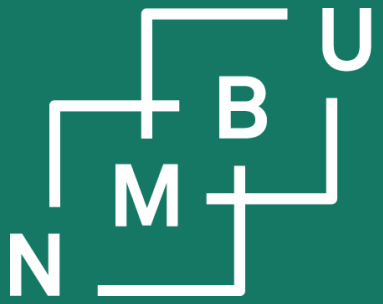
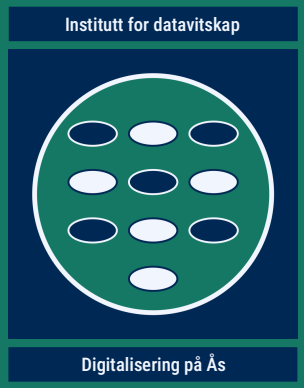


Noregs miljø- og biovitskaplege universitet



INF203

June advanced programming project

3 Uncertainty, validation, and testing

Different kinds of tests

Unit tests

Test one piece of code, *e.g.*, one method, for right arguments → return value.

Integration tests

Test concrete interactions between parts of the code, do they fit together?

Acceptance tests

Holistic validation: Run the complete code/system, do y/n correctness checks.

Regression tests

Added once a bug is detected and fixed. Check that the bug does not return.

What did we overlook?

Unit tests

Test one piece of cod → return value.

Integ

Test co y fit together?

Acceptan

Holistic vali omplete em, do y/n correctness checks.

Regression tests

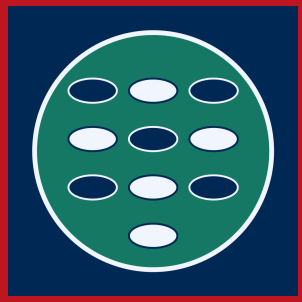
Added once a bug is detected and fixed. Check that the bug does not return.

```
import pytest

@pytest.mark.parametrize(
    "input1, input2, input3",
    [(1, 2, 3), (1, 3, 4), (3.1, 4.0, 7.1)]
)
def testAddition(input1, input2, input3):
    assert input1 + input2 == pytest.approx(input3)
```

Why would we never actually do this?

Noregs miljø- og biovitskaplege universitet



3 Validation

3.3 Formal program analysis

3.4 Holistic validation

3.5 Auto- and decorrelation

3.6 Time series block averaging

INF203

13th June 2025

Preconditions and postconditions

For purposes of formal analysis, the program flow is analysed step by step, e.g., at the instruction (statement) level, at the level of blocks of code that form a coherent unit, or at the level of functions or methods.

Precondition: State of the program at a point directly before the considered unit. This may include assumptions taken from the design contract or specification.

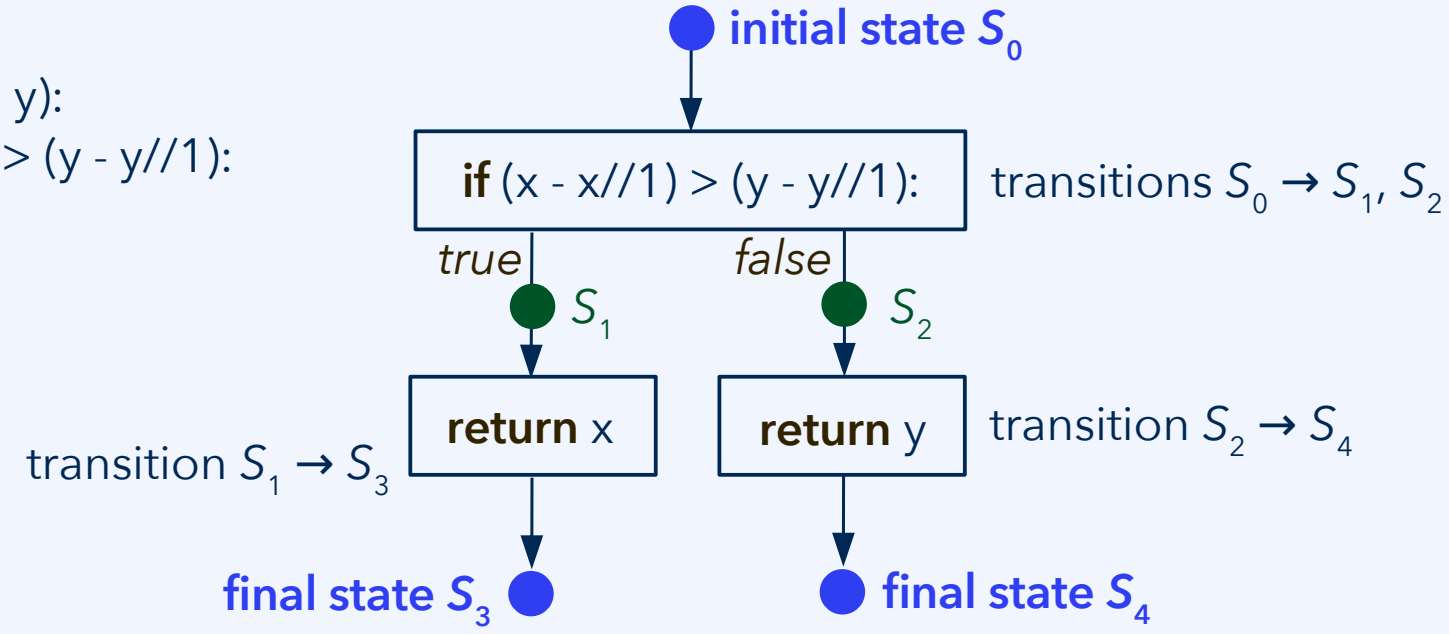
Postcondition: State of the program at a point directly after the considered unit, assuming that the precondition was fulfilled at the point directly before it.

Example

As part of a development project, we need a function grtfrac(x, y) that takes **two floating-point arguments** and returns the one with the greater fractional part; *e.g.*, grtfrac(2.7, 3.6) is to return 2.7, because ".7" is greater than ".6". **In design by contract, the caller, not the called method needs to guarantee the precondition.**

Preconditions and postconditions

```
def grtfrac(x, y):
    if (x - x//1) > (y - y//1):
        return x
    else:
        return y
```



S_0: x and y are floating-point numbers (by specification).

S_1: x, y as above; the fractional part of x is greater than that of y.

S_2: x, y as above; the fractional part of y is greater than that of x, or equal.

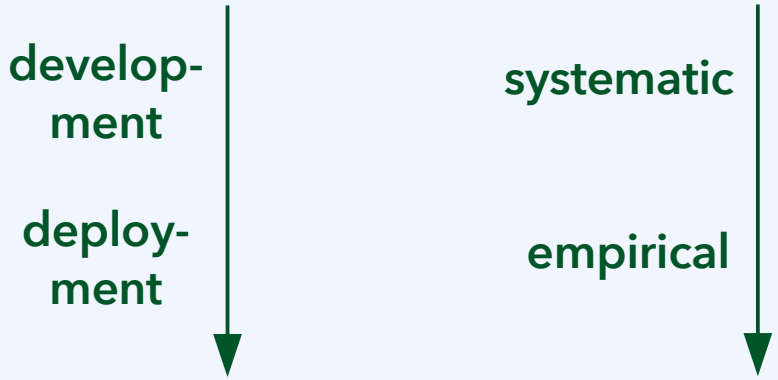
S_3: The fractional part of x is the greater one, and x was returned.

S_4: The fractional part of y is greater (or they are equal); y was returned.

Verification, validation, and testing

Verification: Proof that the developed product complies with its specification.

- Where possible, provide a **rigorous logical/mathematical proof**; alternatively, provide documents following agreed standards/procedures.



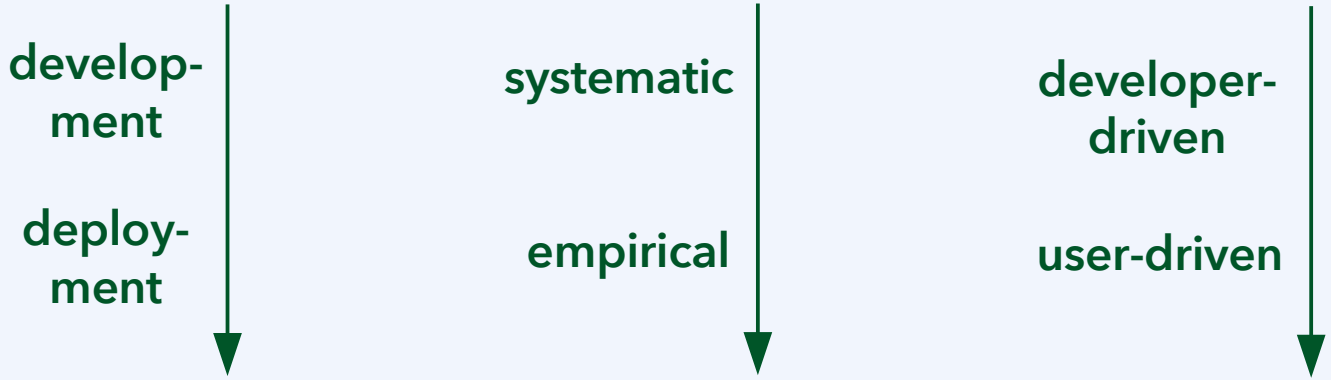
Testing: Use-case driven evaluation of the final (or alpha, or beta) product.

