

# **SD-TSIA204**

## **Statistics : linear models**

**Joseph Salmon**

<http://josephsalmon.eu>

Télécom ParisTech, Institut Mines-Télécom

# Teachers

- **Joseph Salmon** (Assistant Professor) :
  - Positions : Paris Diderot-Paris 7, Duke University, Télécom ParisTech
  - Research themes : high dimensional statistics, aggregation, image processing, optimization
  - Email : *joseph.salmon@telecom-paristech.fr*
  - Office : E410
- **Francois Portier** (Assistant Professor) :
  - Positions : Université catholique de Louvain, Université de Rennes 1, Télécom ParisTech
  - Research themes : sparse regression, bootstrap, semi-parametric statistic, kernel smoothing
  - Email : *fportier@enst.fr*
  - Office : E 302

# Teaching assistants

- **Gower Robert** (Assistant Professor) : Optimization for machine learning, randomized numerical linear algebra, variable metric and quasi-Newton methods
- **Giovanna Varni** (Assistant Professor) : Social Signal Processing (SSP), human-human/human-computer/human-robot interaction, expressive gesture, and interpersonal synchrony
- ▶ **Garcia Alexandre** (PhD Student) : semi-supervised prediction for structured variables, sentiment analysis
- ▶ **Korba Anna** (PhD Student) : ranking, preference learning, voting rules, recommender systems
- ▶ **Ndiaye Eugene** (PhD Student) : sparse estimators, optimization with coordinates descent methods, safe screening rules

# Grades : SD-TSIA204

- Practical 1 : **20% final grade**
  - Problem statement available on Wednesday 06/12/2017 at 1 : 30 PM, deadline 23 : 59 the same day.
  - Single accepted file : **IPython Notebook**
- Practical 2 : **20% final grade**
  - Problem statement available on Wednesday 10/01/2018 at 3 : 15 PM, deadline 23 : 59 the same day.
  - Single accepted file : **IPython Notebook**
- Final exam : **60% note finale**
  - Date : 24/01/2018
  - Format : Quiz + exercises

Rem: Quiz questions samples are available on the course website (cf. Liste de questions section)

**BEWARE : your practical should be personal not copy pasted from your neighbor !!!**

# Practical notation

Practicals are graded on a scale from 0 to **20**, as follows

- ▶ scientific quality of answer **15** pts
- ▶ language/writing quality of answer (spelling, etc.) **2** pts
- ▶ indentation, PEP8 Style, useful comments in code, no/few warnings **2** pts
- ▶ no bug **1** pt (at least  
[https://github.com/agramfort/check\\_notebook](https://github.com/agramfort/check_notebook))
- ▶ one single **.ipynb** file expected, submitted on the “Site pédagogique” of the course ; emailed work will receive a zero score and will not be graded

Late : **no Late work** work will be accepted, unless official reason accepted by Télécom ParisTech's administration ; late work will receive a zero score and will not be graded

# Bonus

**1 pt** out of the **final grade** could be rewarded for any useful contribution to improve this course (slides, codes, etc.)

## Constraints :

- ▶ only the first contribution received on one aspect will be rewarded
- ▶ upload a **.txt** file for each proposition the “Propositions d'amélioration” section of the Site pédagogique
- ▶ detail precisely (code lines, slide page, etc.) the contribution proposed, and what it is improving/solving
- ▶ For typos : a minimum number of 5 typos is required to get the 1pt bonus

# Outline of the course

- Course 1** Joseph Salmon (08/11) : Introduction
- Course 2** Francois Portier (29/11) : Least squares properties
- Course 3** A. K./R. G./A. G./E. N./F. P./J. S. (06/12) :  
**Practical 1** Regression / Regularization
- Course 4** Francois Portier (13/12) : CI / Bootstrap
- Course 5** Francois Portier (20/12) : Ridge/Greedy/Lasso
- Course 6** Joseph Salmon (10/01) : PCA/SVD
- Course 7** G. V./A. K./A. G./E. N./F. P. /J. S. (10/01) :  
**Practical 2** (PCA/SVD)
- Course 8** Joseph Salmon (17/01) : GLM/Logistic regression
- Course 9** (24/01) : **Final Exam**

# Prerequisites

- ▶ **Probability** basis : probability, expectation, law of large number, Gaussian distribution, central limit theorem.  
Books : Foata et Fuchs (1996) (in French) or Murphy (2012, ch.1 and 2)
- ▶ **Optimisation** basis : (differential) calculus, convexity, first order conditions, gradient descent, Newton method  
Lecture : Boyd et Vandenberghe (2004), Bertsekas (1999)



# Prerequisites

- ▶ **Probability** basis : probability, expectation, law of large number, Gaussian distribution, central limit theorem.  
Books : Foata et Fuchs (1996) (in French) or Murphy (2012, ch.1 and 2)
- ▶ **Optimisation** basis : (differential) calculus, convexity, first order conditions, gradient descent, Newton method  
Lecture : Boyd et Vandenberghe (2004), Bertsekas (1999)
- ▶ (bi-)linear algebra basis : vector space, norms, inner product, matrices, determinants, diagonalization  
Lecture : Horn et Johnson (1994)

# Prerequisites

- ▶ **Probability** basis : probability, expectation, law of large number, Gaussian distribution, central limit theorem.  
Books : Foata et Fuchs (1996) (in French) or Murphy (2012, ch.1 and 2)
- ▶ **Optimisation** basis : (differential) calculus, convexity, first order conditions, gradient descent, Newton method  
Lecture : Boyd et Vandenberghe (2004), Bertsekas (1999)
- ▶ **(bi-)linear algebra** basis : vector space, norms, inner product, matrices, determinants, diagonalization  
Lecture : Horn et Johnson (1994)
- ▶ **Numerical linear algebra** : linear system resolution, Gaussian elimination, matrix factorization, conditioning, etc.  
Lecture : Golub et VanLoan (2013), Applied Numerical Computing par L. Vandenberghe

# Prerequisites

- ▶ **Probability** basis : probability, expectation, law of large number, Gaussian distribution, central limit theorem.  
Books : Foata et Fuchs (1996) (in French) or Murphy (2012, ch.1 and 2)
- ▶ **Optimisation** basis : (differential) calculus, convexity, first order conditions, gradient descent, Newton method  
Lecture : Boyd et Vandenberghe (2004), Bertsekas (1999)
- ▶ **(bi-)linear algebra** basis : vector space, norms, inner product, matrices, determinants, diagonalization  
Lecture : Horn et Johnson (1994)
- ▶ **Numerical linear algebra** : linear system resolution, Gaussian elimination, matrix factorization, conditioning, etc.  
Lecture : Golub et VanLoan (2013), Applied Numerical Computing par L. Vandenberghe

# Algorithmic aspects : some advice

Python installation : use **Conda** / **Anaconda**

Rem: you are on your own for this (or use the school machines)

Recommended tools : **Jupyter** / **IPython Notebook** (mandatory for your practical) **IPython** with a text editor e.g., **Atom**, **Sublime Text**, **Visual Studio Code**, etc., for larger projects

- ▶ **Python, Scipy, Numpy** :

[http://perso.telecom-paristech.fr/~gramfort/liesse\\_python/](http://perso.telecom-paristech.fr/~gramfort/liesse_python/)

- ▶ **Pandas** : <http://pandas.pydata.org/>

- ▶ **scikit-learn** : <http://scikit-learn.org/stable/>

Rem: for practicals, bring your own machine if you prefer but install your Python environment upfront

## General advice

- ▶ Use version control system for your work : **Git** (e.g., **Bitbucket**, **Github**, etc.)
- ▶ Use clean way of writing code/ presenting your code  
Example : **PEP8** for Python (use for instance **AutoPEP8**, <https://github.com/kenko000/jupyter-autopep8>)
- ▶ Learn from good examples :  
<https://github.com/scikit-learn/>,  
<http://jakevdp.github.io/>, etc.

# Outline

## Syllabus, grades, etc.

- Teaching staff

- Grades and bonus

## 1D Least squares

- Introduction : visualization / Python

- Modeling

- Mathematical Formulation

- Centering - scaling

- Likelihood

# Outline

Syllabus, grades, etc.

Teaching staff

Grades and bonus

## 1D Least squares

Introduction : visualization / Python

Modeling

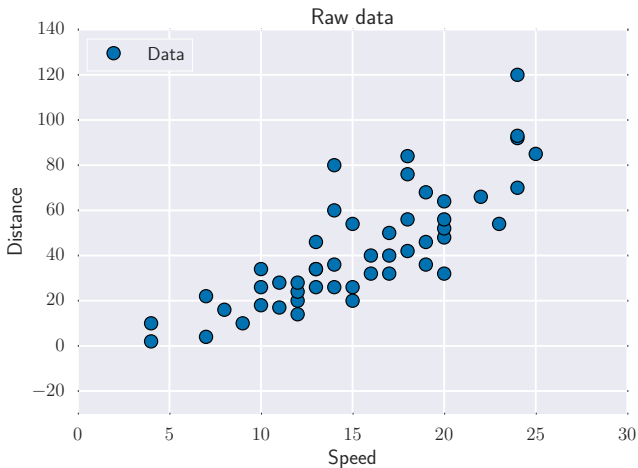
Mathematical Formulation

Centering - scaling

Likelihood

## A 2D starting example

Example : braking distance for cars as a function of speed  
( $n = 50$  measurements)



Dataset *cars* : <https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/cars.html>  
<https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/cars.html>



## A 2D starting example

Example : braking distance for cars as a function of speed  
( $n = 50$  measurements)



Dataset *cars* : <https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/cars.html>  
<https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/cars.html>

## Python command

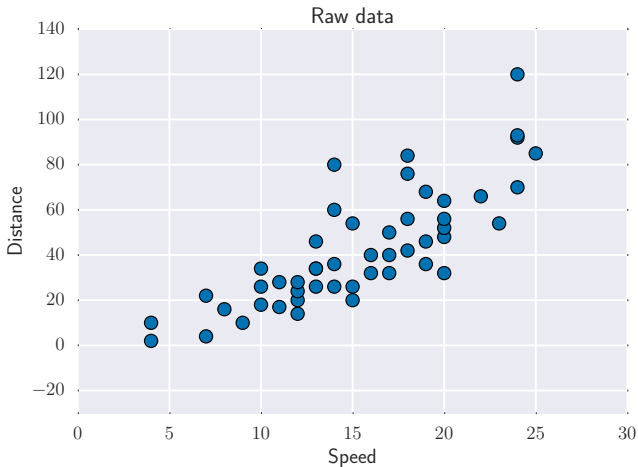
```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.linear_model as lm
# Load data
url = 'cars.csv'
dat = pd.read_csv(url)
y = dat['dist']
X = dat[['speed']] # sklearn needs X to have 2 dim.

skl_linmod = lm.LinearRegression(fit_intercept=False)
skl_linmod.fit(X, y) # Fit regression model

fig = plt.figure(figsize=(8, 6))
plt.plot(X, y, 'o', label="Data")
plt.plot(X, skl_linmod.predict(X),
         label="OLS-sklearn-no-intercept")
plt.legend(loc='upper left')
plt.show()
```

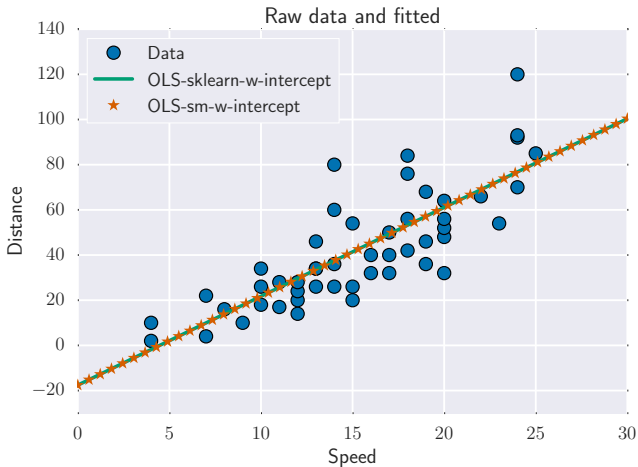
## A 2D starting example : with an intercept

Example : braking distance for cars as a function of speed ( $n = 50$  measurements)



## A 2D starting example : with an intercept

Example : braking distance for cars as a function of speed ( $n = 50$  measurements)



## Python commands : with intercept

```
import statsmodels.api as sm

# data, fitted, etc
y = dat['dist']
X = dat[['speed']]
X = sm.add_constant(X)
results = sm.OLS(y,X).fit()

# plot
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(X['speed'], y, 'o', label="data")
ax.plot(X['speed'], results.fittedvalues,
        linewidth=3, label="OLS-sm-w-intercept")
ax.legend(loc='best')
```

Alternative : use `lm.LinearRegression(fit_intercept=True)`

# Outline

Syllabus, grades, etc.

