“Regression testing” of scientific workflows: introduction to *TestKraken*

Regression testing

Regression testing:

- testing changes to computer programs to make sure that the older code still works with the new changes
- you don't have to knows the expected result, the assumption is that the past results were correct

Regression testing:

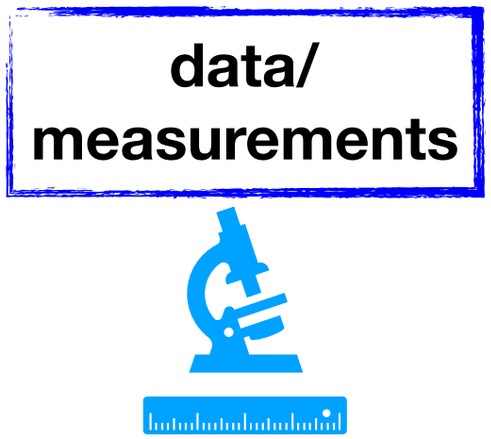
- testing changes to computer programs to make sure that the older code still works with the new changes
- you don't have to knows the expected result, the assumption is that the past results were correct

“Generalized regression” testing of scientific workflows:

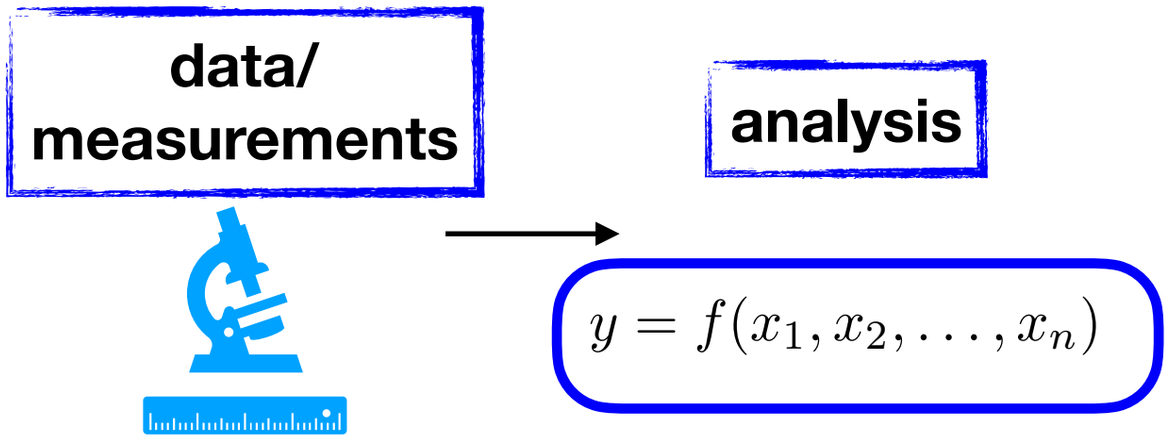
- testing changes to computer programs to make sure that the code still works (and gives a similar results) with changes in code, *computational* environment, or data
- the assumption is that the past results provide a “good” approximation of the *truth*

