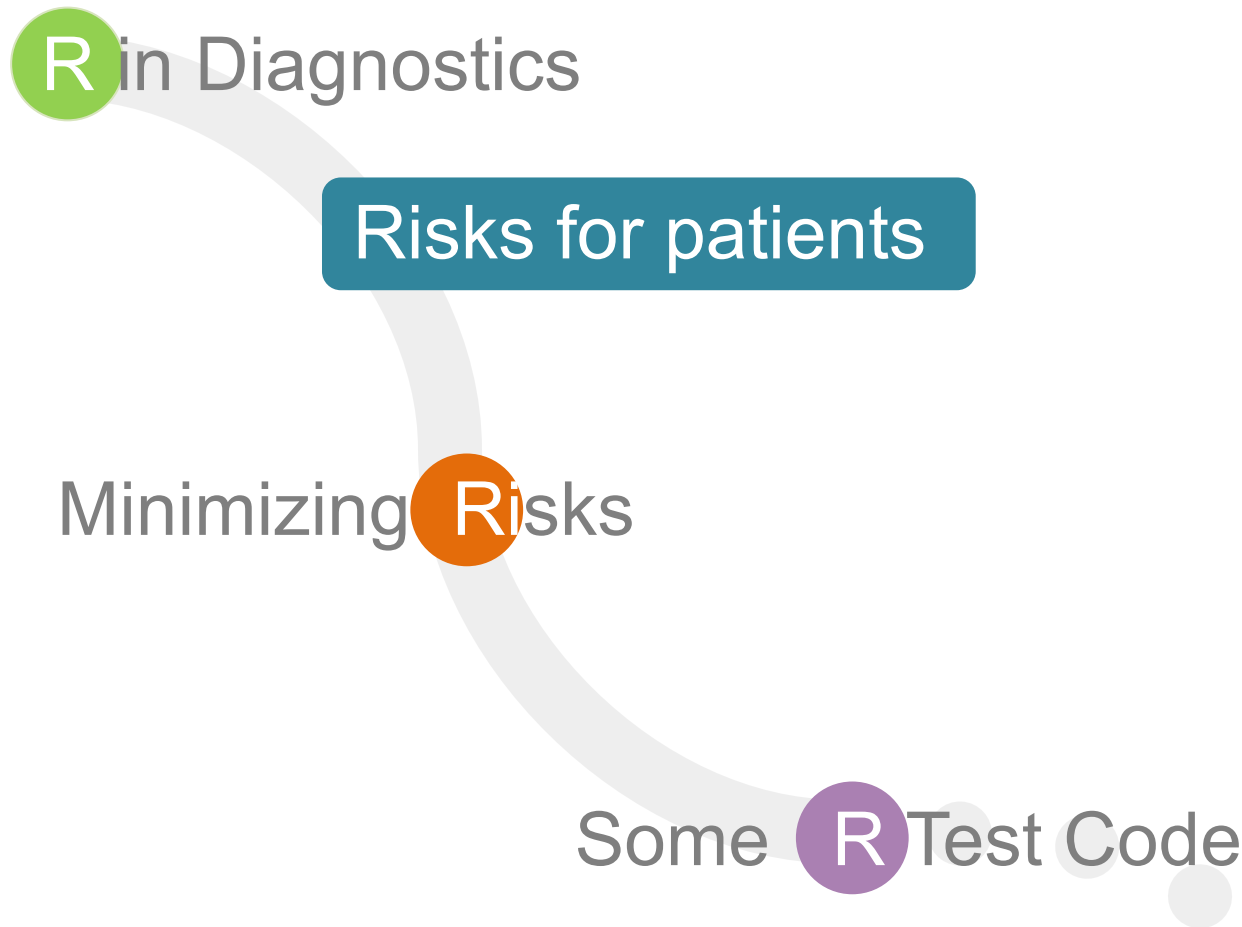
R in Diagnostics



Influenced by:

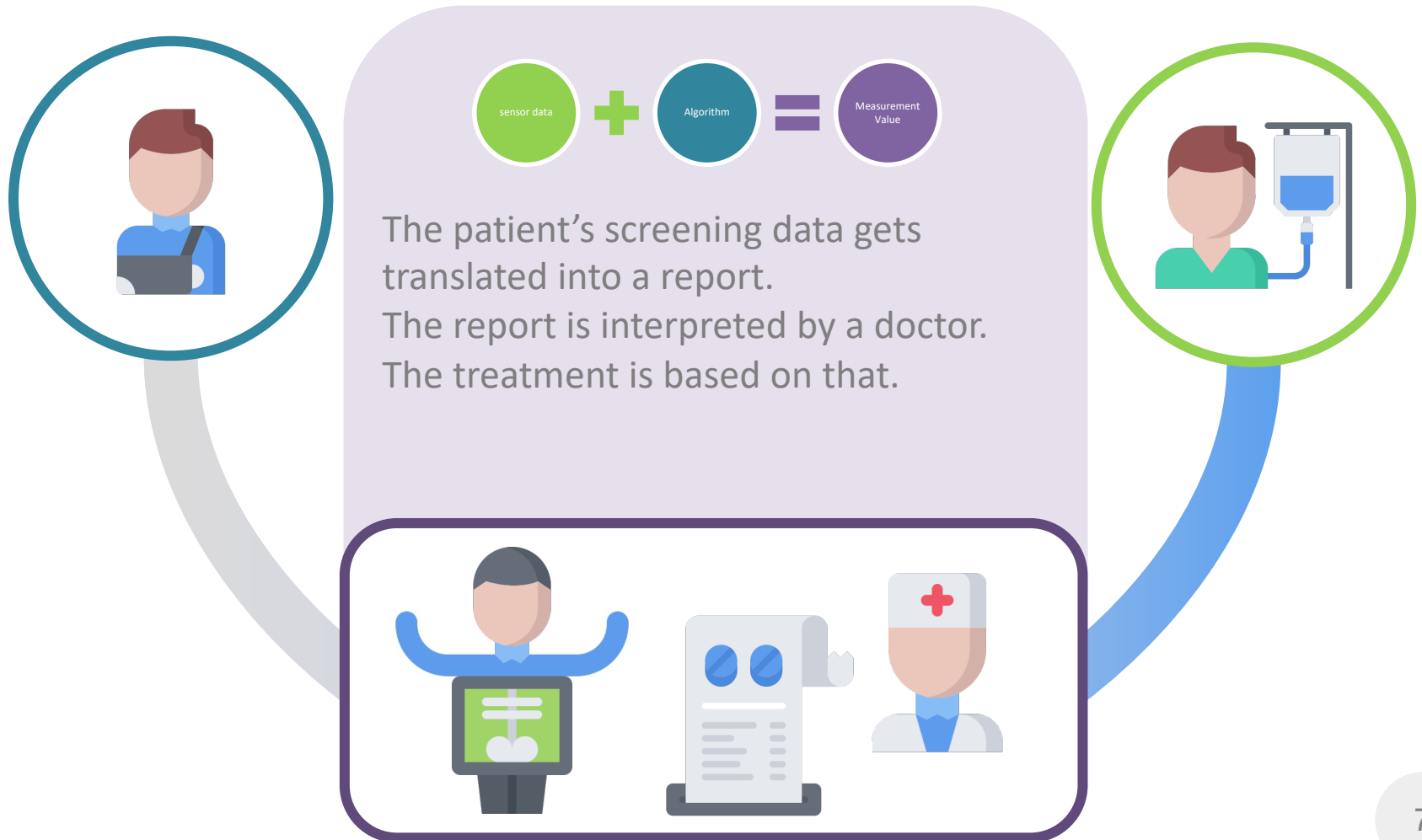
- Reagent
- Reagent Lot
- Device
- Lab factors
- Developer
- Test Case library
- Computing environment (speed)
- Blood of patient

