# Workshop
# Data processing

Experimental Project

Master Engineering Physics

Tristan Gilet
Microfluidics Lab, ULiège

# Contents

**Learn with an example**

1. Image processing: segmentation, filtering, measurements

2. Data processing: selection, model fitting

3. Statistical analysis: descriptive statistics, t-test, ANOVA

Practice with your preliminary data

# An example

Captured at 2000fps
Scale unknown

**Final goal**

Analyze the image, and infer the radius of both drops

# An example

Captured at 2000fps
Scale unknown

**Final goal**

Analyze the image, and infer the radius of both drops

**Strategy**

1. Detect the drop position vs. time by image processing
   ➔ Image J: https://imagej.nih.gov/ij/
   (all functions ➔ equiv. in Matlab)

2. Fit a theoretical model of free fall on the data, and
   deduce the scale from the acceleration of gravity
   ➔ Matlab

# Image processing

Results

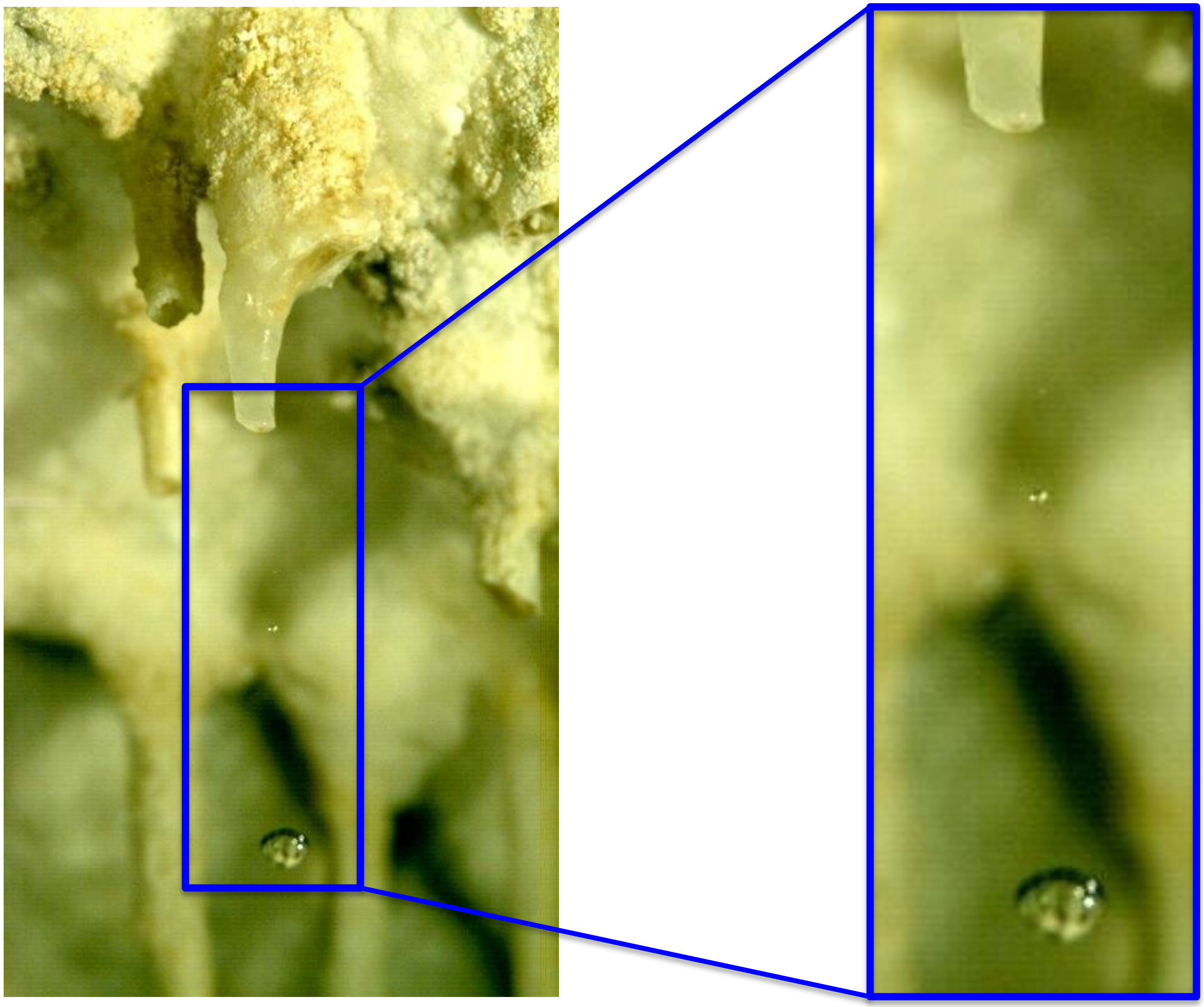
| | Area | X | Y | BX | BY | Width | Height | Circ. | Slice | AR | Round | Solidity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 689 | 29 | 74.3 | 207.4 | 71 | 205 | 7 | 5 | 1.0 | 261 | 1.4 | 0.7 | 0.9 |
| 690 | 1018 | 83.1 | 380.1 | 62 | 363 | 40 | 36 | 0.7 | 261 | 1.2 | 0.8 | 0.9 |
| 691 | 41 | 57.9 | 52.0 | 53 | 49 | 10 | 6 | 0.8 | 262 | 2.4 | 0.4 | 0.9 |
| 692 | 24 | 74.5 | 209.2 | 72 | 206 | 5 | 6 | 1.0 | 262 | 1.1 | 0.9 | 0.9 |
| 693 | 1014 | 82.8 | 383.4 | 62 | 367 | 39 | 35 | 0.8 | 262 | 1.2 | 0.9 | 0.9 |
| 694 | 46 | 58.1 | 52.1 | 53 | 49 | 11 | 6 | 0.8 | 263 | 2.2 | 0.5 | 0.9 |
| 695 | 20 | 74.4 | 211.4 | 72 | 209 | 5 | 5 | 1.0 | 263 | 1.1 | 0.9 | 0.9 |
| 696 | 1003 | 83.0 | 386.7 | 62 | 370 | 41 | 35 | 0.8 | 263 | 1.2 | 0.9 | 0.9 |
| 697 | 46 | 58.0 | 52.0 | 53 | 49 | 10 | 6 | 0.9 | 264 | 2.0 | 0.5 | 0.9 |
| 698 | 32 | 73.8 | 213.2 | 70 | 210 | 7 | 6 | 1.0 | 264 | 1.2 | 0.9 | 0.9 |
| 699 | 991 | 83.2 | 389.7 | 62 | 374 | 41 | 34 | 0.8 | 264 | 1.2 | 0.8 | 0.9 |

**Goal** Detect objects of interest, then calculate their properties (position, size, shape, etc.)

**Strategy**

1. Image preparation
2. Background removal to highlight the objects
3. Segmentation to select the objects
4. Filtering to remove noise
5. Measurement of object properties + storage

# Image preparation



e.g. Rotation, cropping, resizing
(./Image/)

# Background generation

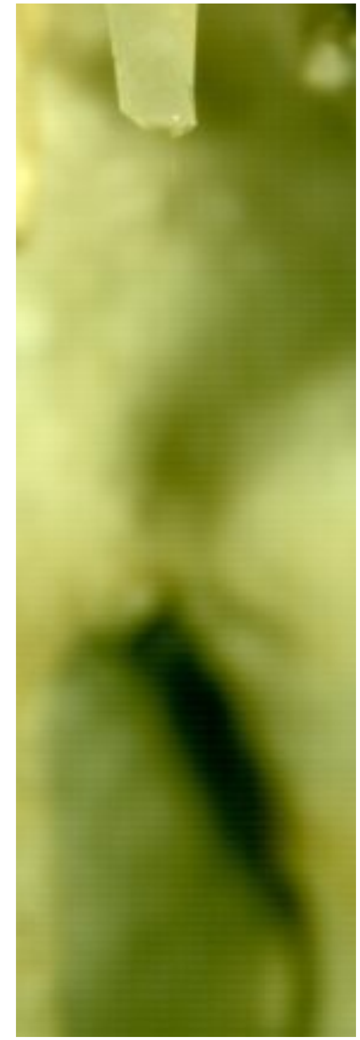
./Image/Duplicate

./Image/Stack/Z-project

$1^{st}$ frame

Last frame

Time average

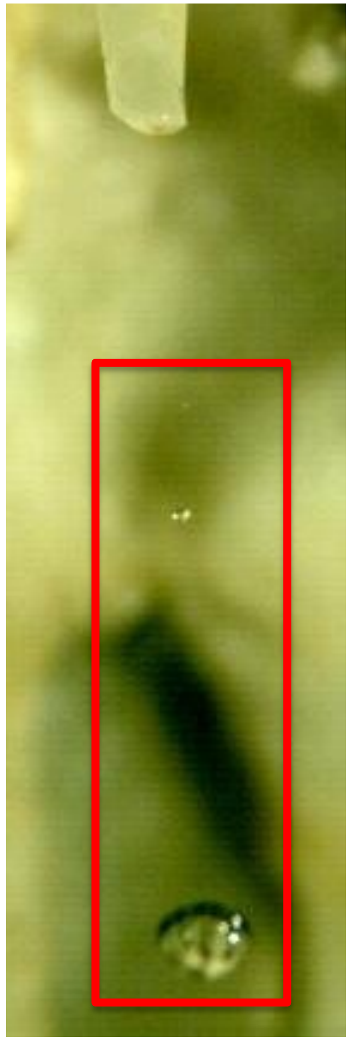


Time median

# Background generation

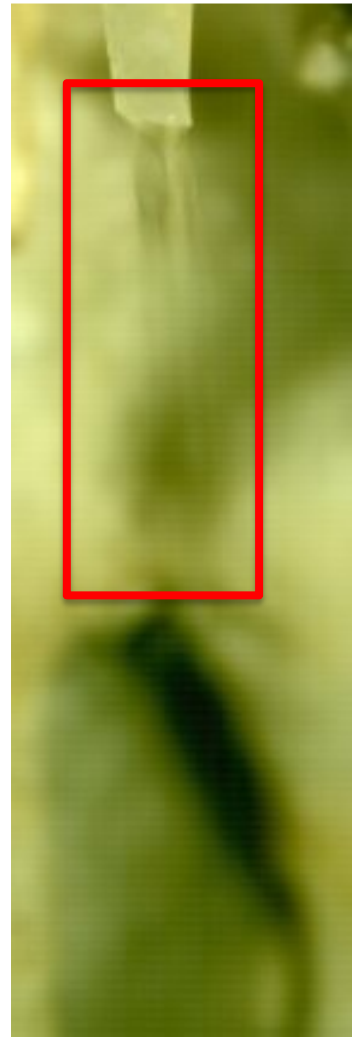
./Image/Duplicate

1st frame



Last frame

./Image/Stack/Z-project



Time average

Time median

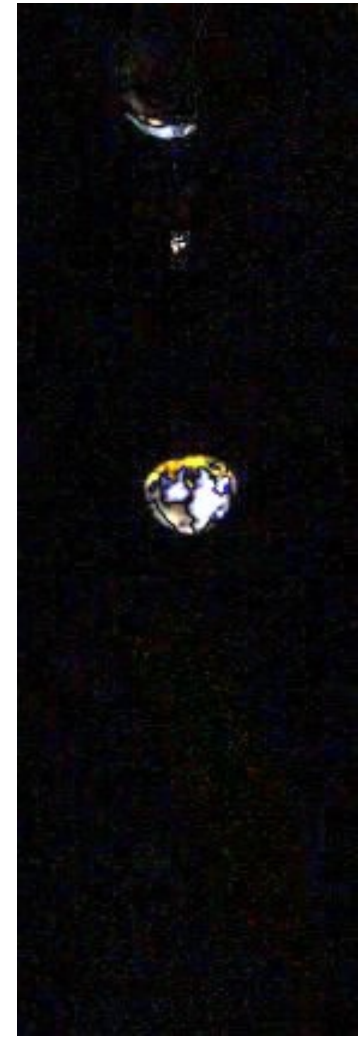
# Background subtraction

./Process/ImageCalculator

Original

Background

Subtraction



Difference (➔ abs)

