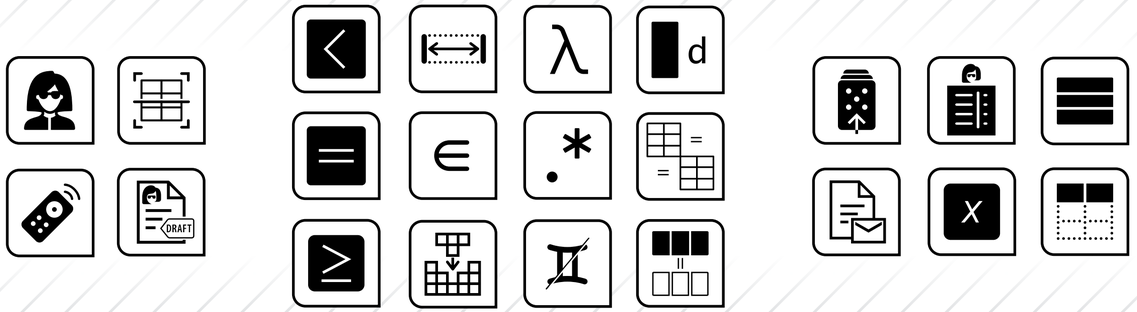
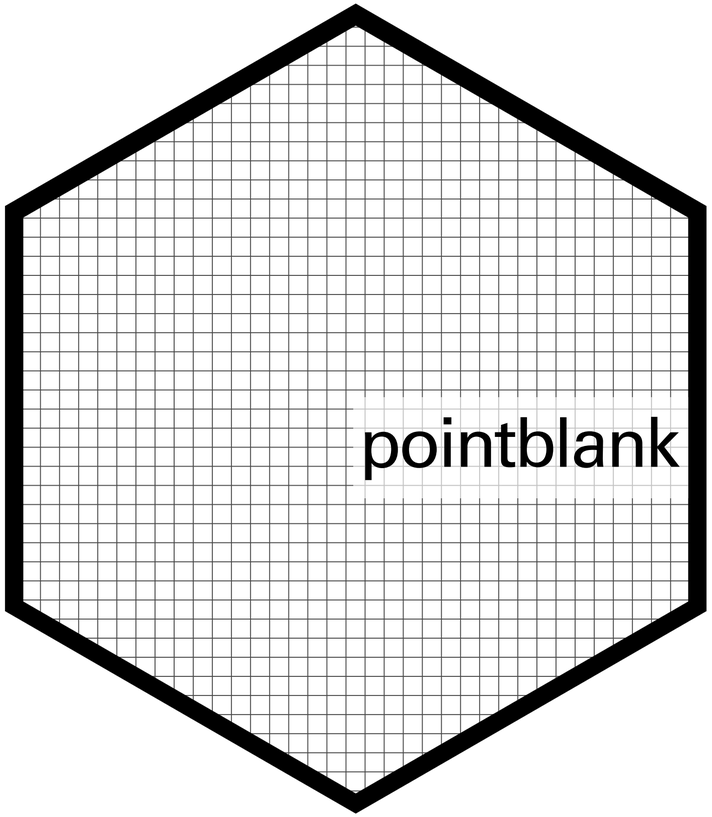
Validating Data Tables With the **pointblank** Package





rich-iannone

@riannone

rich@rstudio.com

Data Validation in pointblank

PRIMARY WORKFLOWS

You really need to understand and get ahead of **data quality** issues.

You need to **check your data** before it proceeds further down a pipeline.

SECONDARY WORKFLOWS

Validating data tables in **testthat**-type **unit tests**.

Data checks to get **logical values** for programming.

OVERALL DESIGN CONSIDERATIONS FOR PACKAGE

Work with **local tables** and **database tables** with minimal changes in the API.

Provide extra tools for **understanding** new local and remote datasets.

Have reporting outputs translated to multiple **spoken languages**.
EN ▪ FR ▪ DE ▪ IT ▪ ES

Give a lot of attention to making the package **docs and examples** the best they can be.

The Data Quality Workflow

You really need to
understand and get on top of
your **data quality**.

This centers around data quality reporting that
summarizes the results of **validation steps**.

The reporting can be stored, published,
exported, and transformed.

This workflow aligns with a key tenet of the 📦:
discover, communicate, and help solve DQ issues.