Teaching staff

Grades and bonus

## 1D Least squares

Introduction : visualization / Python

Modeling

Mathematical Formulation

Centering - scaling

Likelihood

# Modeling I

Observations :  $(y_i, x_i)$ , for  $i = 1, \dots, n$

Linear model or linear regression hypothesis assume :

$$y_i \approx \theta_0^* + \theta_1^* x_i$$

- ▶  $\theta_0^*$  : intercept (unknown)
- ▶  $\theta_1^*$  : slope (unknown)

Rem: both parameters are unknown from the statistician

## Definition

- ▶  $y$  is an **observation** or a variable to explain
- ▶  $x$  is a **feature** or a covariate

# Notation interpretation

## Example : dataset *cars*

- ▶  $n = 50$
- ▶  $y_i$  : braking time for  $i$ -th car
- ▶  $x_i$  : speed of  $i$ -th car
- ▶  $y$  : the observation is the car's braking time
- ▶  $x$  : the feature/covariate is the car's speed

Linear model / Linear regression hypothesis : assume that braking time is proportional to speed

---

**Exo:** use `describe()` from Pandas to get a rough data summary

---



# Modeling II

Let us give a precise meaning to the sign  $\approx$  :

## Probabilistic model

$$y_i = \theta_0^* + \theta_1^* x_i + \varepsilon_i,$$

$$\varepsilon_i \stackrel{i.i.d}{\sim} \varepsilon, \text{ for } i = 1, \dots, n$$

$$\mathbb{E}(\varepsilon) = 0$$

where i.i.d. means “independent and identically distributed”

## Interpretation

$\varepsilon_i = y_i - \theta_0^* - \theta_1^* x_i$  : represent the error between the theoretical model and the observations, represented by random variables  $\varepsilon_i$  centered (often referred to as **white noise**).

Rem: motivation for the random nature of the noise – measurement noise, transmission noise, in-population variability, etc.

# Modeling III

$$y_i = \theta_0^* + \theta_1^* x_i + \varepsilon_i$$

## Definition

We call

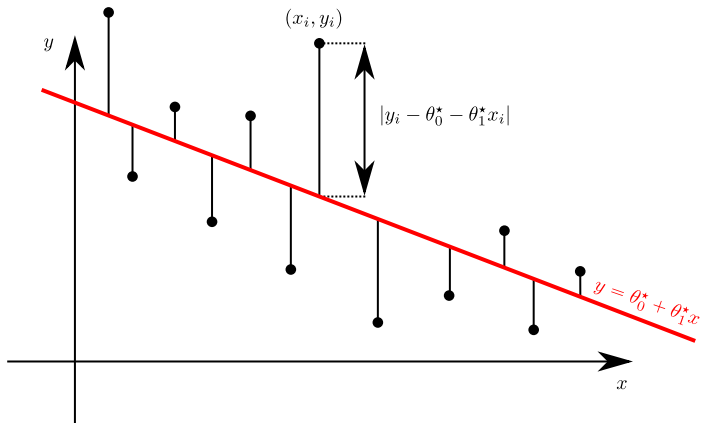
- ▶ **intercept** the scalar  $\theta_0^*$  (■ ■ : *ordonnée à l'origine*)
- ▶ **slope** the scalar  $\theta_1^*$  (■ ■ : *pente*)

## Goal

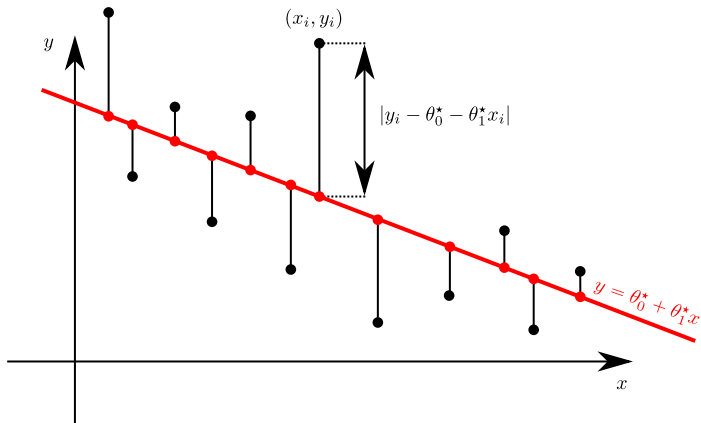
Estimate  $\theta_0^*$  and  $\theta_1^*$  (unknown) by  $\hat{\theta}_0$  and  $\hat{\theta}_1$  relying on observations  $(y_i, x_i)$  for  $i = 1, \dots, n$

Rem: The “hat” notation is classical in statistics for referring to estimators

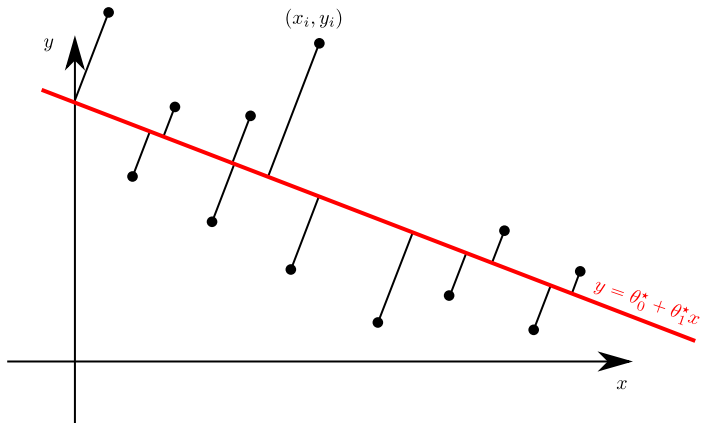
# Least squares : visualization



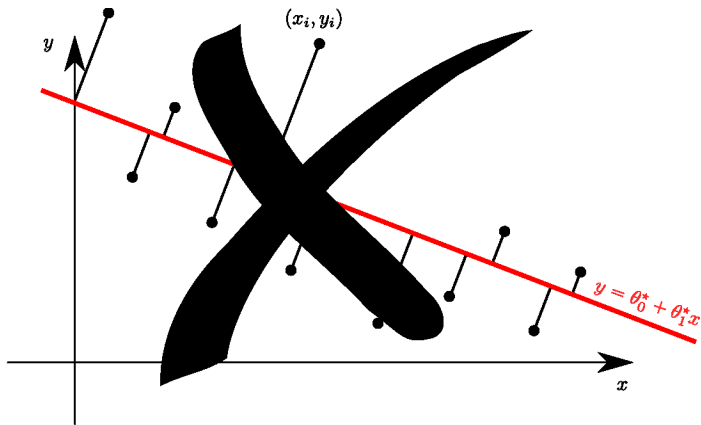
# Least squares : visualization



## (Total) Least squares : visualization



## (Total) Least squares : visualization



# Least squares : mathematical formulation

## Definition

The **least squares** estimator is defined as :

$$(\hat{\theta}_0, \hat{\theta}_1) \in \arg \min_{(\theta_0, \theta_1) \in \mathbb{R}^2} \sum_{i=1}^n (y_i - \theta_0 - \theta_1 x_i)^2$$

- ▶ it is also referred to as “ordinary least squares” (OLS)
- ▶ an original motivation for the squares is computational : first order conditions only require solving a linear system
- ▶ a solution always exists : minimizing a **coercive** continuous function (coercive :  $\lim_{\|x\| \rightarrow +\infty} f(x) = +\infty$ )

Rem: write «  $\in \arg \min$  » as long as you do not know if the solution is unique

## Least square authorship (controversial)



Adrien-Marie Legendre :  
"Nouvelles méthodes pour la  
détermination des orbites des comètes",  
1805



Carl Friedrich Gauss :  
"Theoria Motus Corporum Coelestium  
in sectionibus conicis solem  
ambientium" 1809



# Historical / robust detour

## Definition

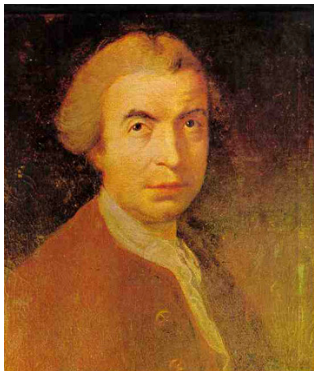
The **least absolute deviation** (LAD) estimator reads :

$$(\hat{\theta}_0, \hat{\theta}_1) \in \arg \min_{(\theta_0, \theta_1) \in \mathbb{R}^2} \sum_{i=1}^n |y_i - \theta_0 - \theta_1 x_i|$$

Rem: hard to compute without computer ; requires an optimization solver for non-smooth function (or a Linear Programming solver)

Rem: more robust to outliers (🇫🇷 : *données aberrantes*)

## Least absolute deviation authorship



Ruđer Josip Bošković :“???",  
1757



Pierre-Simon de Laplace  
“Traité de mécanique céleste”,  
1799

# Outline

Syllabus, grades, etc.

Teaching staff

Grades and bonus

## 1D Least squares

Introduction : visualization / Python

Modeling

**Mathematical Formulation**

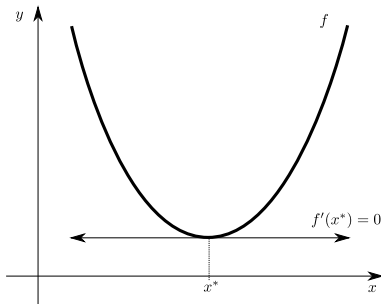
Centering - scaling

Likelihood

# Local minimum : first order condition

## Theorem : Fermat's rule

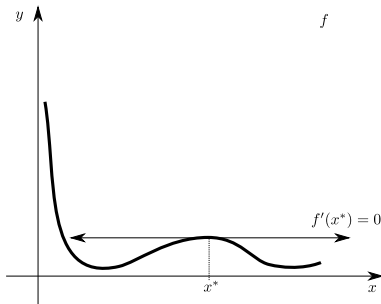
If  $f$  is differentiable, then at a local minimum  $x^*$  the gradient of  $f$  vanishes at  $x^*$ , i.e.,  $\nabla f(x^*) = 0$ .



# Local minimum : first order condition

## Theorem : Fermat's rule

If  $f$  is differentiable, then at a local minimum  $x^*$  the gradient of  $f$  vanishes at  $x^*$ , i.e.,  $\nabla f(x^*) = 0$ .

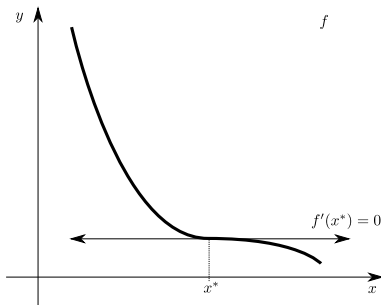


Rem: sufficient condition when  $f$  is convex !

# Local minimum : first order condition

## Theorem : Fermat's rule

If  $f$  is differentiable, then at a local minimum  $x^*$  the gradient of  $f$  vanishes at  $x^*$ , i.e.,  $\nabla f(x^*) = 0$ .



Rem: sufficient condition when  $f$  is convex !

## Back to least squares

$$\hat{\boldsymbol{\theta}} = (\hat{\theta}_0, \hat{\theta}_1) \in \arg \min_{(\theta_0, \theta_1) \in \mathbb{R}^2} \frac{1}{2} \sum_{i=1}^n (y_i - \theta_0 - \theta_1 x_i)^2$$

For least squares, minimize the function of two variables :

$$f(\theta_0, \theta_1) = f(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^n (y_i - \theta_0 - \theta_1 x_i)^2$$

First order condition / Fermat's rule :

$$\begin{cases} \frac{\partial f}{\partial \theta_0}(\hat{\boldsymbol{\theta}}) = \sum_{i=1}^n (y_i - \hat{\theta}_0 - \hat{\theta}_1 x_i) = 0 \\ \frac{\partial f}{\partial \theta_1}(\hat{\boldsymbol{\theta}}) = \sum_{i=1}^n (y_i - \hat{\theta}_0 - \hat{\theta}_1 x_i) x_i = 0 \end{cases}$$

---

**Exo:** Is  $f$  convex? help : a sum of convex functions is convex

---

## Calculus continued

Usual mean notation :  $\bar{x}_n = \frac{1}{n} \sum_{i=1}^n x_i$  and  $\bar{y}_n = \frac{1}{n} \sum_{i=1}^n y_i$

With that, Fermat's rule states (dividing by  $n$ ) :

$$\begin{cases} \frac{\partial f}{\partial \theta_0}(\hat{\theta}) = \sum_{i=1}^n (y_i - \hat{\theta}_0 - \hat{\theta}_1 x_i) = 0 \\ \frac{\partial f}{\partial \theta_1}(\hat{\theta}) = \sum_{i=1}^n (y_i - \hat{\theta}_0 - \hat{\theta}_1 x_i) x_i = 0 \end{cases}$$

$\Leftrightarrow$

$$\begin{cases} \hat{\theta}_0 = \bar{y}_n - \hat{\theta}_1 \bar{x}_n & \text{(CNO1)} \\ \hat{\theta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x}_n)(y_i - \bar{y}_n)}{\sum_{i=1}^n (x_i - \bar{x}_n)^2} & \text{(CNO2)} \end{cases}$$