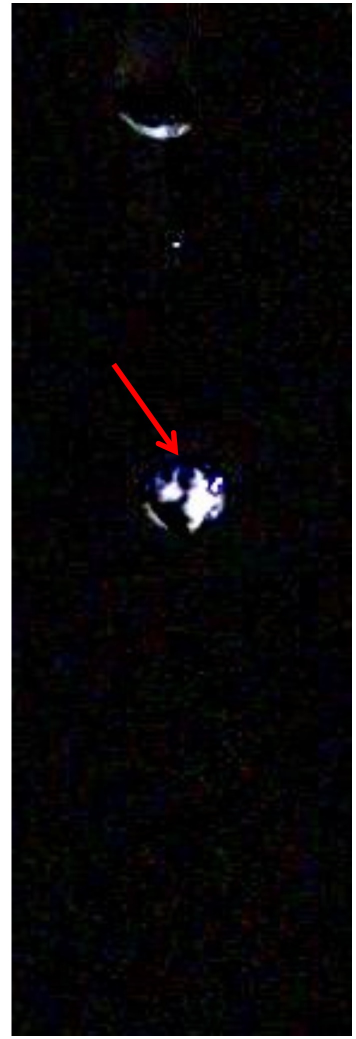
Increase contrast ➔ ./Image/Adjust/BrightnessContrast: Max = 64

# Background subtraction

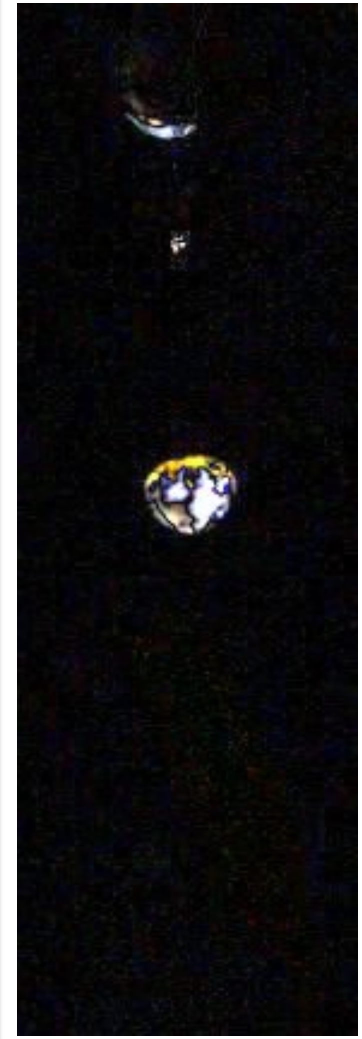
./Process/ImageCalculator

Original

Background



Subtraction



Difference (➔ abs)

Increase contrast ➔ ./Image/Adjust/BrightnessContrast: Max = 64

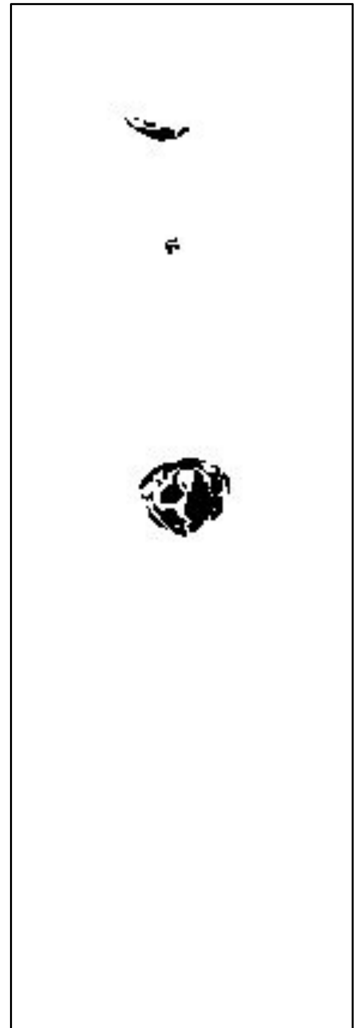
# Segmentation

./Image/Type/8-bit

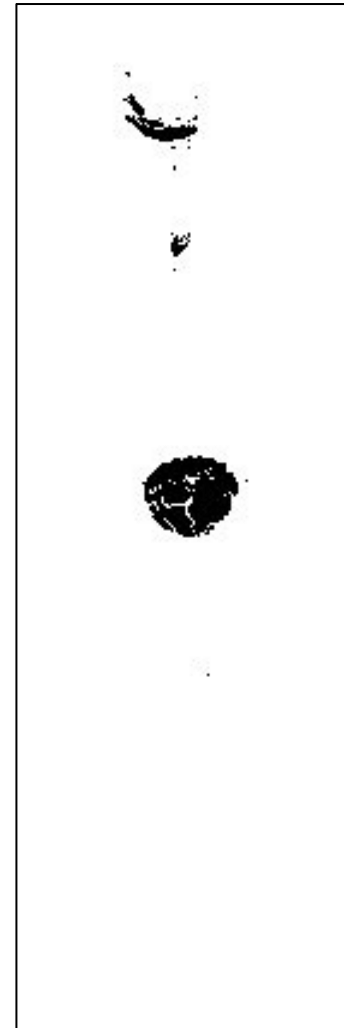
./Image/Adjust/Threshold

RGB
3x [0, 255]

Gray level
[0, 255]



Threshold
[100, 255]



Threshold
[40, 255]

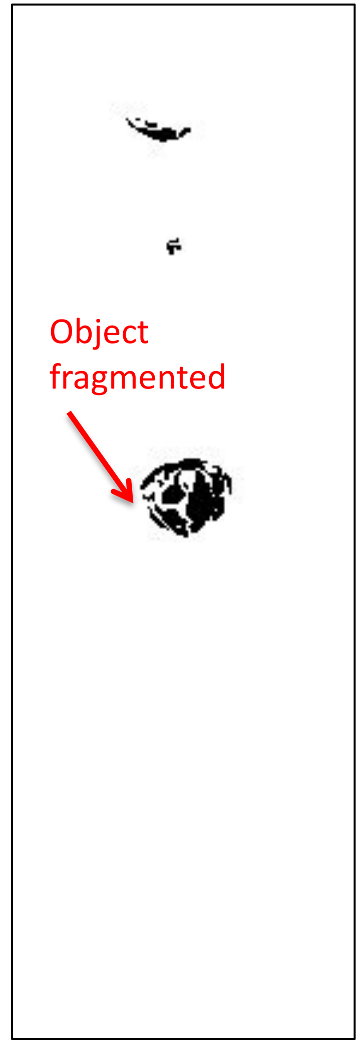
# Segmentation

./Image/Type/8-bit

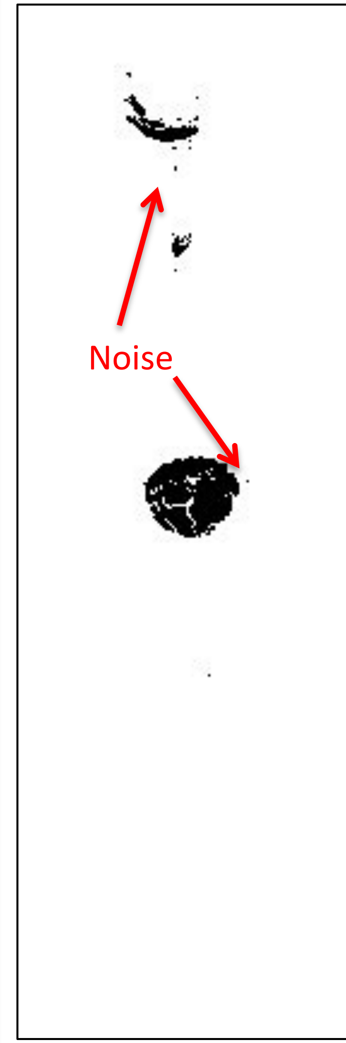
./Image/Adjust/Threshold

RGB
3x [0, 255]

Gray level
[0, 255]




Threshold
[100, 255]

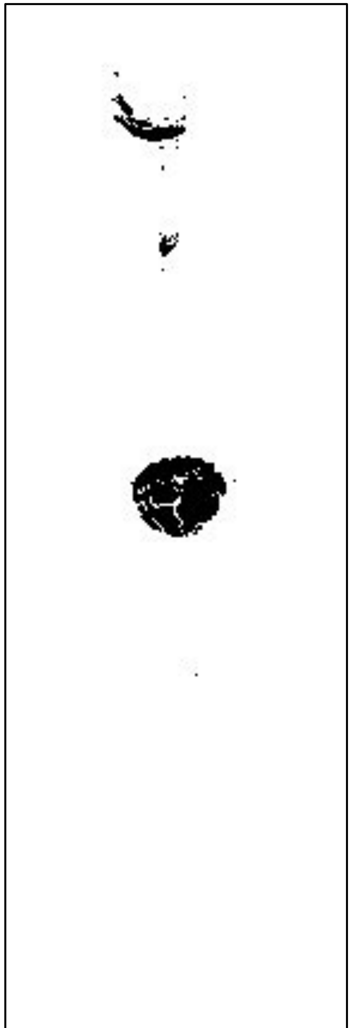



Threshold
[40, 255]

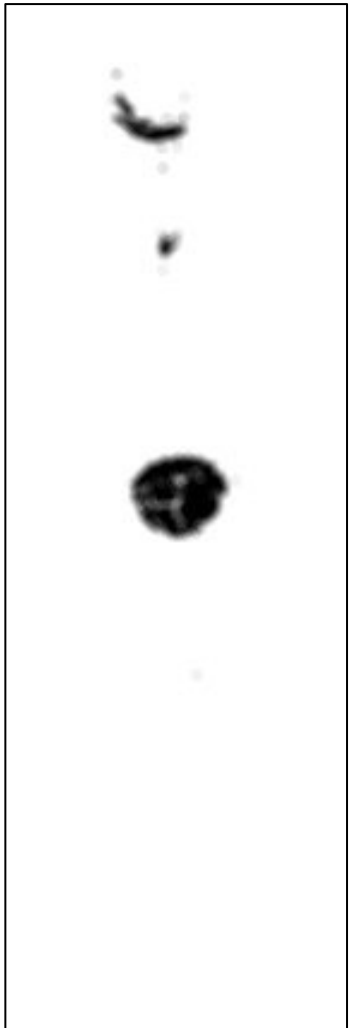
# Filtering

= convolution of the image matrix by a filtering kernel

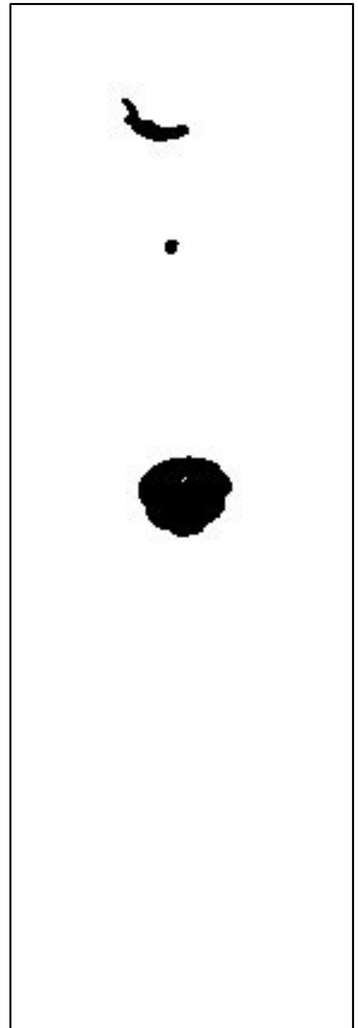
./Process/Filters/



Original B&W
(noisy)



