[Please select]
[Please select]

EUROPEAN CENTRAL BANK

EUROSYSTEM

Ib Hansen
ECB

A Toolbox (ModelFlow) managing and recycling models in Python

Place the subtitle here.

Disclaimer

- *The viewpoints and conclusions stated are the responsibility of the author, and do not necessarily reflect the views of ECB or Danmarks Nationalbank.*
- *Authors' calculations, for illustration purposes only: some model inputs are intentionally muted or randomized.*
- *Results presented should be taken as purely hypothetical model calculations prepared to illustrate the features of the framework and model.*

Agenda

- Why
- What is a model
- How to solve
- Models on Python
- Demo
- Wrap up

Run on virtual Machine
https://mybinder.org/v2/gh/IbHansen/modelflow/master

Take care with Internet Explorer

Why?

- From system wide to systemic focus
- Integration of submodels
- Feedback - including contemporaneous
  - Real economy
  - Banks
  - Other financial institutions
  - Markets
- Fast and agile development
- Simulate more years/quarters
- Many data sources
- Fast execution, Fast turnaround
  - Enables new possibilities
    - Goal seek
    - Attribution
    - Stochastic simulation

What is a Model?

Endogeneous   Exogeneous

$$y_t^1 = f^1(\cdot, y_t^2 \ldots, y_t^n \ldots y_{t-r}^1 \ldots, y_{t-r}^n, x_t^1 \ldots x_t^k, \ldots x_{t-s}^1 \ldots, x_{t-s}^k)$$
$$y_t^1 = f^1(y_t^1 \ldots, y_t^n \ldots y_{t-r}^1 \ldots, y_{t-r}^n, x_t^1 \ldots x_t^k, \ldots x_{t-s}^1 \ldots, x_{t-s}^k)$$
$$\vdots$$
$$y_t^n = f^n(y_t^1 \ldots, y_t^{n-1} \ldots y_{t-r}^1 \ldots, y_{t-r}^n, x_t^1 \ldots x_t^r, x. . .{}_{t-s}^1 \ldots, x_{t-s}^k)$$

If we have a model:

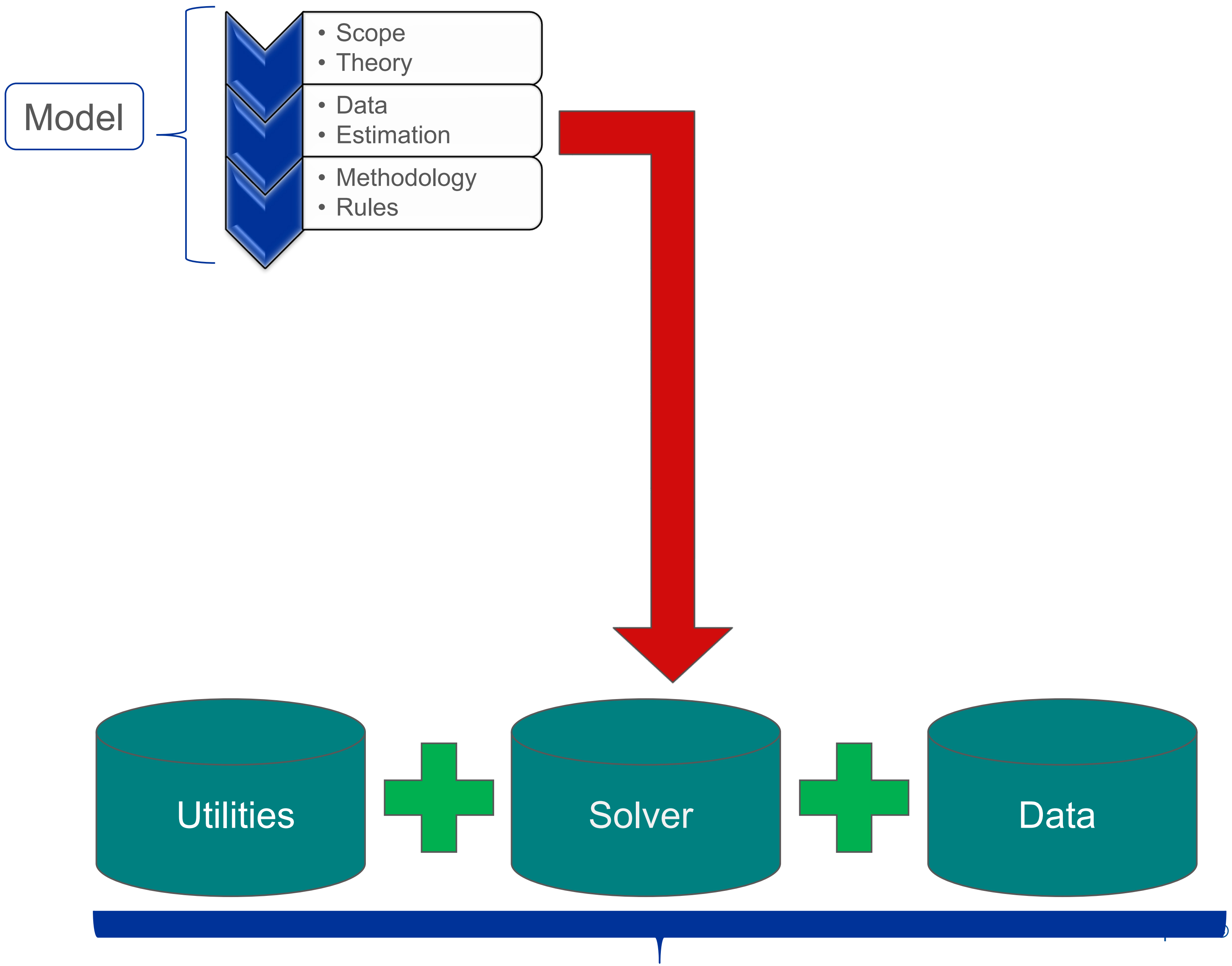
$$\mathbf{y}_t = \mathbf{F}(\mathbf{y}_t \cdots \mathbf{y}_{t-r}, \mathbf{x}_t \cdots \mathbf{x}_{t-s})$$