The Data Quality Workflow

This report-based workflow begins with creating the agent.

```
create_agent()
```

The **agent** is an integral part of the **data quality workflow**.

The Data Quality Workflow

The **agent** is given the **target table**…

`actions`

`end_fns`

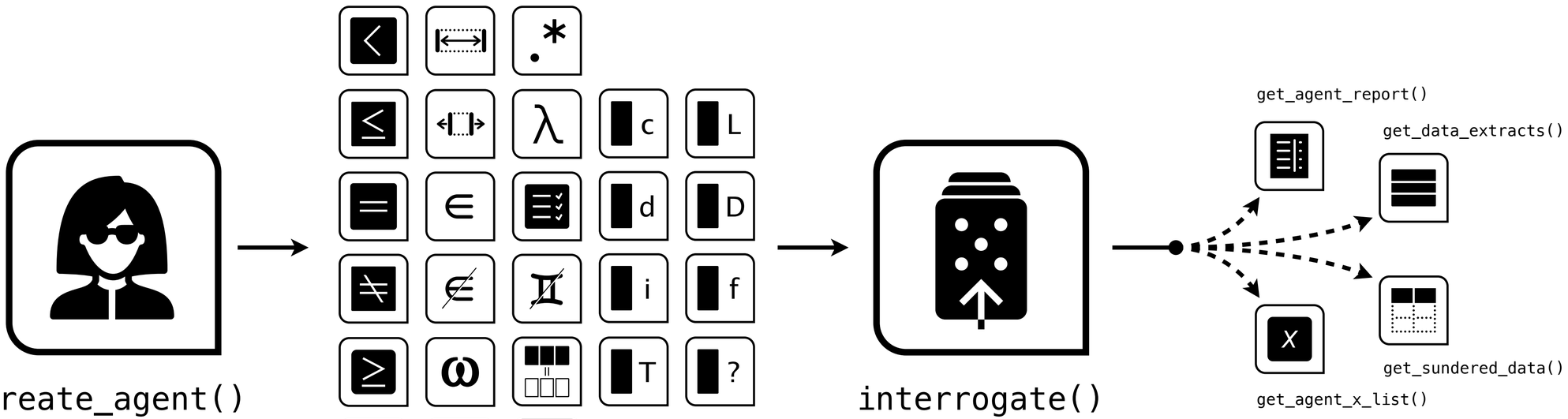
`lang`

…and some directives on interrogation.

`create_agent()`

The **agent** is an integral part of the **data quality workflow**.

