Sensor interpolation data wrangling

Simon Dobson

School of Computer Science
University of St Andrews
Scotland UK

simon.dobson@st-andrews.ac.uk
https://simondobson.org
https://github.com/simoninireland

University of St Andrews | FOUNDED 1413

Introduction

We're interested in complex systems and sensors

- How do we deal with errors and failures?
- Can we develop better strategies for collection and analysis?

We're currently doing experiments into the issues

This talk explores the process we're going through

- How to study sensor error
- A huge volume of preparatory work
- An unsatisfyingly small number of results
- Hopes as to what will pay off in the future

Acknowledgements

Collaborators

- Muffy Calder, Julie McCann, Michael Fisher
- Peter Mann, Yasmeen Rafiq, Lei Fang, Michele Sevegnani, Sven Linker
- Juan Ye, Danilo Pianini, Mirko Viroli

Partially funded by the UK EPSRC under grant number EP/N007565/1 (Science of Sensor Systems Software, S4)

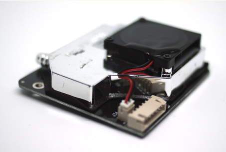
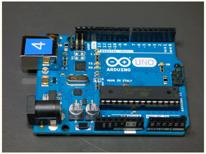
EPSRC
Engineering and Physical Sciences
Research Council

Structure of this talk

Sensor systems

Increasing numbers of sensors

- Dedicated sensors

- "Casual" sensors attached to other things, like cellphones
- Often aggregated into sensor networks

A torrent of data being returned all the time

- How to we interpret it? How do we justify the costs of its collection and storage?

Characteristics

Things you may know about sensors

- Varying accuracy and precision
- Wildly varying costs and power requirements

Characteristics

Things you may know about sensors

- Varying accuracy and precision
- Wildly varying costs and power requirements

Things you may *not* know about sensors

- They fail. A lot
- Limited physical lifetime
- Mechanical degradation from weathering, plants, animal activity, ...
- Cost and power can affect placement decisions

Consequences

Lifespan

- We will tend to leave expensive sensors in the field as long as possible, to try to extract maximum value from them

⇒ What happens to the results as they degrade?

⇒ How does (should) this affect our decision-making?

Placement

- Placement is often defined by where we *can* put sensors, rather than by where we might *want* to put them

⇒ What are the consequences of taking readings from "imperfect" locations? (More may not be better [1])

[1] D. Pianini, S. Dobson, and M. Viroli. Self-stabilising target counting in wireless sensor networks using Euler integration. In *Proceedings of the Eleventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'17)*, pages 11–20, September 2017. doi: 10.1109/SASO.2017.10

Structure of this talk

Background

Studying placement and error

Data wrangling

Results

Conclusions

Scientific question

What are the effects of sensor placement and error on the analytic approaches we use to interpret the data collected?

Scientific question

What are the effects of sensor placement and error on the analytic approaches we use to interpret the data collected?

Not simply on the raw data

- There's almost always substantial post-collection analysis
- We need to understand the impacts of error on what processes see *post facto*
- Different sensitivities to different issues

Experimental approaches

Put sensors in the field and let them decay

- (This is what we wanted to do)
- It's difficult to persuade someone to fund it…

Experimental approaches

Put sensors in the field and let them decay ✗

- (This is what we wanted to do)
- It's difficult to persuade someone to fund it…

Build a mathematical or simulation model

- There's really not enough known
- Assumptions would be in some senses arbitrary

Experimental approaches

Put sensors in the field and let them decay ✗

- (This is what we wanted to do)
- It's difficult to persuade someone to fund it…

Build a mathematical or simulation model ✗

- There's really not enough known
- Assumptions would be in some senses arbitrary

Find a dataset that's amenable to synthetic error

- Something that's dense enough to support "fake" failure and error
- Cause problems deliberately, in a controlled way

Experimental approaches

Put sensors in the field and let them decay ✗

- (This is what we wanted to do)
- It's difficult to persuade someone to fund it…

Build a mathematical or simulation model ✗

- There's really not enough known
- Assumptions would be in some senses arbitrary