Then we have a solution, if we can find y_t^* so that:

$$\mathbf{y}_t^* = \mathbf{F}(\mathbf{y}_t^* \cdots \mathbf{y}_{t-r}, \mathbf{x}_t \cdots \mathbf{x}_{t-s})$$

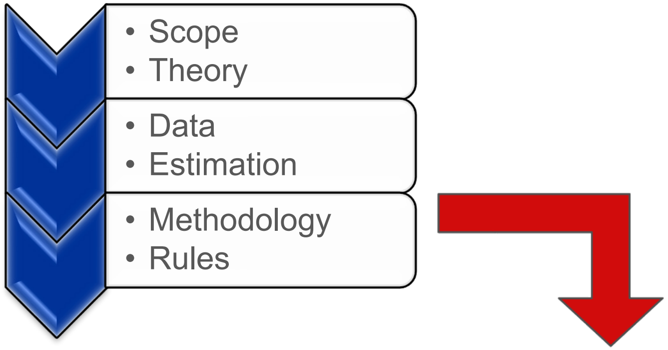
Particular easy if no contemporaneous feedback
Else iterative methods are necessary.
Gauss or Newton type

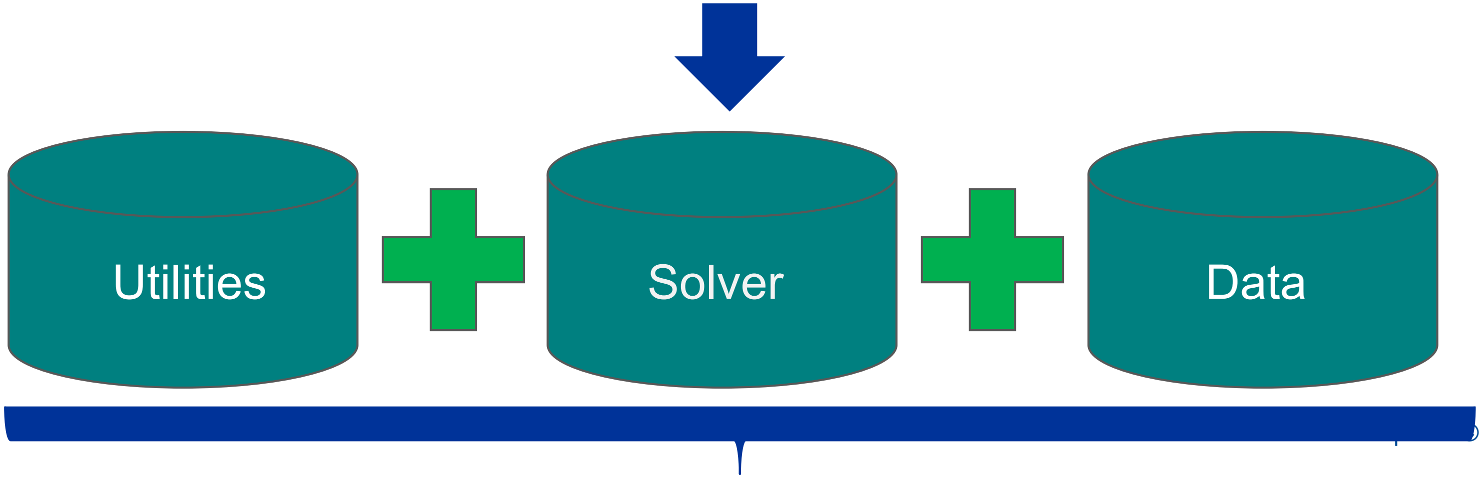
From model to engine/solver



From model to engine/solver