Mean – R=2
➔ gray



Median – R=2
➔ B&W

Kernel – R=2

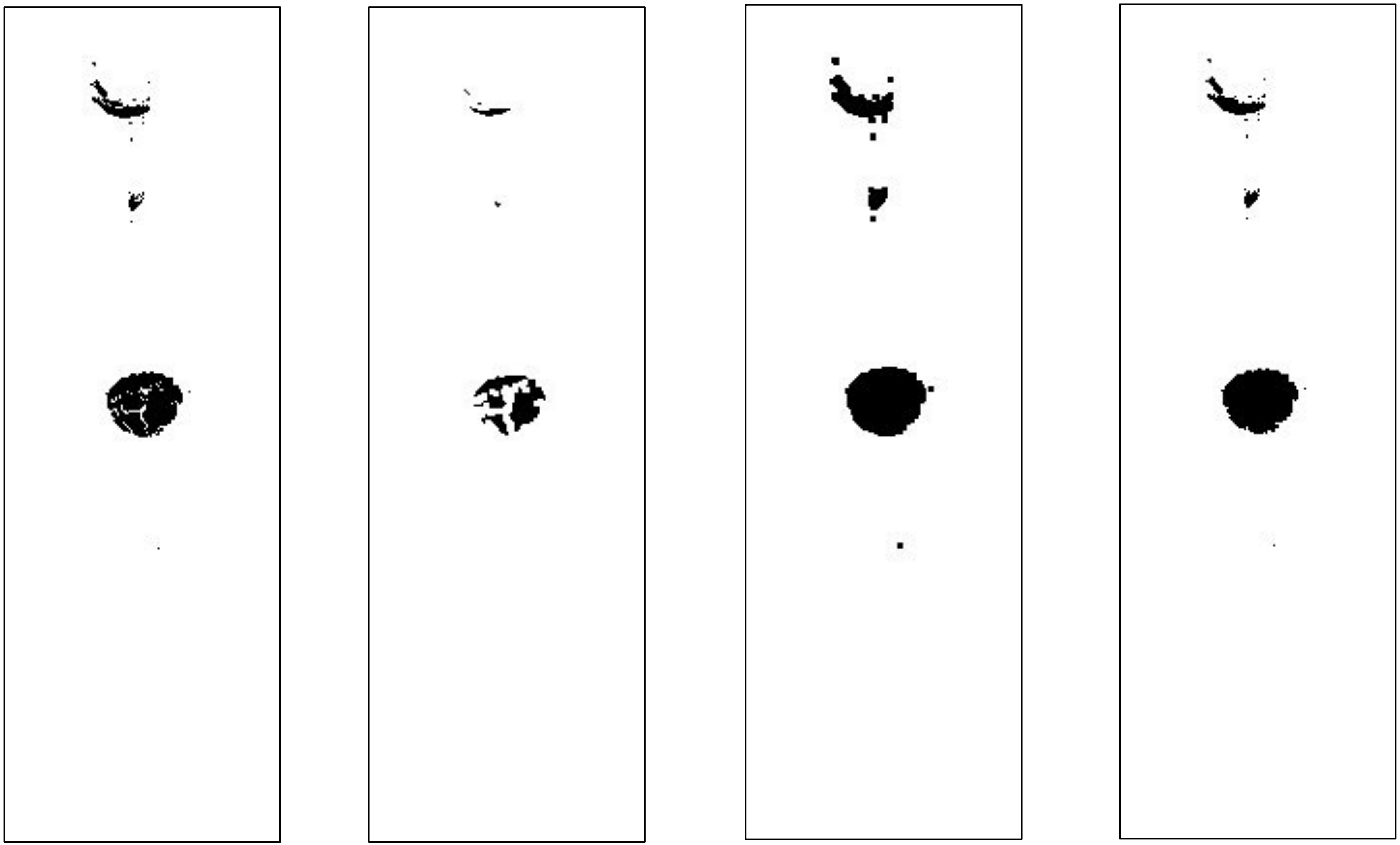
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

➔ Each pixel is replaced by the mean / median / … of the 21 neighboring pixels

# Filtering – morphological operations

= special convolutions (B&W ➔ B&W)

./Process/Binary/



Original B&W (noisy)

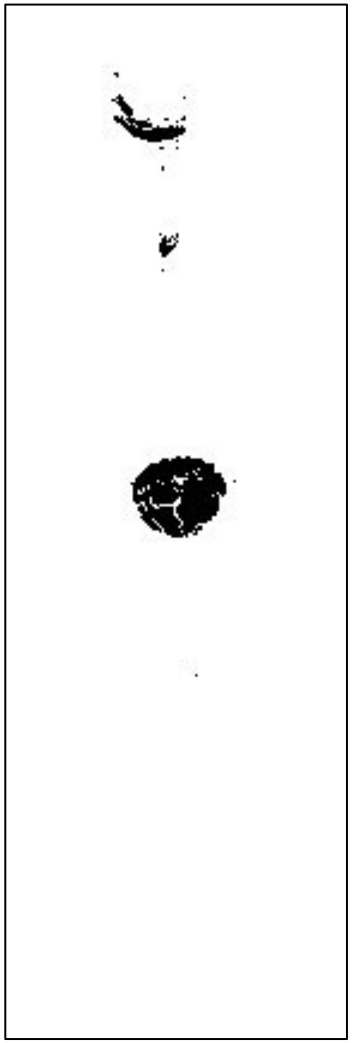
Erode = remove one layer

Dilate = add one layer

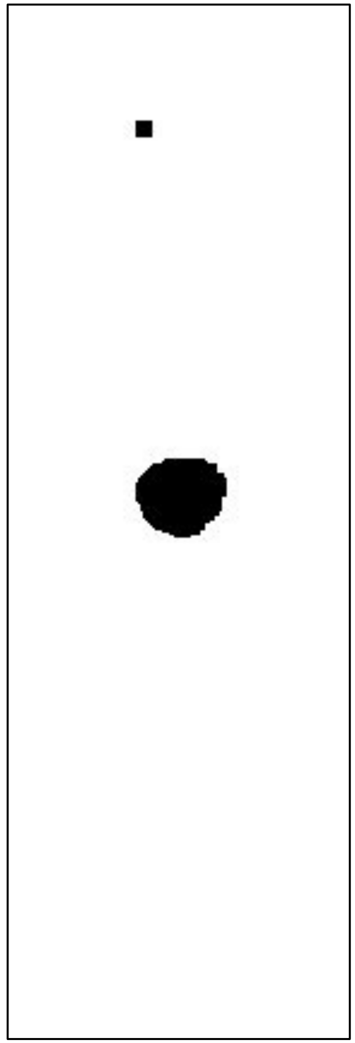
Fill holes

# Filtering – combinations

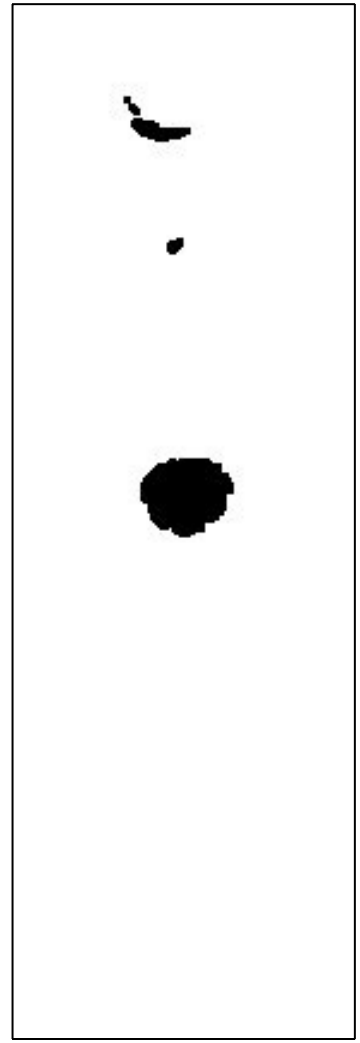
Succession of filters (Median, Erode, Dilate, Fill holes)      Size +/- conserved if #(E) = #(D)



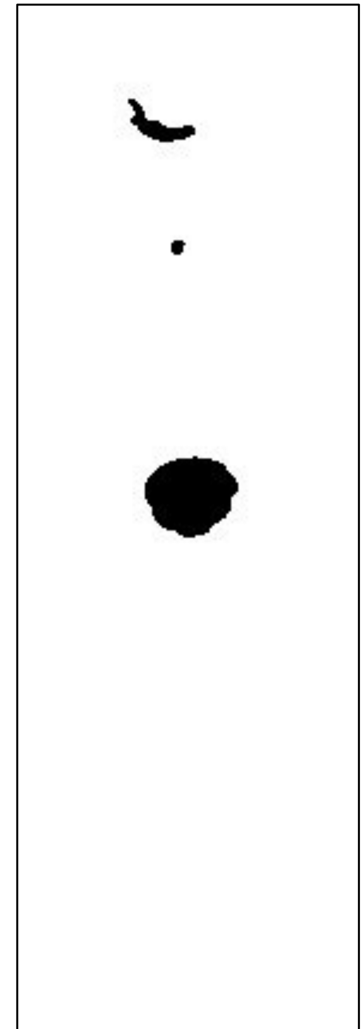
Original B&W (noisy)



DFEEEEDDD



FED



MF

# Measurements

./Analyze/SetMeasurements

➔ e.g. Select Area, Centroid, Bounding rectangle, Shape descriptors, Stack position

./Analyze/AnalyzeParticles

Additional noise removal ➔ Size: 10-inf $\text{pixel}^2$

Possible constraint on circularity $C = 4\pi \frac{\text{Area}}{\text{Perimeter}^2}$

Results

| | Area | X | Y | BX | BY | Width | Height | Circ. | Slice | AR | Round | Solidity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 689 | 29 | 74.3 | 207.4 | 71 | 205 | 7 | 5 | 1.0 | 261 | 1.4 | 0.7 | 0.9 |
| 690 | 1018 | 83.1 | 380.1 | 62 | 363 | 40 | 36 | 0.7 | 261 | 1.2 | 0.8 | 0.9 |
| 691 | 41 | 57.9 | 52.0 | 53 | 49 | 10 | 6 | 0.8 | 262 | 2.4 | 0.4 | 0.9 |
| 692 | 24 | 74.5 | 209.2 | 72 | 206 | 5 | 6 | 1.0 | 262 | 1.1 | 0.9 | 0.9 |
| 693 | 1014 | 82.8 | 383.4 | 62 | 367 | 39 | 35 | 0.8 | 262 | 1.2 | 0.9 | 0.9 |
| 694 | 46 | 58.1 | 52.1 | 53 | 49 | 11 | 6 | 0.8 | 263 | 2.2 | 0.5 | 0.9 |
| 695 | 20 | 74.4 | 211.4 | 72 | 209 | 5 | 5 | 1.0 | 263 | 1.1 | 0.9 | 0.9 |
| 696 | 1003 | 83.0 | 386.7 | 62 | 370 | 41 | 35 | 0.8 | 263 | 1.2 | 0.9 | 0.9 |
| 697 | 46 | 58.0 | 52.0 | 53 | 49 | 10 | 6 | 0.9 | 264 | 2.0 | 0.5 | 0.9 |
| 698 | 32 | 73.8 | 213.2 | 70 | 210 | 7 | 6 | 1.0 | 264 | 1.2 | 0.9 | 0.9 |
| 699 | 991 | 83.2 | 389.7 | 62 | 374 | 41 | 34 | 0.8 | 264 | 1.2 | 0.8 | 0.9 |

➔ Save as .txt file, to be imported in Matlab for further processing

# Contents

**Learn with an example**

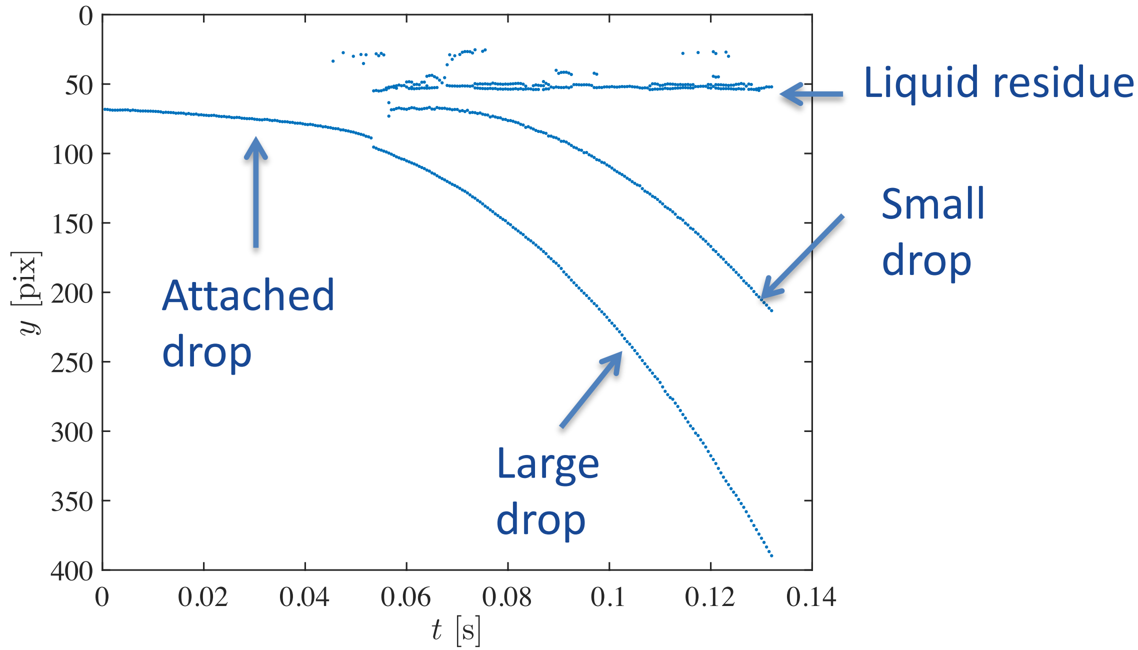
1. Image processing: segmentation, filtering, measurements
2. **Data processing: selection, model fitting**
3. Statistical analysis: descriptive statistics, t-test, ANOVA

Practice with your preliminary data

# Data import in Matlab

```
data=dlmread('Results.txt','\t',1,0);
A=data(:,2);                          % Area [pix^2]
y=data(:,4);                          % Vertical position [pix]
W=data(:,7);                          % Width [pix]
H=data(:,8);                          % Height [pix]
t=data(:,10)/2000;                    % Time [s]
```

