

Calibration

+ modules

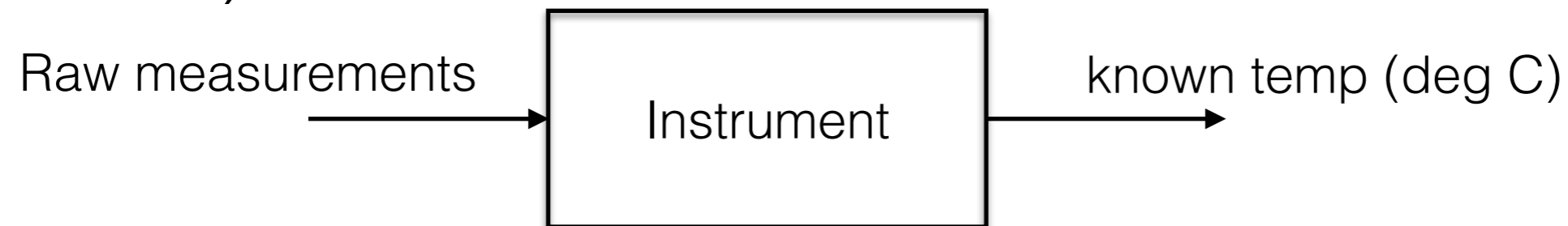
Calibration

Calibration is the act of comparing **measured** values under test to a set of **known** values.

- Calibration assumes that the **measured** values depend only on the **known** value.
- Calibrating allows future measurements to be transformed into unknown quantities of interest.

Temperature measurement

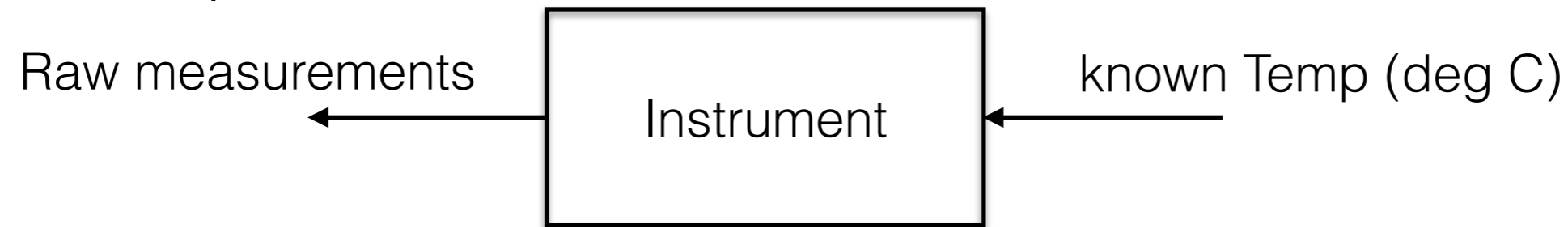
- **Raw measurements:** Resistance, voltage or digital number.
- **Unknown values:** Water temp in deg C (or Kelvin).



$f(x)$ is known

Temperature calibration

- **Raw measurements:** Here we measure resistance, voltage or digital number.
- **Known values:** Water temperature in deg C (or kelvin).