Find a dataset that's amenable to synthetic error ✓

- Something that's dense enough to support "fake" failure and error
- Cause problems deliberately, in a controlled way

What we decided to do

1. Take a dataset that's been collected using a recognised methodology, and interpolated using an approach that accepted as "good enough"
2. Change the sample set, re-interpolate, and compare with the original
   - *Failure* – Remove some fraction of nodes: are some failures worse than others?
   - *Error* – Change the value at some fraction of nodes: are some errors more disruptive than others?
   - *Placement* – Remove some nodes at points one would expect to be "good observations": are these nodes in places whose omission significantly changes the results?

What we decided to do

1. Take a dataset that's been collected using a recognised methodology, and interpolated using an approach that accepted as "good enough"
2. Change the sample set, re-interpolate, and compare with the original
   - *Failure* – Remove some fraction of nodes: are some failures worse than others? ⇐ This is where we are so far
   - *Error* – Change the value at some fraction of nodes: are some errors more disruptive than others?
   - *Placement* – Remove some nodes at points one would expect to be "good observations": are these nodes in places whose omission significantly changes the results?

Dataset desiderata

Large-scale

- Enough points for individual nodes not to dominate

Dense

- Enough points that we can remove some and still have a dense network

Some notion of ground truth

- An interpretation of the samples, for example by interpolating to a finer granularity
- (We also need to be able to reproduce this interpretation, at least to some level)

Data sources

Climate science has a lot of datasets with (some of) the properties we need datasets

- UK Met Office CEDA MIDAS Archive: 150+ stations, extensive historical archive, a bit sparse in places
- Scottish EPA “tipping buckets”: 280+ stations, geographically limited, about 20 years’ of data from a varying sub-set of stations
- UK EPA: 950+ stations, live and recent data only

Data sources

Climate science has a lot of datasets with (some of) the properties we need datasets

- UK Met Office CEDA MIDAS Archive: 150+ stations, extensive historical archive, a bit sparse in places
- Scottish EPA "tipping buckets": 280+ stations, geographically limited, about 20 years' of data from a varying sub-set of stations
- UK EPA: 950+ stations, live and recent data only

Comments

- Density, stability, and longevity are a hard ask

Interpretation sources

A common interpretation

- CEH-GEAR interpolation [2] , whole UK at 1km resolution back to the 19th century using a stable and well-respected algorithm (now pretty much the global standard)

[2] V. Keller, M. Tanguy, I. Prosdocimi, J. Terry, O. Hitt, S. Cole, M. Fry, and D. Morris. CEH-GEAR: 1km resolution daily and monthly areal rainfall estimates for the UK for hydrological and other applications. *Earth System Science Data*, 7:143–155, 2015. doi: 10.5194/essd-7-143-2015

Interpretation sources

A common interpretation

- CEH-GEAR interpolation [2] , whole UK at 1km resolution back to the 19th century using a stable and well-respected algorithm (now pretty much the global standard)

Comments

- Don't link to the raw dataset underlying the interpolation, or identify the actual stations

[2] V. Keller, M. Tanguy, I. Prosdocimi, J. Terry, O. Hitt, S. Cole, M. Fry, and D. Morris. CEH-GEAR: 1km resolution daily and monthly areal rainfall estimates for the UK for hydrological and other applications. *Earth System Science Data*, 7:143–155, 2015. doi: 10.5194/essd-7-143-2015

First take-away

> You can't always get what you want.
>
> — M. Jagger

Datasets are always compromised

- They weren't collected with you in mind
- Often have varying histories, and inadequate metadata
- Missing values aren't always noted properly
- The sensors have errors – funnily enough – that often get through

Data formats

There are, fortunately, standard data formats

- For example, NetCDF [3] represents large multi-dimensional arrays efficiently
- (Although its string handling is terrible)
- Great metadata support
- Language bindings

[3] Unidata. Network Common Data Format (NetCDF). Technical report, University Corporation for Atomospheric Research (UCAR), 2019. URL http://doi.org/10.5065/D6H70CW6