The story of RTest



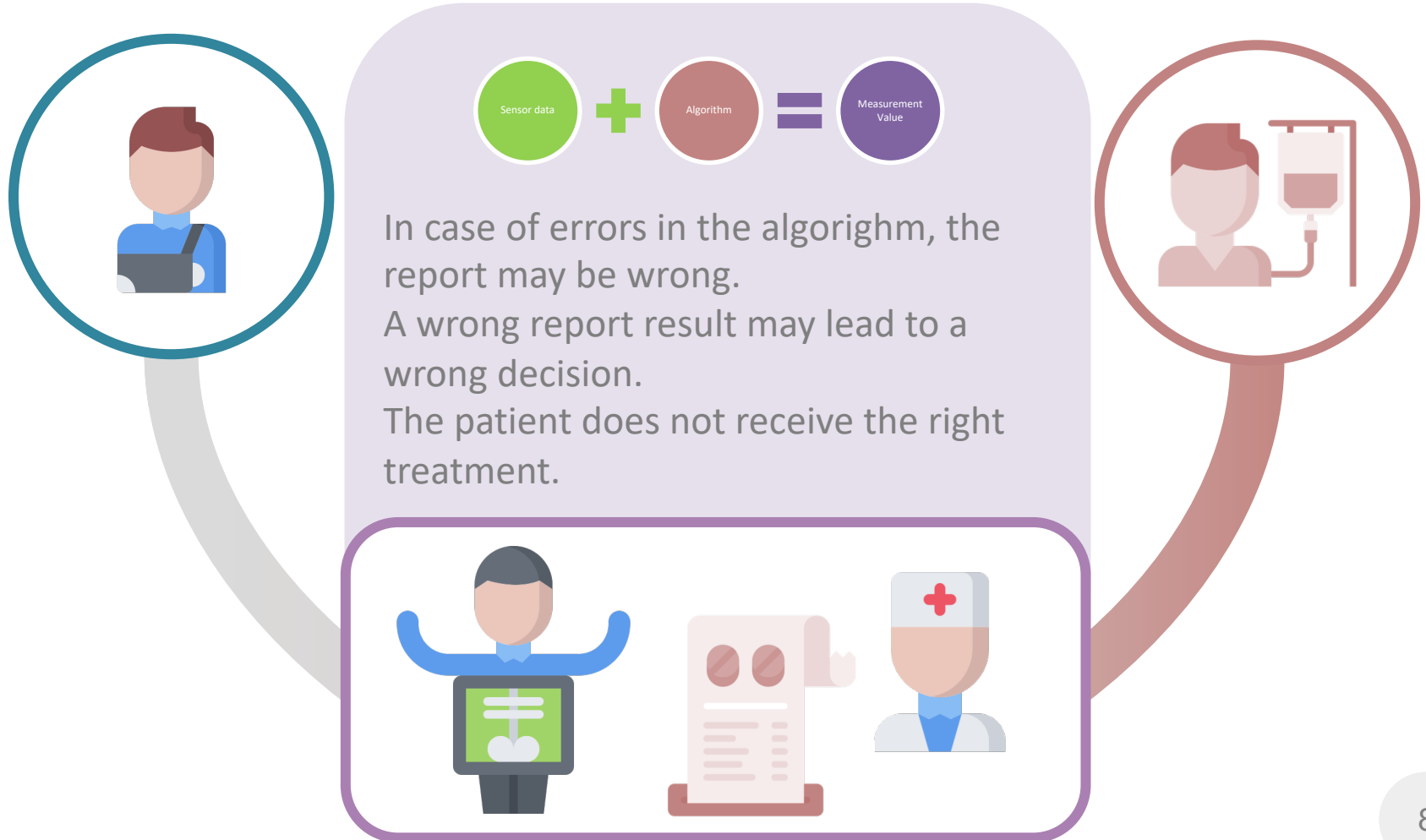
Risks for patients

Any treatment decision is based on Diagnostics reports

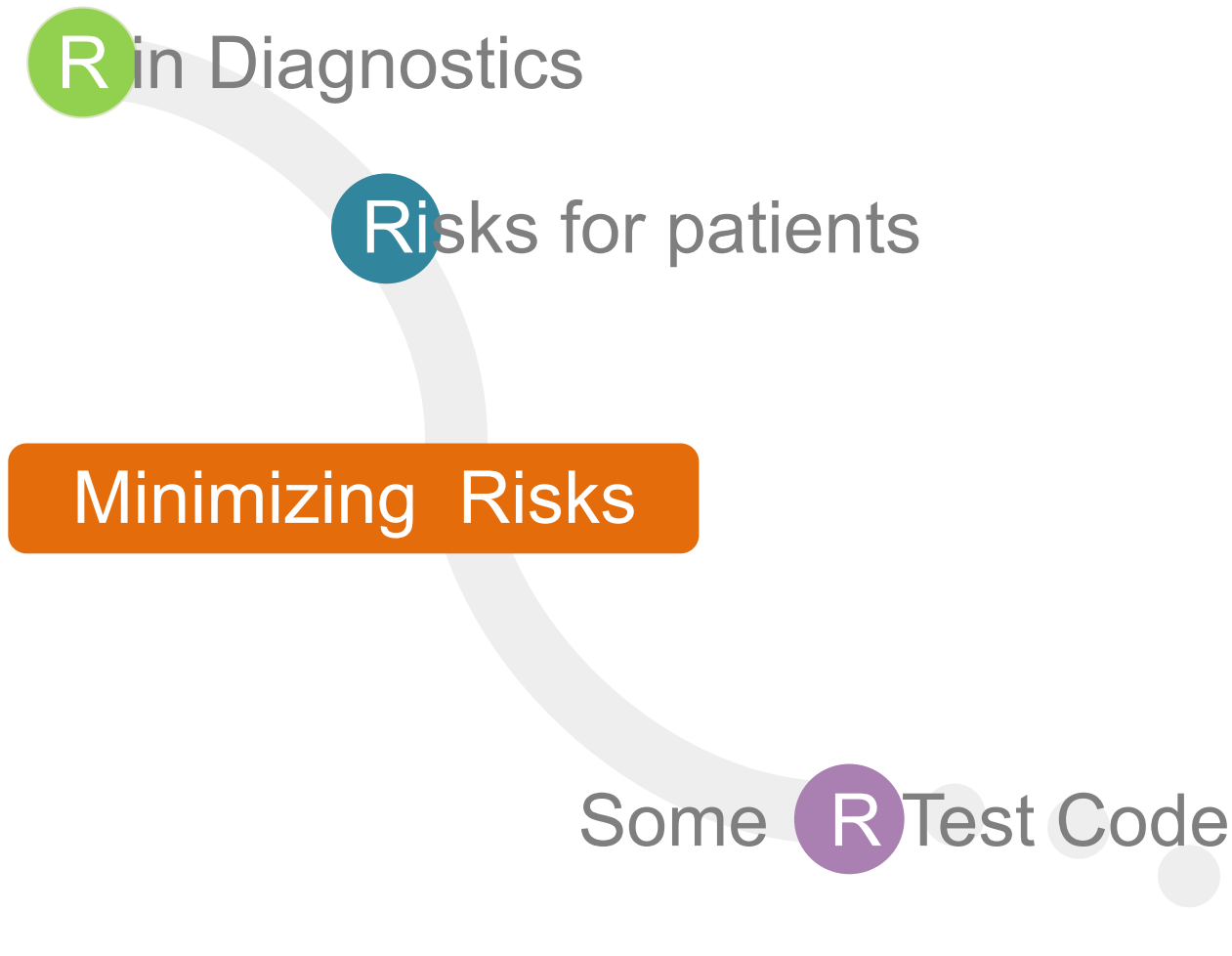


Risks for patients