$f(x)$ = is not known

How to compute $f(x)$

1. Choose a function that makes sense. If instrument response is linear, choose a line.

2. $y = mx + b$

3. $T_{\text{Actual}} = mT_{\text{Arduino}} + b$. This is our $f(x)$

4. Another option is the Steinhart-Hart equation:

$$\frac{1}{T} = A + B \ln(R) + C \ln(R)^3$$

Steinhart-Hart Equation

sometimes called fit parameters

$$\frac{1}{T} = A + B \ln(R) + C \ln(R)^3$$

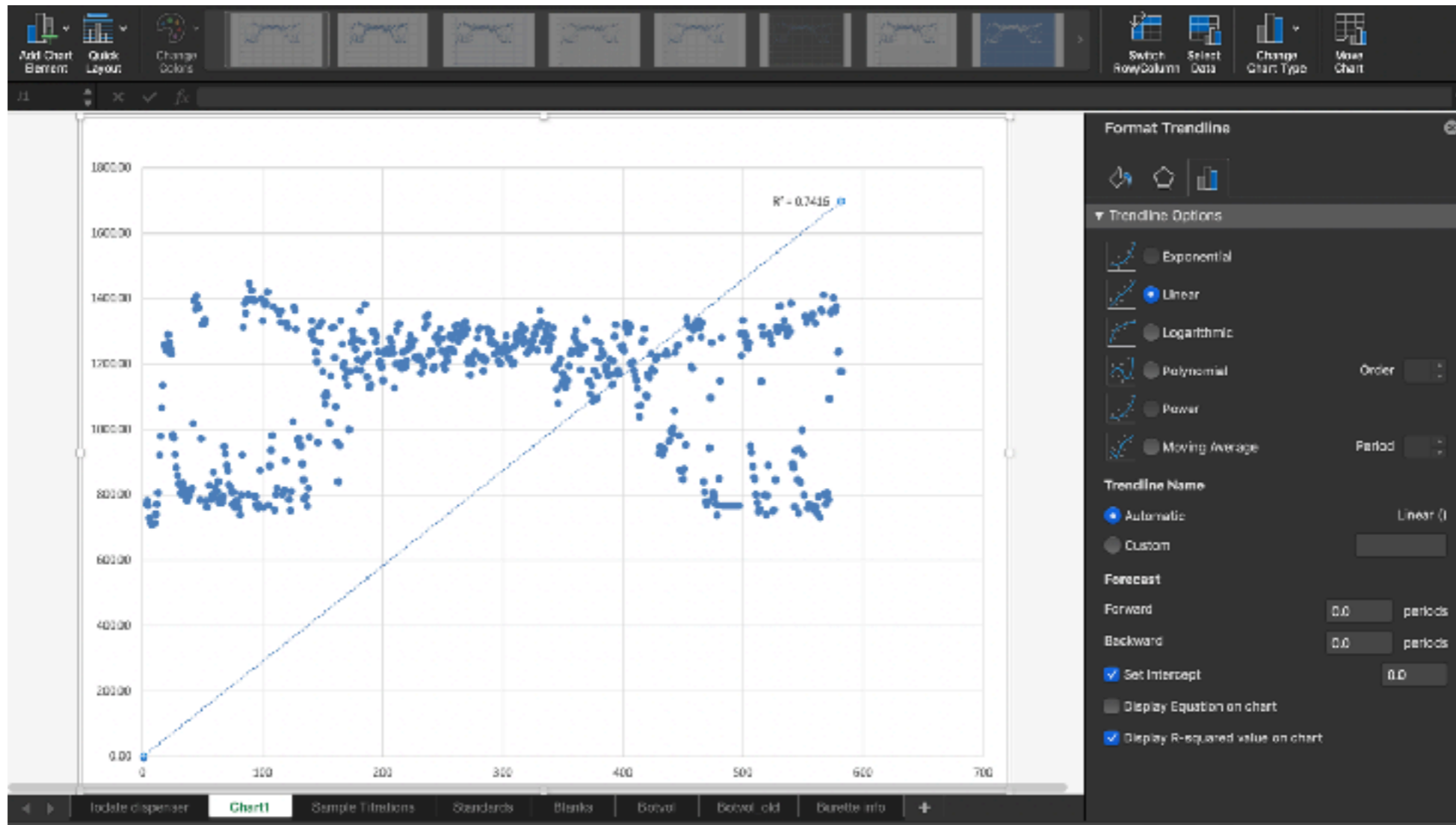
- In this case our function is $f(R)$, Resistance.
- S-H is a polynomial of form $a + bx + cx^2 + dx^3 + zx^n \dots$
- For at-home measurements, we assumed A,B,C were known. But we want $f(R)$ to be accurate, so we calibrate.
- For the calibration, we know T and R. Then we **solve for A,B,C using curve fitting.**
- S-H is linear with respect to the parameters, A, B, C.