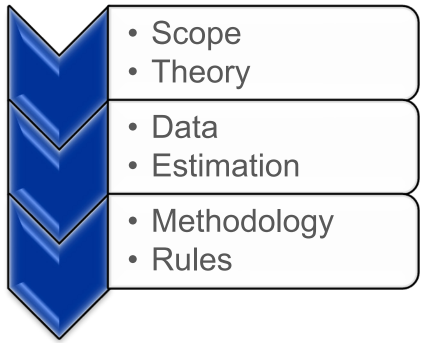
Code the model

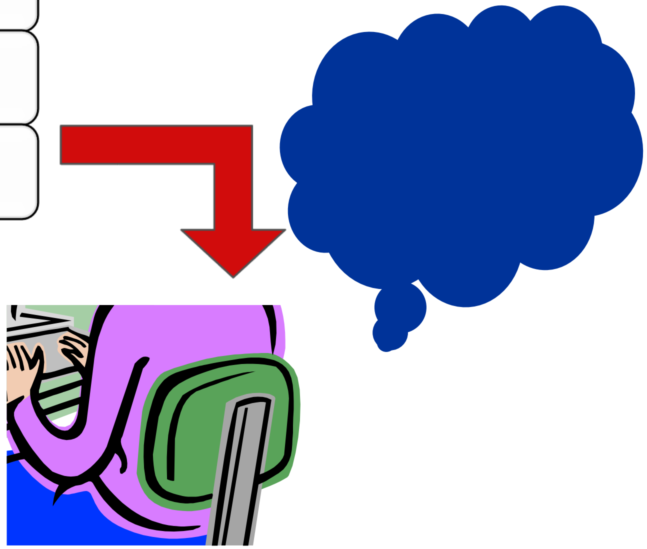




From model to engine/solver

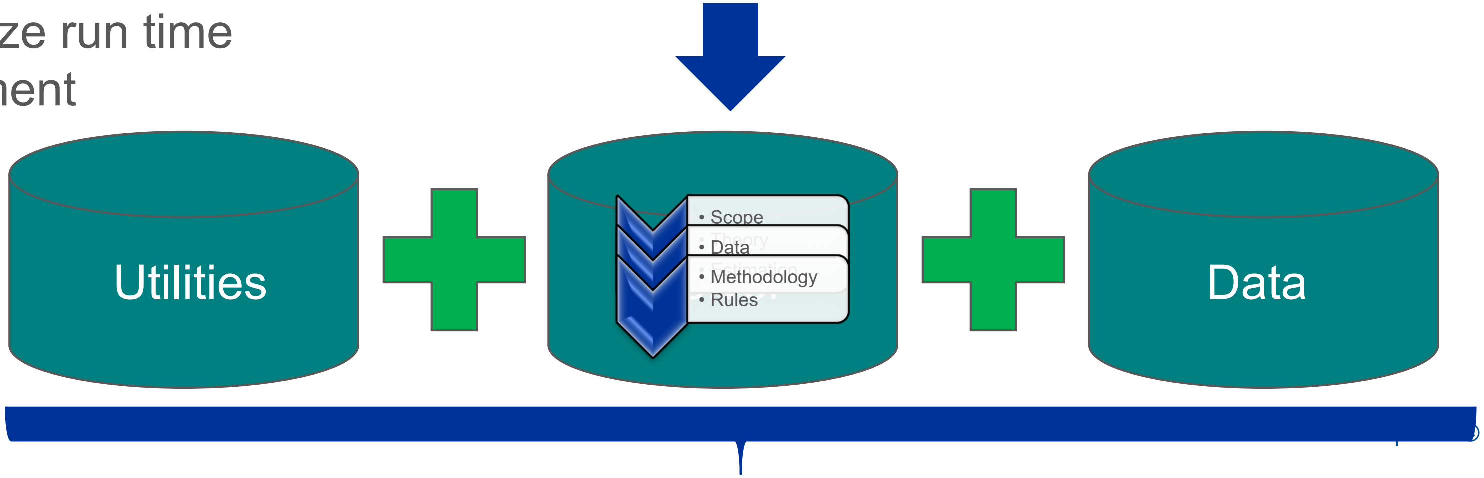
Code the model





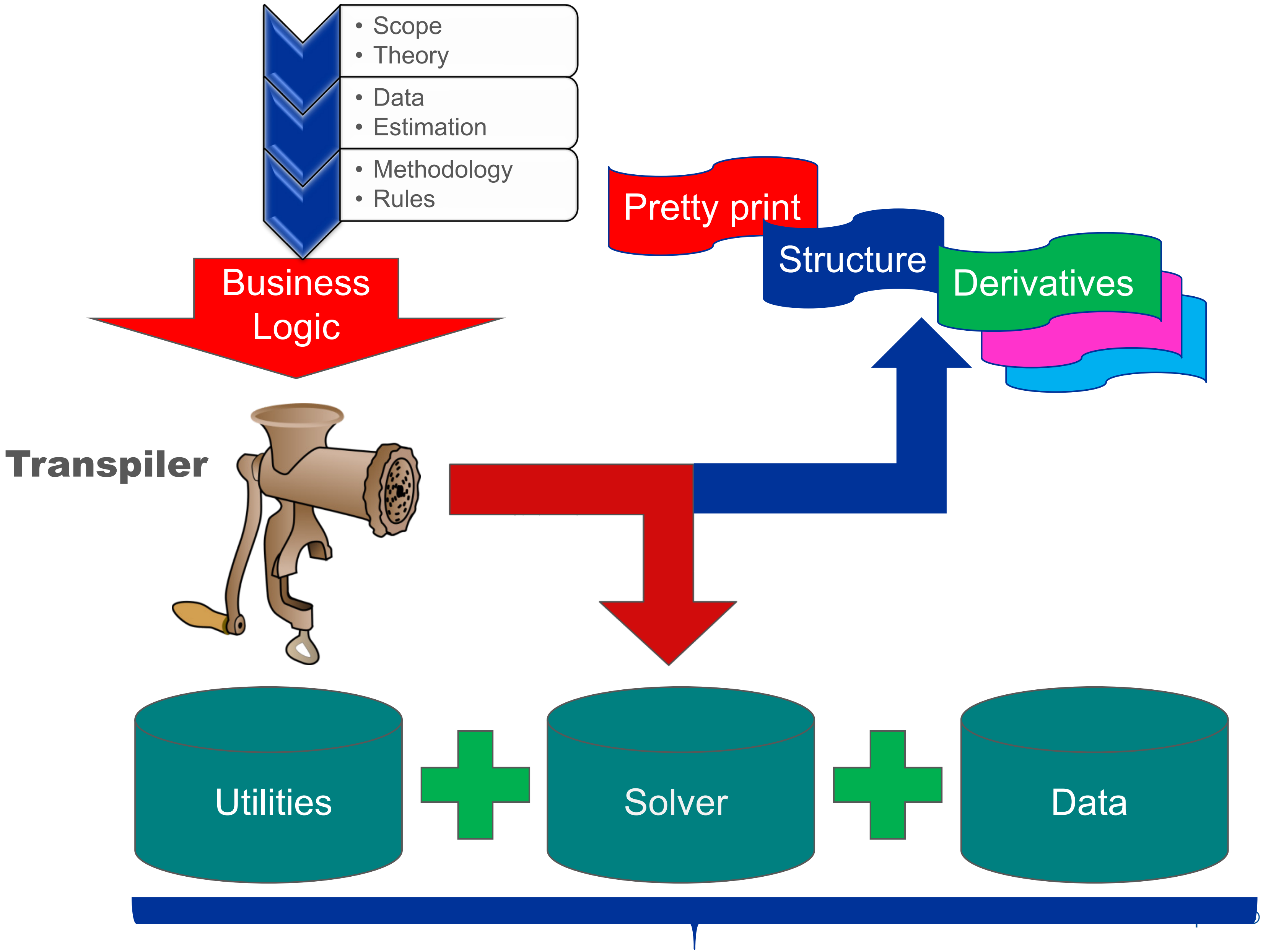
Issues:

- The solver and the model drift apart
- Difficult to:
    - Combine models
    - Debug
    - Change
    - Quality check
    - Maintain
    - Optimize run time
    - Document