In case of a wrong algorithm patients treatments can go wrong



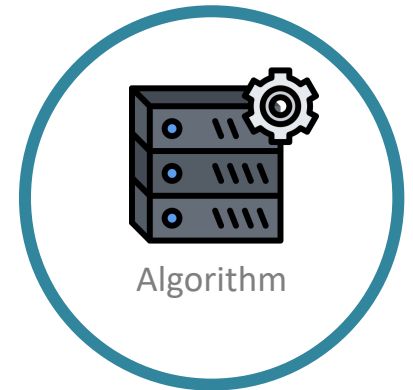
The story of RTest



Minimizing Risks

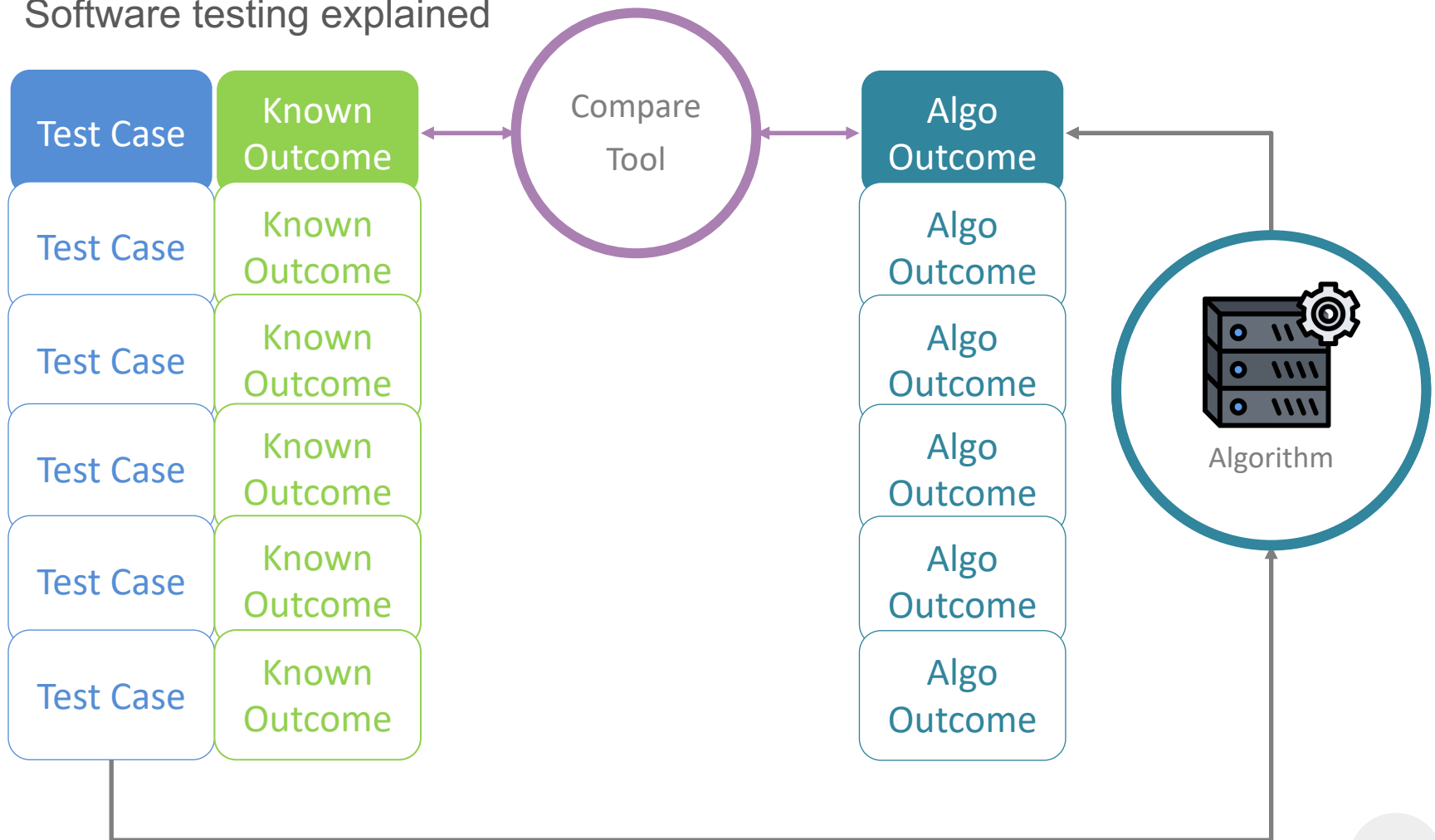
Software testing explained

Test Case	Known Outcome
Test Case	Known Outcome
Test Case	Known Outcome
Test Case	Known Outcome
Test Case	Known Outcome
Test Case	Known Outcome



Minimizing Risks

Software testing explained



Minimizing risks

How to improve testing?