Scientific workflows

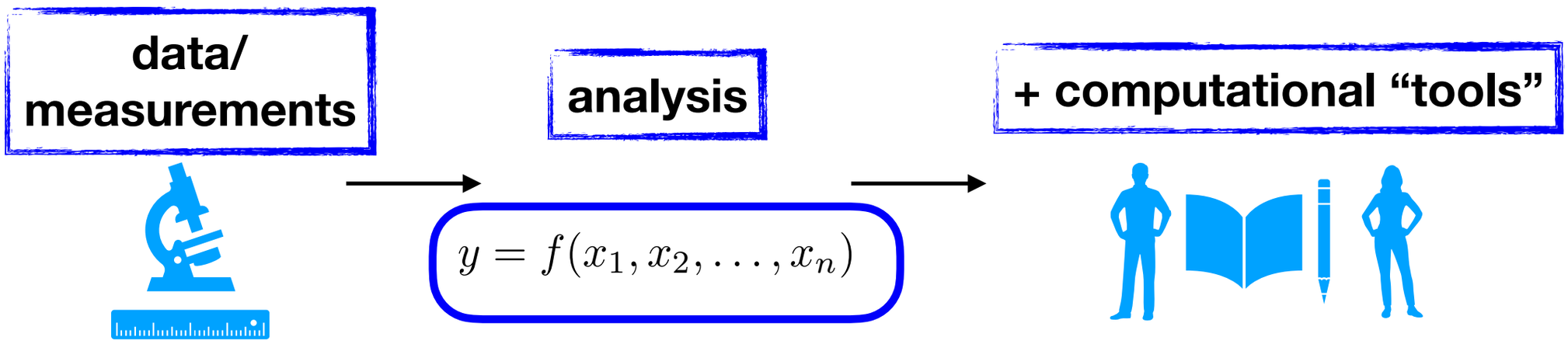
A scientific workflow



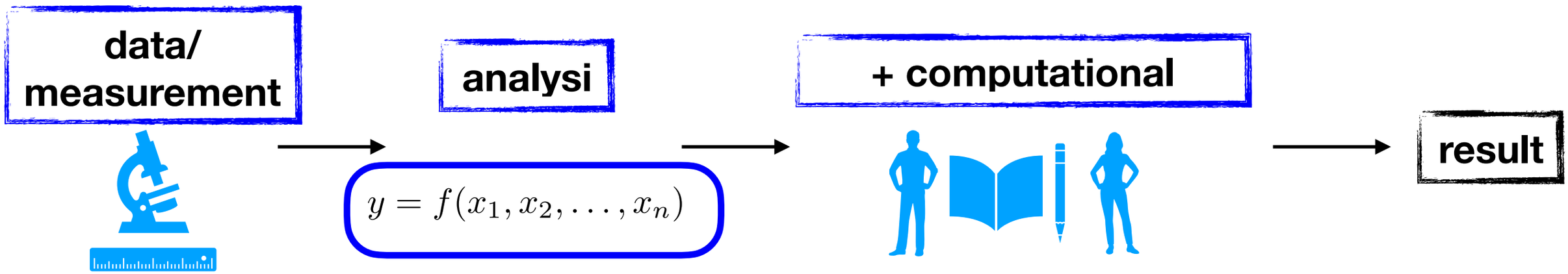
A scientific workflow



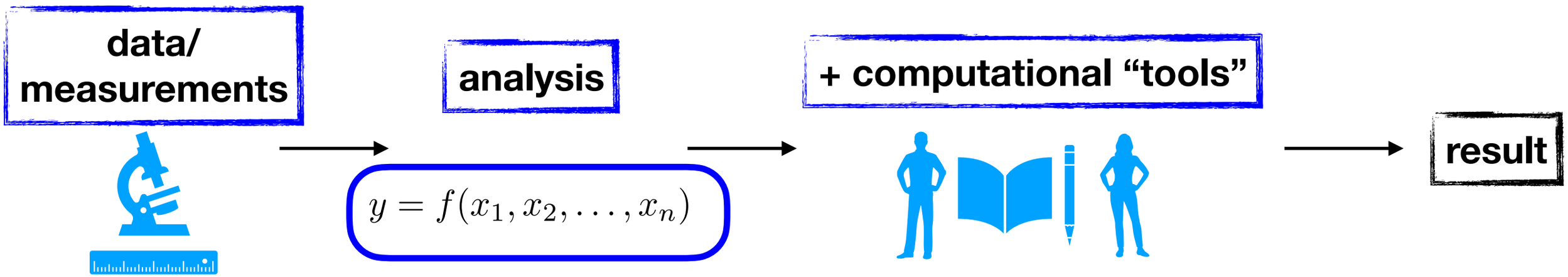
A scientific workflow



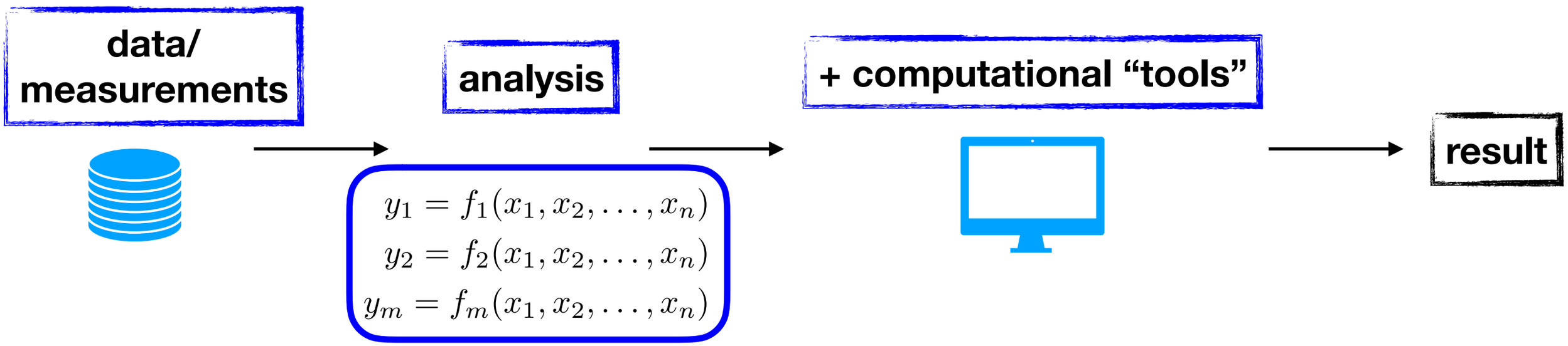
A scientific workflow



A scientific workflow



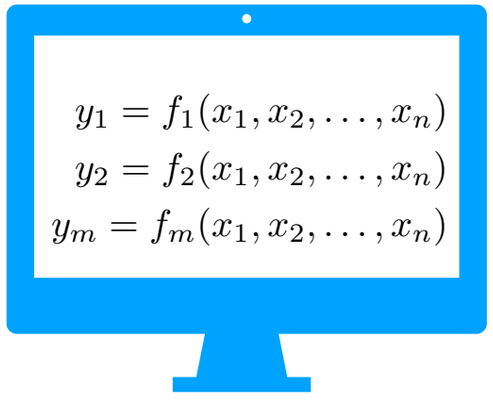
And if you a lot of data and/or complex analysis



A scientific workflow

datasets

analyses / software



result

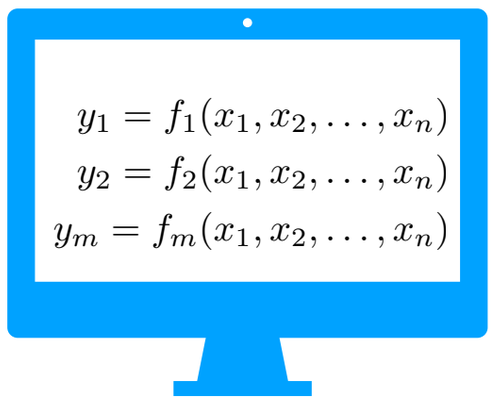
In an ideal world…

ideally:
- enough data
- good resolution
- well defined
- …

ideally:
- exact analytical solution
- enough computational resources
- …

datasets

analyses / software



The result

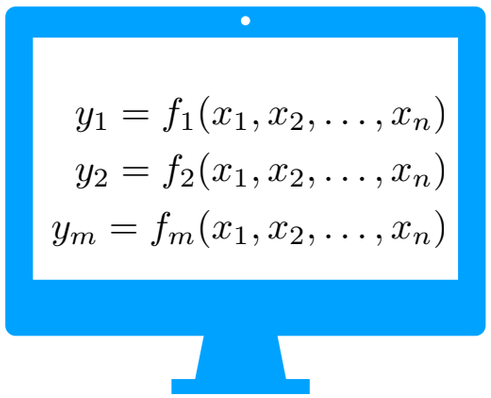
In the real world…

usually:
- not enough data
  (hopefully a representative group)
- could be collected in various places
  (hopefully well described)
- …

usually:
- not known exact solution
  (hopefully numerical methods fits the problem)
- too complex system to calculate everything
- …