From model to engine/solver

Transpile machine readable business logic



Separation of concerns

Edsger W. Dijkstra in his 1974 paper "On the role of scientific thought“

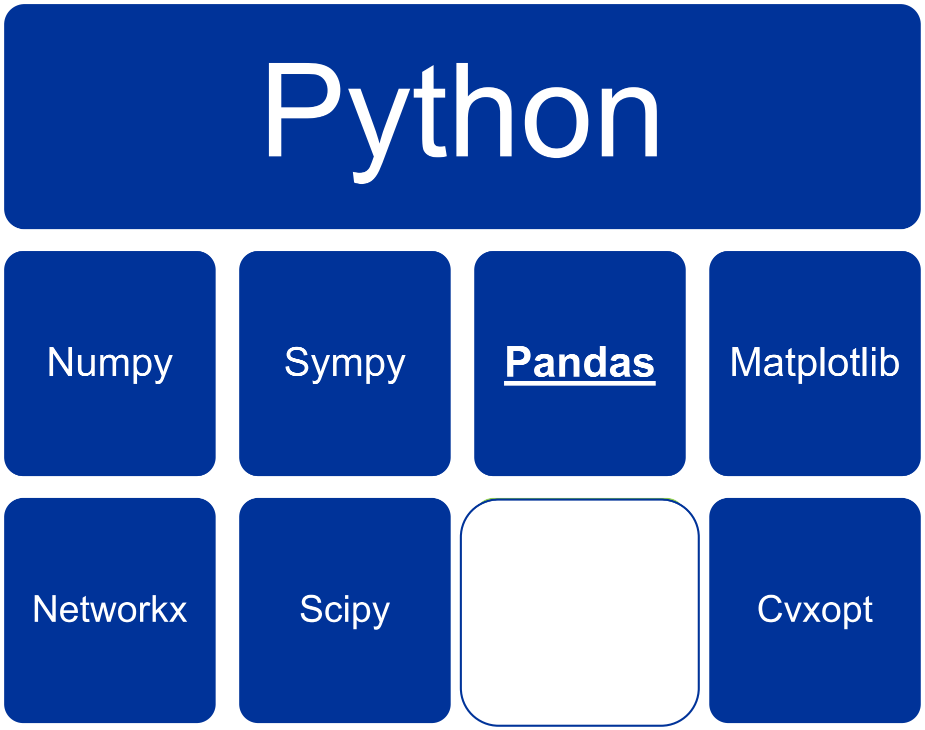
Why Python?

- Easy to learn
- Suitable for rapid development
  - Supported by many libraries
  - Multiple programming styles are supported (procedural, object-oriented, functional, etc.)
  - Interpreted rather than compiled. (but)
- Elegant syntax
  - Easy to read and easy to remember