Sampling freq.




```
figure(1);
plot(t,y,'.');
xlabel('$t$ [s]','fontsize',FS,'fontname','Times','interpreter','latex');
ylabel('$y$ [pix]','fontsize',FS,'fontname','Times','interpreter','latex');
set(gca,'fontsize',FS-2,'YDir','reverse','fontname','Times');
print -depsc ./Figures/Fig1_ty.eps;
```

# Data selection / partition

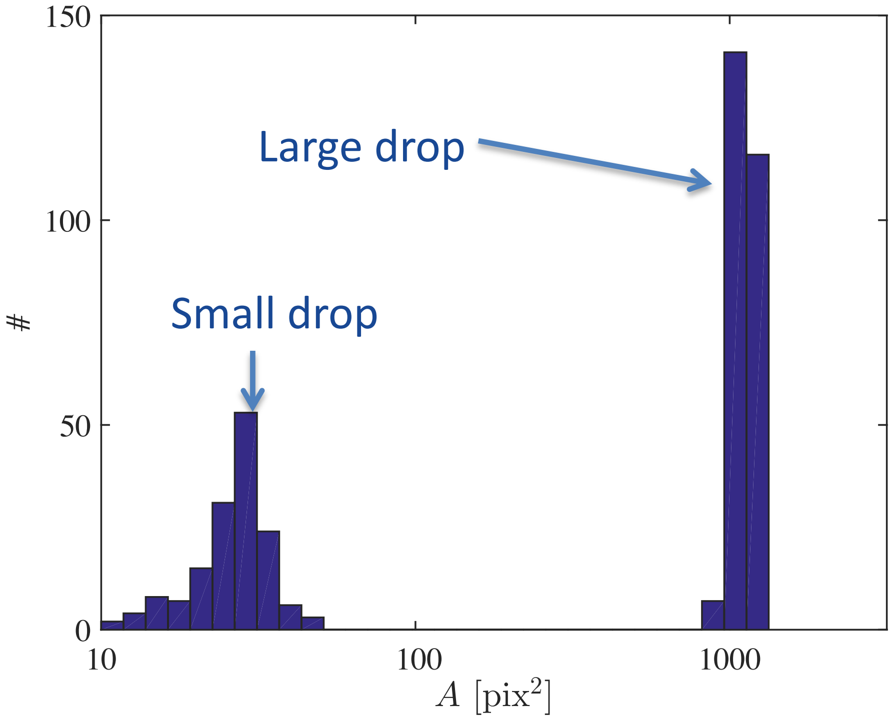
f0=find(y>60); → Exclude the liquid residue, based on vertical position

# Data selection / partition

`f0=find(y>60);` → Exclude the liquid residue, based on vertical position

Then check size histogram to distinguish and separate both drops

```
hist(log10(A(f0)),30);
```

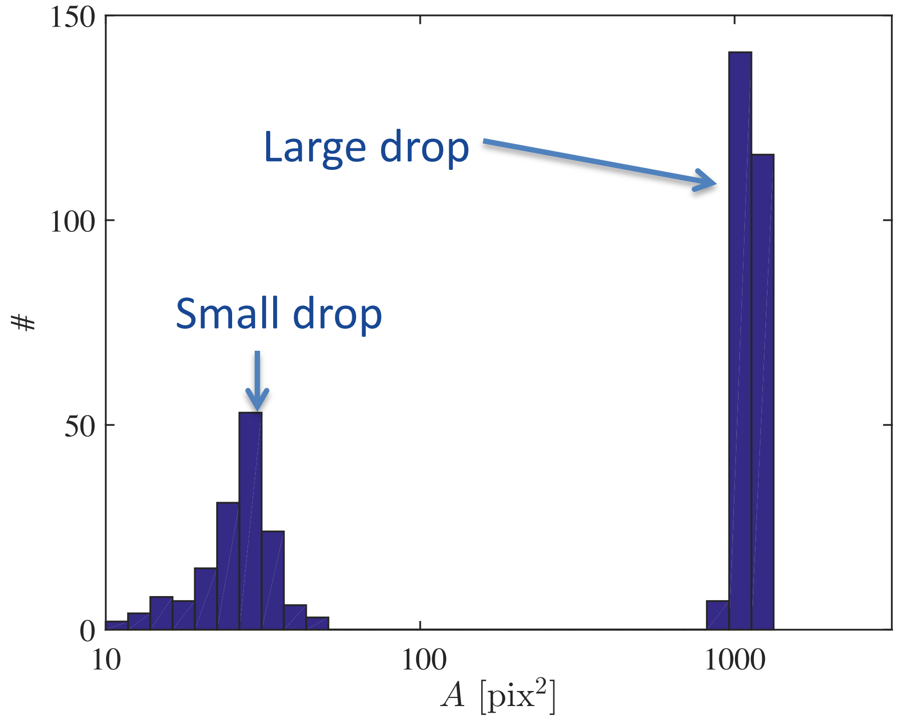


# Data selection / partition

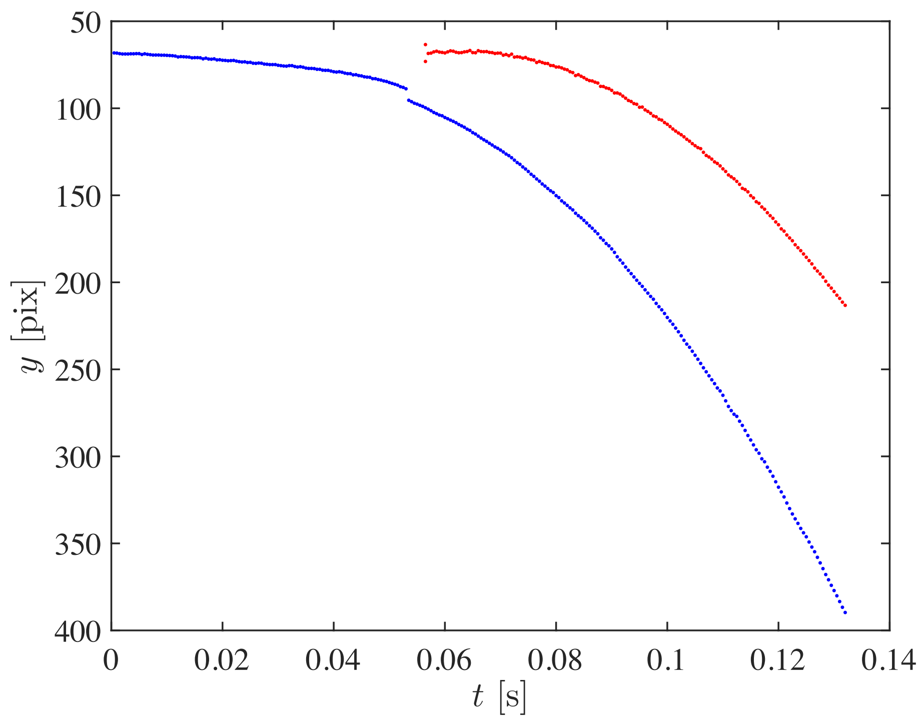
```
f0=find(y>60);
```
→ Exclude the liquid residue, based on vertical position

Then check size histogram to distinguish and separate both drops

```
hist(log10(A(f0)),30);
```




```
f1=find(A>100); f01=intersect(f0,f1);
f2=find(A<100); f02=intersect(f0,f2);
plot(t(f01),y(f01),'b.'); plot(t(f02),y(f02),'r.');
```




The partition based on area works to distinguish both drops.

➔ Tracking (more complicated) not needed

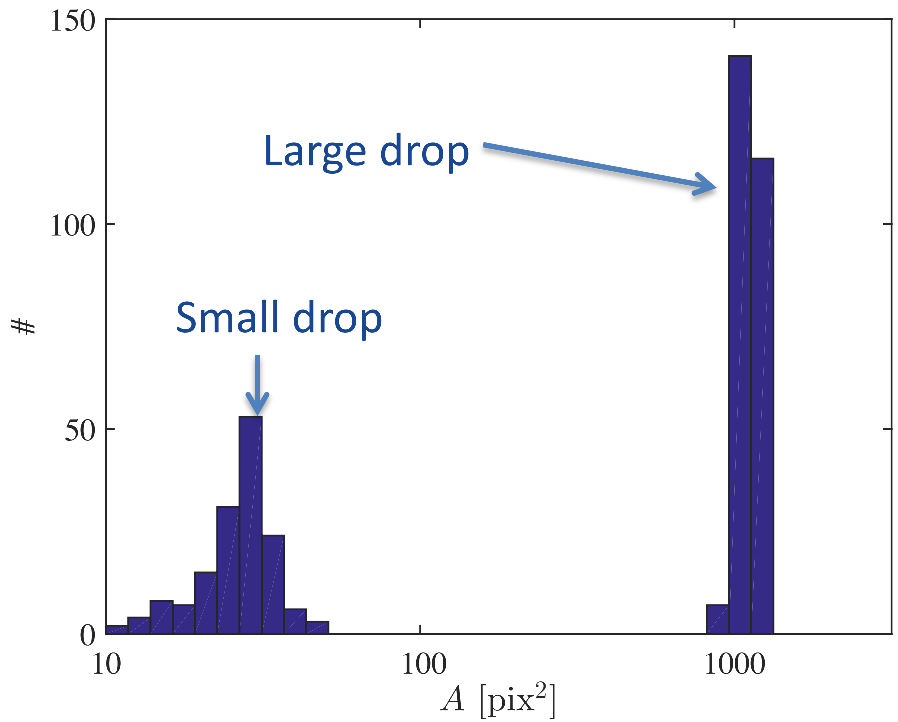
# Data selection / partition

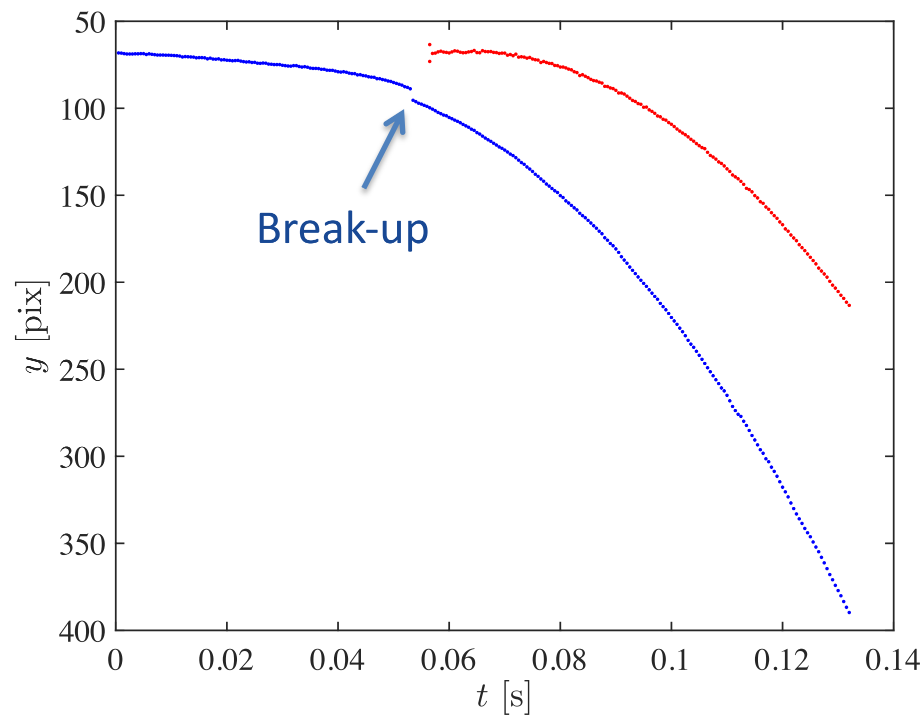
```
f0=find(y>60);
```
→ Exclude the liquid residue, based on vertical position

Then check size histogram to distinguish and separate both drops

```
hist(log10(A(f0)),30);
```

```
f1=find(A>100); f01=intersect(f0,f1);
f2=find(A<100); f02=intersect(f0,f2);
plot(t(f01),y(f01),'b.'); plot(t(f02),y(f02),'r.');
```