1. More test cases
2. More **diverse** people writing and reading test cases
3. Easier and regular execution of test cases

Minimizing risks

How to improve testing?

1. More test cases

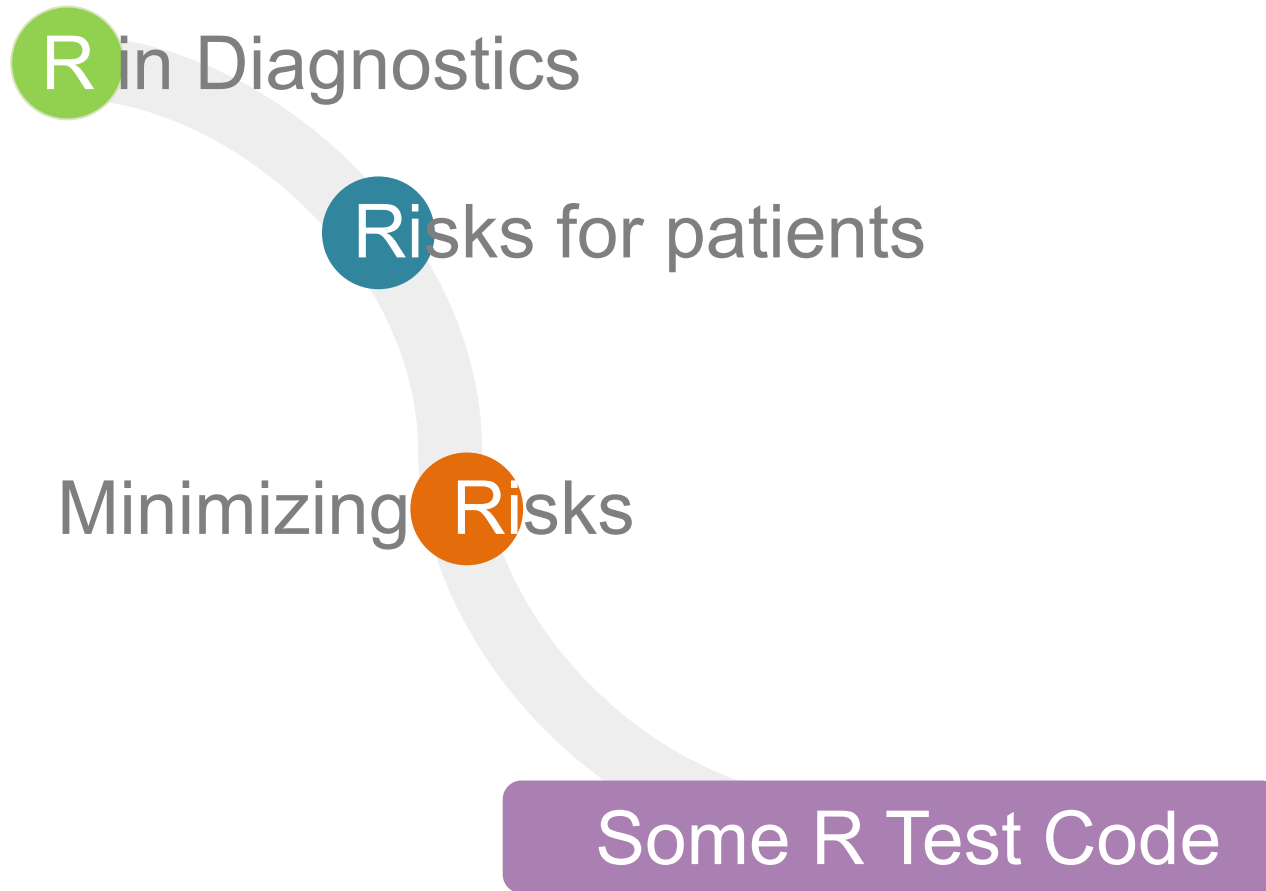
2. More **diverse** people writing and reading test cases