  - Highly expressive,
  - Allowing you to get more done with less code
- Interface to:
  - Data-management systems to pull and push data
  - Model systems [Matlab | C | R | SAS | Fortran …] to link with (legacy) models.
  - Presentation systems [Excel | Word | Powerpoint …]

Source: http://quant-econ.net/_static/pdfs/py-quant-econ.pdf

The Python ecosystem …

has (nearly) everything: Data wrangling, plotting, transformation, optimization …..



However:

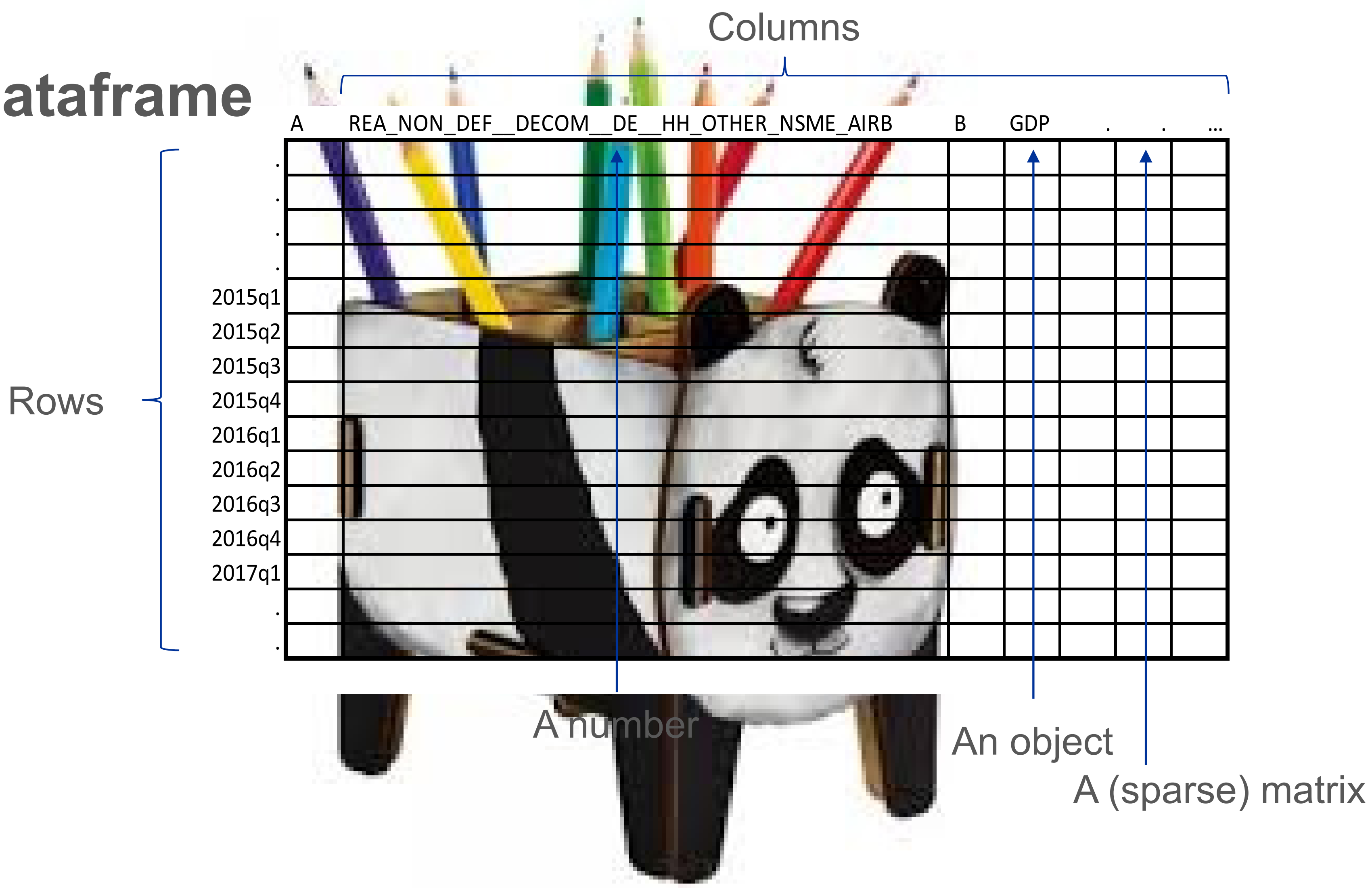
Not models with:

- **Lagged variables**
- **Simultaneous equation**

No (template) Business logic language

Solution:
A model toolbox

Dataframe


Columns
A
REA_NON_DEF__DECOM__DE__HH_OTHER_NSME_AIRB
B
GDP
. . ...
Rows
2015q1
2015q2
2015q3
2015q4
2016q1
2016q2
2016q3
2016q4
2017q1
A number
An object
A (sparse) matrix

ModelFlow

- Using Python libraries:
- Agile model definition language
  - Macro and stress test models
  - Large model
  - Network models
  - Embed optimization
- Solve the model
- Compare, visualize and analyse results
- Invert the model (targets and instruments)
- Attribution analysis
- Evaluate stability of the linearized model
- Visualize model structure in adjacency matrix or graphs
- Walk the dependency graph with data or attribution

How do we get there?

fmodel

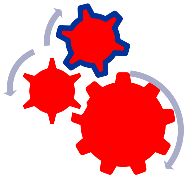
Input_scenario_1

Input_scenario_2

```
model   = create_model(fmodel)
result1 = model(input_scenario_1)
result2 = model(input_scenario_1)
```

model

A class



result_1

result_2

Model structure

Comparision

Model Inversion

Attribution

Plot

Differentiate formulars

Heatmaps

Swarm plot

On boarding a model in Python

Template Business logic

Latex script

Matlab function

Model Aremos

Formulars in Excel sheet

Python function

Coefficient from Excel

Coefficient from Matlab

Business Logic

Language-independent Model specification

Solver ++ (Model class)

Targets and instruments (Tinbergen)