Error

1. We want $T_{\text{Actual}} - f(\text{Resistance}) = 0$.
2. In reality, $T_{\text{Actual}} - f(\text{Resistance}) = \text{err}$.
3. Error leads to a misfit between the actual Temp and the calibration. We seek to minimize the misfit.
4. We can make many measurements of T_{Bath} and T_{Model} . This allows us to solve the calibration equation in a least-square sense.

General curve fitting

Are you familiar with this?



If so, you have already done curve fitting.

General curve fitting

- Mathematical basis of regression analysis (Excel makes this easy).
- Jumping off point for the field of Inverse Modeling.
- Extremely versatile tool, because many nonlinear equations can be linearized.
- How do we get the best fit? Minimize the model-data difference (sometimes called Chi² minimization)

$$\chi^2 = \left| \underline{\mathbf{A}} \cdot \mathbf{x} - \hat{y}_{obs} \right|^2$$

General linear least squares

- A – the “design” matrix; with $m > n$.
Alternately called the “model”.

$$\underline{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix}$$

- x – the parameters
- $$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

- If a linear fit,

$$x = \begin{bmatrix} m & b \end{bmatrix}^T$$

$$\chi^2 = |\underline{A} \cdot x - \hat{y}_{obs}|^2$$

- Note: Elements in design matrix don't have to be linear – only model has to be linear w.r.t. the parameters (x).

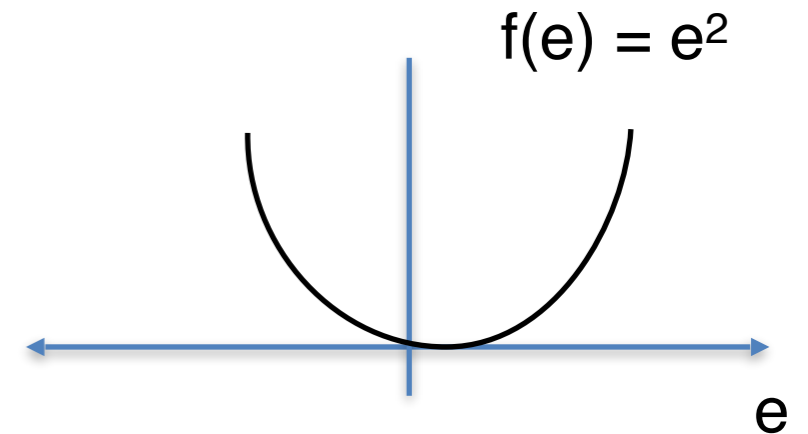
$$\underline{A} = \begin{bmatrix} a^3 & \cdots & \cos(\theta) \\ \vdots & \ddots & \vdots \\ 1/a & \cdots & \sinh(\phi) \end{bmatrix}$$

General linear least squares

- Procedure: Find minimum in model data difference.