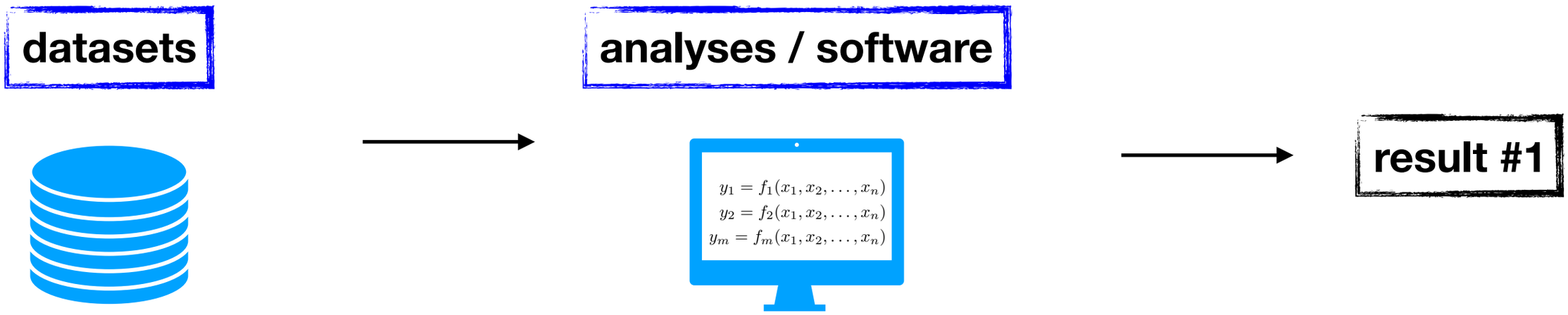
datasets

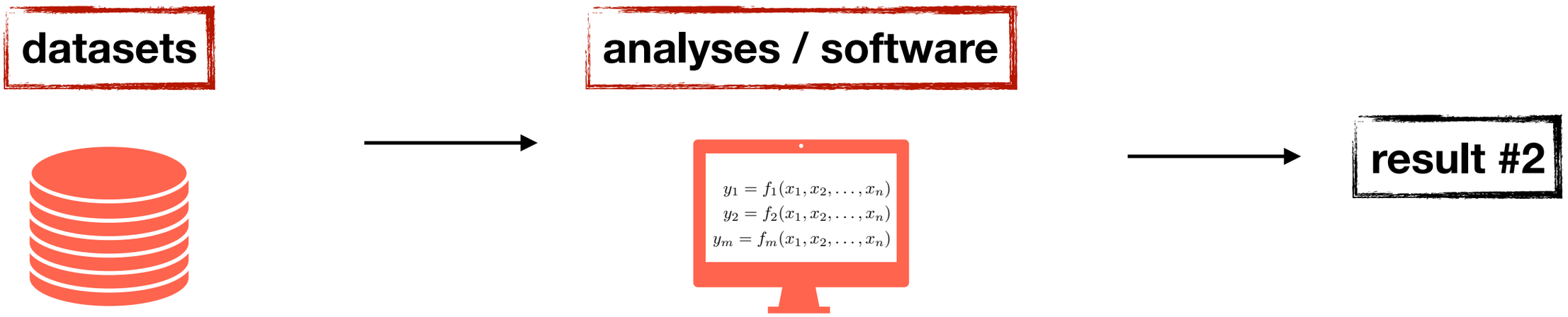
analyses / software



A result

In the real world…

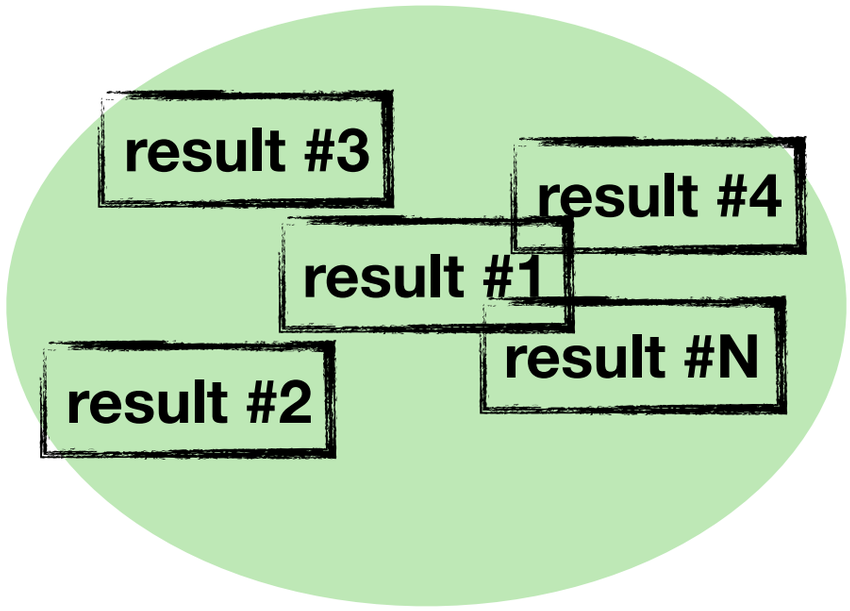




Testing and Reproducibility

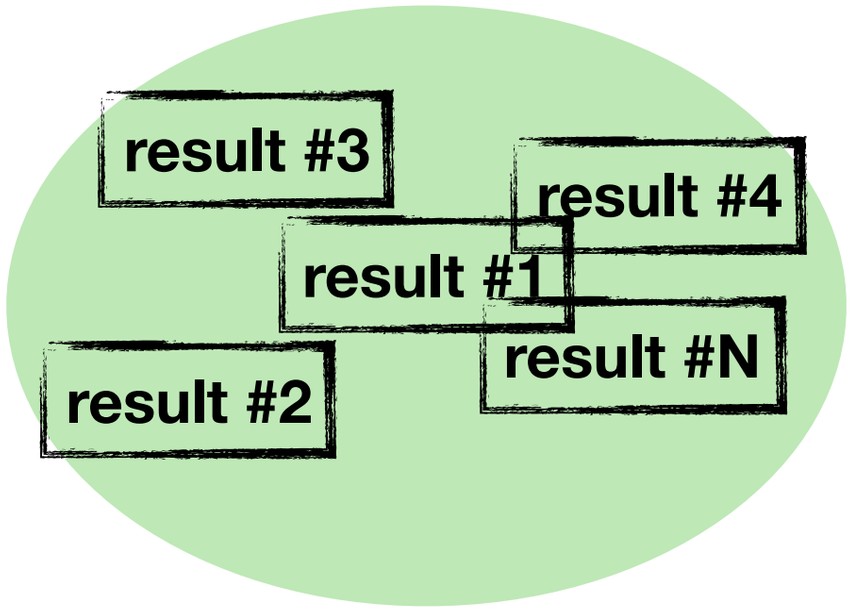
Are they similar..?

they will be likely not exactly the same, but might be very close



Are they similar..?