---

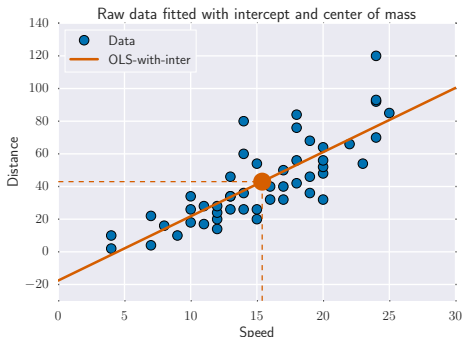
**Exo:** Prove that (CNO2) holds if and only if  $\mathbf{x} = (x_1, \dots, x_n)^\top$  is non constant, i.e.,  $\mathbf{x}$  is not proportional to  $\mathbf{1}_n = (1, \dots, 1)^\top \in \mathbb{R}^n$

---



# Center of gravity and interpretation

$$(\text{CNO1}) \Leftrightarrow (\bar{x}_n, \bar{y}_n) \in \{(x, y) \in \mathbb{R}^2 : y = \hat{\theta}_0 + \hat{\theta}_1 x\}$$



- ▶  $\overline{speed} = 15.4$
- ▶  $\overline{dist} = 42.98$
- ▶  $\hat{\theta}_0 = -17.579095$  intercept (negative!)
- ▶  $\hat{\theta}_1 = 3.932409$  slope

Physical interpretation : the cloud of points' center of gravity belongs to the (estimated) regression line

## Vector formulation

Notation :  $\mathbf{x} = (x_1, \dots, x_n)^\top$  and  $\mathbf{y} = (y_1, \dots, y_n)^\top$

$$(\text{CNO2}) \Leftrightarrow \hat{\theta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x}_n)(y_i - \bar{y}_n)}{\sum_{i=1}^n (x_i - \bar{x}_n)^2}$$

$$(\text{CNO2}) \Leftrightarrow \hat{\theta}_1 = \text{corr}_n(\mathbf{x}, \mathbf{y}) \cdot \frac{\sqrt{\text{var}_n(\mathbf{y})}}{\sqrt{\text{var}_n(\mathbf{x})}}$$

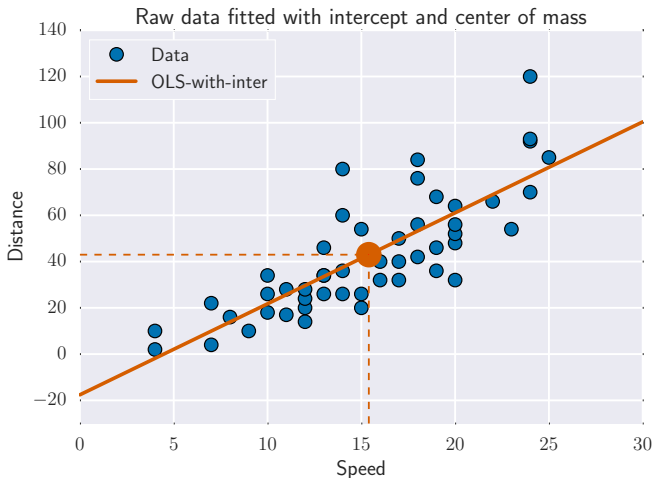
where  $\text{corr}_n(\mathbf{x}, \mathbf{y}) = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x}_n)(y_i - \bar{y}_n)}{\sqrt{\text{var}_n(\mathbf{x})} \sqrt{\text{var}_n(\mathbf{y})}}$

and  $\text{var}_n(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n (z_i - \bar{z}_n)^2$  (for any  $\mathbf{z} = (z_1, \dots, z_n)^\top$ )

respectively **empirical correlation** **empirical variances**

## Back to the *cars* example

Line slope :  $\text{corr}_n(\mathbf{x}, \mathbf{y}) \cdot \frac{\sqrt{\text{var}_n(\mathbf{y})}}{\sqrt{\text{var}_n(\mathbf{x})}} = 3.932409$ .



# Outline

Syllabus, grades, etc.

Teaching staff

Grades and bonus

## 1D Least squares

Introduction : visualization / Python

Modeling

Mathematical Formulation

Centering - scaling

Likelihood

# Centering

**Centered** model :

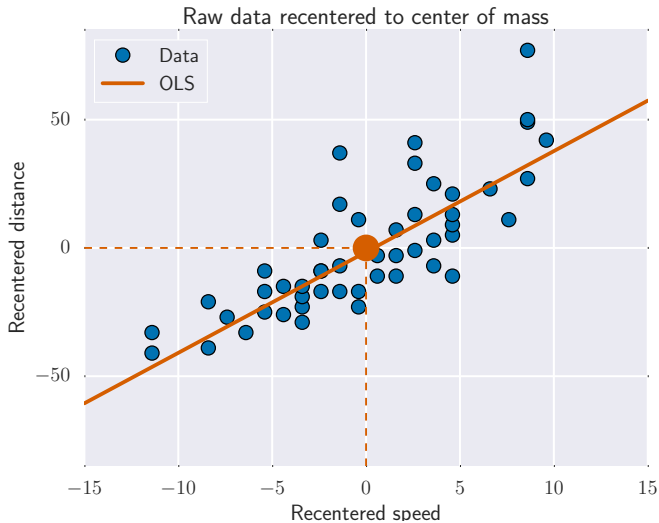
$$\text{Write for any } i = 1, \dots, n : \begin{cases} x'_i = x_i - \bar{x}_n \\ y'_i = y_i - \bar{y}_n \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}' = \mathbf{x} - \bar{x}_n \mathbf{1}_n \\ \mathbf{y}' = \mathbf{y} - \bar{y}_n \mathbf{1}_n \end{cases}$$

and  $\mathbf{1}_n = (1, \dots, 1)^\top \in \mathbb{R}^n$ , then solving the OLS with  $(\mathbf{x}', \mathbf{y}')$  leads to

$$\begin{cases} \hat{\theta}'_0 = 0 \\ \hat{\theta}'_1 = \frac{\frac{1}{n} \sum_{i=1}^n x'_i y'_i}{\frac{1}{n} \sum_{i=1}^n x_i'^2} \end{cases}$$

Rem: equivalent to choosing the cloud of points' center of mass as origin, i.e.,  $(\bar{x}'_n, \bar{y}'_n) = (0, 0)$

## Centering (II)




## Centering and interpretation

Consider the coefficient  $\hat{\theta}'_1$  ( $\hat{\theta}'_0 = 0$ ) for centered points  $\mathbf{y}'$ ,  $\mathbf{x}'$ , then :

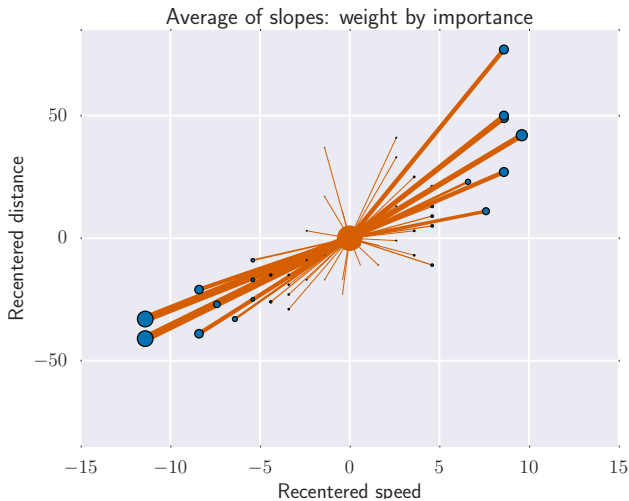
$$\hat{\theta}'_1 \in \arg \min_{\theta_1} \sum_{i=1}^n (y'_i - \theta_1 x'_i)^2 = \arg \min_{\theta_1} \sum_{i=1}^n x'^2_i \left( \frac{y'_i}{x'_i} - \theta_1 \right)^2$$

Interpretation :  $\hat{\theta}'_1$  is a weighted average of the slopes  $\frac{y'_i}{x'_i}$

$$\hat{\theta}'_1 = \frac{\sum_{i=1}^n x'^2_i \frac{y'_i}{x'_i}}{\sum_{j=1}^n x'^2_j}$$

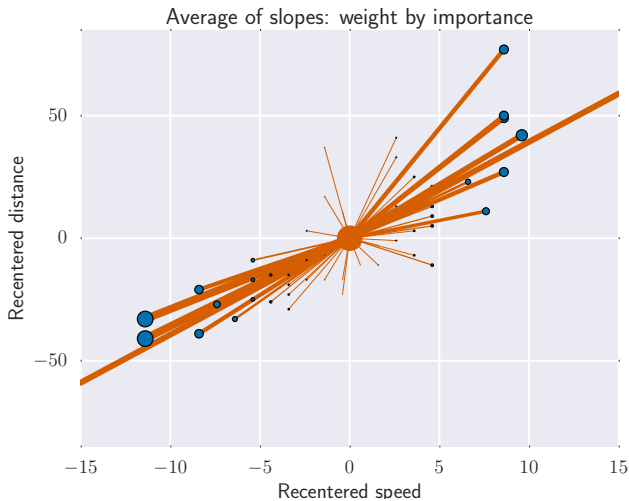
Influence of extreme points : weights proportional to  $x'^2_i$  ;  
connected to the **leverage** (  : levier ) effect

# Extreme points – leverage effect

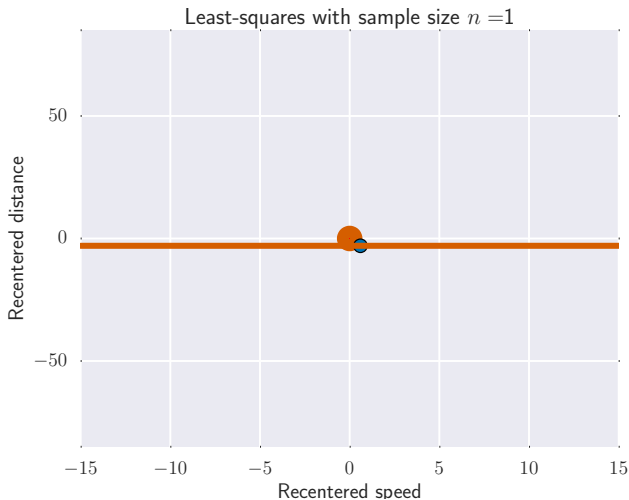




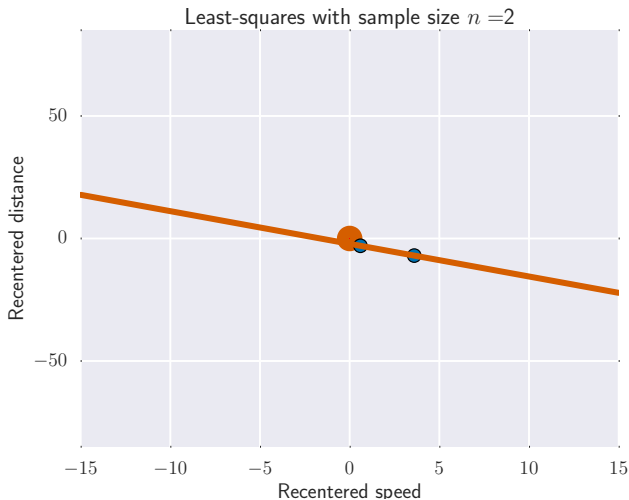
# Extreme points – leverage effect



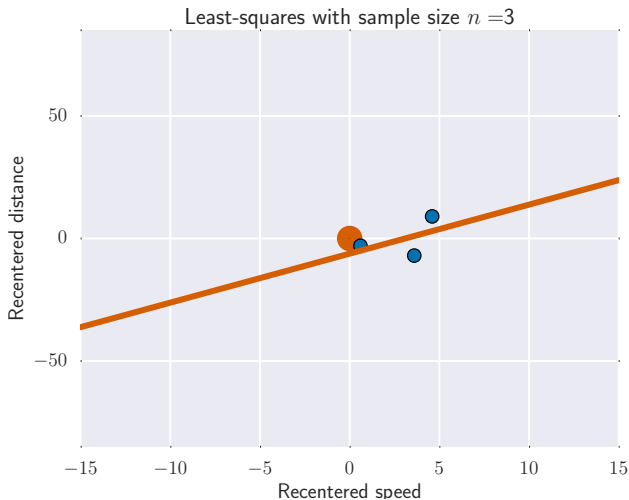
## Extreme points – leverage effect (II)