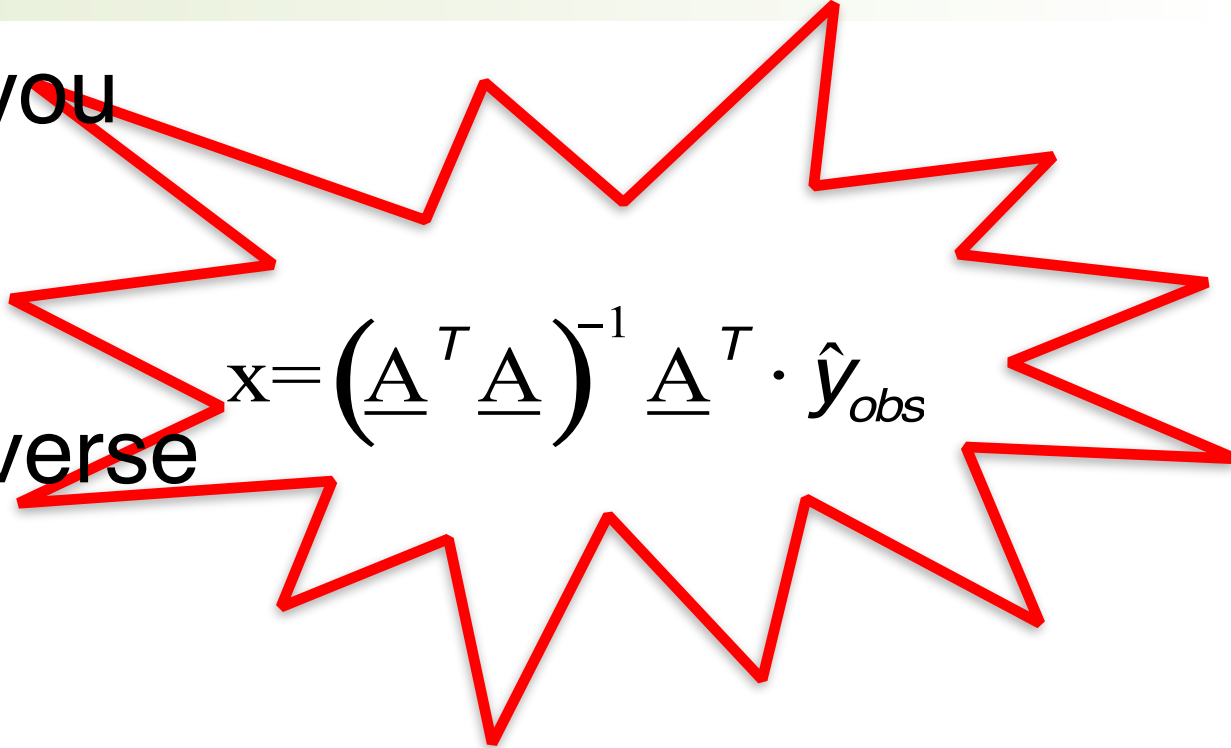
$$\frac{\partial \chi^2}{\partial \mathbf{x}} = 0 \quad \chi^2 = \left| \underline{\mathbf{A}} \cdot \mathbf{x} - \hat{y}_{obs} \right|^2$$

- This is guaranteed to find a minimum, b.c. the function is a quadratic.