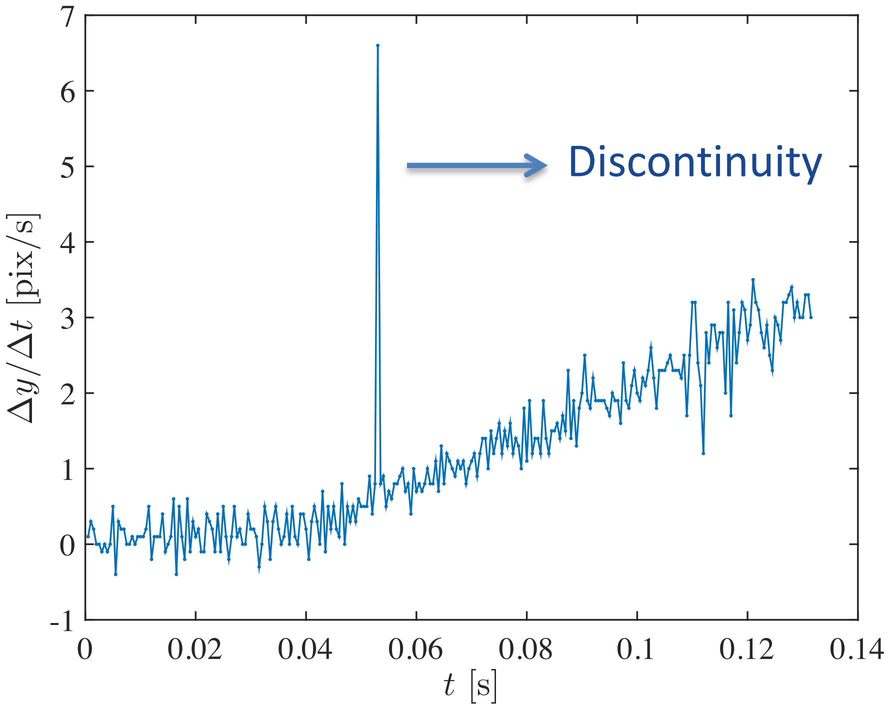




The partition based on area works to distinguish both drops.
➔ Tracking (more complicated) not needed

# Data selection / partition

```
dy=diff(y(f01));
plot(t(f01(1:end-1)),dy,'.-');
f3=find(dy>5);
t0=t(f01(f3));
```

→ Time of break-up

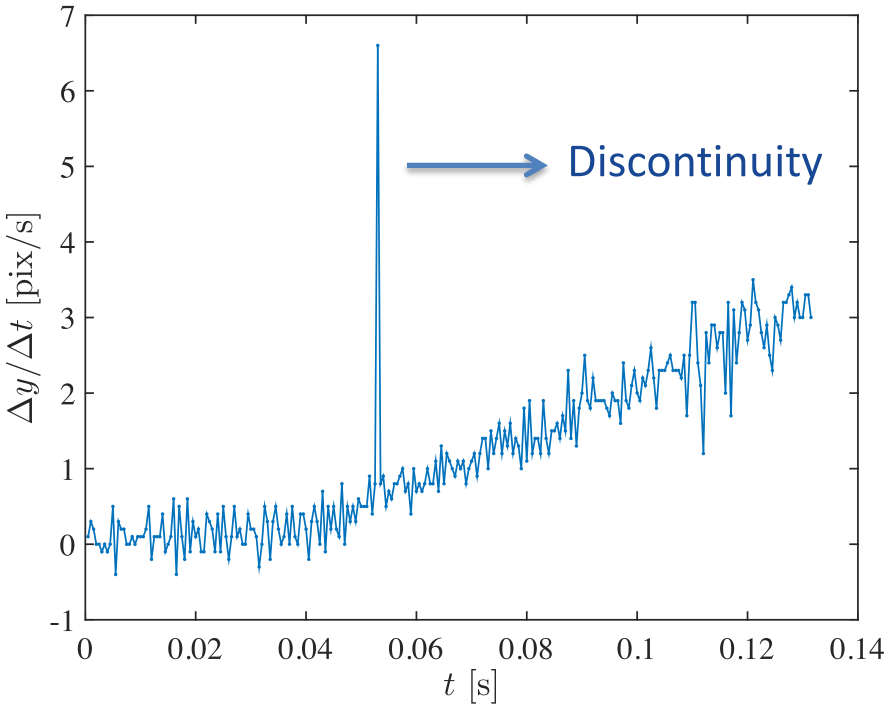


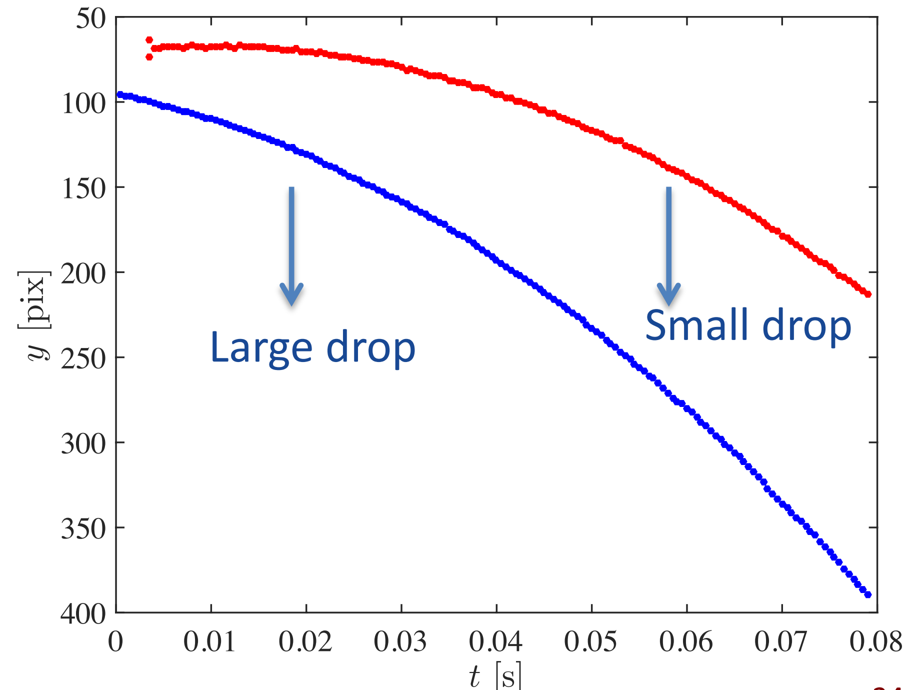
# Data selection / partition

```
dy=diff(y(f01));
plot(t(f01(1:end-1)),dy,'.-');
f3=find(dy>5);
t0=t(f01(f3));
t=t-t0;
f31=f01(t(f01)>0); f32=f02(t(f02)>0);
t1=t(f31); t2=t(f32); y1=y(f31); y2=y(f32);
```

→ Time of break-up (for t0=t(f01(f3));)

→ Free fall selected (for f31=f01(t(f01)>0); f32=f02(t(f02)>0);)



# Acceleration of gravity

Deduced from *y*, through finite differences

$$y_{n+1} = y_n + v_n(\Delta t) + \frac{a_n}{2}(\Delta t)^2$$

$$y_{n-1} = y_n - v_n(\Delta t) + \frac{a_n}{2}(\Delta t)^2$$

$$\Rightarrow a_n = \frac{2(y_{n+1} + y_{n-1} - 2y_n)}{(\Delta t)^2}$$

$y$ [pix]
$a_n$ = acceleration [pix/s$^2$]
$c$ = size [m] of one pixel
$g$ = 9.81 [m/s$^2$]

$$c = \frac{g}{\mathrm{mean}(a_n)}$$

# Acceleration of gravity

Deduced from *y*, through finite differences

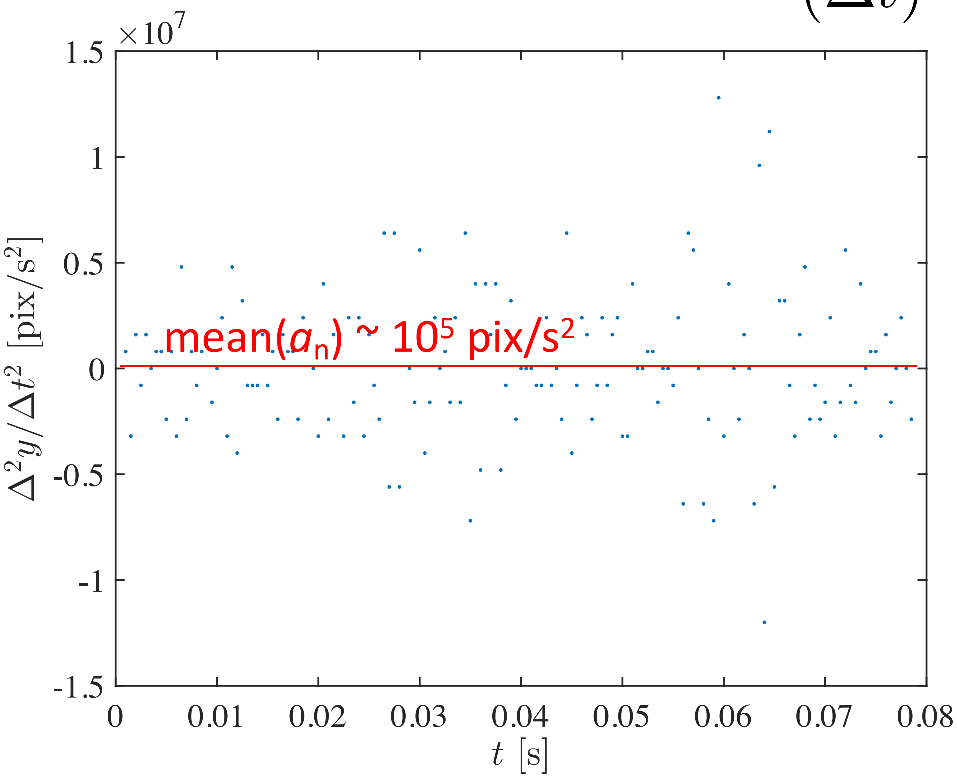
$$y_{n+1} = y_n + v_n(\Delta t) + \frac{a_n}{2}(\Delta t)^2$$

$$y_{n-1} = y_n - v_n(\Delta t) + \frac{a_n}{2}(\Delta t)^2$$

$$\Rightarrow a_n = \frac{2(y_{n+1} + y_{n-1} - 2y_n)}{(\Delta t)^2}$$

*y* [pix]
$a_n$ = acceleration [pix/s$^2$]
*c* = size [m] of one pixel
*g* = 9.81 [m/s$^2$]

$$c = \frac{g}{\text{mean}(a_n)}$$




```
ac=(y1(3:end)+y1(1:end-2)-2*y1(2:end-1))*2./(t1(2:end-1)-t1(1:end-2)).^2;
plot(t1(2:end-1),ac,'.'); hold on;
plot([min(t1),max(t1)],mean(ac)*[1 1],'r');
c=9.81/mean(ac);
```

It yields
*c* = 87 µm/pix, from the large drop
*c* = 29µm/pix, from the small drop

Why ?

# Acceleration of gravity

Deduced from $y$, through finite differences

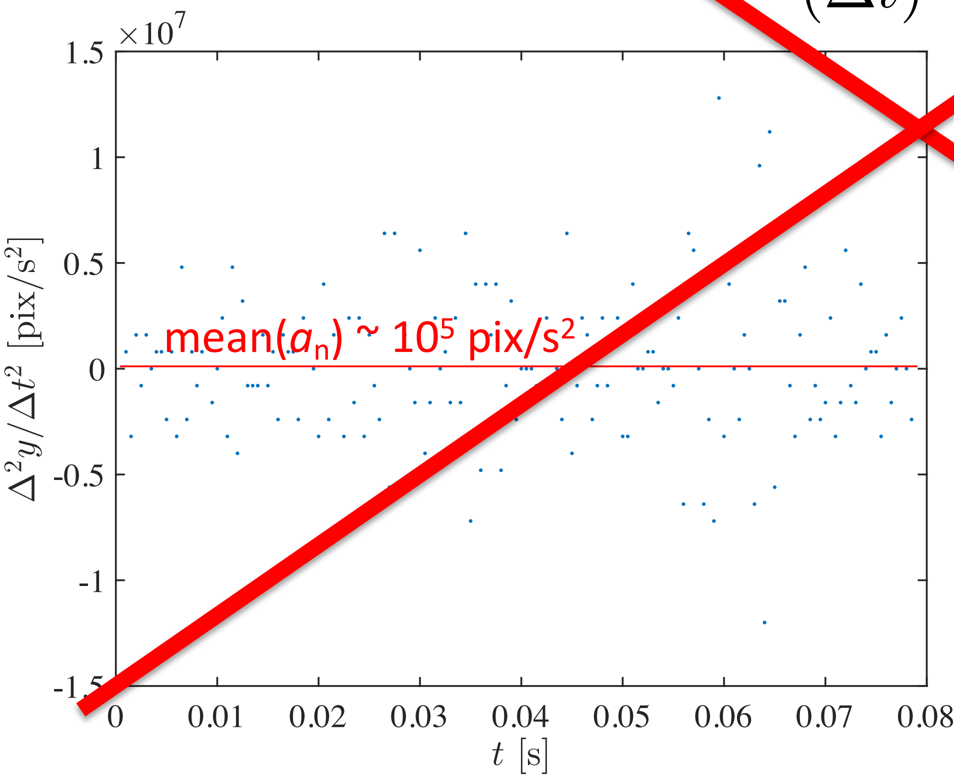
$$y_{n+1} = y_n + v_n(\Delta t) + \frac{a_n}{2}(\Delta t)^2$$

$$y_{n-1} = y_n - v_n(\Delta t) + \frac{a_n}{2}(\Delta t)^2$$

$$\Rightarrow a_n = \frac{2(y_{n+1} + y_{n-1} - 2y_n)}{(\Delta t)^2}$$

$y$ [pix]
$a_n$ = acceleration [pix/s$^2$]
$c$ = size [m] of one pixel
$g$ = 9.81 [m/s$^2$]

$$c = \frac{g}{\mathrm{mean}(a_n)}$$




```
ac=(y1(3:end)+y1(1:end-2)-2*y1(2:end-1))*2./(t1(2:end-1)-t1(1:end-2)).^2;
plot(t1(2:end-1),ac,'.'); hold on;
plot([min(t1),max(t1)],mean(ac)*[1 1],'r');
c=9.81/mean(ac);
```

It yields
$c$ = 87 µm/pix, from the large drop
$c$ = 29µm/pix, from the small drop

Why ?