Post-Interrogation Operations



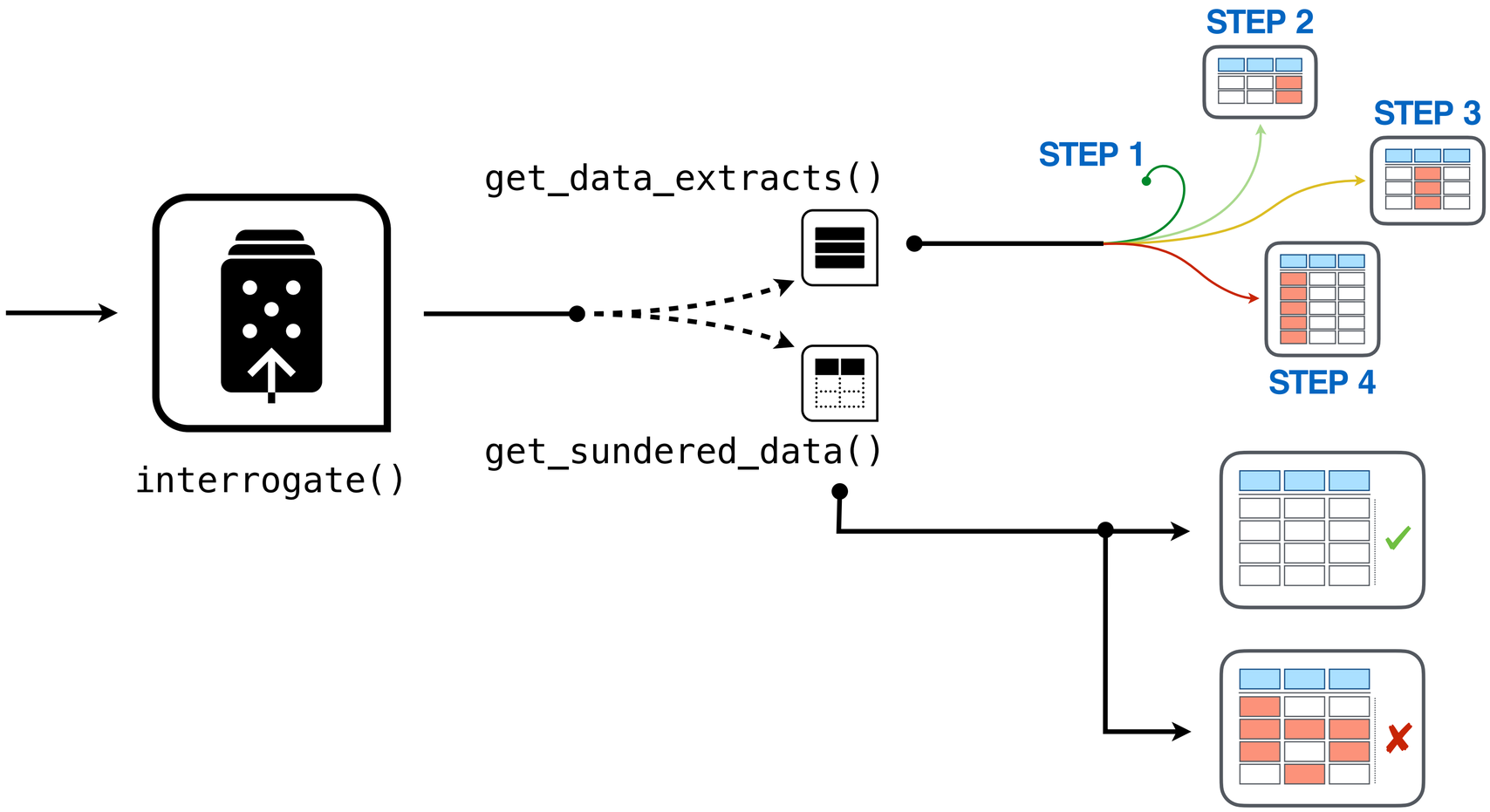
The **agent** is an integral part of the **data quality workflow**.

Use validation functions to generate a **validation plan**.

The **agent** performs the **interrogation**. Final step of workflow.

Use **post-interrogation** functions for reporting and root-cause analysis.

Post-Interrogation Operations



Get **table fragments** for failing test units (from row-based validations)

Get two table pieces, split across the **passing** and **failing** test units.

Data Validation with a Simple Table

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

Let’s start with a simple table

5 rows, 3 columns

Data Validation with a Simple Table

VALIDATION RULES BASED ON DOMAIN KNOWLEDGE

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

simple table

5 rows, 3 columns

1. All values in **c** should be greater than `15`
2. All values in **b** should be either `0` or `1`
3. All values in **a** should fit a pattern of three lowercase letters and a digit
4. Values in **c** must be `≥20` if **b** is `1`; if **b** is `0` then values in **c** must be `<20`
5. Columns **a**, **b**, and **c** should not have any missing values.

validation plan

5 steps

Data Validation with a Simple Table

TRANSLATION OF RULES TO AVAILABLE VALIDATION FUNCTIONS

1. All values in **c** should be greater than `15`
2. All values in **b** should be either `0` or `1`
3. All values in **a** should fit a pattern of three lowercase letters and a digit
4. Values in **c** must be `≥20` if **b** is `1`; if **b** is `0` then values in **c** must be `<20`
5. Columns **a**, **b**, and **c** should not have any missing values.

validation plan

5 steps

≥	`col_vals_gte()`
∈	`col_vals_in_set()`
.*	`col_vals_regex()`
λ	`col_vals_expr() + case_when()`
∅	`col_vals_not_null()`

validation functions

5 `col_vals_*()` functions

Data Validation with a Simple Table

TRANSLATION OF RULES TO AVAILABLE VALIDATION FUNCTIONS

1. All values in **c** should be greater than `15`
2. All values in **b** should be either `0` or `1`
3. All values in **a** should fit a pattern of three lowercase letters and a digit
4. Values in **c** must be `≥20` if **b** is `1`; if **b** is `0` then values in **c** must be `<20`
5. Columns **a**, **b**, and **c** should not have any missing values.