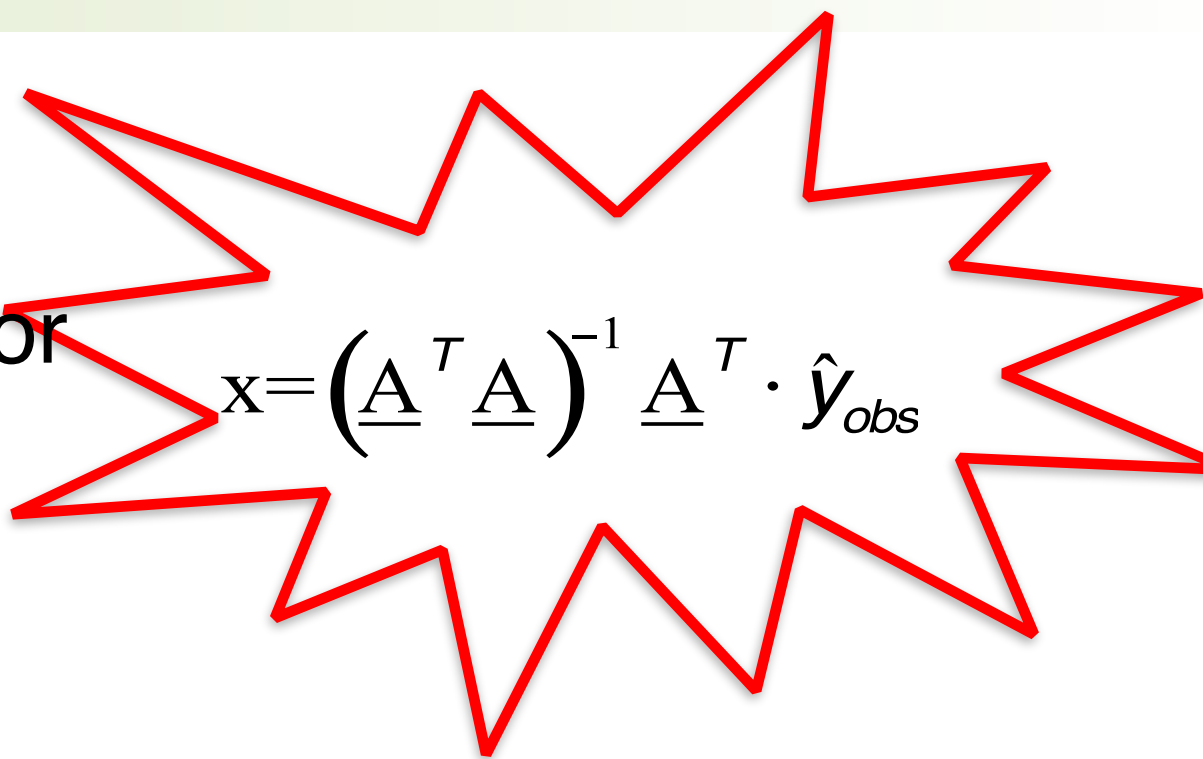
General linear least squares

- This is what Excel does when you choose “add a trendline” ...
- This is also the beginning of inverse modeling.
- All you need is a design matrix that’s linear in its parameters, Python does the rest.


$$\mathbf{x} = \left(\underline{\mathbf{A}}^T \underline{\mathbf{A}} \right)^{-1} \underline{\mathbf{A}}^T \cdot \hat{\mathbf{y}}_{obs}$$

General linear least squares

- IMPORTANT NOTES:
- The solution is only truly an error minimization when $m > n$, ie rows > columns; #equations > #unknowns.
- Solution is exactly determined when $m = n$, but no minimization of error.
- The inverse(A) can be singular and hard to calculate if basis functions aren't truly independent


$$\mathbf{x} = \left(\underline{\mathbf{A}}^T \underline{\mathbf{A}} \right)^{-1} \underline{\mathbf{A}}^T \cdot \hat{\mathbf{y}}_{obs}$$

Checking fit quality

- Coefficient of Determination, R^2 – related to the sum of the square of the residuals