# Model fitting

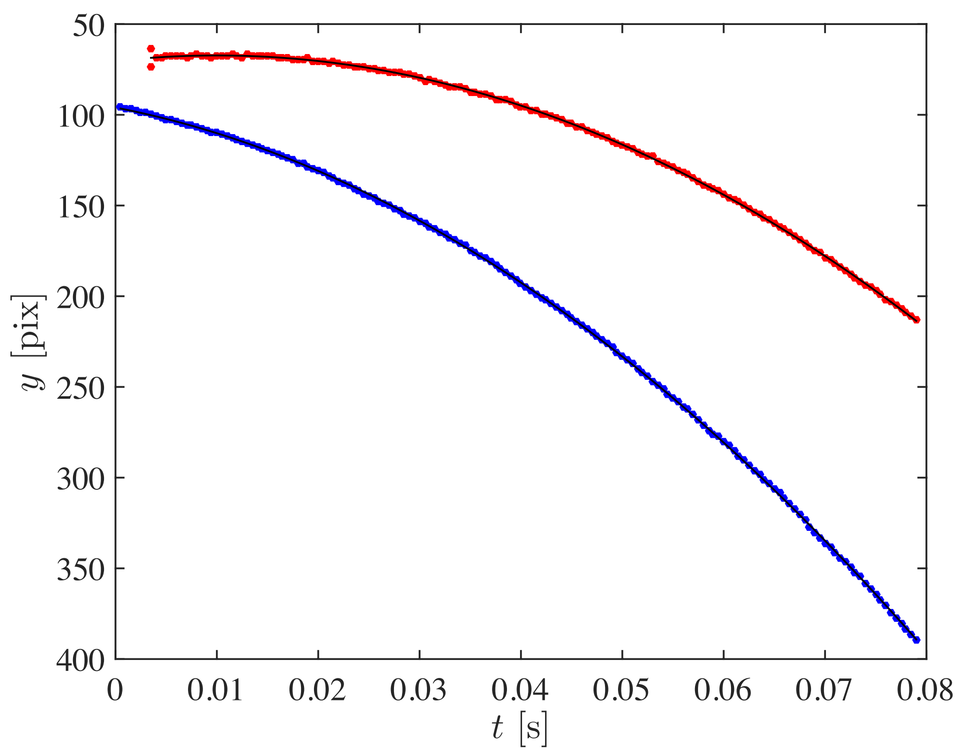
Physical model of free fall (without drag)

$$yc = y_0 c + v_0 t + \frac{g}{2} t^2$$

*y* [pix]
*c* = size [m] of one pixel
$V_0$ = initial speed [m/s]
*g* = 9.81 [m/s$^2$]

Implementation: $$y = p_2 t^2 + p_1 t + p_0 \qquad c = \frac{g}{2p_2}$$

➔ fitting of a 2nd order polynomial (i.e. linear in fitting coefficients $p_i$)

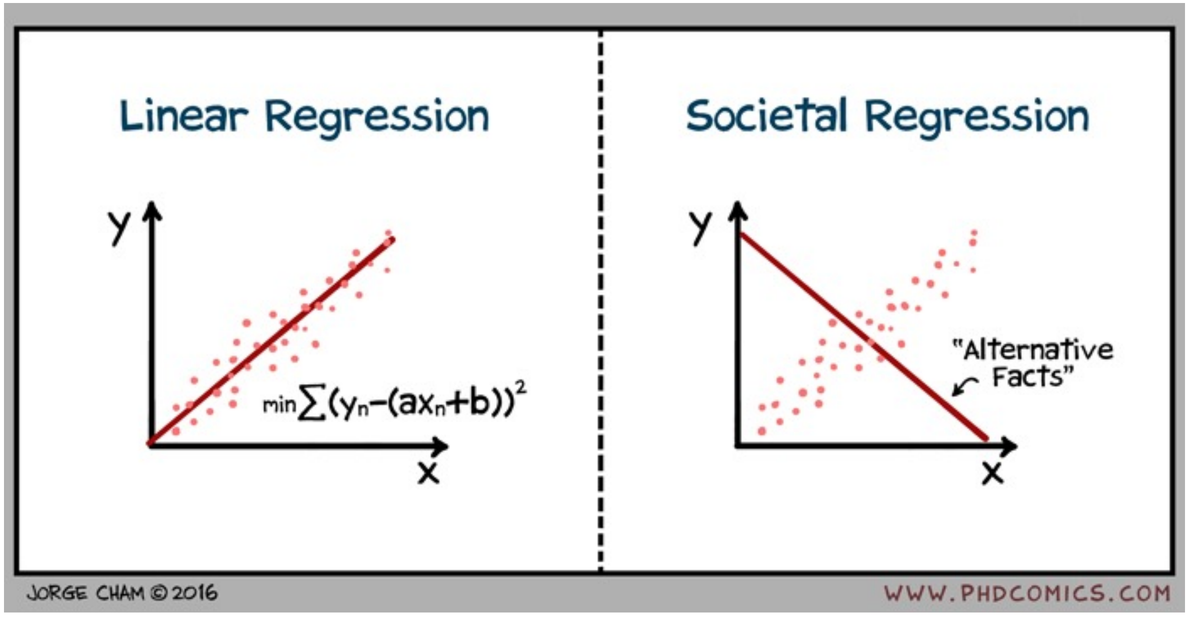



```
p1=polyfit(t1,y1,2); c1=9.81/2/p1(1),
p2=polyfit(t2,y2,2); c2=9.81/2/p2(1),
plot(t1,y1,'b.'); hold on;
plot(t2,y2,'r.');
plot(t1,polyval(p1,t1),'k');
plot(t2,polyval(p2,t2),'k');
```

It yields
*c* = 147 µm/pix, from the large drop
*c* = 160 µm/pix, from the small drop

Why ?

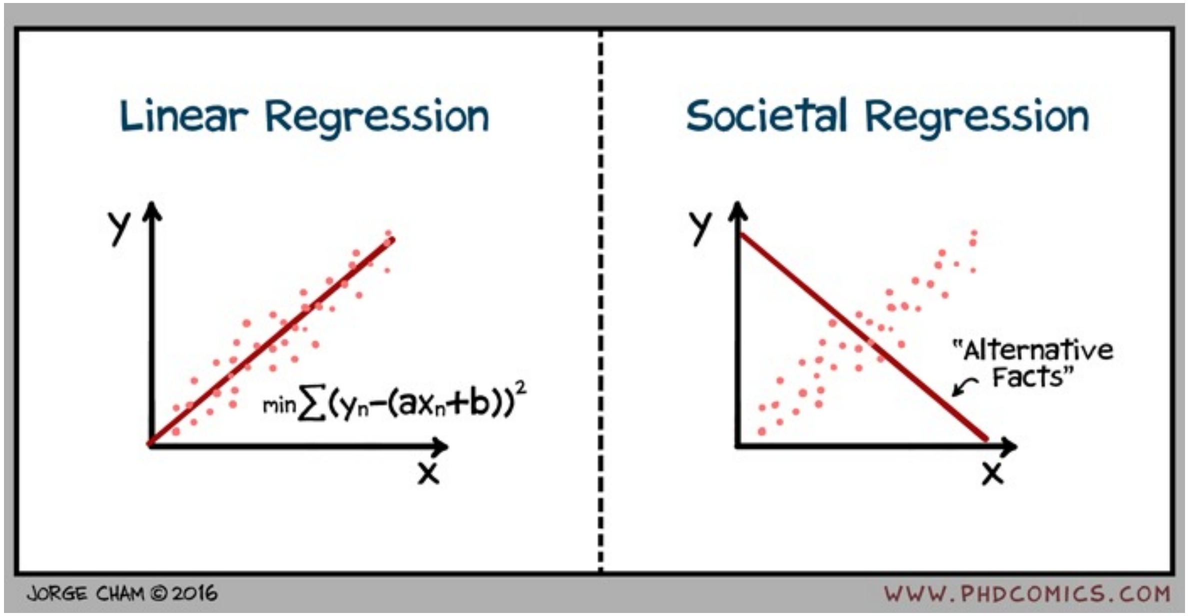
# Linear regression




Linear in the unknown coefficients. E.g. $z(t) \approx ae^{bt} \Rightarrow \ln z \approx \ln a + bt$

$$y = \ln z, \quad X = [1, t], \quad c = [\ln a, b]^T, \quad \boxed{y \approx Xc}$$

# Linear regression




Linear in the unknown coefficients. E.g. $z(t) \approx ae^{bt} \Rightarrow \ln z \approx \ln a + bt$

$$y = \ln z, \quad X = [1, t], \quad c = [\ln a, b]^T, \quad \boxed{y \approx Xc}$$

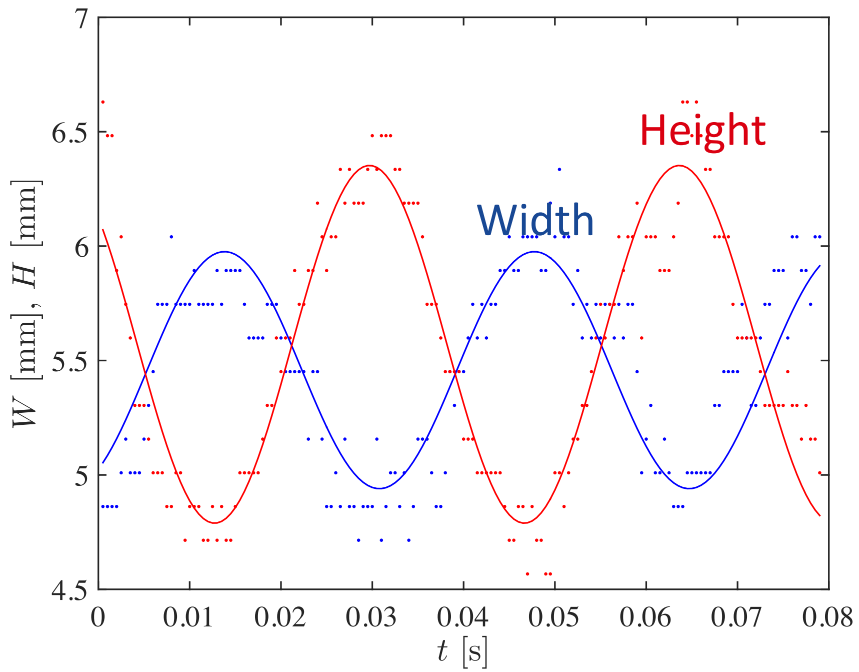
Least min square: $S(c) = ||y - Xc||^2 = y^T y - 2c^T X^T y + c^T X^T X c$

$$\frac{1}{2}\nabla_c S = \boxed{X^T X c - X^T y = 0}$$

➔ Normal equations (linear system)

In Matlab: c = X \ y; (*polyfit* is just a particular case for polynomial fitting)

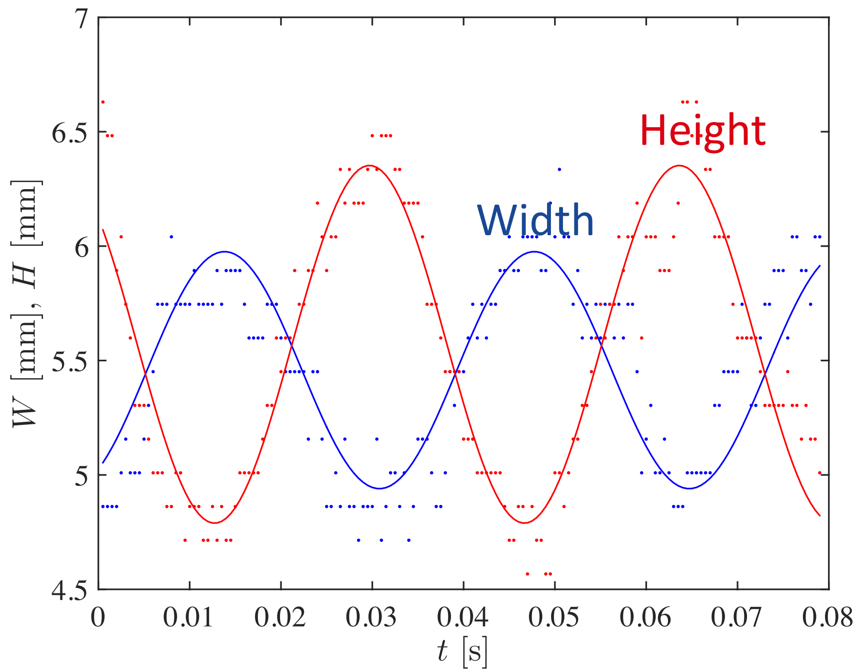
# Drop oscillation – non-linear regression