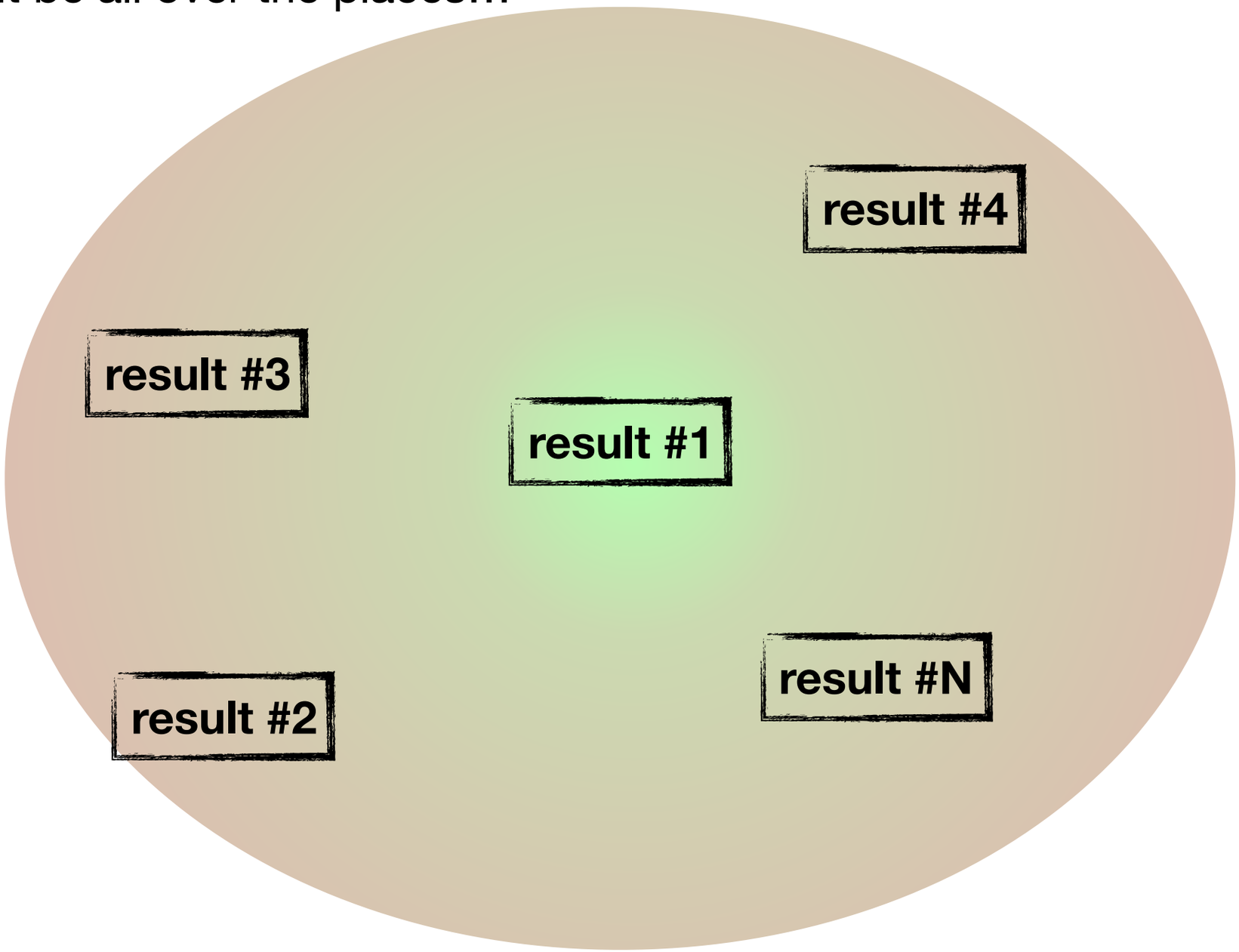
they will be likely not exactly the same, but might be very close



we can talk about an/the answer to the question

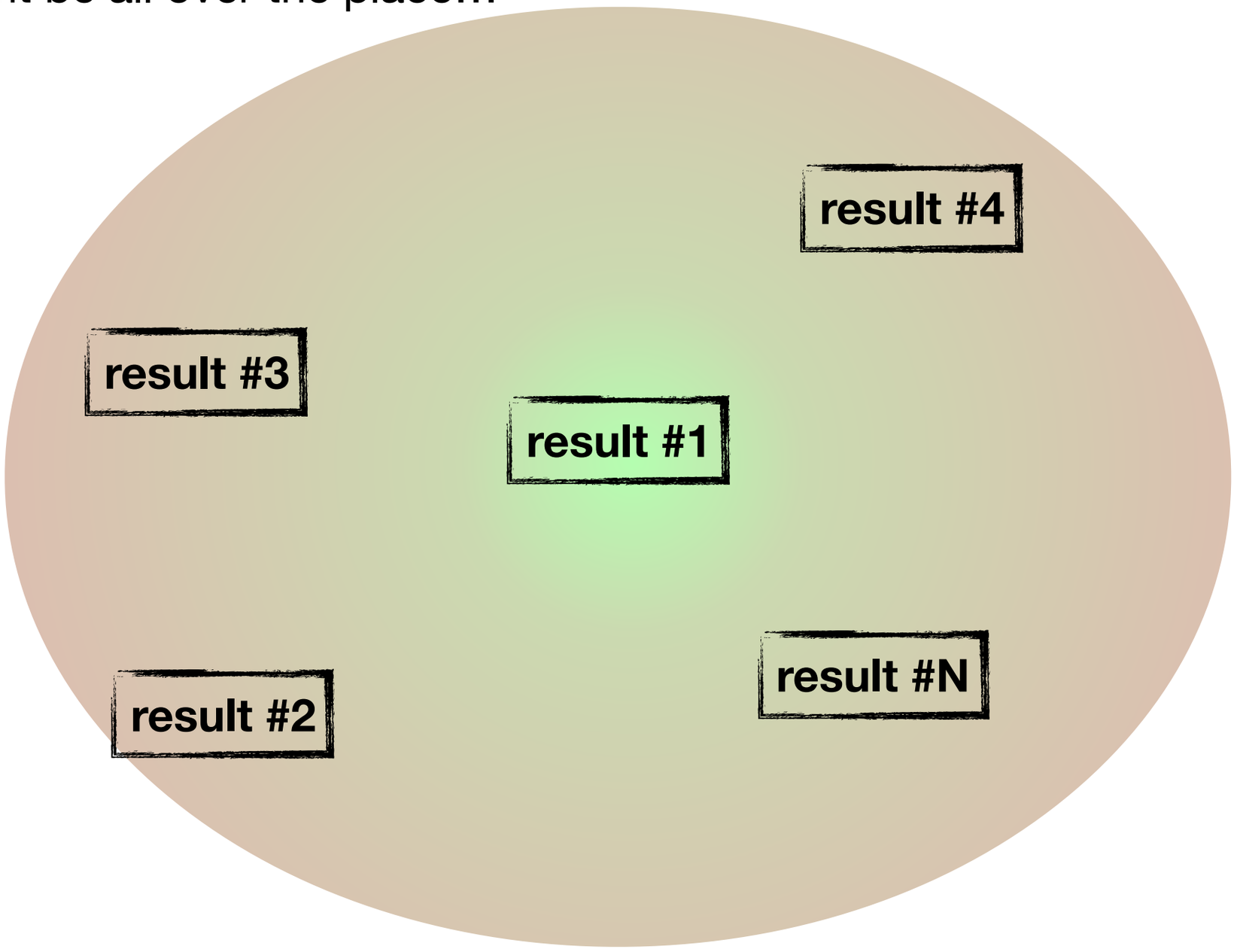
Are they similar..?