- The considered **use cases** should be **representative**.
- They should be as unrelated as possible to any concrete scenarios considered during development, including the validation process.

Verification, validation, and testing

Verification: Proof that the developed product complies with its specification.

- Where possible, provide a **rigorous logical/mathematical proof**; alternatively, provide documents following agreed standards/procedures.



Testing: Use-case driven evaluation of the final (or alpha, or beta) product.

- The considered **use cases** should be **representative**.
- They should be as unrelated as possible to any concrete scenarios considered during development, including the validation process.
- Ideally, **conducted by prospective users**; if unavailable, "play the user."

Verification, validation, and testing

Verification: Proof that the developed product complies with its specification.

- Where possible, provide a **rigorous logical/mathematical proof**; alternatively, provide documents following agreed standards/procedures.

Validation: Empirical evaluation to what extent user the requirements are met.

- All requirements need to be covered and demonstrated at least once.
- Ideally, **requirements** are not identical with the specification. They should be user-oriented; *e.g.*, epics and user stories in a requirements analysis from agile software engineering. **Feedback from users** is needed.

Testing: Use-case driven evaluation of the final (or alpha, or beta) product.

- The considered use cases should be **representative**.
- They should be **as unrelated as possible to** any concrete scenarios considered during development, including **the validation process**.
- Ideally, **conducted by prospective users**; if unavailable, "play the user."

Verification, validation, and testing

Verification: Proof that the developed product complies with its specification.

- Where possible, provide a **rigorous logical/mathematical proof**; alternatively, provide documents following agreed standards/procedures.

Vali ... **met.**

ce.
hould
ysis

Remark

Verification always has the meaning that something is demonstrated to be true, particularly by logical reasoning. **Validation** and **testing** have many meanings to different communities; the distinction here is common in AI (*e.g.*, validation set *vs.* test set).

Testing: Use-case driven evaluation of the final (or alpha, or beta) product.

- The considered use cases should be **representative**.
- They should be **as unrelated as possible to** any concrete scenarios considered during development, including **the validation process**.
- Ideally, **conducted by prospective users**; if unavailable, "play the user."

University of Central Lancashire
UCLan

Verification, validation, and testing

Verification: Proof that the developed product complies with its specification.

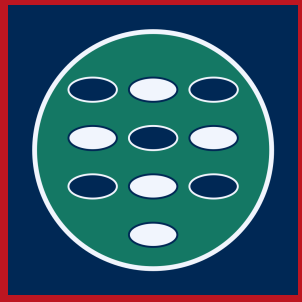
- Where possible, provide a **rigorous logical/mathematical proof**; alternatively, provide documents following agreed standards/procedures.

Note

- The above is what we mean by **formal verification**.
- There can be no verification without a **specification**.
- It can be done by humans, using code or pseudocode.
- It can also be done computationally (**automated verification**); in that case, either the programming language must be restricted severely, or it is only a **model of the program** that can be verified.
- The latter is known as **model checking**. It is limited by the accuracy and extent of the information provided in the model.

Noregs miljø- og biovitskaplege universitet

Institutt for datavitskap



Digitalisering på Ås

3 Validation

3.3 Formal program analysis

3.4 Holistic validation

3.5 Auto- and decorrelation

3.6 Time series block averaging

INF203

13th June 2025

What even is reproducibility?

Reproducibility definitions: Discussed in a review by Plesser.[1]

ACM definitions:

- Repeatability → Same team, same procedure, same lab.
- Replicability → Different team, same procedure, same or different lab.
- Reproducibility → All is different, except for the investigated quantity.

The attempts are successful if "the [same] measurement can be obtained with stated precision," *i.e.*, within the error bars that were part of the data item.

TABLE 1 | Comparison of terminologies. See text for details.

Goodman	Claerbout	ACM
		Repeatability
Methods reproducibility	Reproducibility	Replicability
Results reproducibility	Replicability	Reproducibility
Inferential reproducibility		

But that was only what ACM says. There are a myriad of ideas about this around.

[1]H. E. Plesser, *Frontiers Neuroinform.* **11**: 76, doi:10.3389/fninf.2017.00076, **2018**.

What even is reproducibility?

Reproducibility definitions: Discussed in a review by Plesser.[1]

Consider the case where a validator *b* contradicts findings by *a*:

1) Reseacher *a* did κ and found φ.

2) Reseacher *b* did γ and found $\zeta \neq \varphi$.

[1]H. E. Plesser, *Frontiers Neuroinform.* **11**: 76, doi:10.3389/fninf.2017.00076, **2018**.

What even is reproducibility?

Reproducibility definitions: Discussed in a review by Plesser.[1]

Common formulation and schema for reproducibility claims (RCs):

«Whenever research process κ'' is carried out, it must lead to the outcome φ''.»

Consider the case where a validator *b* contradicts findings by *a*:

1) Reseacher *a* did κ (consistent with κ'') and found φ (consistent with φ'').

   Here, *a* also made the **positive reproducibility claim** $\psi = \Box(\varphi'' \mid \kappa'')$.

2) Reseacher *b* did γ, **consistent with κ''**, and found ζ, **inconsistent with φ''**.

   Here, *b* made the **negative reproducibility claim** $\Diamond(\neg\varphi'' \mid \kappa'') \equiv \neg\Box(\varphi'' \mid \kappa'') \equiv \neg\psi$.

3) What is relevant there is the **contradiction between ψ and $\neg\psi$**.

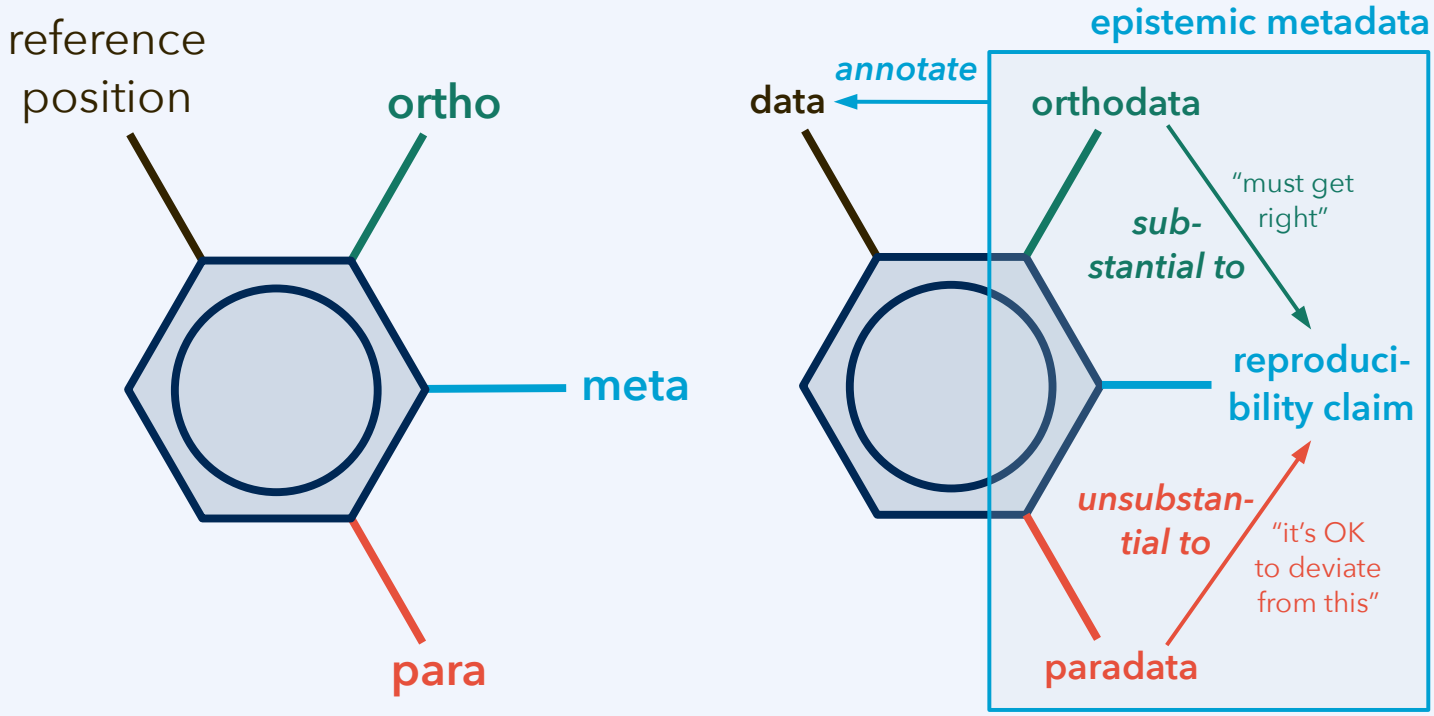
Claim ψ is usually implicit, ascribed to *a* based on unwritten community rules.[2]

[1]H. E. Plesser, *Frontiers Neuroinform.* **11**: 76, doi:10.3389/fninf.2017.00076, **2018**.