Data formats

There are, fortunately, standard data formats

- For example, NetCDF [3] represents large multi-dimensional arrays efficiently
- (Although its string handling is terrible)
- Great metadata support
- Language bindings

…which makes it bizarre that some organisations prefer JSON or CSV

- And not *just* CSV, but CSV where the first rows are metadata and free-text and only later become data…

[3] Unidata. Network Common Data Format (NetCDF). Technical report, University Corporation for Atomospheric Research (UCAR), 2019. URL http://doi.org/10.5065/D6H70CW6

Not actually as perverse as it seems

Support a lot of different use cases

- Web sites presenting live(ish) data
- Small-scale consumption of data from specific places
- Large-scale science

JSON isn't a bad choice for the first two

- …but it's terrible for the third

Access

Not all of this data is properly open-source

- Free for (UK) academic use, varied licences for others

Accessible through the web

- REST APIs (of different kinds)
- …sometimes behind a username/password firewall
- …and sometimes requiring a client-side SSL certificate to be installed first (and frequently)

Different arrangements

- Get data by time, or by station?
- One request per station? One per instrument? One per day? One per month? …

Second take-away

> TANSTAAFL (There ain't no such thing as a free lunch).
>
> Robert Heinlein

Truly open, interoperable data is (largely) a myth

- Understandable, given that *someone* paid for it to be collected and curated
- Every choice is predicated on a use case – and might not work well for others
- Supporting varied use cases requires significant commitment

Structure of this talk

Automating access – acquisition

We resisted the temptations of manual download

- Automate data acquisition
- Essential for reproducibility

We wrote a collection of scripts to hit the API endpoints

- Get the list of available stations
- Grab the data from each station in the form it's presented
- Wrangle it into the form we want

Automating access – standards (again)

We then defined a standard data format to hold the data we acquired

- Took CEH-GEAR's NetCDF model as a basis
- Define a common format for raw data with metadata (we fortunately had some prior experience in this [4])

Metadata
- start :: the start date
- end :: the end date
- resolution :: daily or monthly
- description :: text description
- history :: text timestamp
- source :: the data source URL

Dimensions
- station :: the station number
- time :: the sample point in days since 1800-1-1

Variables
- name(station) :: the station name
- lat(station) :: the station latitude
- long(station) :: the station longitude
- x(station) :: the station easting to the nearest km
- y(station) :: the station northing to the nearest km
- rainfall_amount(time, station) :: rainfall in kg/m^2 (= mm)

[4] S. Dobson, M. Golfarelli, S. Graziani, and S. Rizzi. A reference architecture and model for sensor data warehousing. *IEEE Sensors Journal*, 18, 2018. doi: 10.1109/JSEN.2018.2861327

The interpolation algorithm

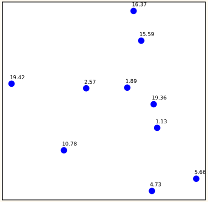
Natural nearest neighbour interpolation

- Discrete points s_i, each with a sampled rainfall $\mathcal{R}(s_i)$
- Within a boundary $\mathcal{B}$

The interpolation algorithm

Natural nearest neighbour interpolation

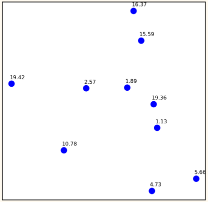
- Discrete points s_i, each with a sampled rainfall $\mathcal{R}(s_i)$
- Within a boundary $\mathcal{B}$



The interpolation algorithm

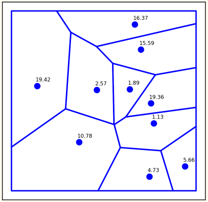
Natural nearest neighbour interpolation

- Discrete points s_i, each with a sampled rainfall $\mathcal{R}(s_i)$
- Within a boundary $\mathcal{B}$



Divide-up the space

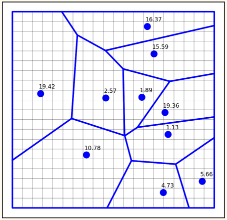
- The Voronoi diagram $\mathcal{V}$
- For a sample point s_i, the Voronoi cell $\mathcal{V}(s_i)$ is the set of points $p \in \mathcal{B}$ that lie closer to s_i than to any other s_j