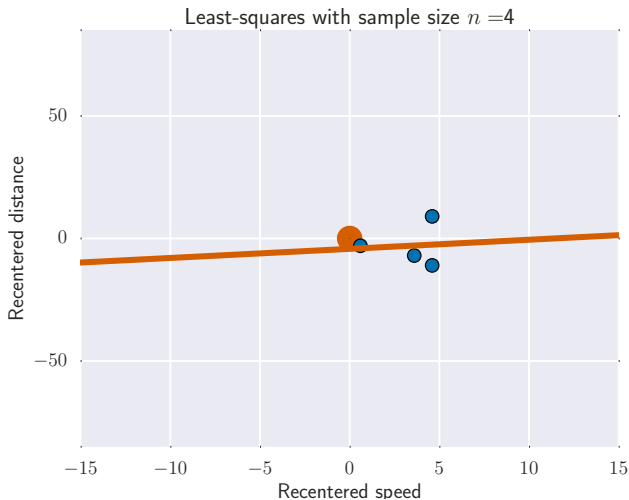
## Extreme points – leverage effect (II)



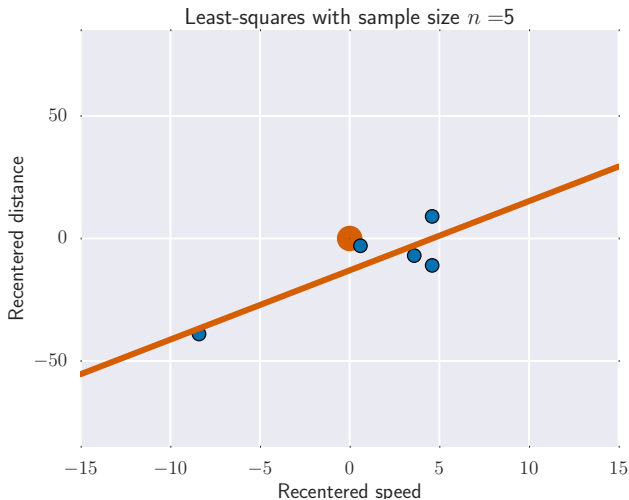
## Extreme points – leverage effect (II)



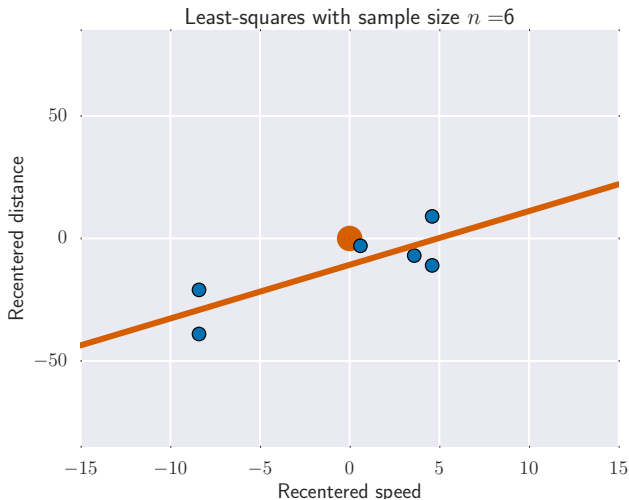
## Extreme points – leverage effect (II)



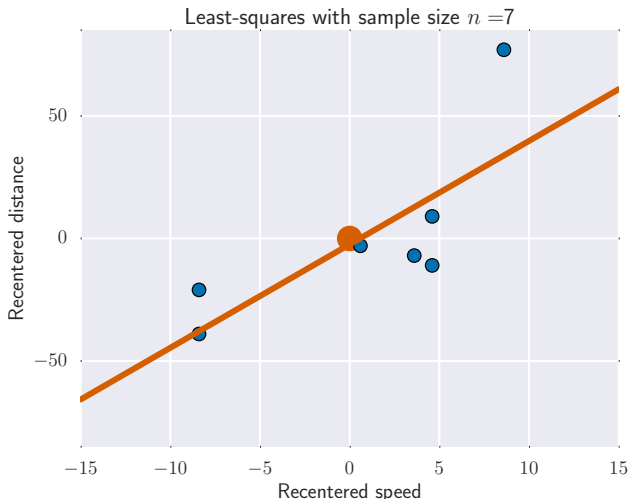
## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)

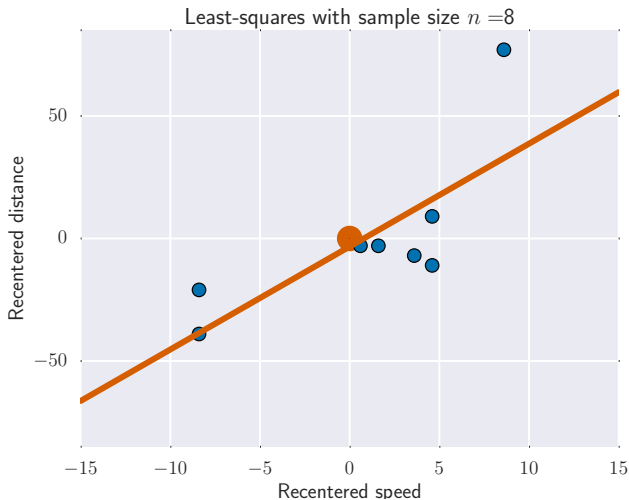


## Extreme points – leverage effect (II)

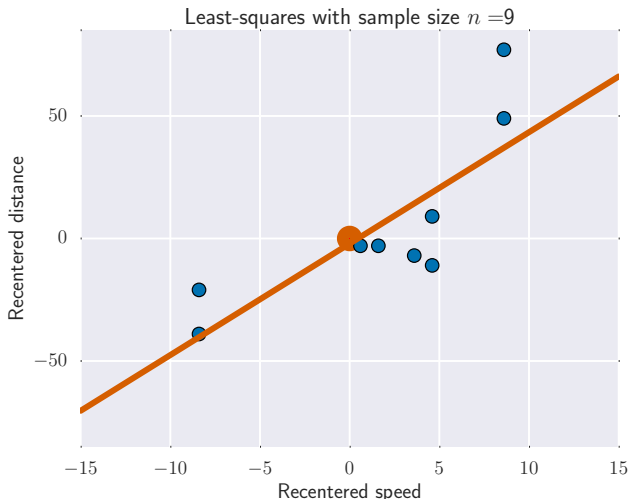




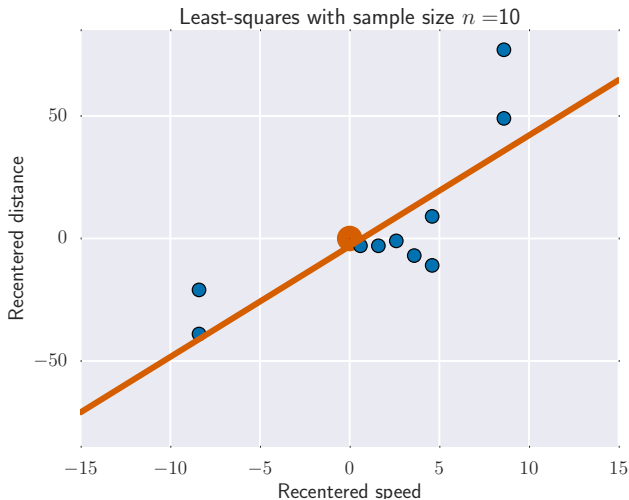
## Extreme points – leverage effect (II)



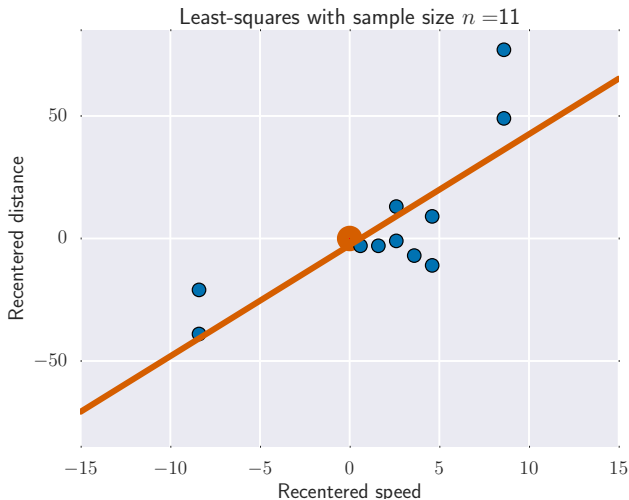
## Extreme points – leverage effect (II)



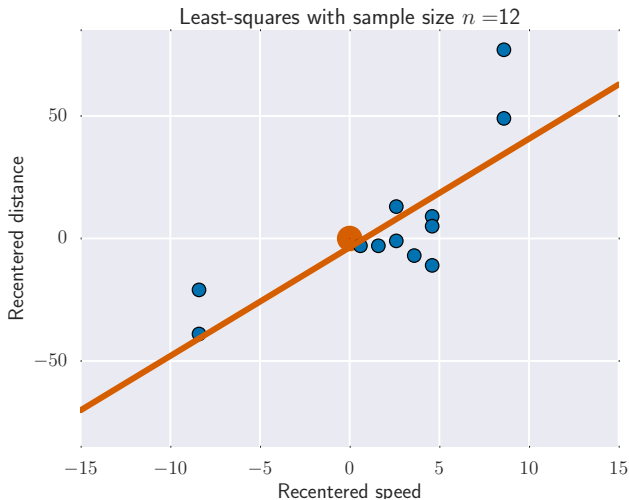
## Extreme points – leverage effect (II)



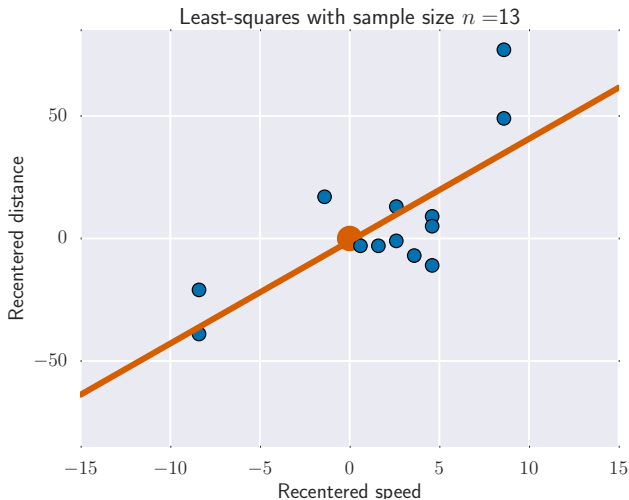
## Extreme points – leverage effect (II)



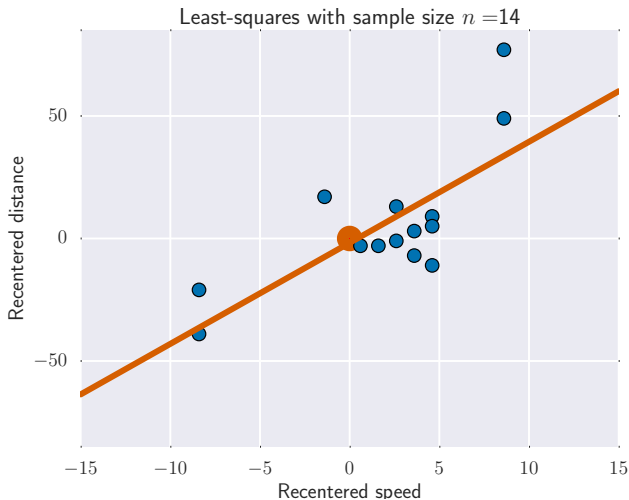
## Extreme points – leverage effect (II)



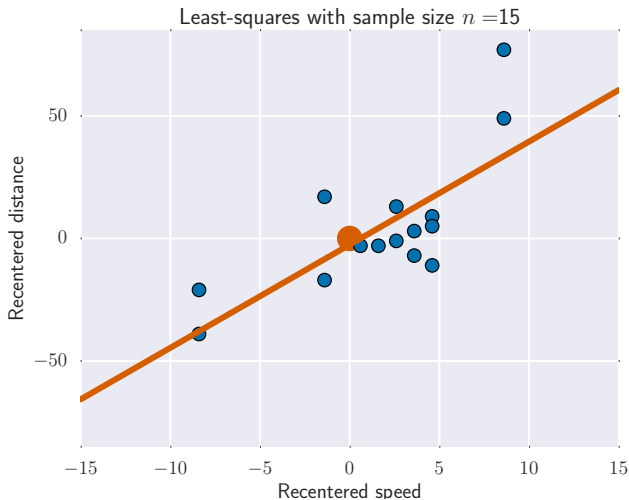
## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)