Drop width $W(t) = c_1 \cos(\omega t + \varphi) + c_2$ ➔ not linear in $\omega$ and $\varphi$

Equivalent and better: $W(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t) + c_3$ ➔ not linear in $\omega$

# Drop oscillation – non-linear regression




Drop width $W(t) = c_1 \cos(\omega t + \varphi) + c_2$ ➔ not linear in $\omega$ and $\varphi$

Equivalent and better: $W(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t) + c_3$ ➔ not linear in $\omega$

**Trick: Screening in the space of non-linear parameters**

for $\omega$ = … : …

“solve linear problem in ($c_1$, $c_2$, $c_3$)”, and store residual min($S$)

end;

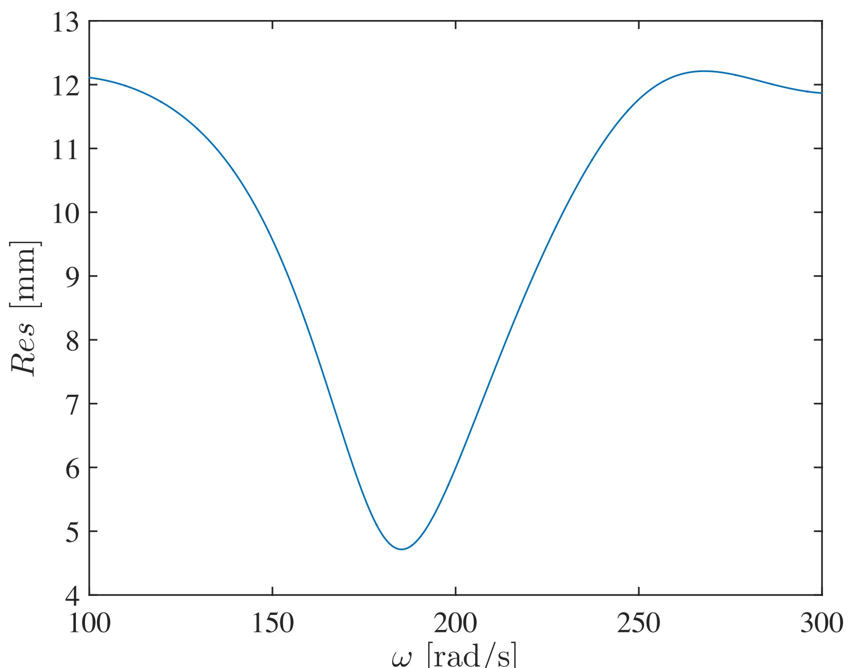
Then select the $\omega$ which gave the smallest residual

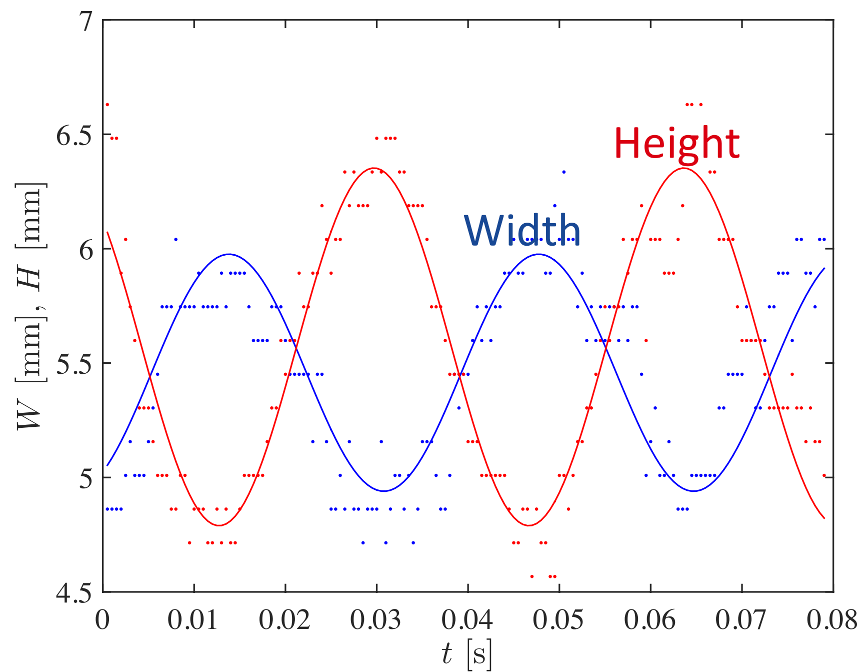
Also possible with iterative method (optimization)

# Drop oscillation – non-linear regression

```
om=linspace(100,300,1001);
Res=[];
for i0=1:numel(om),
    X=[cos(om(i0)*t(f31)), sin(om(i0)*t(f31)),ones(size(f31))];
    cW=X\W(f31);
    Res(i0)=norm(W(f31)-X*cW);
end;
i0=find(Res==min(Res));
XW=[cos(om(i0)*t(f31)), sin(om(i0)*t(f31)), ones(size(f31))];
cW=XW\W(f31);
plot(t(f31),XW*cW,'b');
```

Why is the width amplitude smaller than the height amplitude ?



# Contents

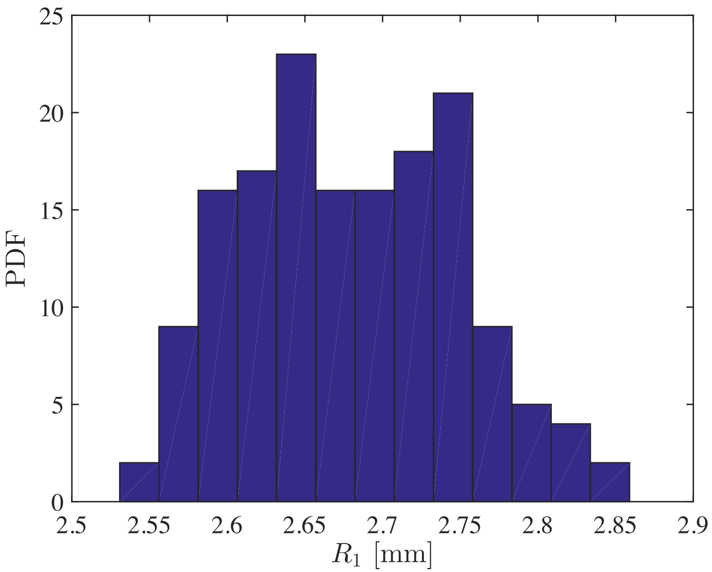
**Learn with an example**

1. Image processing: segmentation, filtering, measurements
2. Data processing: selection, model fitting
3. **Statistical analysis: descriptive statistics, t-test, ANOVA**

Practice with your preliminary data

# Drop size – Descriptive statistics

- Equivalent radius *R* [m] calculated from area *A* [pix²]: $R \simeq \sqrt{Ac^2/\pi}$

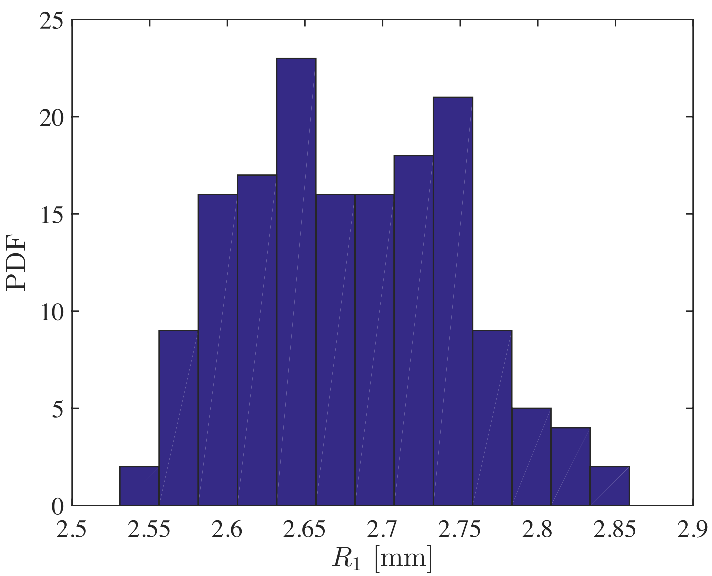



hist(R1*1e3,13);

➔ Depends on bins

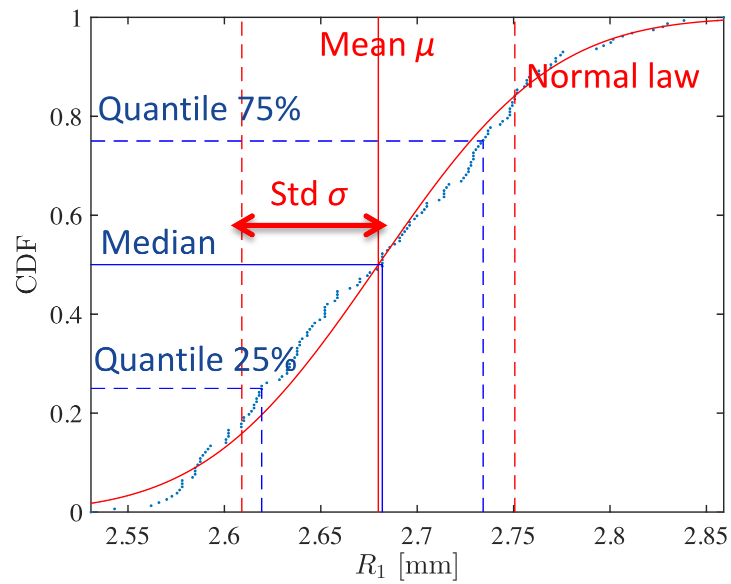
# Drop size – Descriptive statistics

- Equivalent radius *R* [m] calculated from area *A* [pix²]: $R \simeq \sqrt{Ac^2/\pi}$
- Reminder: Mean $\mu$, variance, standard deviation $\sigma$, median, quantile, box plot



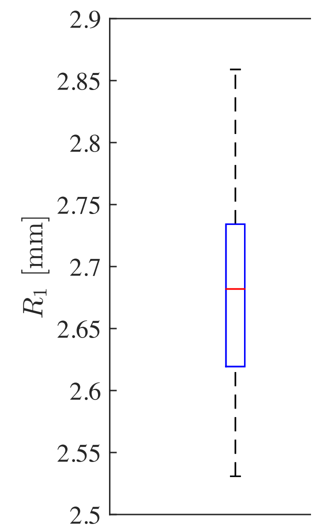

hist(R1*1e3,13);

➔ Depends on bins




plot(sort(R1),linspace(0,1,numel(R1)),'.');

➔ Independent of bins

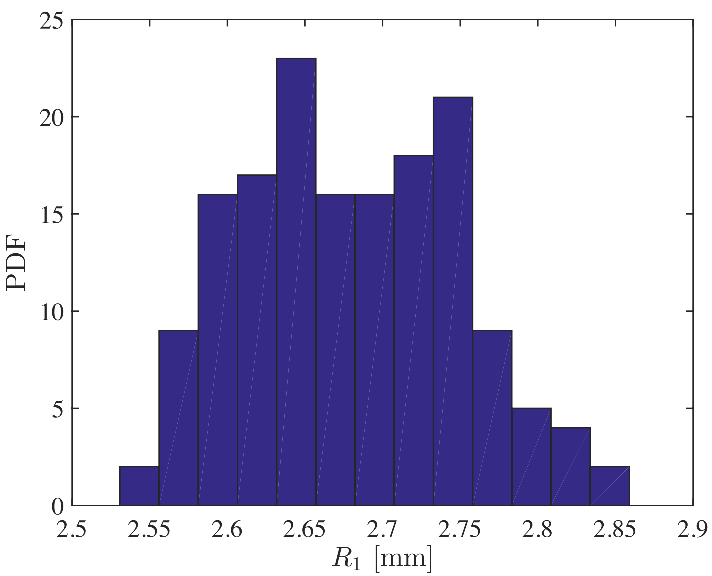



boxplot(R1);