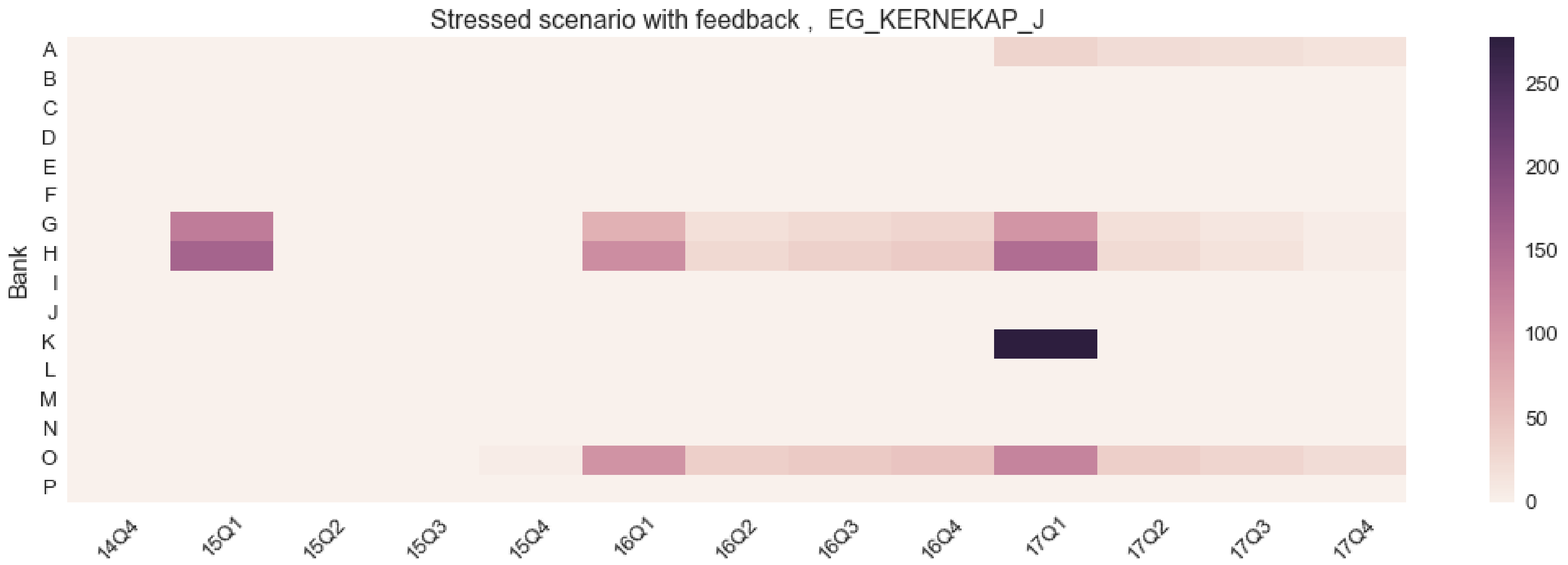
- *ModeL:*
  - $\mathbf{y}_t = \mathbf{F}(\mathbf{x_t})$
- Think of a condensed model (**G**) with a few endogenous variables: **targets** and a few exogenous variables: **instrument** variables. All the rest of the exogenous variables are fixed:
  - $\bar{y}_t = G(\bar{\mathbf{x}}_t\ )$
- Invert G and we get LedoM (ModeL reversed). A model where instruments are functions of targets:
  - $\bar{\mathbf{x}}_t^* = \mathbf{G}^{-1}(\bar{\mathbf{y}}_t^*)$
- LedoM can be solved numerically (next slide):
  - Slim to fit experiments (required leverage)
  - Fatten to fit experiments (required capital injection)
  - Reverse stress test (how much stress should be applied for specific bank(s) to go below a threshold CET1 ratio)
  - and more

Solving LedoM

– $\bar{\mathbf{x}}_t^* = \mathbf{G}^{-1}(\bar{\mathbf{y}}_t^*)$ can be found using *Newton–Raphson* method:

- $$\bar{\mathbf{x}}_t^i = \bar{\mathbf{x}}_t^{i-1} + \mathbf{J}^{-1}\left(\bar{\mathbf{y}}_t^* - \bar{\mathbf{y}}_t^{i-1}\right)$$
  $$\bar{\mathbf{y}}_t^i = \mathbf{G}\left(\bar{\mathbf{x}}_t^i\right)$$
  $$Until\ abs\left(\bar{\mathbf{y}}_t^* - \bar{\mathbf{y}}_t^{i-1}\right) < \varepsilon$$
- $J_{\mathrm{t}} = \frac{\partial \bar{G}_t}{\partial \bar{x}_t}$ is the Jacobian matrix of **G**
- $J_{\mathrm{t}} \approx \frac{\Delta \bar{G}_t}{\Delta \bar{x}_t}$ A lot of simulations – but it works
- J_{t} should be invertible. ➔ same number of instruments and targets (and more)
- One instrument can be a scalable package of several exogenous variables

Need for capital in order to maintain lending capacity in a hypothetical stress scenario
(Based on MonaS (a Danish Model))



DEMO

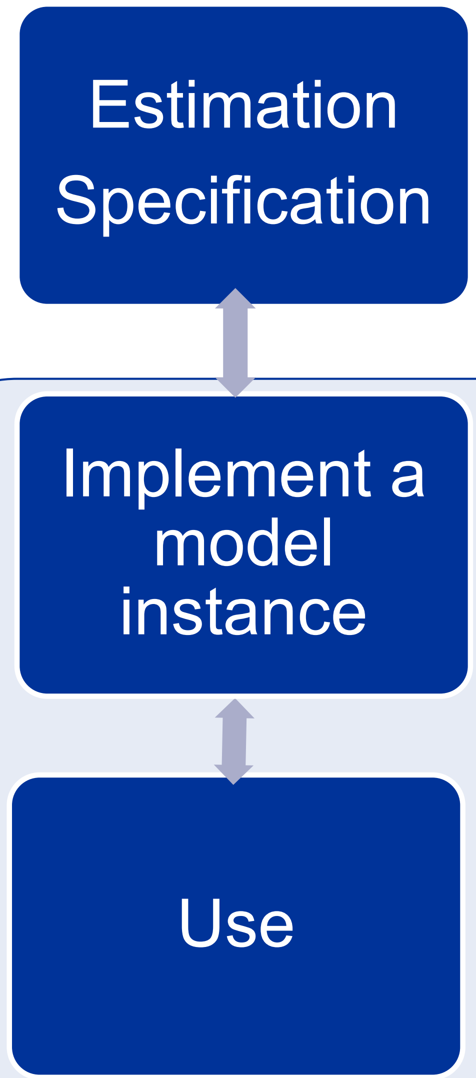
- Macro prudential perspective (aka the BEAST)
  - Specified in Dynare
  - 88k equations
- Contagio model
  - Specified in Business logic with Matrices.
  - 2600 x 2600 large exposures
  - Run after MPP
  - But integration with MPP is in the works

So we got.