NNI – synthetic points

Interpolation grid

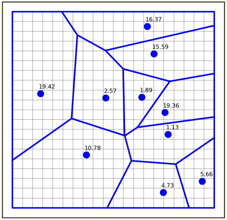
- Construct a grid of points at which to interpolate the samples
- The samples and the grid constitute the "map"



NNI – synthetic points

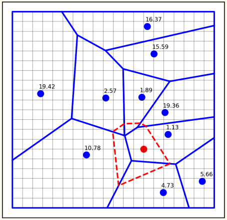
Interpolation grid

- Construct a grid of points at which to interpolate the samples
- The samples and the grid constitute the "map"



Construct synthetic points and cells

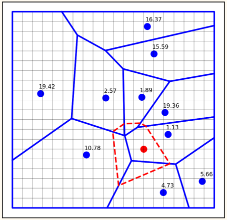
- For each interpolation point d_{xy}, construct a new Voronoi diagram $\mathcal{D}$ with a cell $\mathcal{D}(s')$ for each point $s' \in \{s_i\} \cup \{d_{xy}\}$



NNI – values

Compute the interpolated samples

- For each interpolated point d_{xy}, the interpolated rainfall $\mathcal{R}^{int}(d_{xy})$ is the sum of $\mathcal{R}(s_i)$ times the fraction of $\mathcal{D}(d_{xy})$ that overlaps $\mathcal{V}(s_i)$, for all sample points s_i
- Only neighbouring cells contribute any value



Complexity

This is, as you can imagine, quite a lot of computation

- Geometric calculations for every interpolation point
- Takes hours for even small examples
- Impractical to do this for every set of samples

Third take-away

> Science: that feeling you get when you realise that that thing you don't understand isn't actually understood by anyone.

The paper may not tell you what you need to know

- The authors might regard computation as just a mechanism, not what their readers will be interested in
- The authors may be using someone else's code that they don't understand
- The authors may be keeping their sauce secret

Tensor formulation

The weights that each sample contributes to each interpolation point are fixed for a given set of samples points

- Given a map, we can pre-compute the weights
- For each interpolation point d_{xy} there is a vector w_{xy} of weights, $|w_{xy}| = |s_i|$

A *tensor* capturing the interpolation of a given map

- A 3d block of numbers
- Each entry T_{xyi} is the weight given to the value $\mathcal{R}(s_i)$ in computing the interpolated value at d_{xy}

Interpolation

Interpolation is just linear algebra

- Apply the tensor in a particular way to a vector of observations, one per sample point

Take the dot product of the weights vector (one-form) at each interpolation point with the vector of samples

- Given a tensor $\mathcal{T}$ and vector of samples $\mathcal{R}(s_i)$, produce a matrix $\mathcal{G}$ where $\mathcal{G}_{xy} = \sum_i \mathcal{T}_{xyi} \cdot \mathcal{R}(s_i)$
- Expensive if done using that standard maths routines (`numpy.dot`)

Optimisation – sparseness

Most weights are zero: an interpolation point typically uses the weights of about 6 observations

- Optimise to extract the non-zero elements
- Can interpolate rainfall over the whole of England at 1km resolution from the 980+ EPA stations in about 15s (single 3.8GHz Intel i7 core)

```
# create the result grid
grid = numpy.zeros((self._tensor.shape[0],
                    self._tensor.shape[1]))

# apply the tensor, optimising for sparseness
for i in range(grid.shape[0]):
  for j in range(grid.shape[1]):
    # extract indices of the non-zero elements
    # of each weighting row
    nz = numpy.nonzero(self._tensor[i, j, :])[0]

    # compute the weighted sum
    if len(nz) > 0:
      # sparse dot product, including
      # only the non-zero elements
      grid[i, j] = numpy.dot(self._tensor[i, j, nz],
                             samples[nz])
```