[2]In *Proc. FOIS 2023*, *pp.* 302–317, doi:10.3233/faia231136, **2023**.

Orthodata and paradata in reproducibility claims[1]

Reproducibility definitions: Discussed in a review by Plesser.[2]

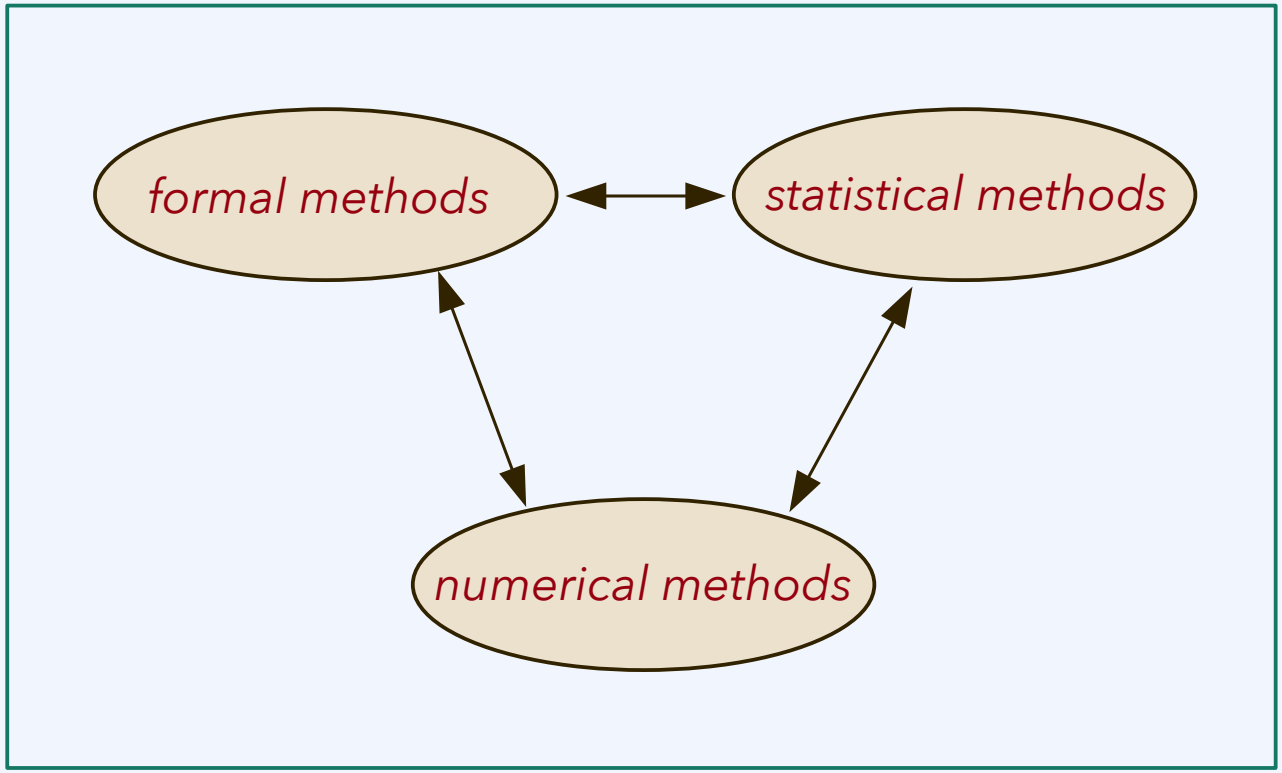


[1]*Epistemic metadata in molecular modelling: Second-stage case-study*, doi:10.5281/zenodo.7608074, **2023**.

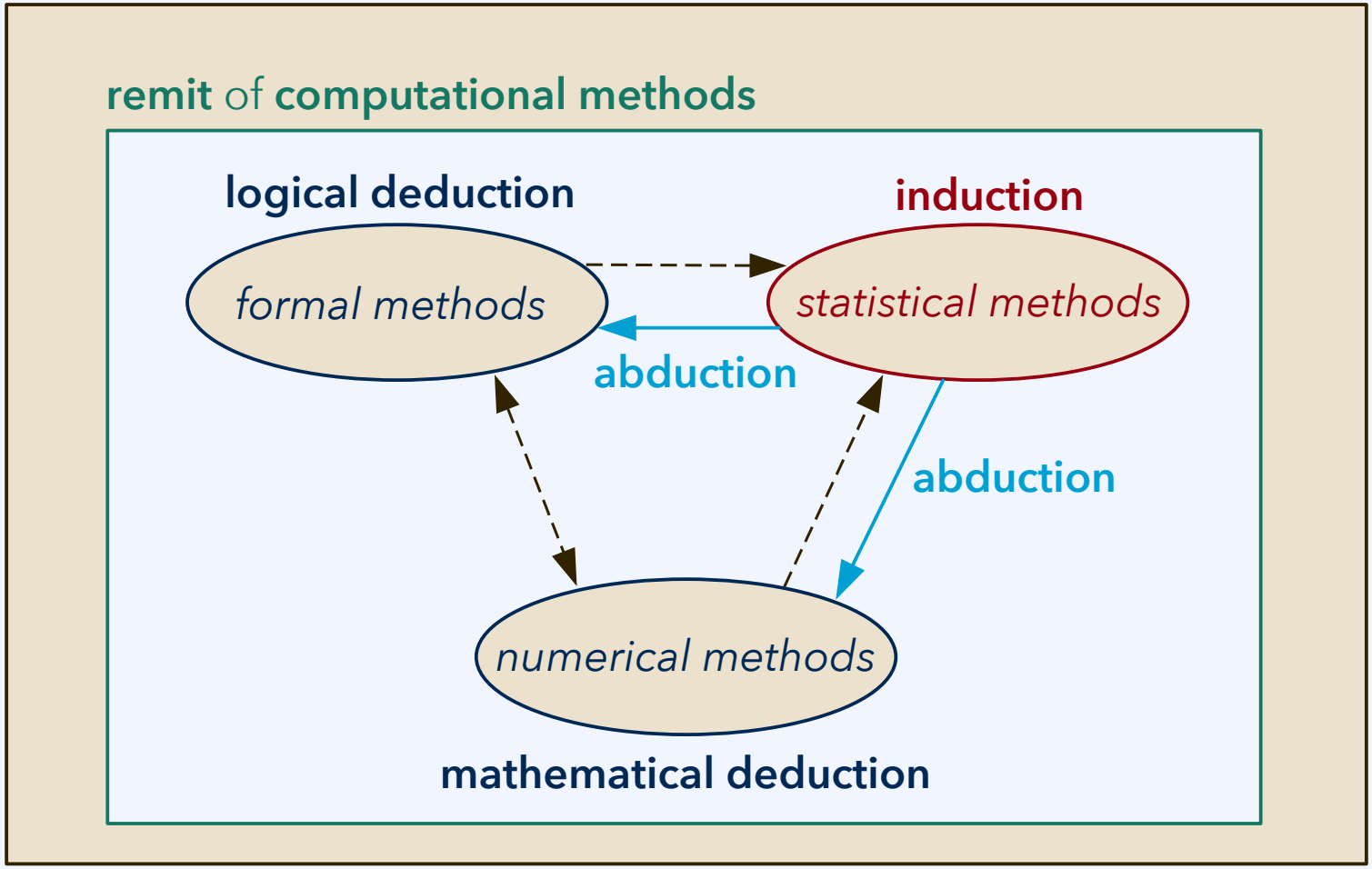
[2]H. E. Plesser, *Frontiers Neuroinform.* **11**: 76, doi:10.3389/fninf.2017.00076, **2018**.

Modes of reasoning

remit of **computational methods**



Modes of reasoning



human cognition employing **abduction**, **deduction**, and **induction**

Why don't more people use formal verification[1]?

While "these tools seem to have the potential to boost the reliability of codes, they are not widely adopted. […] So, from both a scientific and an epistemological perspective, it seems even more legitimate today to ask: *«If this stuff is so good, why isn't it used more?»* (Rushby 1997, 18).[2] Why does the scientific community not seize on this type of practice and why does it continue to shun one of the best tools for increasing confidence in scientific code?"

[1]C. Imbert, V. Ardourel, *Philos. Sci.* **90**(2): 376–394, doi:10.1017/psa.2022.78, **2023**.

[2]J. Rushby, in *Safety and Reliability of Software Based Systems*, *pp.* 1–42, Springer, **1997**.

Why don't more people use formal verification[1]?

While "these tools seem to have the potential to boost the reliability of codes, they are not widely adopted. […] **Why […]?**"

From own experience: For example, in MC simulation. Even if you formally prove that the Metropolis criterion is implemented correctly, that the random number generator is sufficiently random, *etc.*

> … *the whole process of model parameterization, simulation, and analysis of the results is far too complex. It is inaccessible to formal verification.*

Verifying parts could be nice, but it cannot replace "holistic validation." In effect, here, formal verification is nice but useless.

"[…] insights by Lenhard[2] (2018) about the tendency of the modularity of computational codes to erode […]"

[1]C. Imbert, V. Ardourel, *Philos. Sci.* **90**(2): 376–394, doi:10.1017/psa.2022.78, **2023**.