- Agile model definition language
  - Macro and stress test models
  - Large model
  - Network models
- Solve the models - fast
- Compare, visualize and analyse results
- Invert the model (targets and instruments)
- Attribution analysis
- Evaluate stability of the linearized model
- Visualize model structure in adjacency matrix or graphs
- Walk the dependency graph with data or attribution

Thank You

Different needs. Different tools



Model estimation/calibration
Using the tools appropriate for the different subcomponents

A model management framework

- Model implementation
  - Onboard models
  - Data management
  - Solving
- Model analysing
- Result analysis
  - Comparing
  - Attribution analysis
  - Visualization
- Release management

Model source transformation

From Latex to Business logic

```
Residual net equity (own funds) is obtained by subtracting a debt stock $B$ and
$L$, bonds $BO$ and reserves $R$. $B$ contains deposits $D$, wholesale funding $

\begin{equation}
\label{eq:E}
E=A-B-LLR
\end{equation}

\begin{equation}
\label{eq:Ectd}
E=\underbrace{L+BO+R}_{\equiv A}-\underbrace{\left(D+W+F\right)}_{\equiv B}-LLR
\end{equation}
```

Usual output

Residual net equity (own funds) is obtained by subtracting a debt stock B and the aggregate loan loss reserve stock LLR from unweighted total assets A. Assets comprise loans L, bonds BO and reserves R. B contains deposits D, wholesale funding W and central bank funding F.

$$E = A - B - LLR \tag{1}$$

$$E = \underbrace{L + BO + R}_{\equiv A} - \underbrace{(D + W + F)}_{\equiv B} - LLR \tag{2}$$

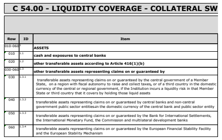
Template Business Logic

```
Do bank $
 Frml E E__{bank} = A__{bank}-B__{bank}-LLR__{bank} $
enddo $
Do bank $
 Frml Ectd E__{bank} = (L__{bank}+BO__{bank}+R__{bank})-((D__{bank}+W__{bank}+F__{bank}))-LLR__{bank}
$
enddo $
```

```
FRML E E__BANK_A = A__BANK_A-B__BANK_A-LLR__BANK_A $
FRML E E__BANK_B = A__BANK_B-B__BANK_B-LLR__BANK_B $
FRML E E__BANK_C = A__BANK_C-B__BANK_C-LLR__BANK_C $
FRML ECTD E__BANK_A = (L__BANK_A+BO__BANK_A+R__BANK_A)-((D__BANK_A+W__BANK_A+F__BANK_A))-LLR__BANK_A $
FRML ECTD E__BANK_B = (L__BANK_B+BO__BANK_B+R__BANK_B)-((D__BANK_B+W__BANK_B+F__BANK_B))-LLR__BANK_B $
FRML ECTD E__BANK_C = (L__BANK_C+BO__BANK_C+R__BANK_C)-((D__BANK_C+W__BANK_C+F__BANK_C))-LLR__BANK_C $
```

Business Logic

From EXCEL Workbook to Business logic



Excel Sheet
Calculation of
Liquidity coverage ratio (LCR)

model.fru

A model:
Calculation of
Liquidity coverage ratio (LCR)

Benefit:
All banks for all time can be calculated in one go
Attribution analysis to find out the sources of changes

Implemented in ModelFlow

- EBA stress test 2016 prototype (600.000 + equations)
  - Macro feedback and many, but not all sub models .
- MPP (88.000 equations gapped from Dynare)
- Contagion model from Large exposures (2600 x 2600 exposure matrixes)
- FRB/US (Macro model of US – in the public domain)
- Liquidity coverage ratio model for Danish Banks
- Danish Stress test model
- Danish macro models: ADAM, Mona, Smec
- MonaS, Mona macro model + Danish Stress test model

Speed.

- Standard Python (interpreted):
  - 4.7 Million floating point operations per seconds (MFLOPS)
- Simple Python programs can be compiles (translated to machine instructions)
- Compiled Python:
  - 767.7 MFLOPS
- A number of libraries which allows for parallelization across cores and/or clusters
- Useful:
  - To invert the model to make
  - Stochastic simulation
- Trade-off: Development effort and compilation time vs. speed