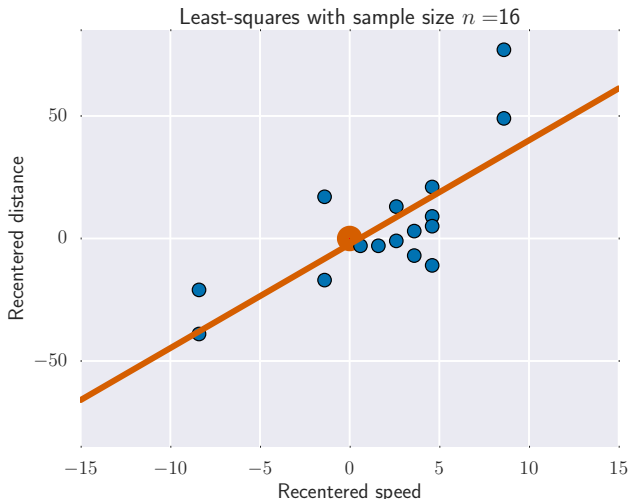


## Extreme points – leverage effect (II)

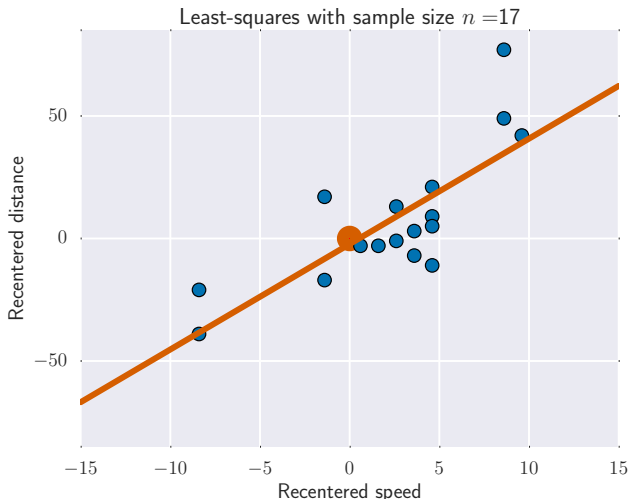




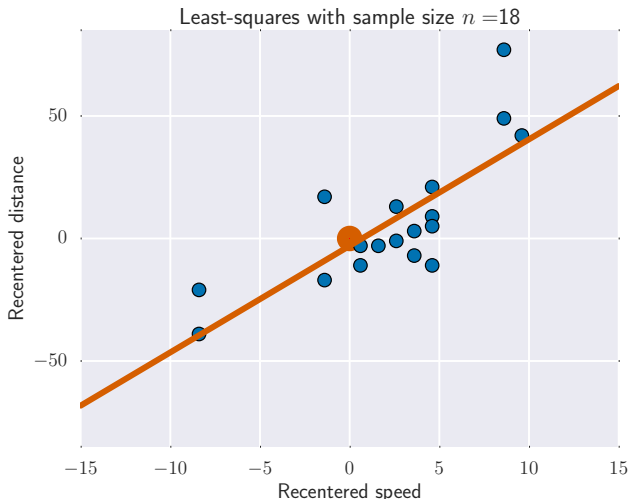
## Extreme points – leverage effect (II)



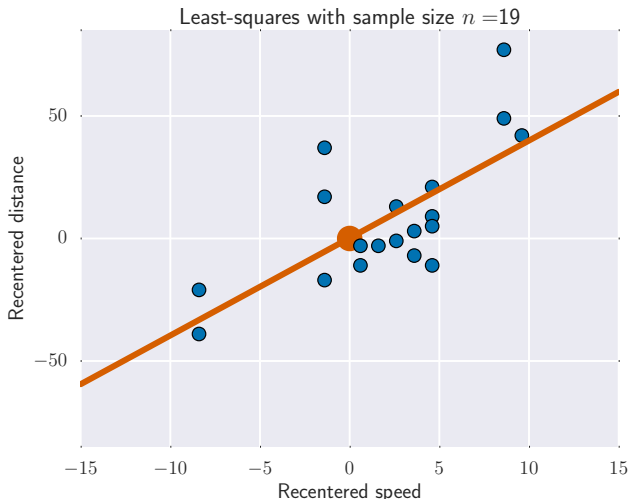
## Extreme points – leverage effect (II)



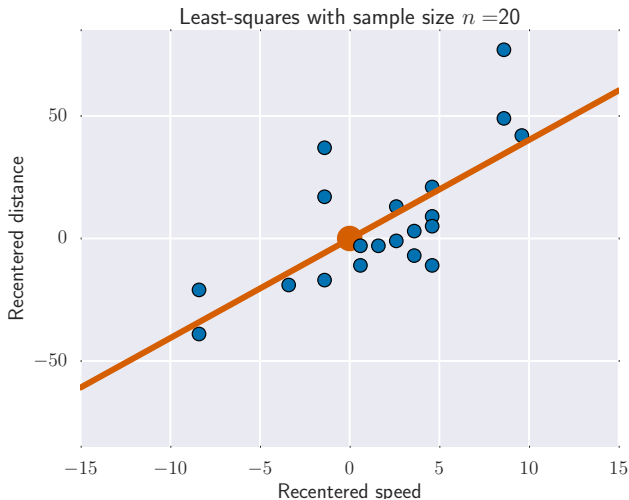
## Extreme points – leverage effect (II)



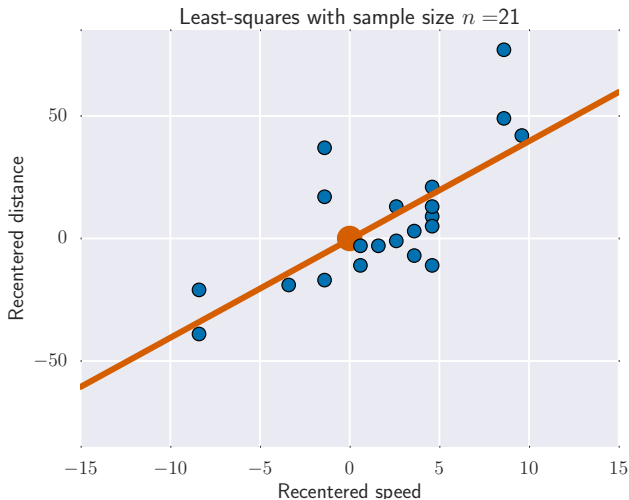
## Extreme points – leverage effect (II)



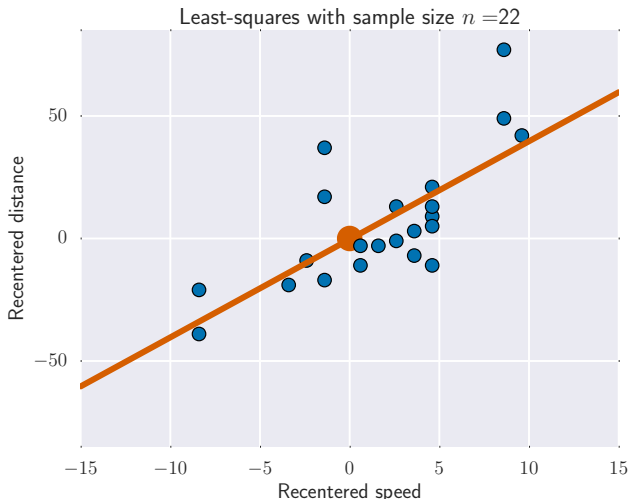
## Extreme points – leverage effect (II)



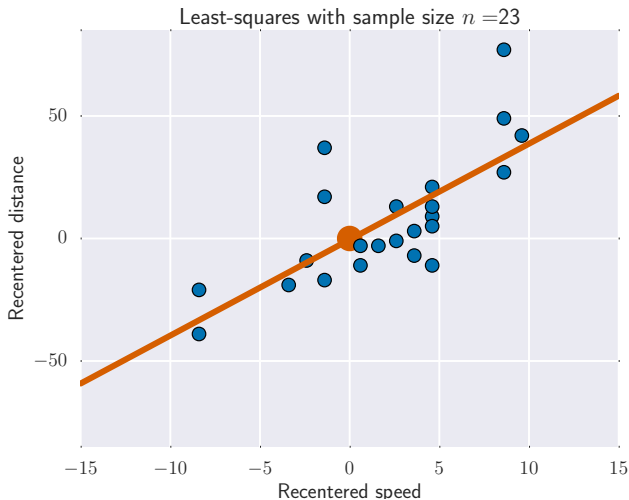
## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)

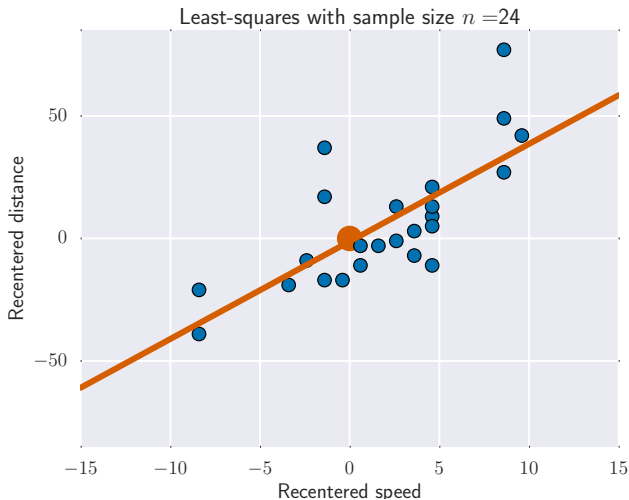


## Extreme points – leverage effect (II)

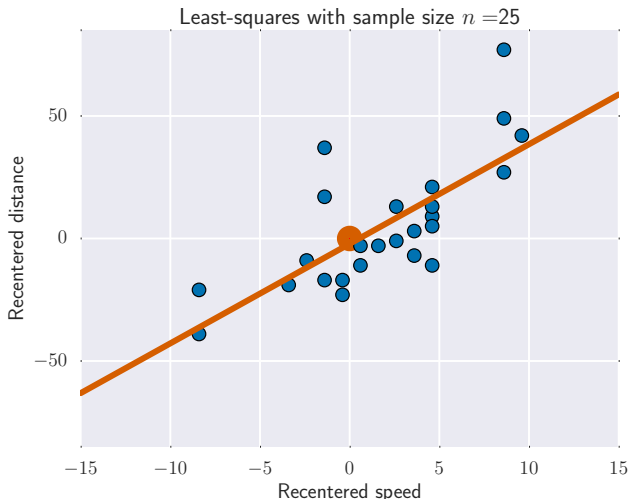




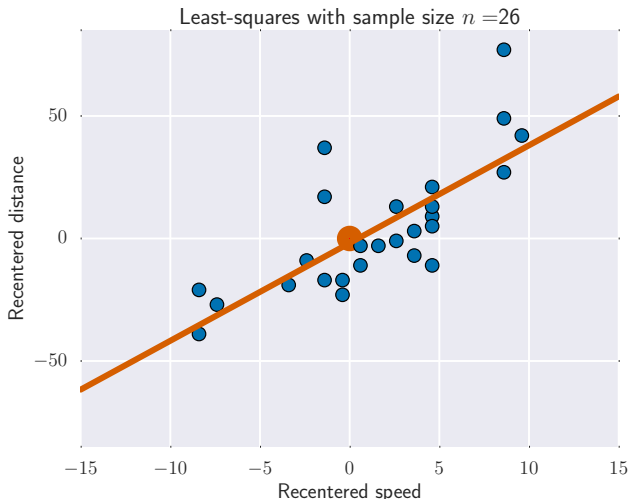
## Extreme points – leverage effect (II)



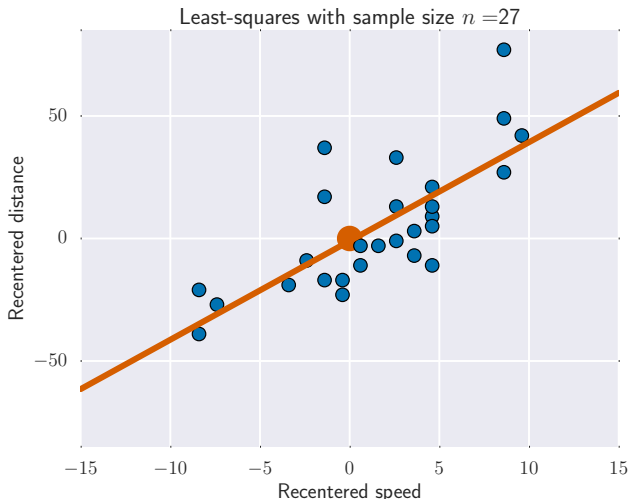
## Extreme points – leverage effect (II)



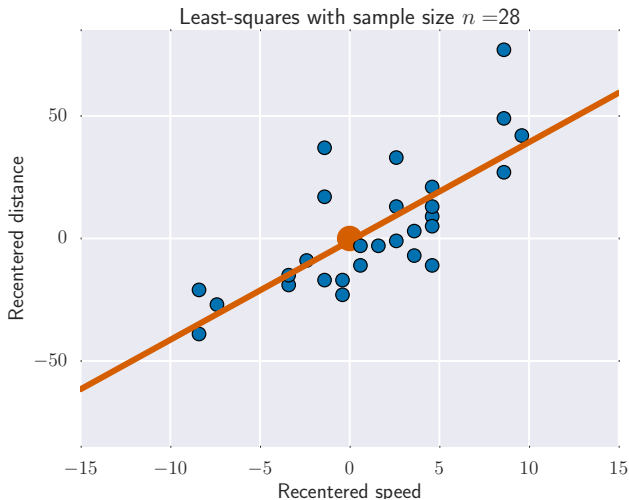
## Extreme points – leverage effect (II)