[2]J. Lenhard, *Philos. Sci.* **85**(5): 832–844, doi:10.1086/699675, **2018**.

Holistic validation: Johannes Lenhard (2018)

> Consider first a *simple brick wall*. It consists of a *multitude of modules*, each with certain form and static properties. These are combined into potentially very large structures. […] like *the auxiliary building of Bielefeld University* in front of my former office that is put together *from container modules*, some of which work as office space, others as restrooms, *etc.* […]
>
> These examples illustrate *how deeply ingrained modularity is in our way of building (larger) objects*. This applies also to the design of complex (software) systems.

J. Lenhard, "Holism, or the erosion of modularity: A methodological challenge for validation," *Philos. Sci.* **85**(5): 832–844, doi:10.1086/699675, **2018**.

Holistic validation: Johannes Lenhard (2018)

Validation is usually conceived in the very same modular structure: *independently validated modules* are *put together in a controlled way* for ensuring the bigger system is also valid.

And I frankly admit that there are well-articulated concepts that would, in principle, ensure software is clearly written, aptly modularized, well maintained, and superbly documented. However, the problem is that *science in principle differs from science in practice*.

The resulting problem for validating models is one of (confirmation) holism.

J. Lenhard, "Holism, or the erosion of modularity: A methodological challenge for validation," *Philos. Sci.* **85**(5): 832–844, doi:10.1086/699675, **2018**.

Holistic validation within the testing framework

Unit tests

Test one piece of code, *e.g.*, one method, for right arguments → return value.

Integration tests

Test concrete interactions between parts of the code, do they fit together?

Acceptance tests

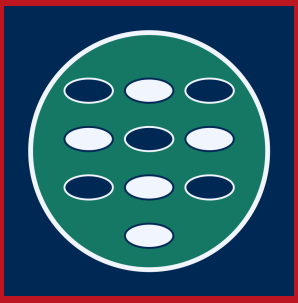
Holistic validation: Run the complete code/system, do y/n correctness checks.

Regression tests

Added once a bug is detected and fixed. Check that the bug does not return.

Noregs miljø- og biovitskaplege universitet

Institutt for datavitskap



Digitalisering på Ås

3 Validation

3.3 Formal program analysis

3.4 Holistic validation

3.5 Auto- and decorrelation

3.6 Time series block averaging

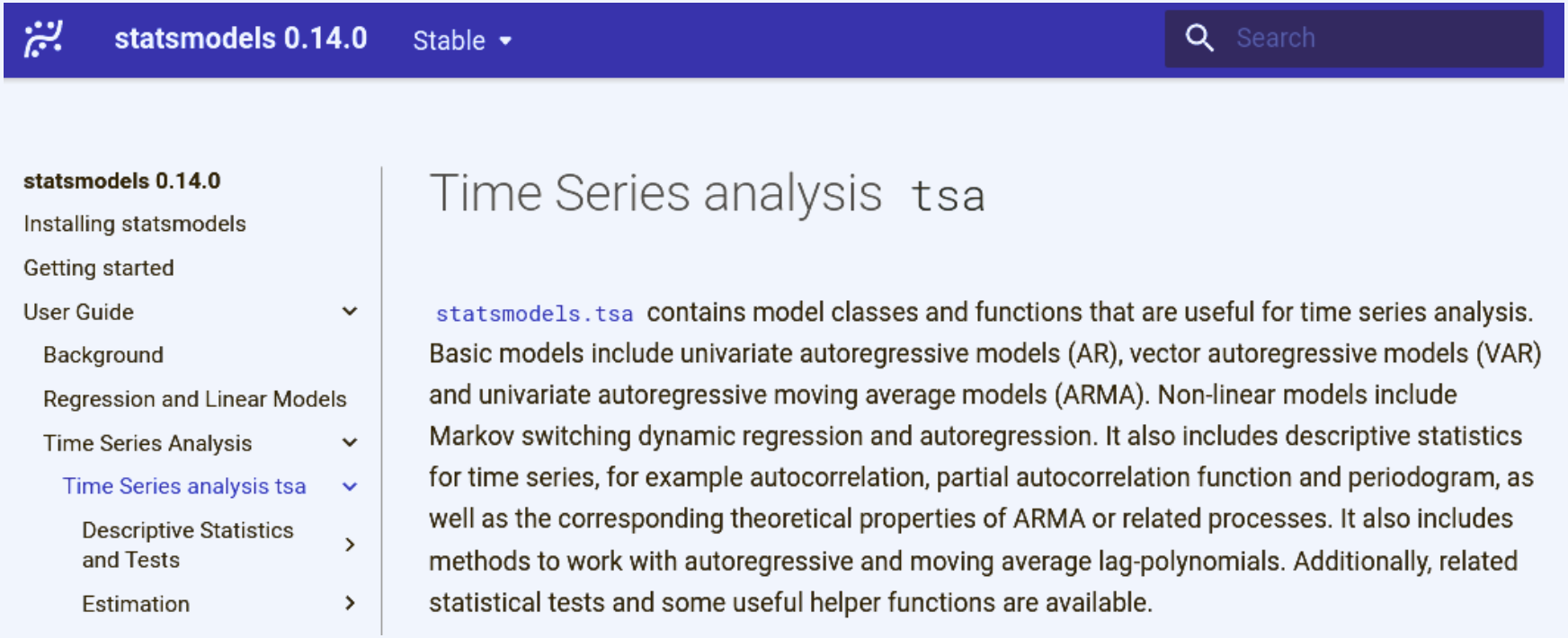
INF203

13th June 2025

Time series in Python

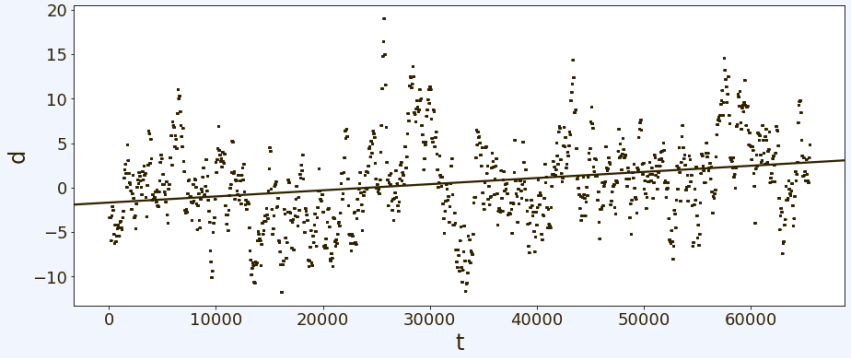
https://www.statsmodels.org/stable/tsa.html

https://www.statsmodels.org/stable/examples/index.html#time-series-analysis



Autocorrelation of time series data

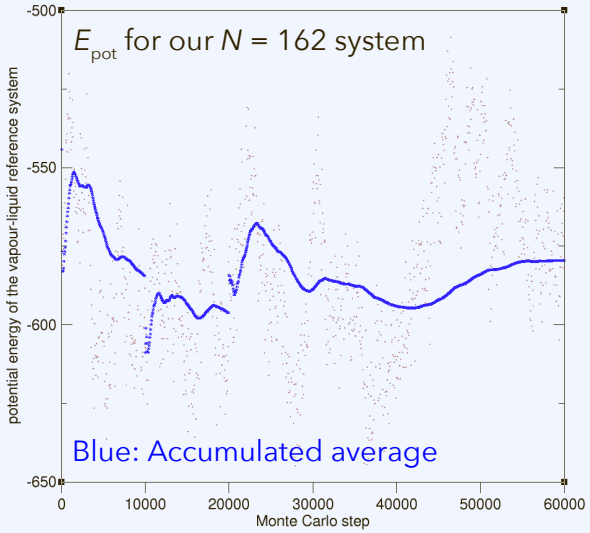
Time series data are **autocorrelated**. This means that *data points taken at times close to each other cannot be regarded as independent* items of information.