# Drop size – Descriptive statistics

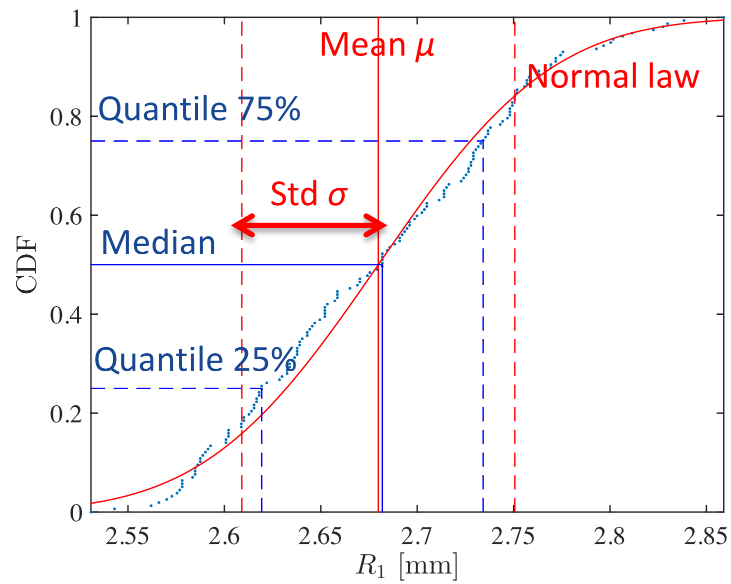
- Equivalent radius *R* [m] calculated from area *A* [pix$^2$]: $R \simeq \sqrt{Ac^2/\pi}$
- Reminder: Mean $\mu$, variance, standard deviation $\sigma$, median, quantile, box plot
- Variance of the sample mean = variance of the sample / size of the sample

➔ Error on the mean radius (for *N* indep. measurements): $\Delta R = \dfrac{\sigma}{\sqrt{N}}$



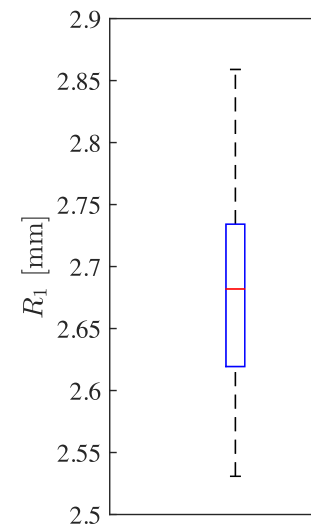

hist(R1*1e3,13);

➔ Depends on bins




plot(sort(R1),linspace(0,1,numel(R1)),'.');

➔ Independent of bins




boxplot(R1);

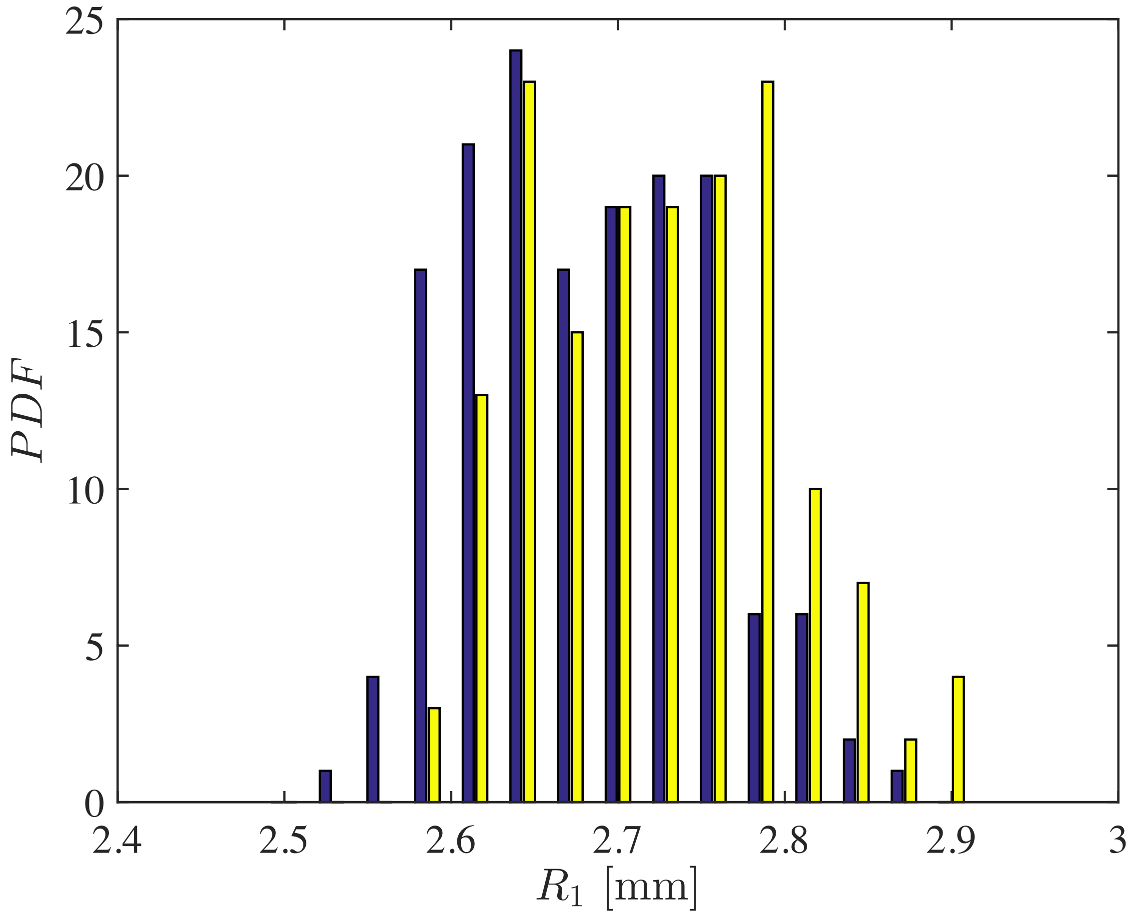
# Does the filtering method influence the results ?

Comparison of PDFs of $R_1$, from both MF and DFEEEEDDD filters

MF ➔ $R_1 = 2.68 \pm 0.07$ mm

DFEEEEDDD ➔ $R_1 = 2.72 \pm 0.07$ mm

Is this difference significant ?



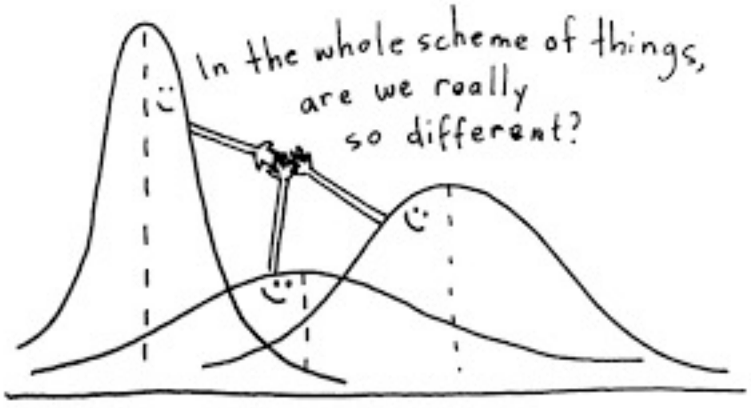
# Welch's two-sample t-test

**One output, 2 samples** (e.g. with one factor that varies)

Two independent (not paired) samples, assumed normal distribution

➔ **Is $m_1$ is significantly different from $m_2$ ?**

| **Sample** | **$X_1$** | **$X_2$** |
|---|---|---|
| Size | $N_1$ | $N_2$ |
| Average | $m_1$ | $m_2$ |
| Standard deviation | $s_1$ | $s_2$ |



# Welch's two-sample t-test

**One output, 2 samples** (e.g. with one factor that varies)

| Sample | $X_1$ | $X_2$ |
|---|---|---|
| Size | $N_1$ | $N_2$ |
| Average | $m_1$ | $m_2$ |
| Standard deviation | $s_1$ | $s_2$ |

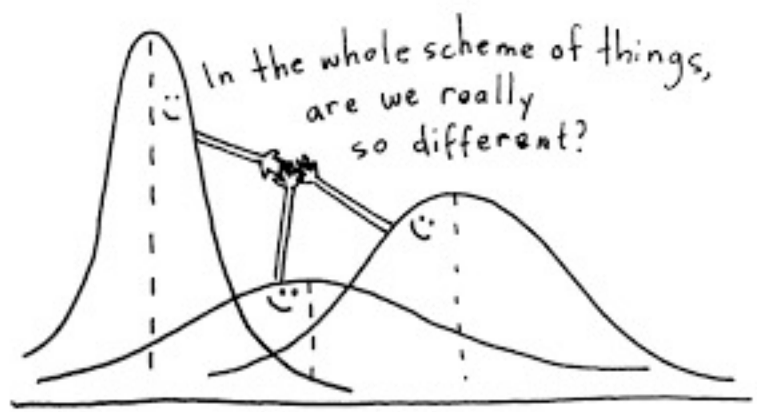
Two independent (not paired) samples, assumed normal distribution

➔ **Is $m_1$ is significantly different from $m_2$ ?**

Method: 2-tailed hypothesis test with a t-distribution
= Welch's two-sample t-test

Practically, in Matlab:
[*h*,*p*] = ttest2(*X1*, *X2*, 'Vartype', 'unequal');

*p* = probability to observe such difference between $m_1$ and $m_2$
if the two samples have the same mean in reality.



# Welch's two-sample t-test

| Sample | $X_1$ | $X_2$ |
|---|---|---|
| Size | $N_1$ | $N_2$ |
| Average | $m_1$ | $m_2$ |
| Standard deviation | $s_1$ | $s_2$ |

**One output, 2 samples** (e.g. with one factor that varies)

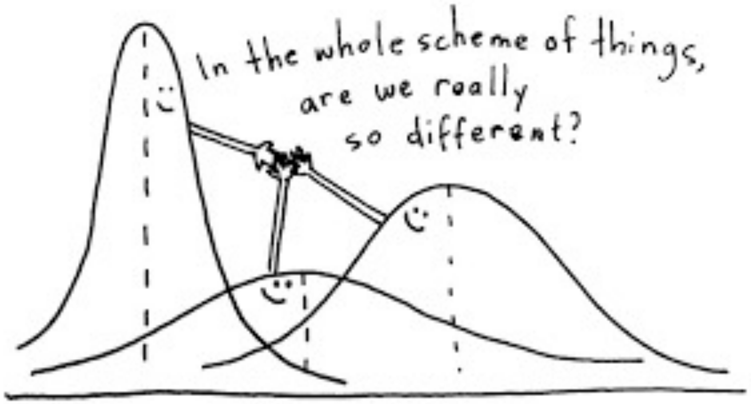
Two independent (not paired) samples, assumed normal distribution

➔ **Is $m_1$ is significantly different from $m_2$ ?**

Method: 2-tailed hypothesis test with a t-distribution
= Welch's two-sample t-test

Practically, in Matlab:
[*h*,*p*] = ttest2(*X1*, *X2*, 'Vartype', 'unequal');

*p* = probability to observe such difference between $m_1$ and $m_2$
if the two samples have the same mean in reality.
$\alpha$ = significance level (= 5% by default)




**Conclusion**

- $p>\alpha$ ➔ $h = 0$: we cannot conclude that $m_1 \neq m_2$
- $p<\alpha$ ➔ $h = 1$: we conclude that $m_1 \neq m_2$ , with less than 5% chance of being wrong

Power *P* of the test = probability to have $h$=1 if $m_1 \neq m_2$
Practically in Matlab: *P* = sampsizepwr('t2', [*m1*, *s1*], *m2*, [], *N2*, 'ratio', *N1*/*N2*);
➔ tells you if the samples are sufficiently large to conclude

# ANOVA

**One output, several factors**

ANOVA (ANalysis Of VAriance) test

➔ check if factors *g* have any influence on a measurement *y*

➔ Generalization of the Welch's test to many variables

Example in Matlab:

```
y = [52.7 57.5 45.9 44.5 53.0 57.0 45.9 44.0]';
g1 = [1 2 1 2 1 2 1 2];
g2 = {'hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo'};
g3 = {'may';'may';'may';'may';'june';'june';'june';'june'};
p = anovan(y,{g1,g2,g3})
```

ANOVA ➔ p-factor associated to factor *g* = probability that *g* has no influence on *y*

Here: p(*g1*) = 0.42 p(*g2*) = 0.003 p(*g3*) = 0.91

So *g2* does necessarily influence *y*, while *g1* and *g3* do not significantly influence *y*.

# Contents

Learn with an example

1. Image processing: segmentation, filtering, measurements
2. Data processing: selection, model fitting
3. Statistical analysis: descriptive statistics, t-test, ANOVA

**Practice with your preliminary data**