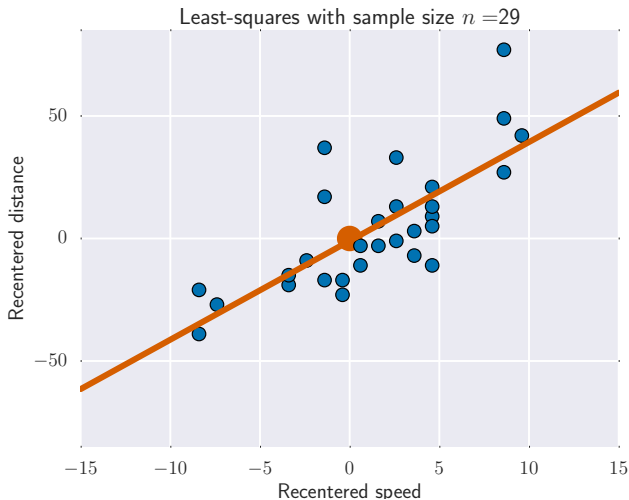
## Extreme points – leverage effect (II)



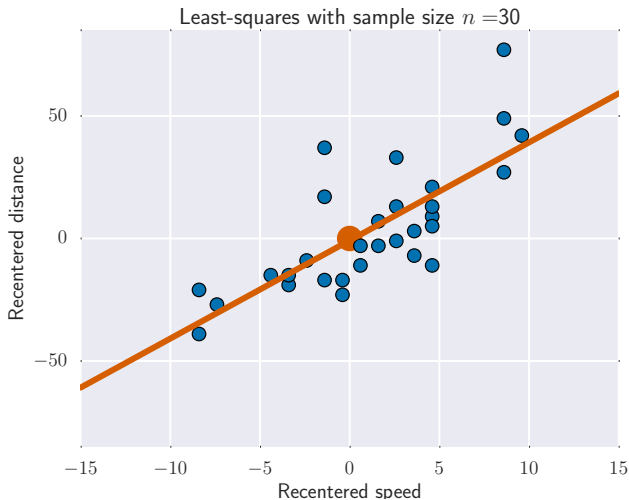
## Extreme points – leverage effect (II)



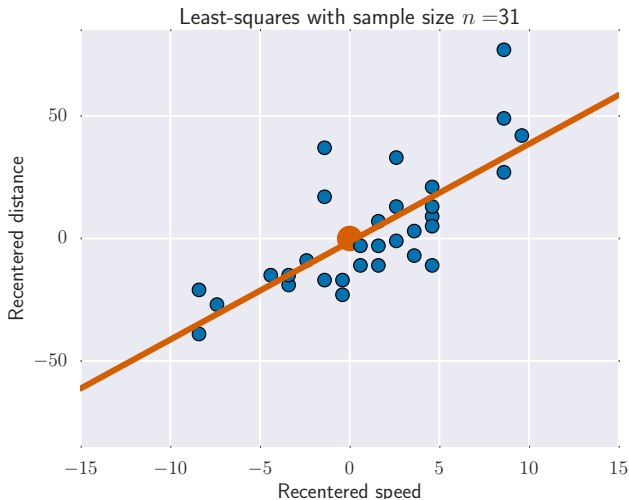
## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)

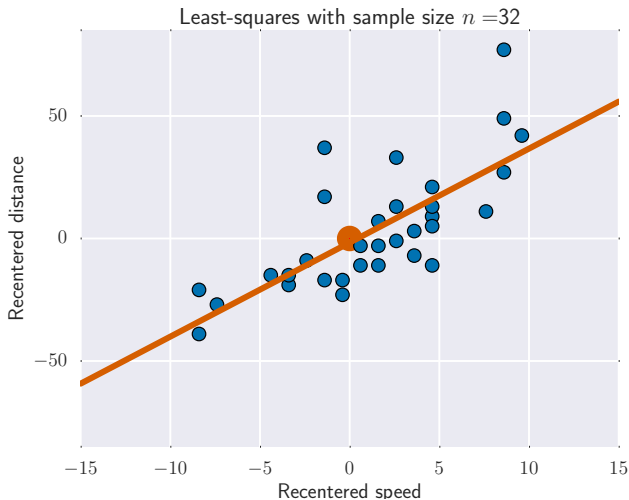


## Extreme points – leverage effect (II)

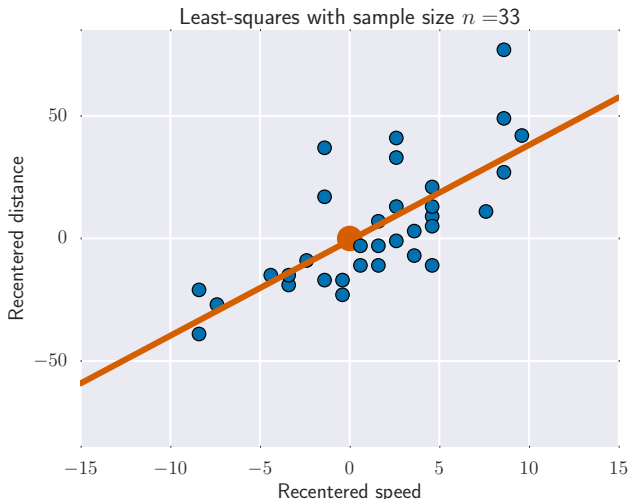




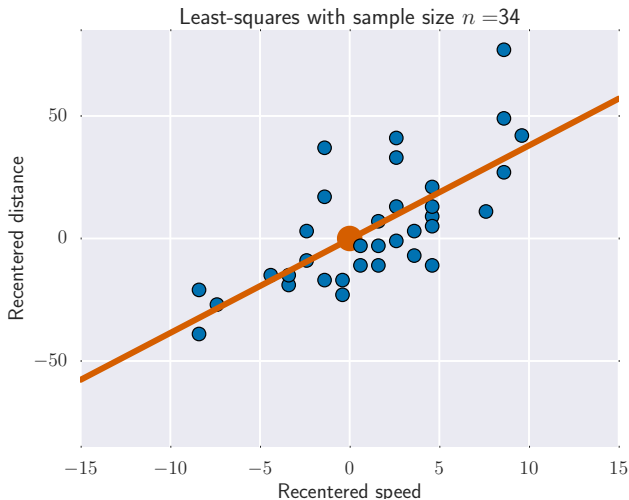
## Extreme points – leverage effect (II)



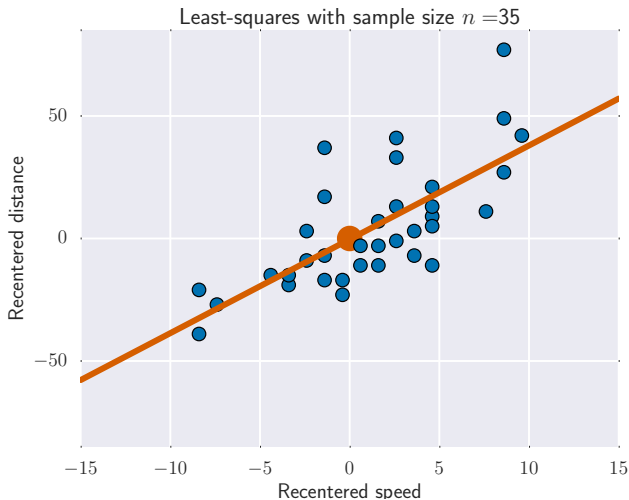
## Extreme points – leverage effect (II)



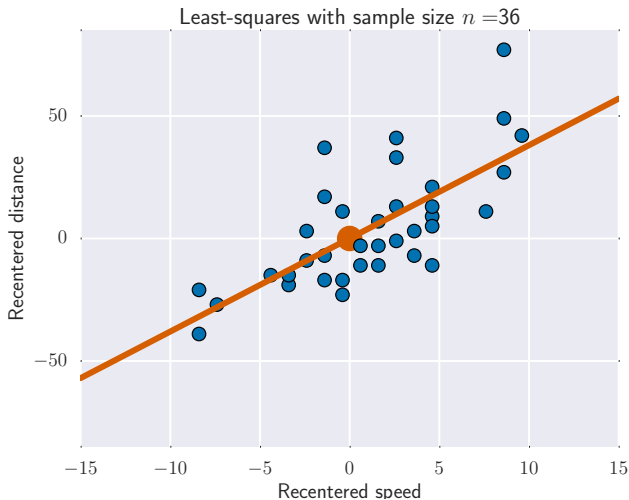
## Extreme points – leverage effect (II)



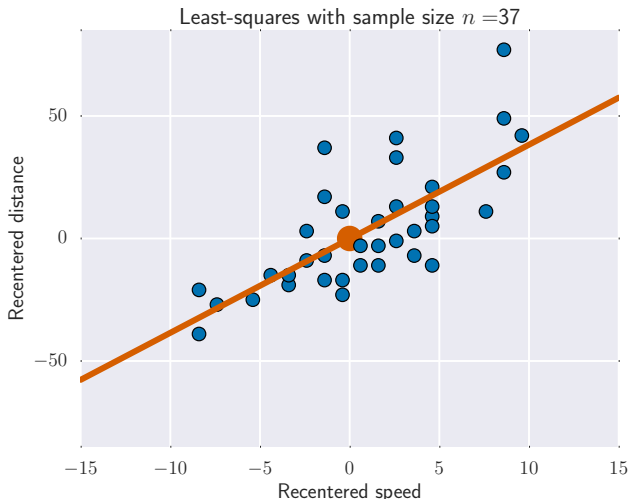
## Extreme points – leverage effect (II)



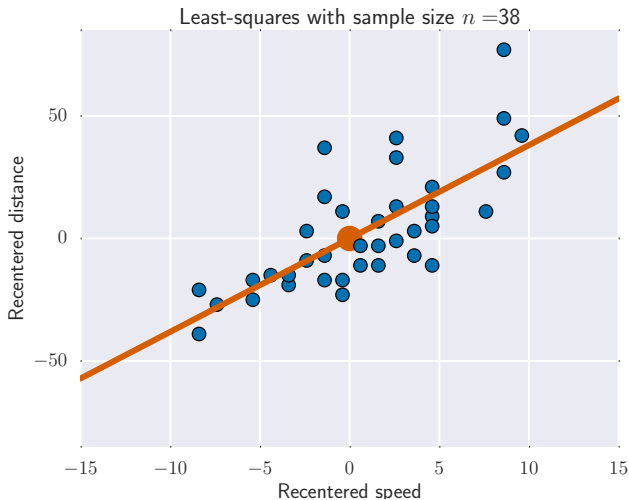
## Extreme points – leverage effect (II)



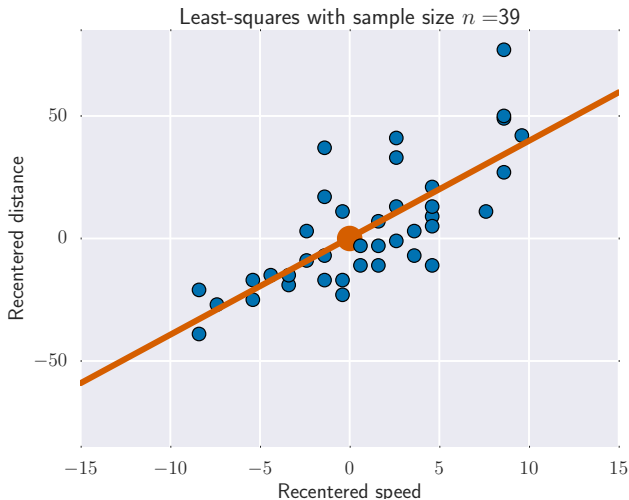
## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)