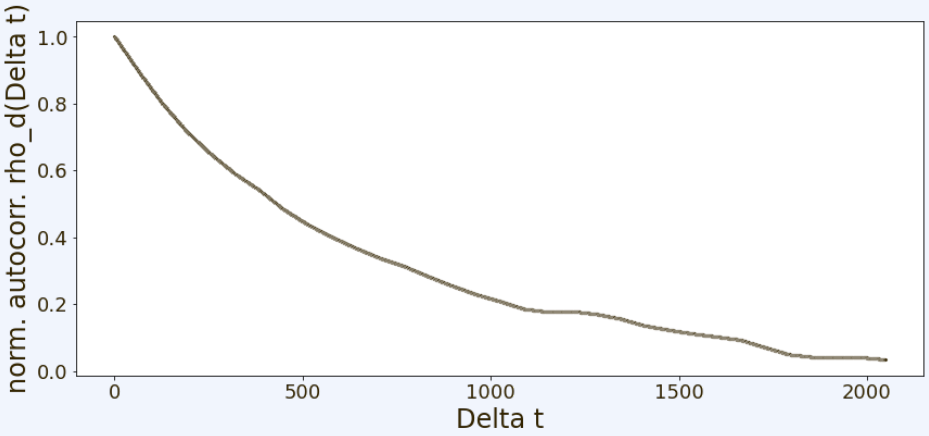


Assume we are given time series data $d(t)$:

autocorrelation $R(\Delta t) = \langle d(t)\, d(t+\Delta t)\rangle$
autocovariance $\langle [d(t) - \langle d\rangle]\,[d(t+\Delta t) - \langle d\rangle]\rangle$

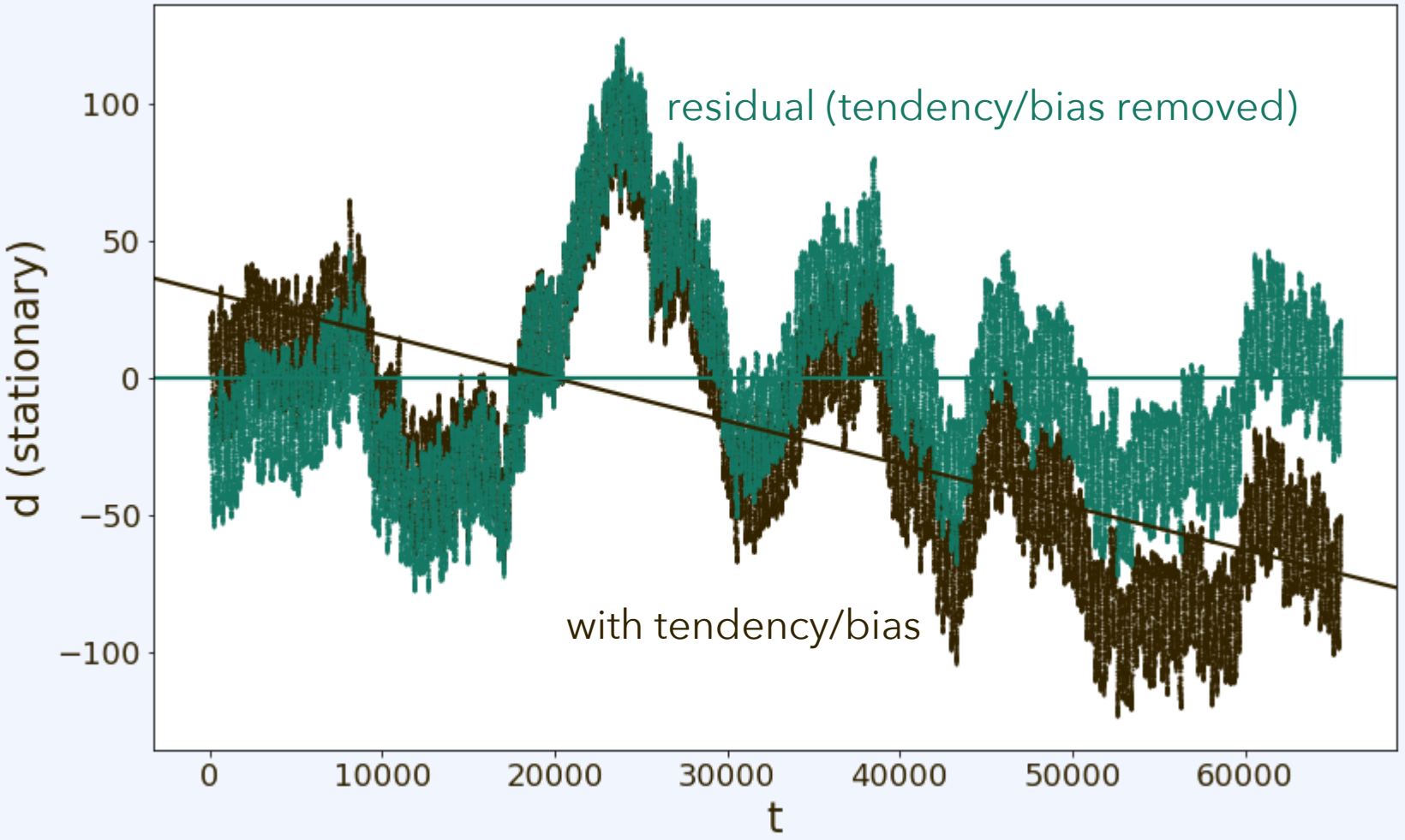
Often **normalized** by Var(d) to yield $\rho(\Delta t)$.





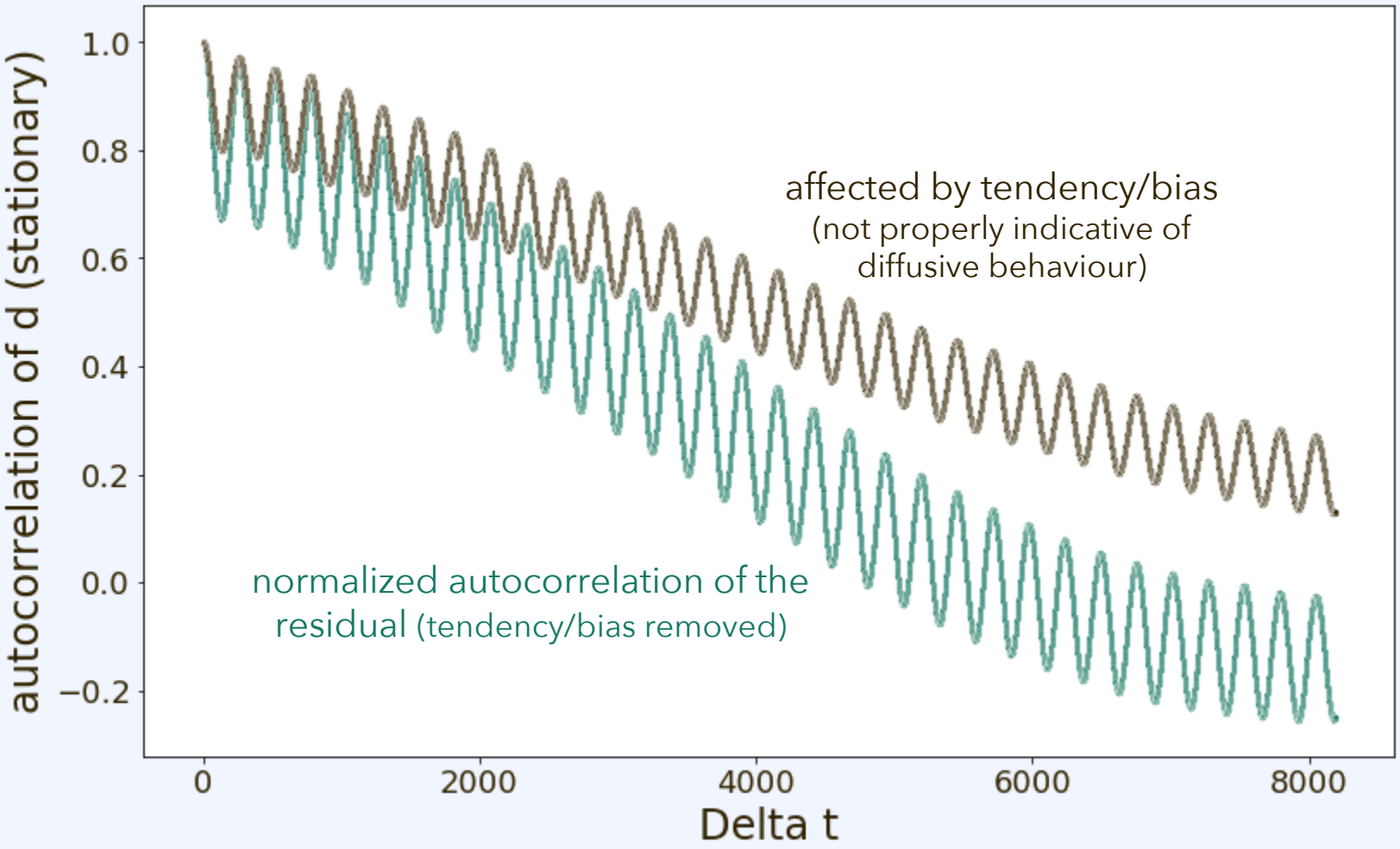
Autocorrelation function (normalized autocovariance)

autocorrelation-statsmodels.ipynb



Autocorrelation function (normalized autocovariance)