but they might be all over the places…



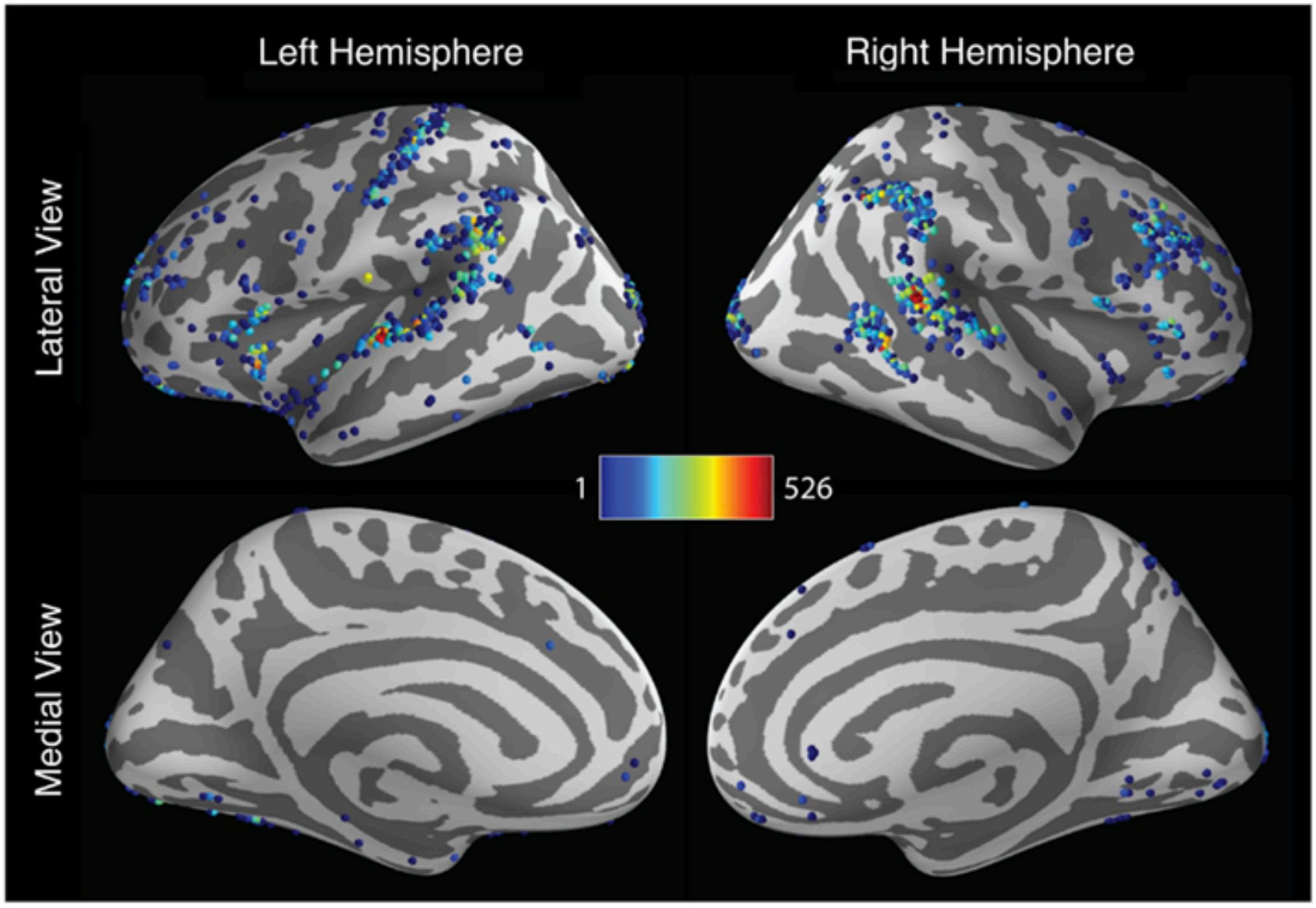
Are they similar..?