Fourth take-away

> In theory there is no difference between theory and practice. But in practice, there is.
>
> J.L.A. van de Snepscheut

Big data often *requires* early optimisation

- Hard to get anywhere without optimised code
- …even in order to do meaningful tests
- Some of the speed-ups are quite astonishing: small individual improvements, but millions of repetitions

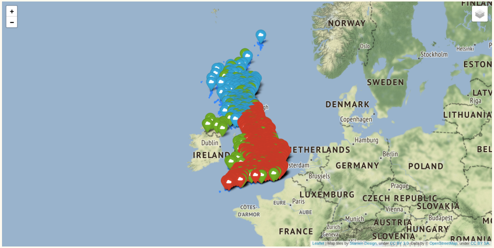
A lot of this code also paralellises well

Structure of this talk

Loading the datasets

We could now, finally, start work

- The full set of rain gauges available from EPA, SEPA, and CEDA MIDAS

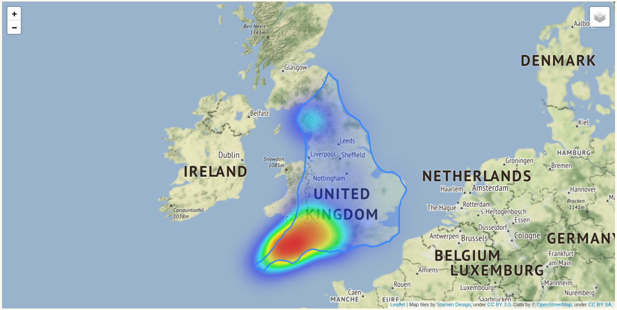


We choose *one* of these datasets to work with

- The EPA ("live") set is the densest, with 980+ stations
- (SEPA + CEDA MIDAS has about 500+ stations but a better historical archive, and would also be a reasonable choice)

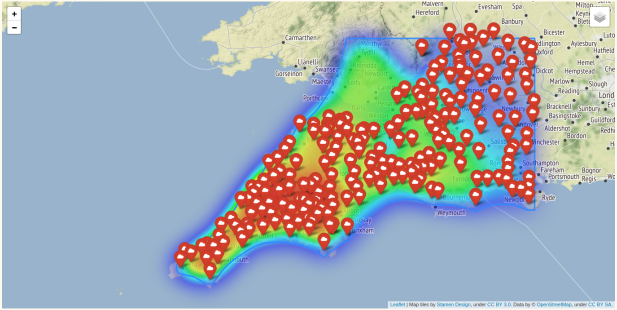
Interpolating England's rainfall

For one particular day (2022-03-11)



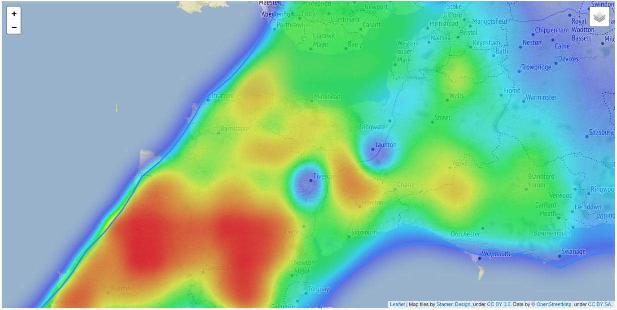
Focus on Cornwall and the south-west

About 220 stations



Focus on Cornwall and the south-west

Without the stations



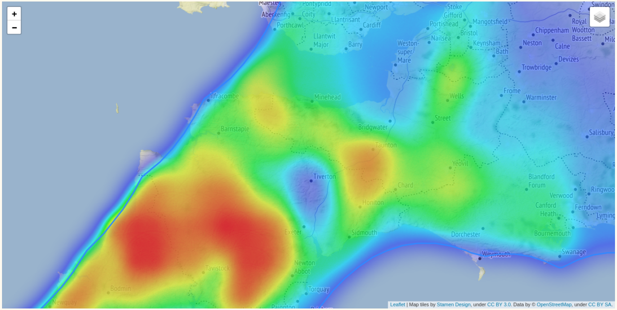
How does failure affect the interpolation?