autocorrelation-statsmodels.ipynb



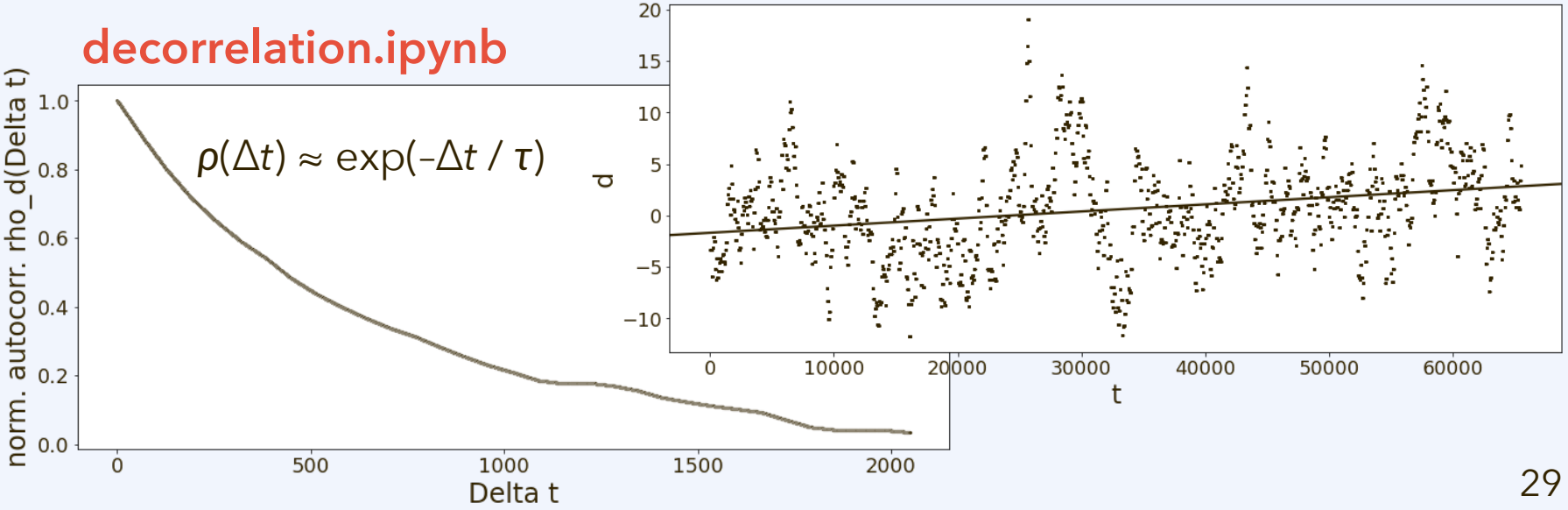
Decorrelation time

A typical approach to uncertainty estimation is based on:

- Some set of data that are representative of the phenomenon;
- Separation of data points into training, validation, and test data.

But: Different data *points on a time series are not independent* tests, they are correlated – this is exactly what is expressed by the autocorrelation function.

decorrelation.ipynb



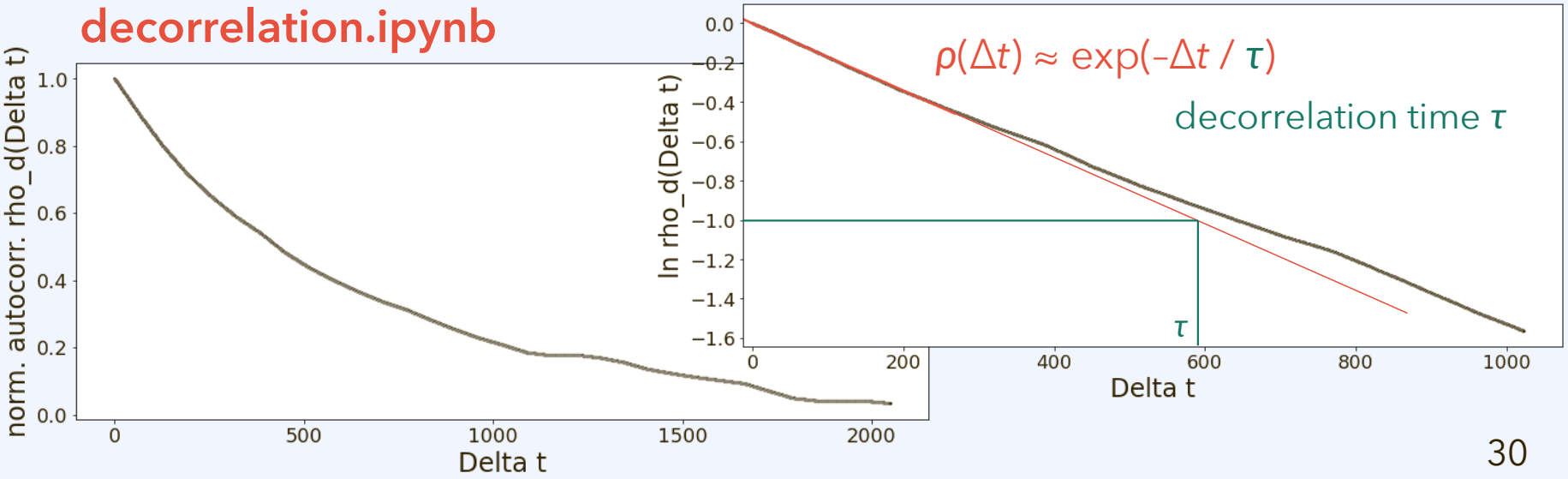
Decorrelation time

A typical approach to uncertainty estimation is based on:

- Some set of data that are representative of the phenomenon;
- Separation of data points into training, validation, and test data.

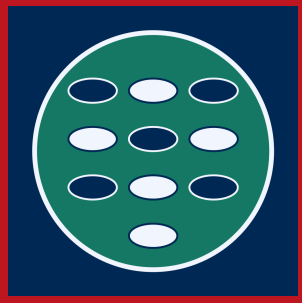
But: Different data *points on a time series are not independent* tests, they are correlated – this is exactly what is expressed by the autocorrelation function.

decorrelation.ipynb



Noregs miljø- og biovitskaplege universitet

Institutt for datavitskap



Digitalisering på Ås

3 Validation

3.3 Formal program analysis
3.4 Holistic validation
3.5 Auto- and decorrelation
3.6 Time series block averaging

INF203

13th June 2025

Uncertainty in time series

Our approach to uncertainty estimation can now be based on:

- Some set of data that are representative of the phenomenon;
- Separation of data points such that they are *decorrelated*.

Once Δt exceeds 3τ, the normalized autocorrelation is small, $\rho(\Delta t) < 0.05$. We can *average over blocks* with size 3τ (or more) and then treat each of these **block averages** as independent data points.

Warning: This only works if the autocorrelation *actually is* decaying.

It will lead to mistakes where there is a strong correlation over long timespans, such as when analysing a periodic signal. If your data points are not really decorrelated, you can never treat them as independent items of information.

Uncertainty in time series

Our approach to uncertainty estimation can now be based on:

- Some set of data that are representative of the phenomenon;
- Separation of data points such that they are *decorrelated*.

Once Δt exceeds 3τ, the normalized autocorrelation is small, $\rho(\Delta t) < 0.05$. We can *average over blocks* with size 3τ (or more) and then treat each of these **block averages** as independent data points. This is called **block averaging**.

N_b such blocks correspond to N_b–1 independent deviations from the mean.

Variance of the block averages: $\sigma_b^2 = (N_b - 1)^{-1} \Sigma (B_i - \langle B \rangle)^2$

Uncertainty based on σ, where $\sigma = N_b^{-1/2} \sigma_b$ from central limit theorem.

A rigorous theory of block averaging was developed by Flyvbjerg and Petersen[1] (which is therefore also called *Flyvbjerg-Petersen block averaging*).

[1]H. Flyvbjerg, H. G. Petersen, *J. Chem. Phys.* **91**: 461–466, doi:10.1063/1.457480, **1989**.