but they might be all over the place…

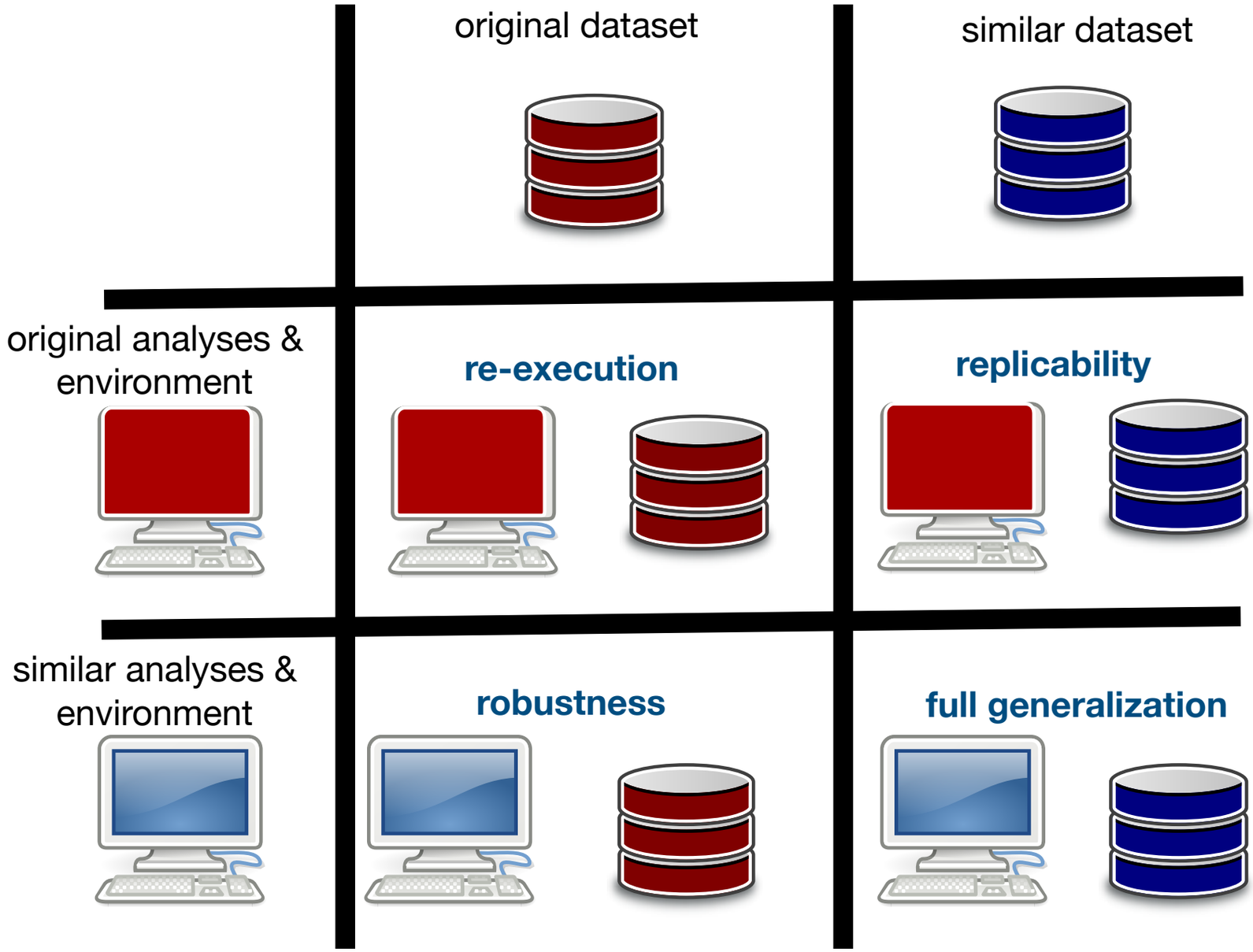


Are we ready to answer the question based on these results..??

Variation in functional activity when the same task data was analyzed by different workflows (*Carp, 2013*)



Reproducibility spectrum

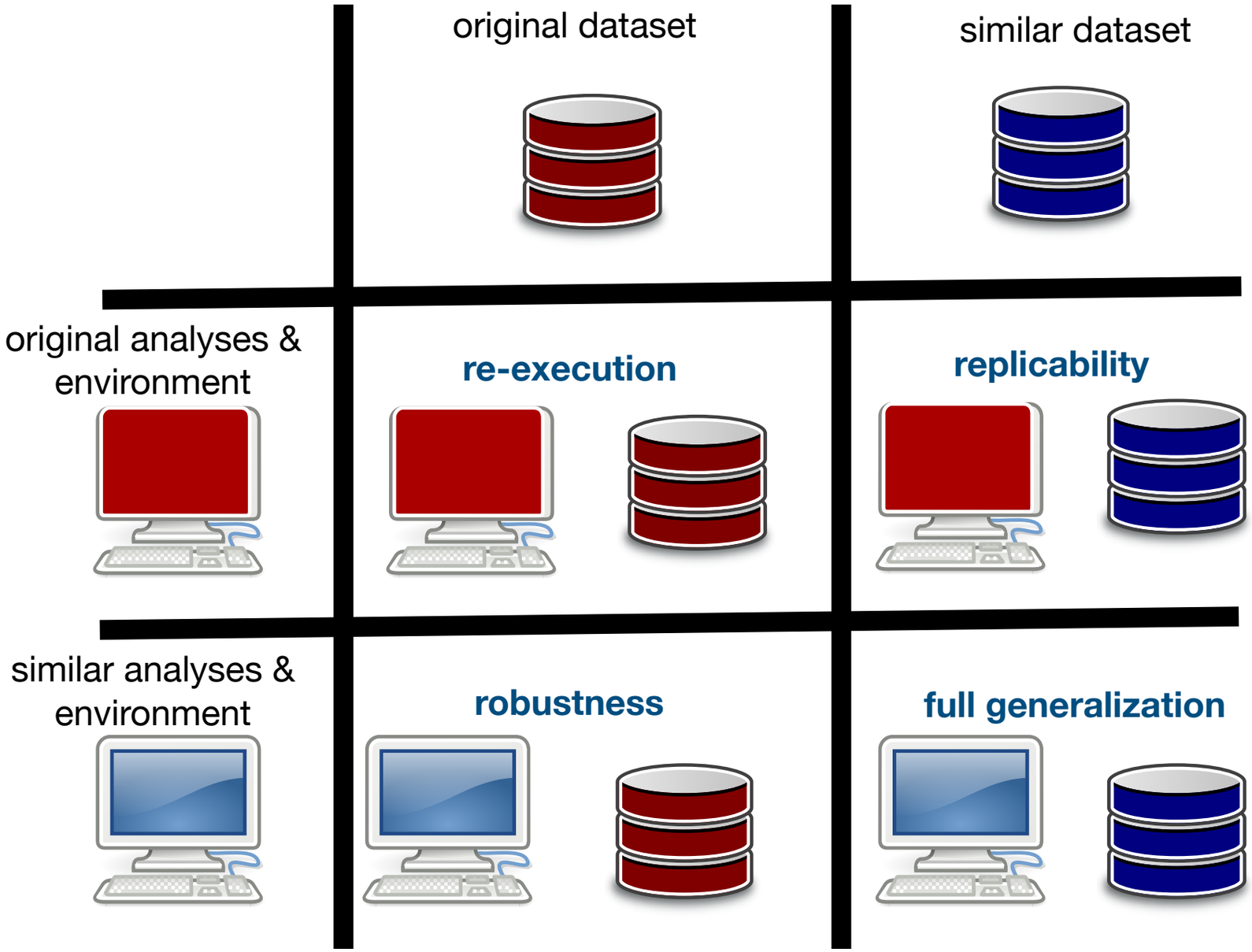


Adapted from multiple sources, including (Whitaker 2016; Drummond 2009; Hong 2015; Goodman, Fanelli, and Ioannidis 2016; Peng 2011).