RTest

3. Easier and regular execution of test cases

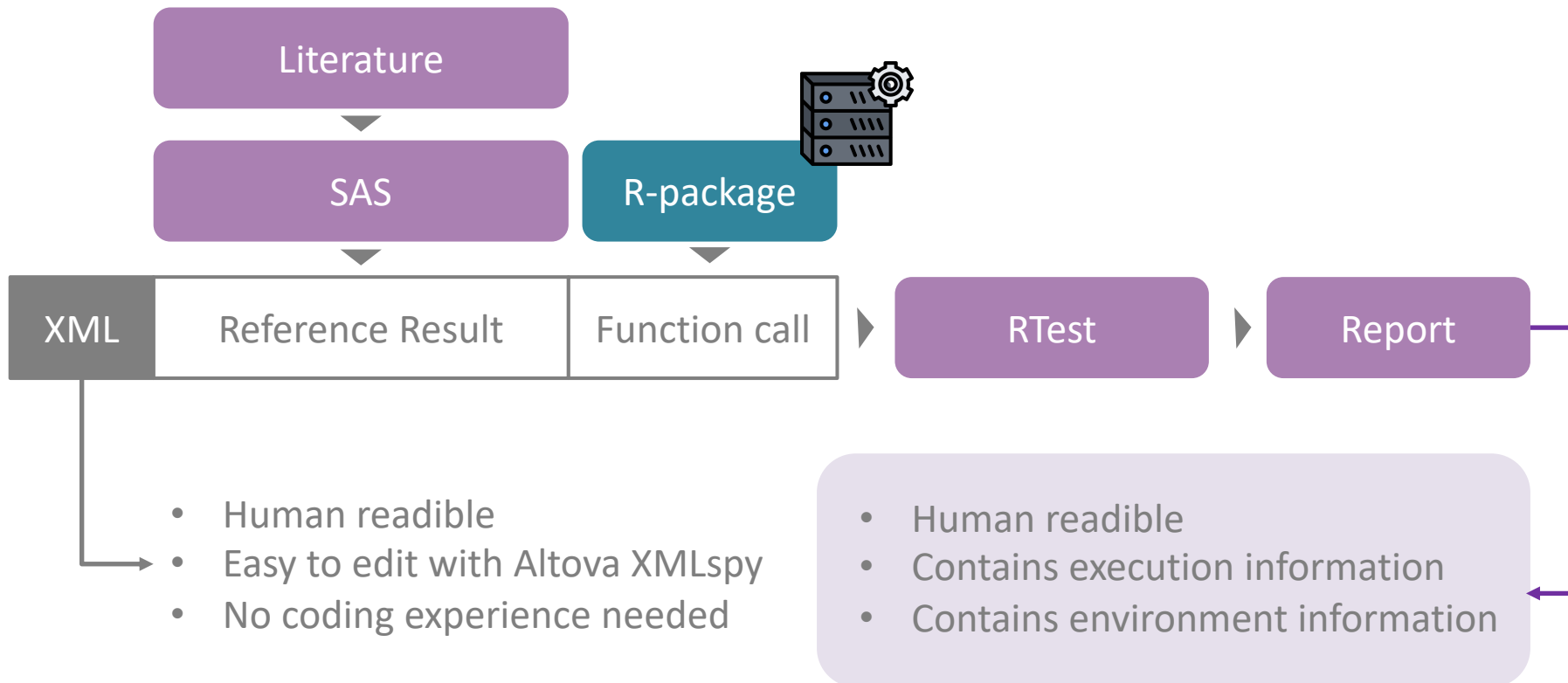
CI / CD e.g. Jenkins

The story of RTest



Some RTest Code

How Rtest works



Some RTest Code

Function to test

```
my_function <- function(  
  data = data.frame(x = c(1,2), y = c(1,2))) {  
    stopifnot(dim(data)[2] == 2)  
    data[, "sum"] <- apply(data, 1,  
      function(x) {sum(x)})  
    return(data)  
  }
```