validation plan

5 steps

```
col_vals_gte(c, 15)
```

```
col_vals_in_set(b, c(0, 1))
```

```
col_vals_regex(a, "[a-z]{3}[0-9]")
```

```
col_vals_expr(~case_when(
  b == 1 ~ c >= 20,
  b == 0 ~ c <  20))
```

```
col_vals_not_null(vars(a, b, c))
```

validation functions

5 `col_vals_*()` functions

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

simple table

5 rows, 3 columns

1 `col_vals_gte(c, 15)`

2 `col_vals_in_set(b, c(0, 1))`

3 `col_vals_regex(a, "[a-z]{3}[0-9]")`

4
```
col_vals_expr(~case_when(
  b == 1 ~ c >= 20,
  b == 0 ~ c <  20))
```

5 `col_vals_not_null(vars(a, b, c))`

validation functions

5 `col_vals_*()` functions

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 1

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

TEST UNITS

1 `col_vals_gte(c, 15)`

INTERROGATE

REPORT

UNITS	PASS	FAIL
5	4 0.8	1 0.2

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 2

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

TEST UNITS

2 ∈ `col_vals_in_set(b, c(0, 1))`

INTERROGATE

REPORT

UNITS	PASS	FAIL
5	4 0.8	1 0.2

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 3

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

TEST UNITS

3 `col_vals_regex(a, "[a-z]{3}[0-9]")`

INTERROGATE

REPORT

UNITS	PASS	FAIL
5	3 0.6	2 0.4

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 4

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

TEST UNITS

4

```
col_vals_expr(~case_when(
  b == 1 ~ c >= 20,
  b == 0 ~ c <  20))
```

INTERROGATE

REPORT

UNITS	PASS	FAIL
4	4 1.0	0 0

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 5 EXPANSION TO THREE DISCRETE STEPS

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

5 `col_vals_not_null(vars(a, b, c))`

MULTIPLE COLUMNS,
WILL EXPAND

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 5 EXPANSION TO THREE DISCRETE STEPS

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

5 `col_vals_not_null(vars(a))`

6 `col_vals_not_null(vars(b))`

7 `col_vals_not_null(vars(c))`

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 5

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

TEST UNITS

5 `col_vals_not_null(vars(a))`

6 `col_vals_not_null(vars(b))`

7 `col_vals_not_null(vars(c))`

INTERROGATE

REPORT

UNITS	PASS	FAIL
5	4 0.8	1 0.2

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 6

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

TEST UNITS

5 `col_vals_not_null(vars(a))`

6 `col_vals_not_null(vars(b))`

7 `col_vals_not_null(vars(c))`

INTERROGATE

REPORT

UNITS	PASS	FAIL
5	5 1.0	0 0

Data Validation with a Simple Table

INTERROGATION OF TABLE USING THE VALIDATION PLAN
STEP 7

a	b	c
yko2	1	23.1
lju7	0	16.3
qib0	1	21.2
sd33	1	24.9
NA	2	NA

TEST UNITS

5 `col_vals_not_null(vars(a))`

6 `col_vals_not_null(vars(b))`

7 `col_vals_not_null(vars(c))`

INTERROGATE

REPORT

UNITS	PASS	FAIL
5	4 0.8	1 0.2

The Data Validation Report

	STEP	UNITS	PASS	FAIL
1	col_vals_gte()	5	4 0.8	1 0.2
2	col_vals_in_set()	5	4 0.8	1 0.2
3	col_vals_regex()	5	3 0.6	2 0.4
4	col_vals_expr()	4	4 1.0	0 0
5	col_vals_not_null()	5	4 0.8	1 0.2
6	col_vals_not_null()	5	5 1.0	0 0
7	col_vals_not_null()	5	4 0.8	1 0.2

For better reporting on data quality, can set thresholds and use side effects.