Uncertainty in time series

block-averaging.ipynb

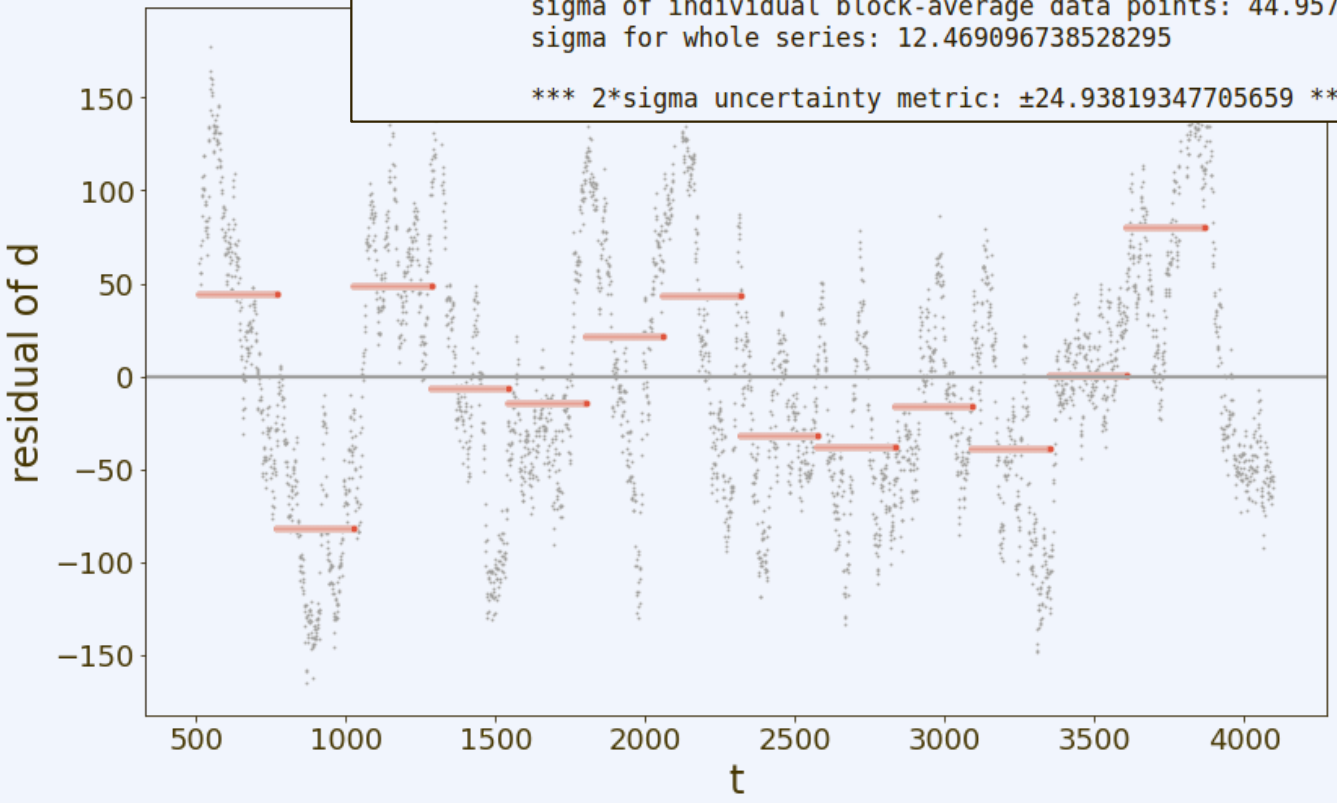
```
In [12]: print("sigma of individual block-average data points:", sigma_block_avg)

         # apply central limit theorem and divide by square root of number of blocks
         sigma_whole = sigma_block_avg / math.sqrt(N_blocks)
         print("sigma for whole series:", sigma_whole)

         # uncertainty metric based on 2*sigma
         print("\n*** 2*sigma uncertainty metric: ±", 2*sigma_whole, " ***", sep="")

         sigma of individual block-average data points: 44.95796764948456
         sigma for whole series: 12.469096738528295

         *** 2*sigma uncertainty metric: ±24.93819347705659 ***
```



Uncertainty in time series

nok-eur.ipynb

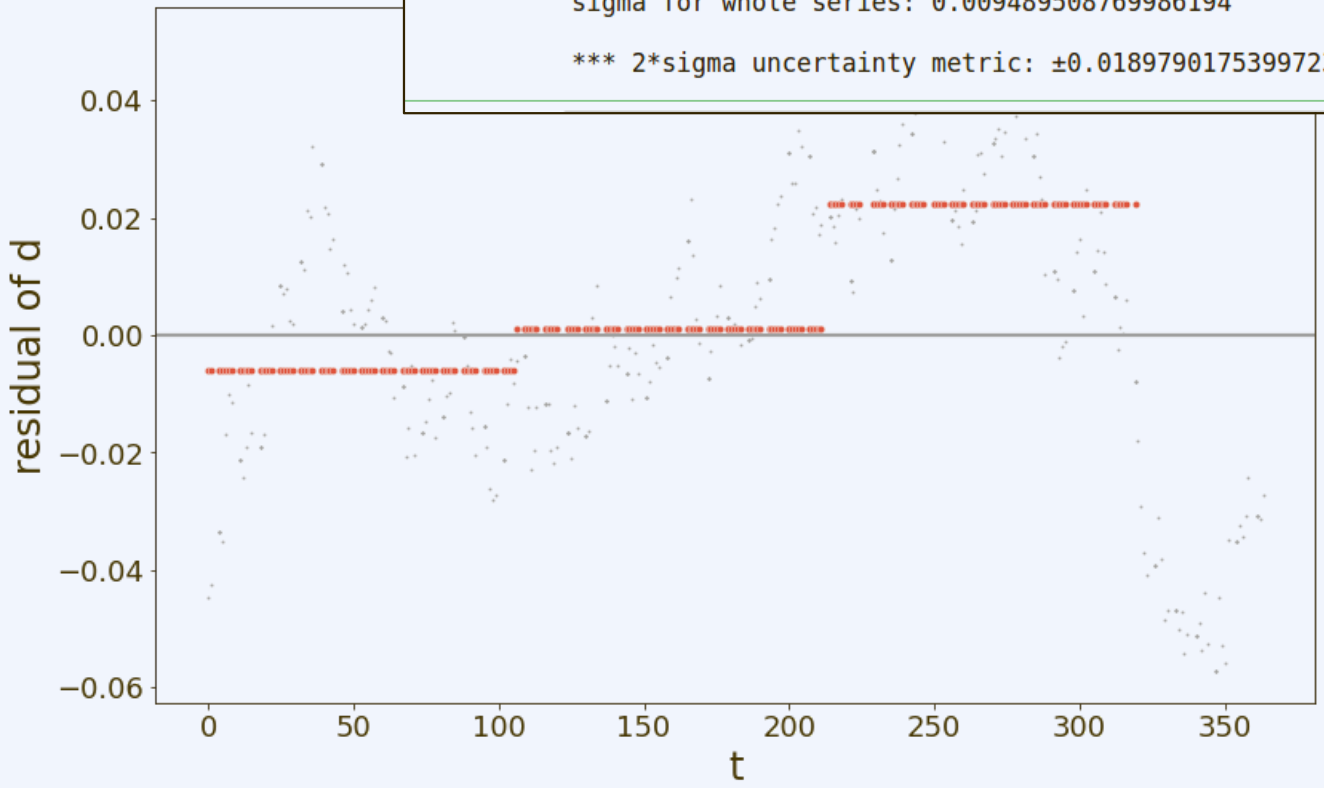
```
In [16]: print("sigma of individual block-average data points:", sigma_block_avg)

         # apply central limit theorem and divide by square root of number of blocks
         sigma_whole = sigma_block_avg / math.sqrt(N_blocks)
         print("sigma for whole series:", sigma_whole)

         # uncertainty metric based on 2*sigma
         print("\n*** 2*sigma uncertainty metric: ±", 2*sigma_whole, " ***", sep="")

         sigma of individual block-average data points: 0.016436311328486528
         sigma for whole series: 0.009489508769986194

         *** 2*sigma uncertainty metric: ±0.018979017539972387 ***
```



Flyvbjerg-Petersen method

The Flyvbjerg-Petersen method works without computing the autocorrelation. It instead analyses convergence behaviour over the number of blocks N_b:

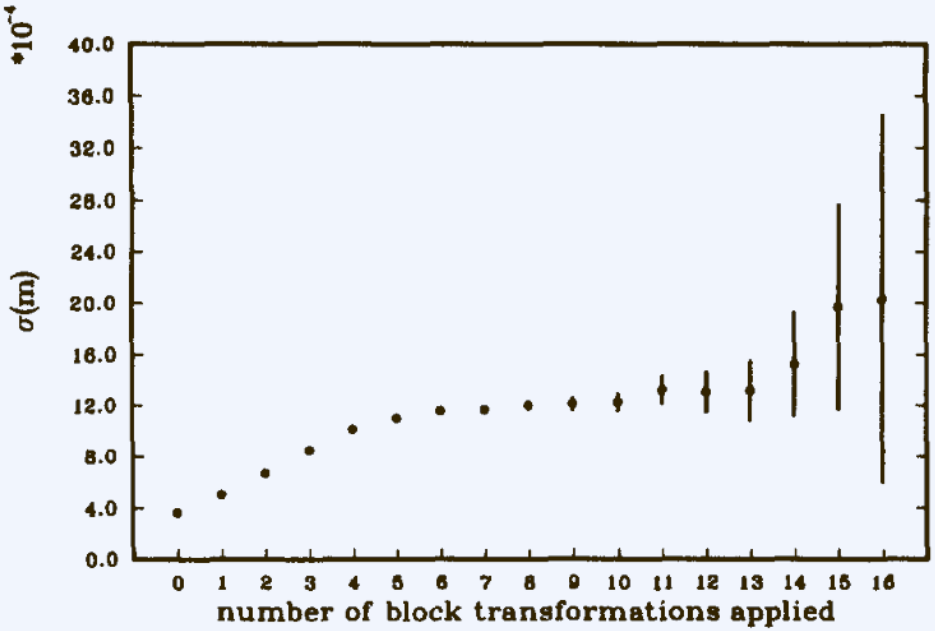
At the fixed point the "blocked" variables $(x'_i)_{i=1,\dots n'}$ are independent Gaussian variables—Gaussian by the central limit theorem, and independent by virtue of the fixed point value of γ'_t. Consequently, we can easily estimate the standard deviation on our estimate $c'_0/(n'-1)$ for $\sigma^2(m)$. It is $(\sqrt{2/(n-1)}\, c'_0/(n'-1)$:

$n' = N_b$

$$\sigma^2(m) \approx \frac{c'_0}{n'-1} \pm \sqrt{\frac{2}{n'-1}} \frac{c'_0}{n'-1}, \qquad (27)$$

$$\sigma(m) \approx \sqrt{\frac{c'_0}{n'-1}} \left(1 \pm \frac{1}{\sqrt{2(n'-1)}}\right). \qquad (28)$$

Knowing this error is a great help in determining whether the fixed point has been reached or not in actual calculations, as we shall see in Sec. VII.