Adds a column «sum» which adds up column 1 and column 2

Some RTest Code

XML test description – defines the reference value and the comparison parameters

```
<my_function test-desc="Test data.frame">
  <params>
    <RTestData_input_data param="data" name="test01" />
  </params>
  <reference>
    <col-defs>
      <coldef name="x" type="numeric" />
      <coldef name="y" type="numeric" />
      <coldef name="sum" type="numeric" />
    </col-defs>
    <row>
      <cell>1</cell>
      <cell>2</cell>
      <cell>3</cell>
    </row>
    <row>
      <cell>1</cell>
      <cell>2</cell>
      <cell>3.5</cell>
    </row>
  </reference>
  <testspec>
    <execution execution-type="silent" />
    <return-value compare-type="equal" diff-type="absolute"
      tolerance="0.001" />
  </testspec>
</my_function>
```

Some RTest Code

XML input data – contains a simple data frame with the definition of column types

```
<input-data>
<data.frame name="test01">
  <col-defs>
    <coldef name="x" type="numeric" />
    <coldef name="y" type="numeric" />
  </col-defs>
  <row>
    <cell>1</cell>
    <cell>2</cell>
  </row>
  <row>
    <cell>1</cell>
    <cell>2</cell>
  </row>
</data.frame>
</input-data>
```

Some RTest Code

The RTest report visualizes the test result in a human-readable way and also describes all reference values.

```
R> RTest.execute(getwd(), "TestCase.xml")
```

Test		No. of Exectations		User	System	Real	Status					
Check return value (data.frame).		13		0.05	0	0.04	FAILED					
#	Test	Received	Expected	Info							Status	
1	Equal Row Number	2	2								SUCCESS	
2	Equal Column Number	3	3								SUCCESS	
#	Test	Column	Received	Expected	Info							Status
3	Equal Column Name	1	x	x								SUCCESS
4	Equal Column Name	2	y	y								SUCCESS
5	Equal Column Name	3	sum	sum								SUCCESS
#	Test	Row	Received	Expected	Info							Status
6	Equal Row Name	1	1	1								SUCCESS
7	Equal Row Name	2	2	2								SUCCESS
#	Test	Row	Column	Received	Data Type	Expected	Data Type	Diff Type	Compare Type	Tolerance	Info	Status
8	Equal Value	1 (1)	1 (x)	1	double	1	double	absolute	equal	0.001		SUCCESS
9	Equal Value	1 (1)	2 (y)	2	double	2	double	absolute	equal	0.001		SUCCESS
10	Equal Value	1 (1)	3 (sum)	3	double	3	double	absolute	equal	0.001		SUCCESS
11	Equal Value	2 (2)	1 (x)	1	double	1	double	absolute	equal	0.001		SUCCESS
12	Equal Value	2 (2)	2 (y)	2	double	2	double	absolute	equal	0.001		SUCCESS
13	Equal Value	2 (2)	3 (sum)	3	double	3.5	double	absolute	equal	0.001	3 not equal to 3.5. 1/1 mismatches [1] 3 - 3.5 == -0.5	FAILED

Some final notes

Where to get the Code?



Install: <https://cran.r-project.org/web/packages/RTest/index.html>

Code: <https://github.com/zappingseb/RTest>

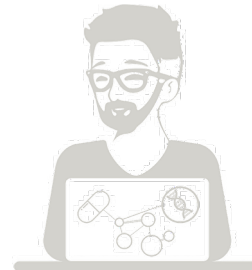
Read: <https://phuse2019.mail-wolf.de>

Testing: <https://travis-ci.org/zappingseb/RTest>

My recommendation for today:

Adrian Waddel on Analyzing Clinical Trials with R
SD09 – 4.30 pm – E104

Thank you



```
#' @email      sebastian@mail-wolf.de
# '
# ' @blog       mail-wolf.de
# '
# ' @bugReport  github.com/zappingseb
# '
# ' @cv         linkedin.com/in/zappingseb
# '
# ' @twitter    twitter.com/zappingseb
```

r

```
#' Roche Diagnostics
# ' Biostats Penzberg, Germany
# '
# ' @co-developer
# '   Sergej Potapov
# '   sergej.potapov@roche.com
```