Introduction to TestKraken

TestKraken

- testing workflow in various environments
- uses *Neurodocker* to create Docker images
- uses Common Workflow Language (CWL) to build a “workflow” with analyses and test
- creates browser-based dashboard with the results summary
- can be used with CircleCI for continuous integration



TestKraken concentrates on testing robustness

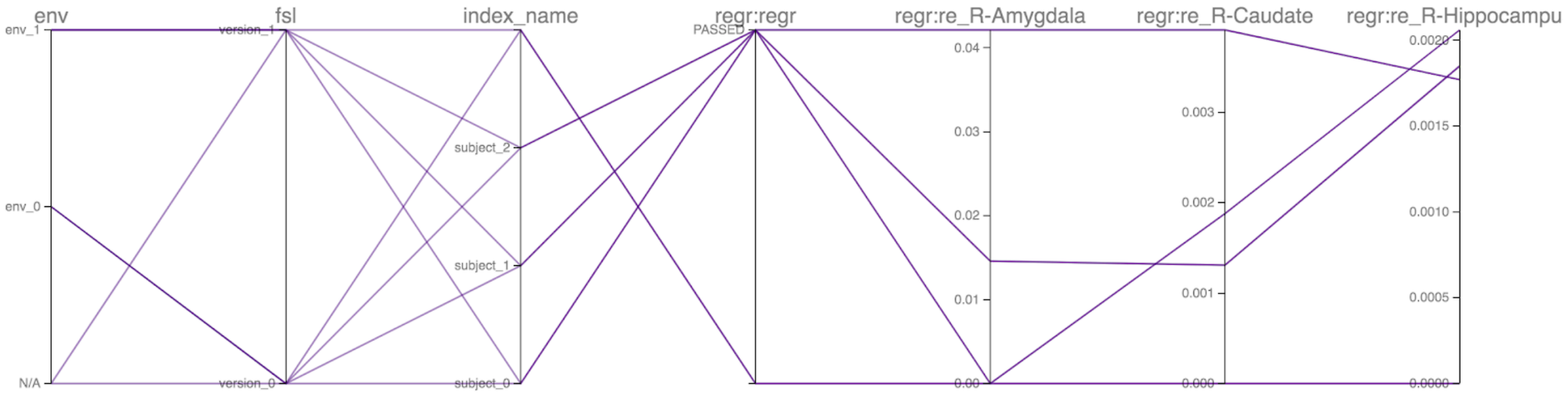
Simple Workflow

- A very simple, re-executable neuroimaging publication (code) https://f1000research.com/articles/6-124/
- Satrajit Ghosh, Jean-Baptiste Poline, David Keator, Yaroslav Halchenko, Adam Thomas, Daniel Kessler, David Kennedy
- used completely open, publicly available data
- ask authors about more!!!

Simple Workflow: results

		Reference Run (Docker)		Mac OSX (10.12.4)			Difference				
Hemisphere	Region	Mean Volume (mm3)	STD	Mean Volume (mm3)	STD	Correlation	Mean Volume (mm3)	STD	Range	Mean Absolute Volume	Percent of Reference
Left	Accumbens	471.2	178.8	468.4	181.0	0.980	2.8	35.7	[-72.0, 75.6]	25.8	5.5
	Amygdala	840.2	257.9	849.2	262.9	0.963	-9.1	70.8	[-303.0, 84.0]	32.1	3.8
	Caudate	3842.4	650.4	3840.5	630.7	0.980	2.0	130.9	[-320.6, 518.4]	56.1	1.5
	Hippocampus	3276.2	794.9	3264.5	783.6	0.997	11.7	62.0	[-147.0, 185.0]	44.2	1.3
	Pallidum	1683.5	316.0	1668.8	313.6	0.995	14.7	30.7	[-16.8, 99.0]	20.2	1.2
	Putamen	4881.1	965.4	4890.2	952.7	0.998	-9.2	59.5	[-145.0, 144.0]	45.4	0.9
	Thalamus	8088.6	1240.8	8108.8	1239.0	0.998	-20.2	71.4	[-135.0, 194.0]	54.5	0.7
Right	Accumbens	389.1	147.6	402.2	148.5	0.966	-13.1	38.5	[-145.5, 32.0]	24.1	6.2
	Amygdala	882.3	297.8	897.6	290.7	0.918	-15.3	119.3	[-464.0, 221.0]	68.2	7.7
	Caudate	3781.0	762.1	3780.3	767.7	0.999	0.7	27.1	[-60.1, 48.0]	20.6	0.5
	Hippocampus	3433.4	796.9	3453.3	793.2	0.997	-19.9	60.6	[-190.0, 78.0]	42.4	1.2
	Pallidum	1694.5	293.8	1695.2	289.5	0.997	-0.7	21.8	[-64.0, 60.9]	12.9	0.8
	Putamen	4965.3	1017.3	4942.8	1008.0	0.993	22.5	118.0	[-201.6, 397.0]	78.4	1.6
	Thalamus. Proper	7723.6	1120.6	7724.3	1129.4	0.999	-0.7	52.9	[-145.0, 118.0]	33.6	0.4
Total	CSF	189856.5	26936.5	190209.3	26668.1	0.999	-352.8	1025.2	[-4274.2, 20.0]	359.7	0.2
	Gray Matter	684781.8	86048.8	684111.7	86173.4	1.000	670.1	2009.7	[-11.3, 7113.4]	676.3	0.1
	White Matter	513866.1	52348.3	513802.8	52341.8	1.000	63.3	467.3	[-541.2, 2192.4]	125.3	0.0
	Brain	1388504.4	132850.8	1388123.8	132962.5	1.000	380.6	1511.6	[-534.0, 6956.6]	425.1	0.0

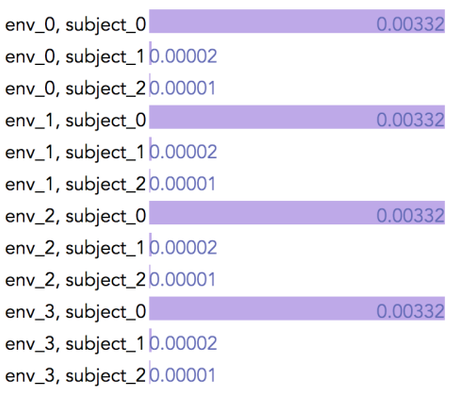
Simple Workflow: dashboard



env | fsl | index_name | regr:regr | regr:re_R-Amygdala | regr:re_R-Caudate | regr:re_R-Hippo