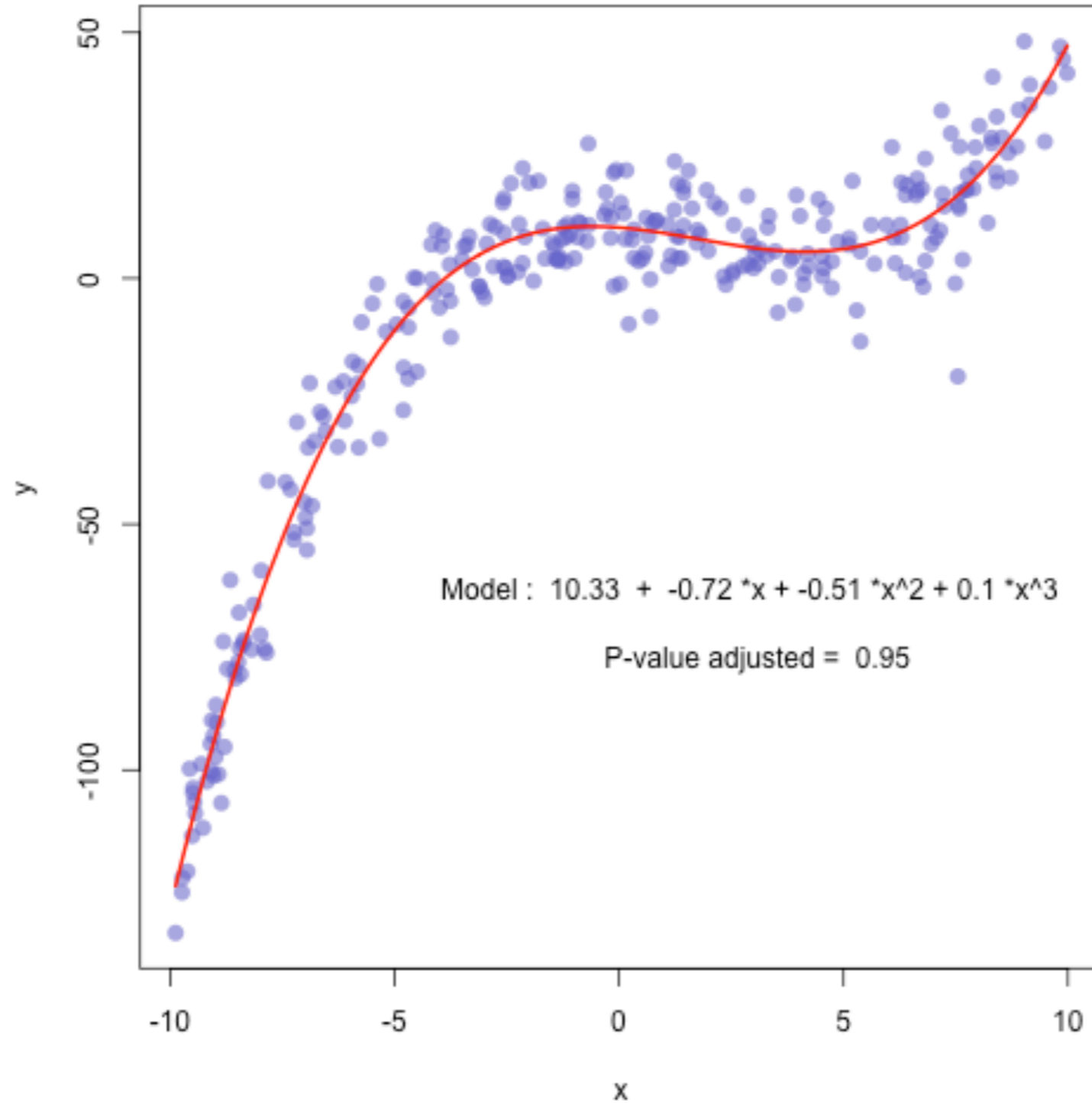
$$\text{residuals} = \left(y_{obs}^i - A \cdot x \right) = \left(y_{obs}^i - y_{mod}^i \right)$$

$$R^2 = 1 - \frac{\sum_{i=1}^N \left(y_{obs}^i - y_{mod}^i \right)^2}{\sum_{i=1}^N \left(y_{obs}^i - \overline{y_{obs}} \right)^2}$$

Caveats

- **We are measuring resistance.**
- Soil also has varying electrical conductivity, so our measurements will be partly influenced by the properties of the soil.
- **Other influences:** Distance between probes, temperature, salinity of the water, etc.

Linear calibration



Functions or 'modules' in Python

Modules are like black boxes (the user doesn't know how the magic happens)



Functions or 'modules' in Python

```
def function_name(Input1,Input2):
```

```
    """
```

```
    doc string (or header)
```

```
    """
```

```
    Output1 = Input1 + Input2    # Here you put the operations you want to perform.
```

```
    Tinv = A +B log(R) + C log(R)^3    # Can be a function or any other task python does.
```

```
    return Input1, Output1 # Here you return the arguments you want to output.
```

Modules are limited in scope

- All the variables created ***inside*** the module disappear after the module is executed.
- The only variables that survive are those that are “output” on the “return” line.
- If an input variable is modified and then output, it will be changed permanently.