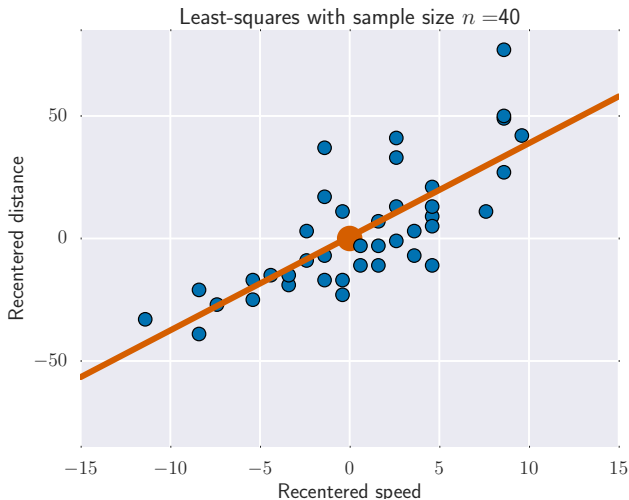


## Extreme points – leverage effect (II)

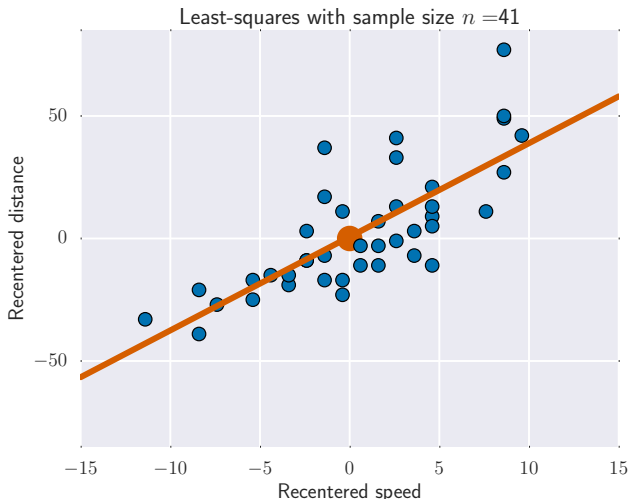




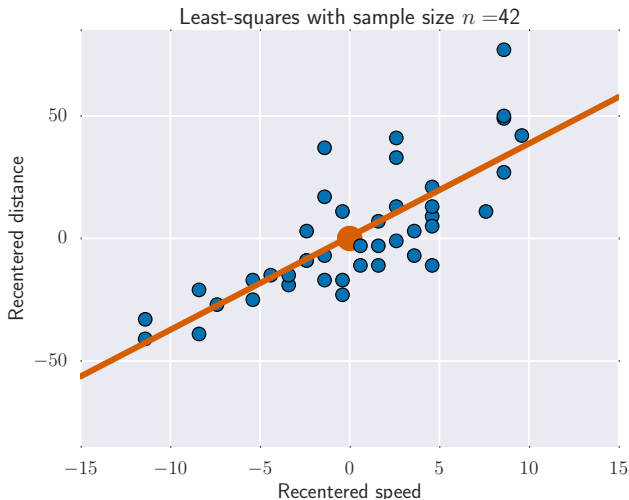
## Extreme points – leverage effect (II)



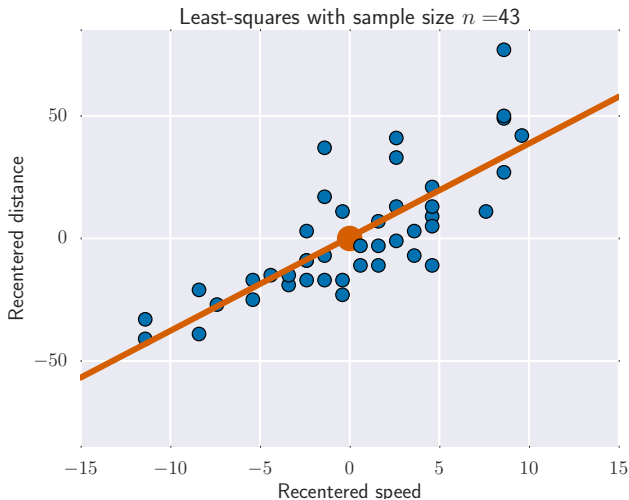
## Extreme points – leverage effect (II)



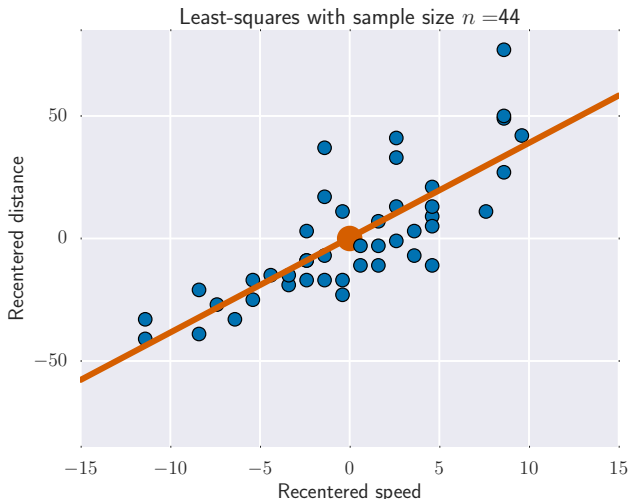
## Extreme points – leverage effect (II)



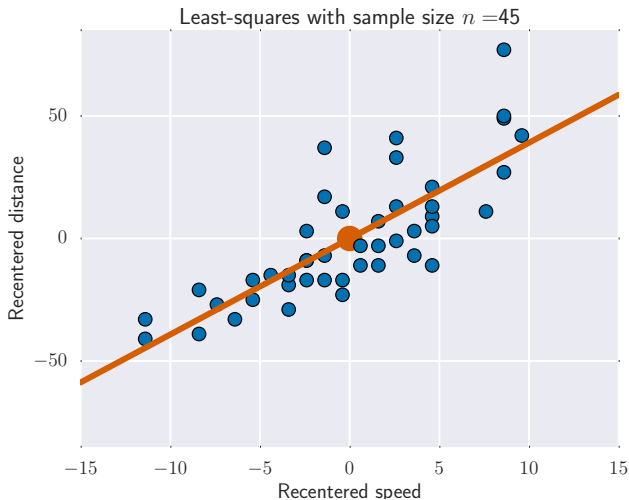
## Extreme points – leverage effect (II)



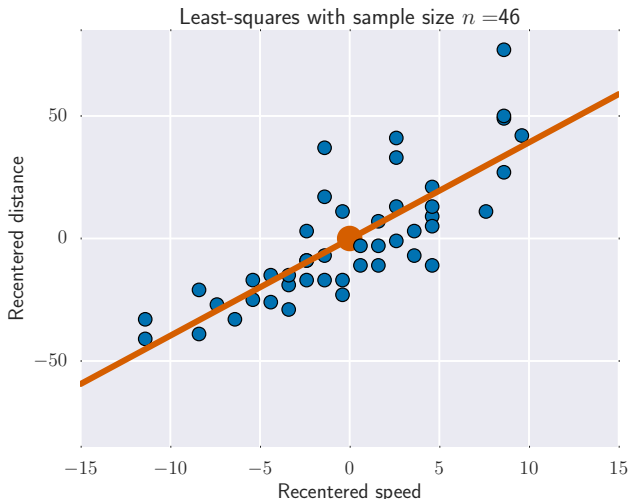
## Extreme points – leverage effect (II)



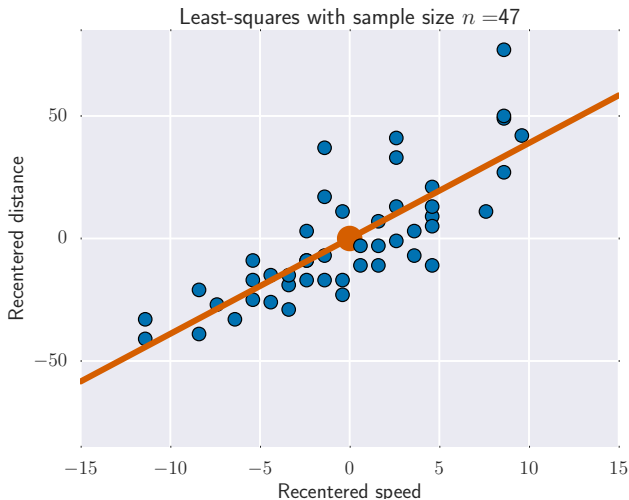
## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)

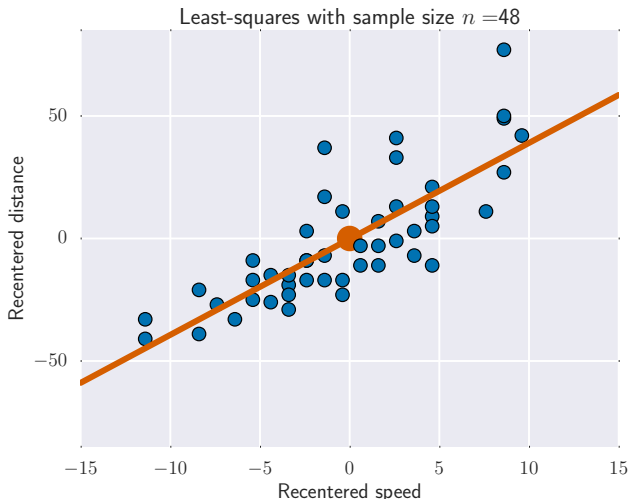


## Extreme points – leverage effect (II)

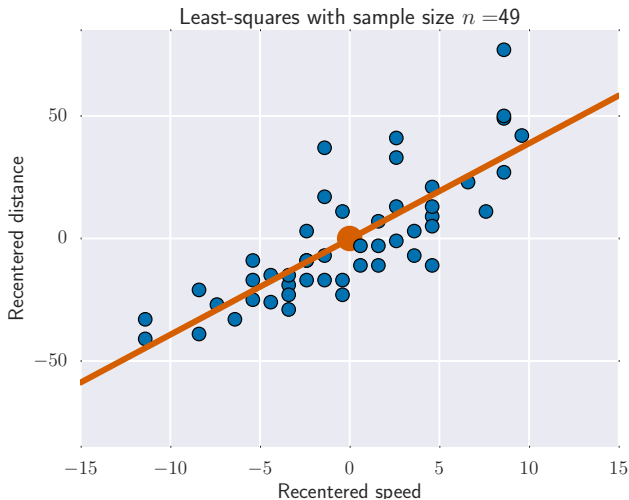




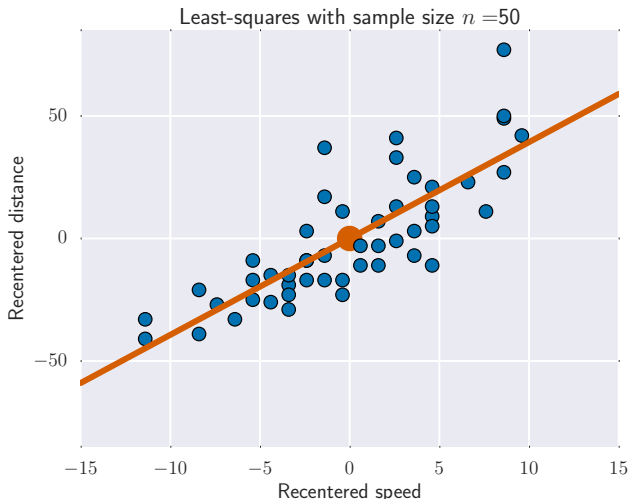
## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)



## Extreme points – leverage effect (II)



## Centering + scaling

Centered-scaled model :

$$\forall i = 1, \dots, n : \begin{cases} x_i'' = (x_i - \bar{x}_n) / \sqrt{\text{var}_n(\mathbf{x})} \\ y_i'' = (y_i - \bar{y}_n) / \sqrt{\text{var}_n(\mathbf{y})} \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}'' = \frac{\mathbf{x} - \bar{x}_n \mathbf{1}_n}{\sqrt{\text{var}_n(\mathbf{x})}} \\ \mathbf{y}'' = \frac{\mathbf{y} - \bar{y}_n \mathbf{1}_n}{\sqrt{\text{var}_n(\mathbf{y})}} \end{cases}$$

Solving OLS with  $(\mathbf{x}'', \mathbf{y}'')$  then

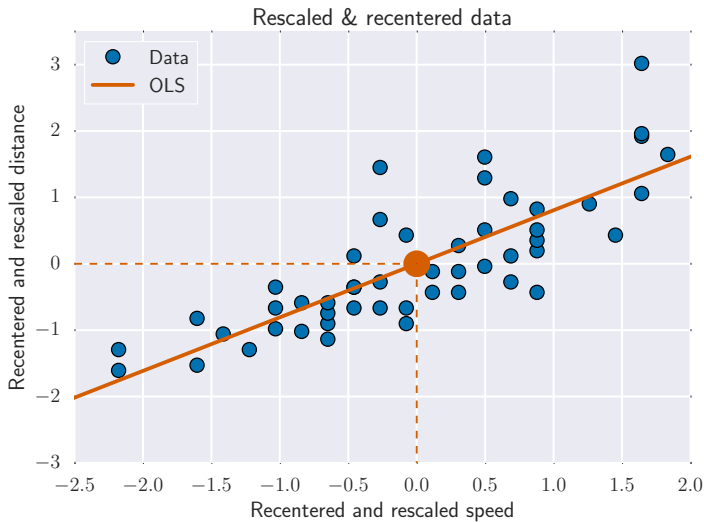
$$\begin{cases} \hat{\theta}_0'' = 0 \\ \hat{\theta}_1'' = \frac{1}{n} \sum_{i=1}^n x_i'' y_i'' \end{cases}$$

Rem: equivalent to choosing the points cloud center of mass as origin and normalize  $\mathbf{x}$  and  $\mathbf{y}$  to have unit **empirical norm**  $\|\cdot\|_n$  :

$$\|\mathbf{x}''\|_n^2 = \frac{1}{n} \sum_{i=1}^n (x_i'')^2 = 1$$

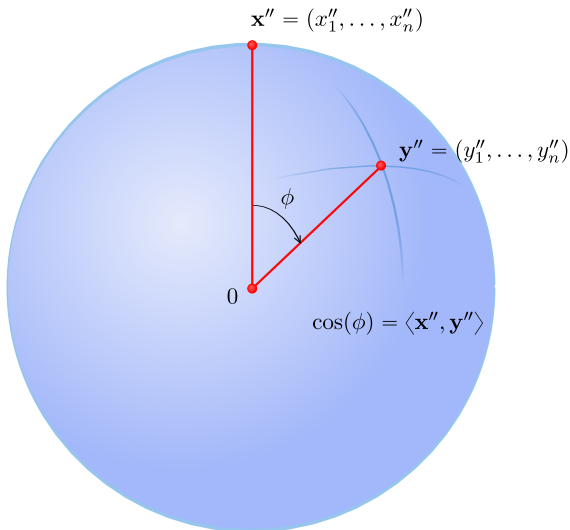
$$\|\mathbf{y}''\|_n^2 = \frac{1}{n} \sum_{i=1}^n (y_i'')^2 = 1$$

# Centering + scaling




# Correlation interpretation (centered-scaled case)

Example :  $n = 3$  and  $\|\mathbf{x}''\|_n^2 = \|\mathbf{y}''\|_n^2 = 1$



# When/why preprocessing ?

Centering  $y$  or using an intercept (or adding a constant feature) is equivalent

Rem: for sparse (  : *creux*) cases centering  $y$  adding a constant feature could be preferred

Scaling features is important :

- ▶ if you want to interpret the coefficients' amplitude in regression (better solution : t-tests)
- ▶ if you want to penalize or regularize coefficients (*cf.* Lasso, Ridge, etc.) a single scale is needed
- ▶ for computing reasons (e.g., store scaling to improve efficiency, improve matrix conditioning, etc.)