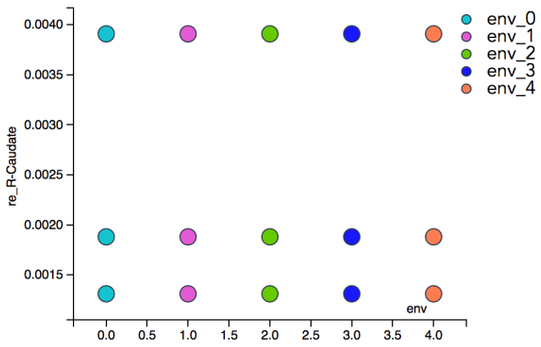
env	fsl	index_name	regr:regr	regr:re_R-Amygdala	regr:re_R-Caudate	regr:re_R-Hi
env_0	version_0	subject_0	PASSED	0	0.00188	0.00206
env_0	version_0	subject_1	PASSED	0.01459	0.00131	0.00185
env_0	version_0	subject_2	PASSED	0.04214	0.00391	0.00177
env_1	version_1	subject_0	PASSED	0	0.00188	0.00206

regr:regr:regr:regr:re_gray



Scatter plot of:

re_R-Caudate vs env



Description of environments

base

version	description
version_0	{'image': 'ubuntu:16.04', 'pkg-manager': 'apt'}
version_1	{'image': 'centos:7', 'pkg-manager': 'apt'}

fsl

version	description
version_0	{'version': '5.0.9'}
version_1	{'version': '5.0.10'}

miniconda

version	description
version_0	{'conda_install': 'python=3.5 nipype pandas requests', 'env_name': 'test', 'activate': 'true'}

TestKraken: preparing workflow for testing

```
workflows4regtests/basic_examples/simple_workflow
├── data_ref
│   └── output
│       ├── AnnArbor_sub16960
│       │   └── segstats.json
│       ├── AnnArbor_sub20317
│       │   └── segstats.json
│       └── AnnArbor_sub38614
│           └── segstats.json
├── parameters.yaml
└── workflow
    └── run_demo_workflow.py
```

TestKraken: preparing parameters.yaml

```yaml
# List all desired combinations of environment specifications. This
# configuration, for example, will produce four different Docker images:
#  1. ubuntu 16.04 + fsl 5.0.9 + miniconda
#  2. ubuntu 16.04 + fsl 5.0.10 + miniconda
#  3. centos 7 + fsl 5.0.9 + miniconda
#  4. centos 7 + fsl5.0.10 + miniconda
env:
  base:
    - image: 'ubuntu:16.04'
      pkg-manager: apt
    - image: 'centos:7'
      pkg-manager: yum
  fsl:
    - version: 5.0.9
    - version: 5.0.10
  miniconda:
    - conda_install:
        - python=3.5
        - nipype
        - pandas
        - requests

# One or more fixed environments to test. These environments are built as defined
# and are not combined in any way. This configuration, for example, will
# produce one Docker image.
fixed_env:
  base:
    image: 'centos:7'
    pkg-manager: yum
  fsl:
    version: 5.0.10
  miniconda:
    conda_install:
      - python=2.7
      - nipype
      - pandas
      - requests
      - bz2file
```

TestKraken: preparing parameters.yaml

```yaml
# The command to run the script.
command: python
# The workflow script. The script must be useable on the command-line.
script: run_demo_workflow.py

# Inputs to the workflow script. The first item is always the CWLType of the
# argument. See https://www.commonwl.org/v1.0/CommandLineTool.html#CWLType for
# all available types.
inputs:
  - [string, --key, 11an55u9t2TAf0EV2pHN0vOd8Ww2Gie-tHp9xGULh_dA]
  - [int, -n, '3']

# Tests to compare the output of the script to reference data.
tests:
  - file:
      - output/AnnArbor_sub16960/segstats.json
      - output/AnnArbor_sub20317/segstats.json
      - output/AnnArbor_sub38614/segstats.json
    # A descriptive name for the test.
    name: regr
    # These scripts are available under `testkraken/testing_functions`.
    script: check_output.py
```

TestKraken: a scientific workflow

```
def div_list(filename):
    """ a simple function for sorting list"""
    random.seed(a=3)
    with open(filename) as json_data:
        listorig = json.load(json_data)

    for i in range(len(listorig)):
        listorig[i] /= 2
        listorig[i] += random.random()/100

    with open('list_final_1.json', 'w') as outfile:
        json.dump(listorig, outfile)
    with open('list_final_2.json', 'w') as outfile:
        json.dump(listorig, outfile)

    with open('list_final.json', 'w') as outfile:
        json.dump(listorig, outfile)


if __name__ == '__main__':
    from argparse import ArgumentParser, RawTextHelpFormatter
    parser = ArgumentParser(description=__doc__,
                            formatter_class=RawTextHelpFormatter)
    parser.add_argument("-f", dest="filename",
                        help="file with a list to sort")
    args = parser.parse_args()

    div_list(args.filename)
```

TestKraken: a test

```
def test_el_list_eq(file_out, file_ref=None, name=None, **kwargs):
    with open(file_out_r) as f:
        try:
            obj_out = json.load(f)
        except:
            obj_out = f.read().strip()

    with open(file_ref_r) as f:
        try:
            obj_ref = json.load(f)
        except:
            obj_ref = f.read().strip()

    report_filename = "report_{}.json".format(name)
    out = {}
    out["rel_error"] = []
    out["abs_error"] = []
    out["index_name"] = []
    for i, el in enumerate(obj_ref):
        out["index_name"].append("el_{}".format(i))
        out["abs_error"].append(round(abs(obj_ref[i] - obj_out[i]), 3))
        out["rel_error"].append(round(1.* abs(obj_ref[i] - obj_out[i]) / obj_ref[i], 3))
    with open(report_filename, "w") as f:
        json.dump(out, f)

if __name__ == '__main__':
    from argparse import ArgumentParser, RawTextHelpFormatter
    parser = ArgumentParser(description=__doc__,
                            formatter_class=RawTextHelpFormatter)
    parser.add_argument("-out", nargs="+", dest="file_out",
                        help="file with the output for testing")
    parser.add_argument("-ref", nargs="+", dest="file_ref",
                        help="file with the reference output")
    parser.add_argument("-name", dest="name",
                        help="name of the test provided by a user")
    args = parser.parse_args()

    test_el_list_eq(**vars(args))
```