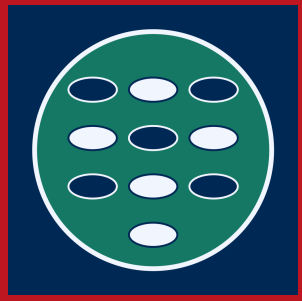
H. Flyvbjerg, H. G. Petersen, "Error estimates on averages of correlated data," *J. Chem. Phys.* 91(1): 461-466, doi:10.1063/1.457480, **1989**.

Observations on block averaging

- When you *indicate an uncertainty* (which you always should), it is also necessary to explain what kind of uncertainty – how it was determined.
- Block averaging is a widespread method for obtaining the *uncertainty of average values from time series*, and datasets similar to time series.
- The variance of block averages is $\sigma_b^2 = (N_b - 1)^{-1} \Sigma (B_i - \langle B \rangle)^2$, where *error bars are often given as* $\pm 2\sigma$, *i.e.*, twice the standard deviation.
- A *prerequisite* for this to work is that the size of a block is large enough so *that different blocks can be considered statistically independent*.
- *You can determine the decorrelation time* to see how large blocks should be, *or you can analyse the convergence behaviour* over N_b.
- In practice, people (who are not data scientists) often ignore the prerequisite, just use some number of blocks, and still call it "Flyvbjerg-Petersen method." This is very bad practice. Don't be like them.

Noregs miljø- og biovitskaplege universitet

Institutt for datavitskap



Digitalisering på Ås

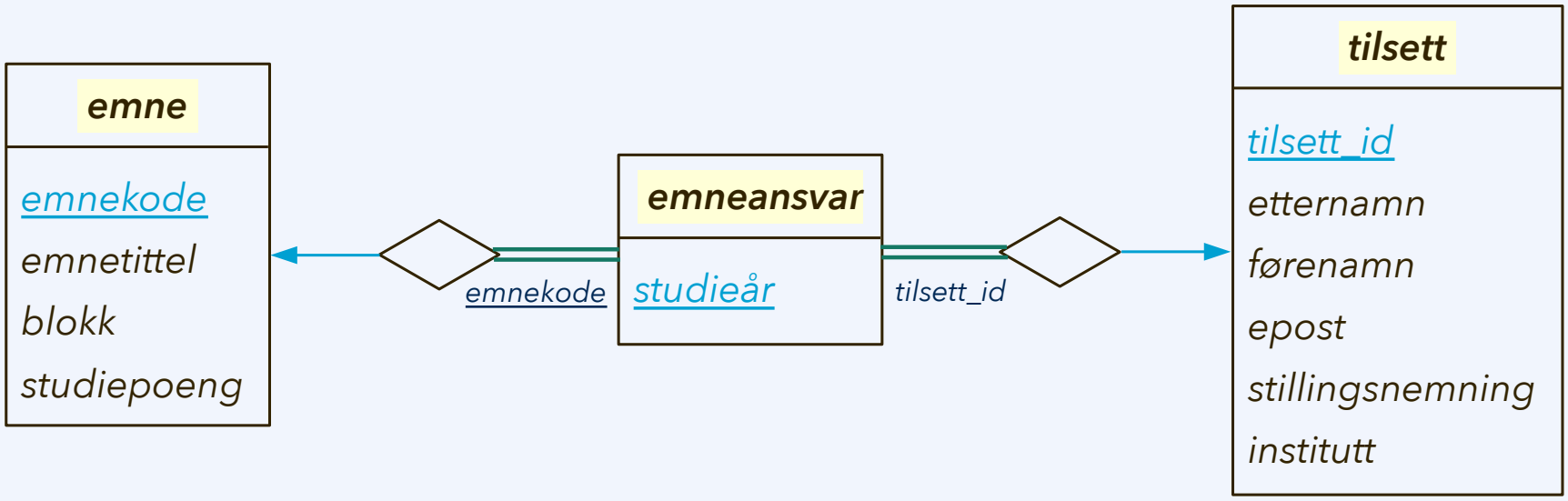
Discussion topic: E-R diagrams

INF203

13th June 2025

Questions about E-R diagrams

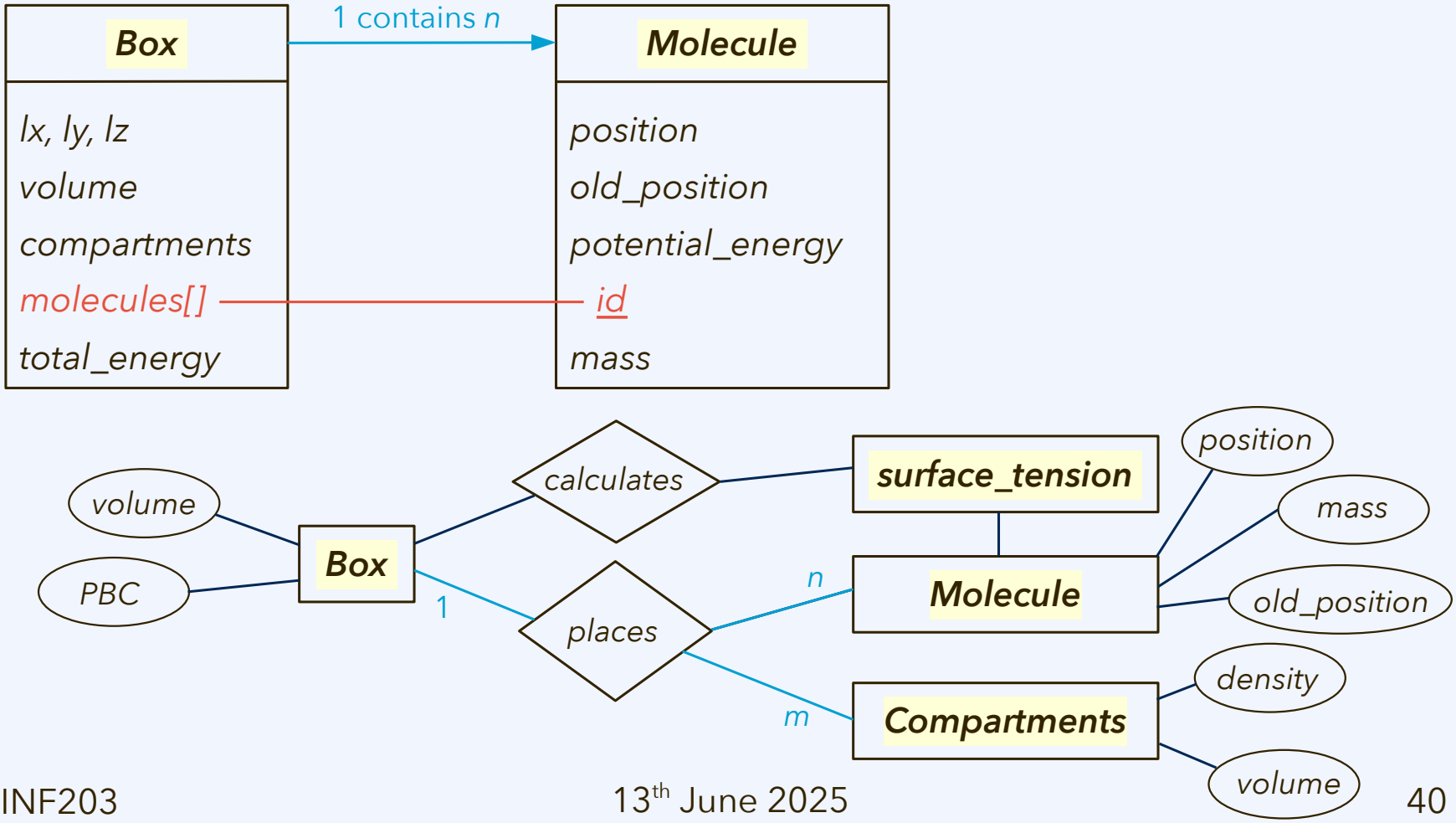
Confer *e.g.* the example:

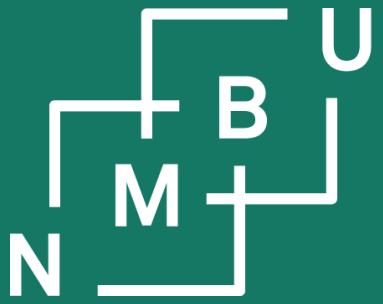


How can we use E-R diagram notation to express our object-oriented data structure design as clearly as possible?

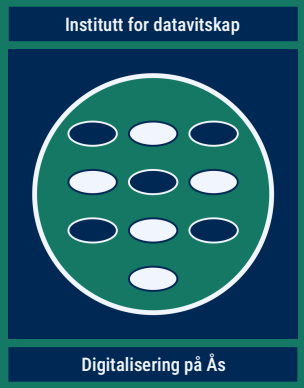
Questions about E-R diagrams

Let us look at some concrete discussion items, *e.g.*, a submission looking like:





Noregs miljø- og biovitskaplege universitet



INF203

June advanced programming project

3 Uncertainty, validation, and testing