Rem: in practice centering/scaling is useful for **estimation** not so much for **prediction** (see next courses)

# Centering with Python

Use centering classes from sklearn, see preprocessing :  
<http://scikit-learn.org/stable/modules/preprocessing.html>

```
from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(X)

print(np.isclose(scaler.mean_, np.mean(X)))

print(np.array_equal(scaler.std_, np.std(X)))

print(np.array_equal(scaler.transform(X),
                     (X - np.mean(X)) / np.std(X)))

print(np.array_equal(scaler.transform([26]),
                     (26 - np.mean(X)) / np.std(X)))
```

Rem: most valuable with pipeline

<http://scikit-learn.org/stable/modules/pipeline.html>



# Definitions

## Prediction

We call **prediction** function the function that associates an estimation of the variable of interest to a new sample. For least squares the prediction is given by :

$$\text{pred}(x_{n+1}) = \hat{\theta}_0 + \hat{\theta}_1 x_{n+1}$$

Rem: often written  $\hat{y}_{n+1}$  (implicit dependence on  $x_{n+1}$ )

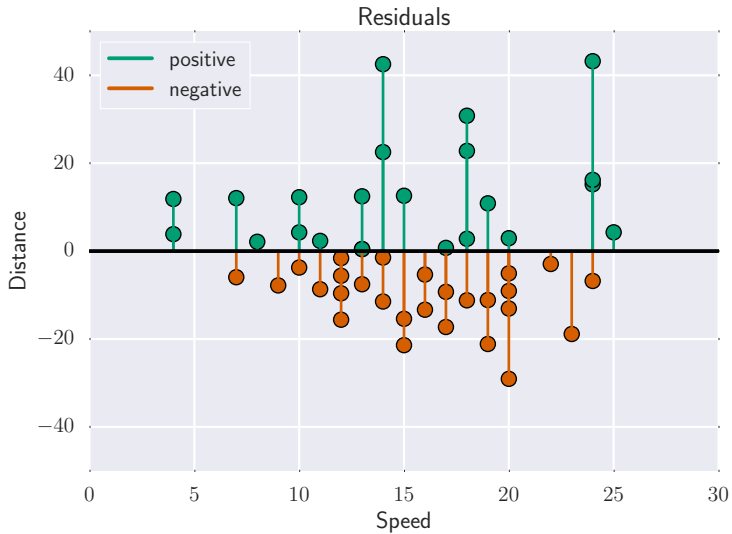
## Residual

The **residual** : difference between observations and predicted values

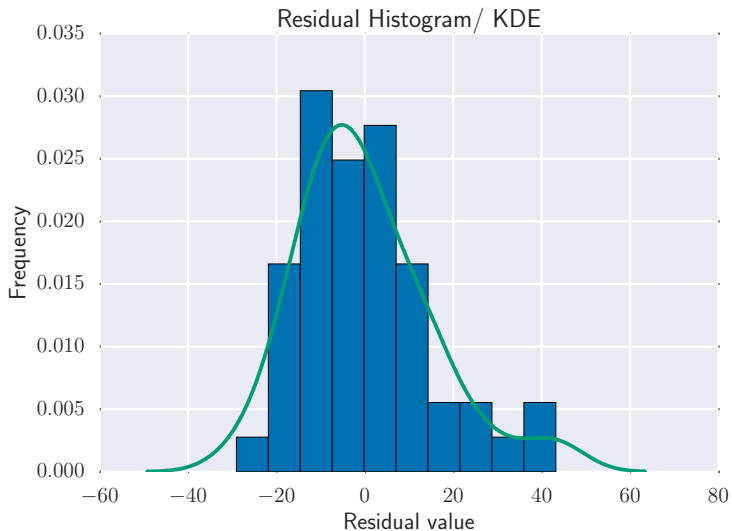
$$r_i = y_i - \text{pred}(x_i) = y_i - \hat{y}_i = y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)$$

Rem: observable estimate of the unobservable statistical error

# Residuals (on cars)



# Residual histograms



## Residuals (continued)

Reminder :  $r_i = y_i - \text{pred}(x_i) = y_i - \hat{y}_i = y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)$

### Properties

Residuals are **centered** :  $\frac{1}{n} \sum_{i=1}^n r_i = 0$

Proof :

## Residuals (continued)

Reminder :  $r_i = y_i - \text{pred}(x_i) = y_i - \hat{y}_i = y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)$

### Properties

Residuals are **centered** :  $\frac{1}{n} \sum_{i=1}^n r_i = 0$

Proof :

$$\frac{1}{n} \sum_{i=1}^n r_i = \frac{1}{n} \sum_{i=1}^n (y_i - \text{pred}(x_i))$$

## Residuals (continued)

Reminder :  $r_i = y_i - \text{pred}(x_i) = y_i - \hat{y}_i = y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)$

### Properties

Residuals are **centered** :  $\frac{1}{n} \sum_{i=1}^n r_i = 0$

Proof :

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n r_i &= \frac{1}{n} \sum_{i=1}^n (y_i - \text{pred}(x_i)) \\ &= \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i) \end{aligned}$$

## Residuals (continued)

Reminder :  $r_i = y_i - \text{pred}(x_i) = y_i - \hat{y}_i = y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)$

### Properties

Residuals are **centered** :  $\frac{1}{n} \sum_{i=1}^n r_i = 0$

Proof :

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n r_i &= \frac{1}{n} \sum_{i=1}^n (y_i - \text{pred}(x_i)) \\ &= \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i) \\ &= \frac{1}{n} \sum_{i=1}^n (y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)) \end{aligned}$$

## Residuals (continued)

Reminder :  $r_i = y_i - \text{pred}(x_i) = y_i - \hat{y}_i = y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)$

### Properties

Residuals are **centered** :  $\frac{1}{n} \sum_{i=1}^n r_i = 0$

Proof :

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n r_i &= \frac{1}{n} \sum_{i=1}^n (y_i - \text{pred}(x_i)) \\ &= \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i) \\ &= \frac{1}{n} \sum_{i=1}^n (y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)) \\ &= \bar{y}_n - (\hat{\theta}_0 + \hat{\theta}_1 \bar{x}_n) = 0 \end{aligned}$$



# Outline

Syllabus, grades, etc.

Teaching staff

Grades and bonus

## 1D Least squares

Introduction : visualization / Python

Modeling

Mathematical Formulation

Centering - scaling

**Likelihood**

# Least squares motivation

- ▶ Computing advantage : computationally heavy methods avoided before computers (e.g., iterative methods)
- ▶ Theoretical advantage : least square analysis easy under simple hypothesis

Example : under additive white Gaussian noise assumption *i.e.*,  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$  the maximum likelihood is equivalent to solving least squares to estimate  $(\theta_0^*, \theta_1^*)$

Rem: for another noise model and/or to limit outliers influence one can solve (see e.g., QuantReg in statsmodels)

$$\hat{\theta} = (\hat{\theta}_0, \hat{\theta}_1) \in \arg \min_{(\theta_0, \theta_1) \in \mathbb{R}^2} \sum_{i=1}^n |y_i - \theta_0 - \theta_1 x_i|$$

# Gaussian likelihood

## Reminder : univariate probability density function (pdf)

We write  $Y \sim \mathcal{N}(\mu, \sigma^2)$ , for a random variable with pdf

$$\varphi_{\mu, \sigma}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mu)^2}{2\sigma^2}\right)$$

Assume :  $y_i \sim \mathcal{N}(\theta_0^* + \theta_1^* x_i, \sigma^2)$ , i.e.,  $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ , then the most **likely** couple  $(\theta_0, \theta_1)$  based on the observations is maximizing the pdf of  $(y_1, \dots, y_n)$

Under an independence hypothesis, this is achieved by solving :

$$(\hat{\theta}_0, \hat{\theta}_1) \in \arg \max_{(\theta_0, \theta_1) \in \mathbb{R}^2} \prod_{i=1}^n \left( \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \theta_0 - \theta_1 x_i)^2}{2\sigma^2}\right) \right)$$

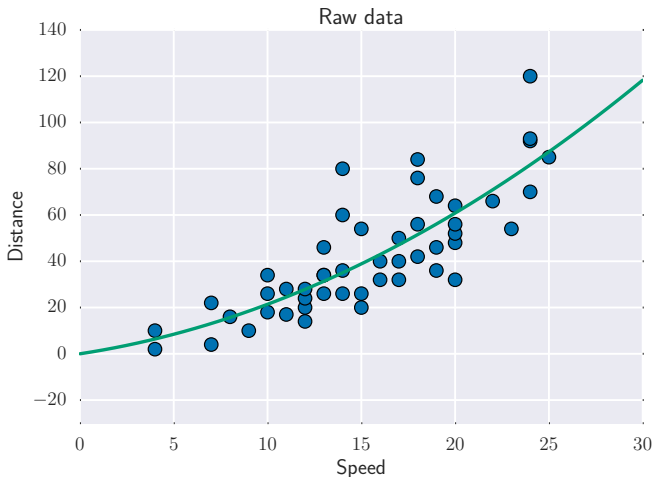
---

**Exo:** check that this is equivalent to the least squares formulation

---

## Discussion : toward multivariate cases

Physical laws (or your driving school memories) would lead to rather pick a **quadratic** model instead of a **linear** one : the OLS can be applied by choosing  $x_i^2$  as features instead of  $x_i$  :



## Web sites and books to go further

- ▶ Datascience in general : Blog + videos by Jake Vanderplas  
<http://jakevdp.github.io/>  
**Homework for next lesson** : watch the following videos  
<http://jakevdp.github.io/blog/2017/03/03/reproducible-data-analysis-in-jupyter/>
- ▶ A few [notebooks](#) of OLS with statsmodels
- ▶ [McKinney \(2012\)](#) about Python for statistics
- ▶ [Lejeune \(2010\)](#) about linear models (in French)
- ▶ Regression course by [B. Delyon](#) (in French, more technical)

# References I

- ▶ D. P. Bertsekas.  
*Nonlinear programming.*  
Athena Scientific, 1999.
- ▶ S. Boyd and L. Vandenberghe.  
*Convex optimization.*  
Cambridge University Press, Cambridge, 2004.
- ▶ B. Delyon.  
Régression, 2015.  
[https://perso.univ-rennes1.fr/bernard.delyon/  
regression.pdf](https://perso.univ-rennes1.fr/bernard.delyon/regression.pdf).
- ▶ D. Foata and A. Fuchs.  
*Calcul des probabilités : cours et exercices corrigés.*  
Masson, 1996.

## References II

- ▶ G. H. Golub and C. F. van Loan.  
*Matrix computations.*  
Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.
- ▶ R. A. Horn and C. R. Johnson.  
*Topics in matrix analysis.*  
Cambridge University Press, Cambridge, 1994.  
Corrected reprint of the 1991 original.
- ▶ M. Lejeune.  
*Statistiques, la théorie et ses applications.*  
Springer, 2010.
- ▶ W. McKinney.  
*Python for Data Analysis : Data Wrangling with Pandas, NumPy, and IPython.*  
O'Reilly Media, 2012.

# References III

- ▶ K. P. Murphy.  
*Machine learning : a probabilistic perspective.*  
MIT press, 2012.