Failure thresholds can be set for three states

W WARNING

S STOP

N NOTIFY

Let's set:

W to 1
S to 2
(**N** not set)

R CODE

```r
action_levels(
  warn_at = 1,
  stop_at = 2
)
```

The Data Validation Report

	STEP	UNITS	PASS	FAIL	W	S	N
1	col_vals_gte()	5	4 0.8	1 0.2	●	○	—
2	col_vals_in_set()	5	4 0.8	1 0.2	●	○	—
3	col_vals_regex()	5	3 0.6	2 0.4	●	●	—
4	col_vals_expr()	4	4 1.0	0 0	○	○	—
5	col_vals_not_null()	5	4 0.8	1 0.2	●	○	—
6	col_vals_not_null()	5	5 1.0	0 0	○	○	—
7	col_vals_not_null()	5	4 0.8	1 0.2	●	○	—

The Data Validation Report

Pointblank Validation

[2022-10-24|13:51:28]

TIBBLE	simple_table	WARN	1	STOP	2	NOTIFY	—

	STEP	COLUMNS	VALUES	TBL	EVAL	UNITS	PASS	FAIL	W	S	N	EXT
1	col_vals_gte()	c	15	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV
2	col_vals_in_set()	b	0, 1	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV
3	col_vals_regex()	a	[a-z]{3}[0-9]	○→	✓	5	3 0.60	2 0.40	●	●	—	CSV
4	col_vals_expr()	—	case_when(b == …	○→	✓	4	4 1.00	0 0.00	○	○	—	—
5	col_vals_not_null()	a	—	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV
6	col_vals_not_null()	b	—	○→	✓	5	5 1.00	0 0.00	○	○	—	—
7	col_vals_not_null()	c	—	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV

2022-10-24 13:51:28 EDT | 1.3 s | 2022-10-24 13:51:30 EDT

The Data Validation Report

	STEP	COLUMNS	VALUES	TBL	EVAL	UNITS	PASS	FAIL	W	S	N	EXT
1	col_vals_gte()	c	15	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV
2	col_vals_in_set()	b	0, 1	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV
3	col_vals_regex()	a	[a-z]{3}[0-9]	○→	✓	5	3 0.60	2 0.40	●	●	—	CSV
4	col_vals_expr()	—	case_when(b == …	○→	✓	4	4 1.00	0 0.00	○	○	—	—
5	col_vals_not_null()	a	—	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV
6	col_vals_not_null()	b	—	○→	✓	5	5 1.00	0 0.00	○	○	—	—
7	col_vals_not_null()	c	—	○→	✓	5	4 0.80	1 0.20	●	○	—	CSV

VALIDATION FUNCTION

VALIDATION STEP INDEX

ASSOCIATED COLUMNS AND VALUES

TABLE MUTATION STATE

TBL EVAL RESULT

TEST UNITS:
TOTAL
PASSING
FAILING

STATES:
WARNING
STOP
NOTIFY

DOWNLOAD EXTRACTS

Code Needed for the Data Validation

R CODE

```r
agent <-
  create_agent(
    tbl = simple_table,
    actions = action_levels(warn_at = 1, stop_at = 2),
  ) %>%
  col_vals_gte(vars(c), 15) %>%
  col_vals_in_set(vars(b), c(0, 1)) %>%
  col_vals_regex(vars(a), "[a-z]{3}[0-9]") %>%
  col_vals_expr(~case_when(
    b == 1 ~ c >= 20,
    b == 0 ~ c < 20
  )) %>%
  col_vals_not_null(vars(a, b, c)) %>%
  interrogate()

agent
```

Learning More About pointblank

You can try out dozens of **pointblank** examples in **RStudio Cloud**

RStudio Cloud | **pointblank Test Drive**

The link is available in the package README and in the project website:

`github.com/rich-iannone/pointblank`

`rich-iannone.github.io/pointblank`

Learning More About pointblank

You can try out dozens of **pointblank** examples in **RStudio Cloud**

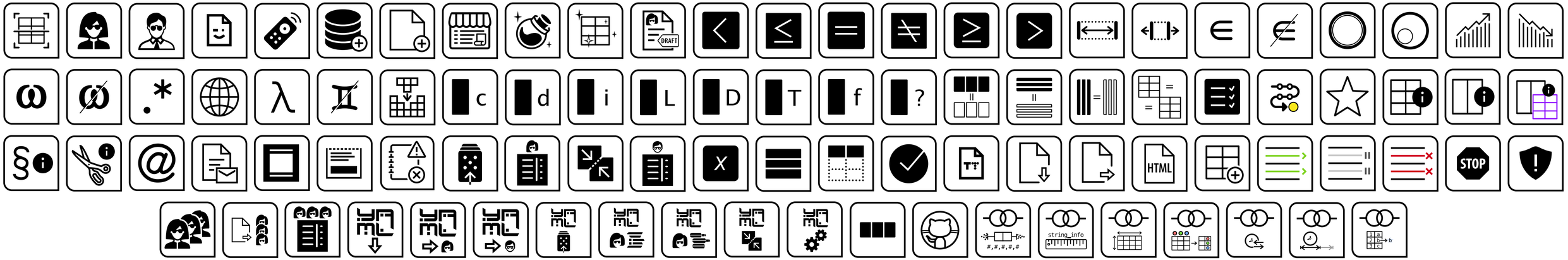
RStudio Cloud | pointblank Test Drive

The link is available in the package README and the project website

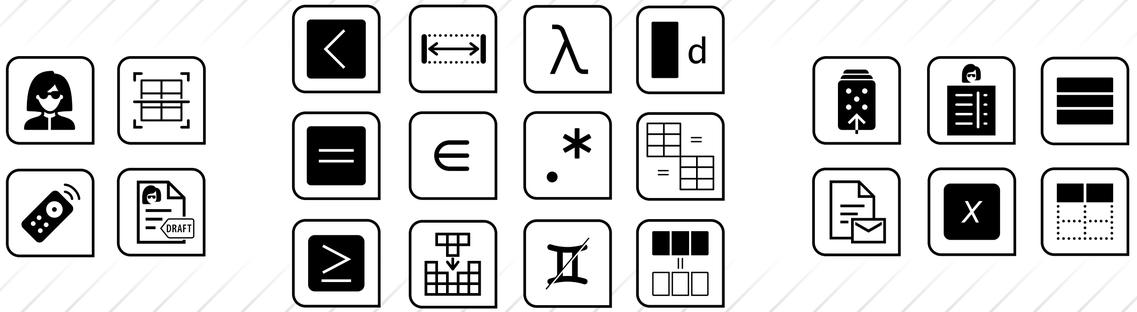
`github.com/rich-iannone/pointblank` `rich-iannone.github.io/pointblank`

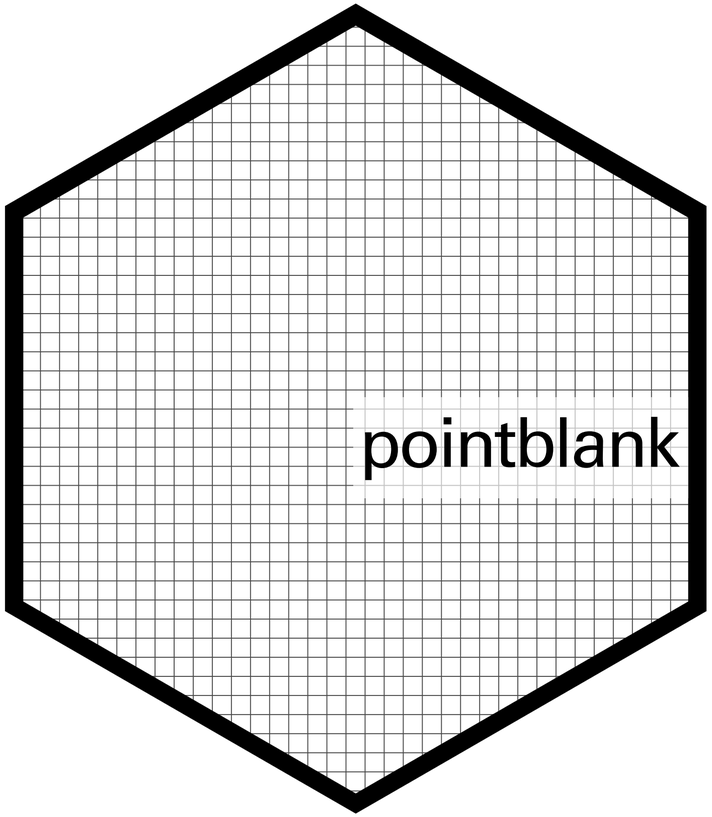
pointblank's *Function Reference* section has per-function info

`https://rich-iannone.github.io/pointblank/reference`



Validating Data Tables With the **pointblank** Package





rich-iannone

@riannone

rich@rstudio.com