Remove 40% of the stations at random

- A scenario of widespread failure or ageing of the rain gauges
- Or, alternatively, a scenario where we're deploying a smaller system to see whether it's "accurate enough" for our needs

What do we think a 40% reduction in observations will do?

How does failure affect the interpolation?

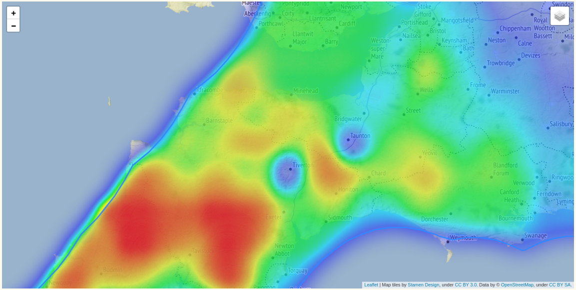


Perhaps not as dramatic as we might have expected

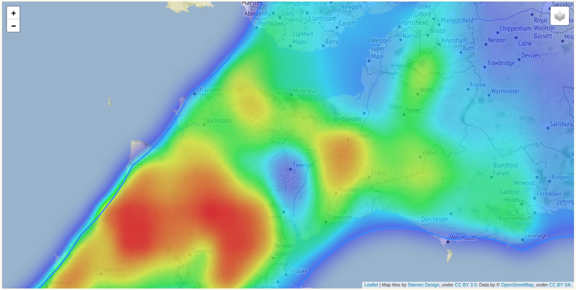
- An overall reduction in observed rainfall (less red)
- …but not uniformly so: if we remove observation of no rain, we *increase* the impact of neighbouring rainy observations

Side by side

All the stations



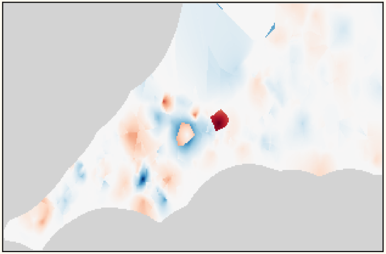
60% of stations



Highlighting the differences

Subtract one map from the other

- Red means higher rainfall in original interpolation
- The differences are quite localised
- Clear that we don't uniformly increase or reduce
- Some dramatic variations



Structure of this talk

Well that was unsatisfying...

There's a *lot* of work just to get started

- Far more than we anticipated
- Necessary experimental computational infrastructure

The groundwork is essential, though

- Understand the data and the techniques
- Have a properly-tested codebase, starting with small toy cases and working up to realistic scale
- (A surprising number of bugs just don't appear on small cases)
- Make everything reproducible, and ideally entirely automated

Next steps

We've talked today about the start of a journey

- An experimental framework with some initial software infrastructure
- The challenges we encountered in practice

We can now hopefully move on to the interesting stuff

- Can we identify the most significant nodes, for removal and error? The ones that maximise divergence?
- What is the minimum set of sensors for a given quality of interpolation?
- We hypothesise that these might be determined by structures within the tensor

References

S. Dobson, M. Golfarelli, S. Graziani, and S. Rizzi. A reference architecture and model for sensor data warehousing. *IEEE Sensors Journal*, 18, 2018. doi: 10.1109/JSEN.2018.2861327.

V. Keller, M. Tanguy, I. Prosdocimi, J. Terry, O. Hitt, S. Cole, M. Fry, and D. Morris. CEH-GEAR: 1km resolution daily and monthly areal rainfall estimates for the UK for hydrological and other applications. *Earth System Science Data*, 7:143–155, 2015. doi: 10.5194/essd-7-143-2015.

D. Pianini, S. Dobson, and M. Viroli. Self-stabilising target counting in wireless sensor networks using Euler integration. In *Proceedings of the Eleventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'17)*, pages 11–20, September 2017. doi: 10.1109/SASO.2017.10.

Unidata. Network Common Data Format (NetCDF). Technical report, University Corporation for Atomospheric Research (UCAR), 2019. URL http://doi.org/10.5065/D6H70CW6.