SD-TSIA204
Statistics : linear models

Ekhine Irurozki
Télécom Paris

Grades : SD-TSIA204

- Practical 2 : **20% final grade**
  - Problem statement available on Wednesday 19/01/2021, deadline 04/02/2021 at 00 :01.
  - Single accepted file : **IPython Notebook**
- Final exam : **80% note finale**
  - Date : 02/02/2021
  - 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)
- 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

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), link par L. Vandenberghe
- **Numerical linear algebra** : Introduction to Applied Linear Algebra – Vectors, Matrices, and Least Squares Lecture : Boyd et Vandenberghe (2018), link par Stephen Boyd and Lieven 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/
- **Pandas** : http://pandas.pydata.org/
- **scikit-learn** : http://scikit-learn.org/stable/

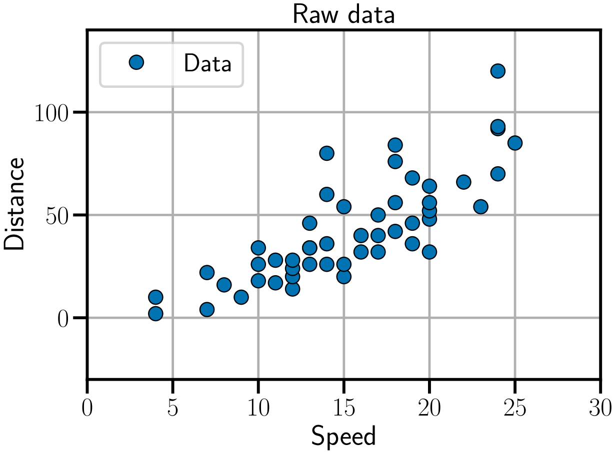
Rem: for practicals, bring your own machine if you prefer but install your `Python` environment upfront

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/kenkoooo/jupyter-autopep8`)
- Learn from good examples :
  `https://github.com/scikit-learn/`,
  `http://jakevdp.github.io/`, etc.

A 2D starting example

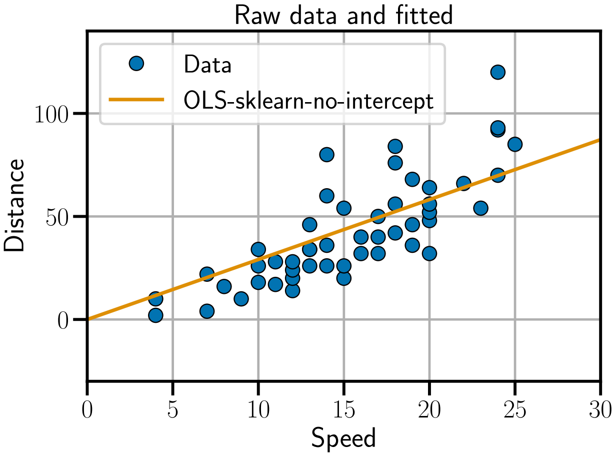
Example : braking distance for cars as a function of speed ($n = 50$ measurements)



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

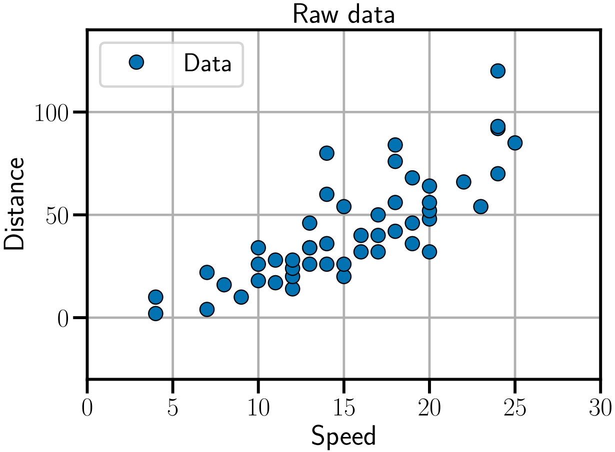
```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.linear_model as lm
# Load data
url = 'cars.csv'
dat = pd.read_csv(url)
y = dat['dist']
X = dat[['speed']] # sklearn needs X to have 2 dim.

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()
```

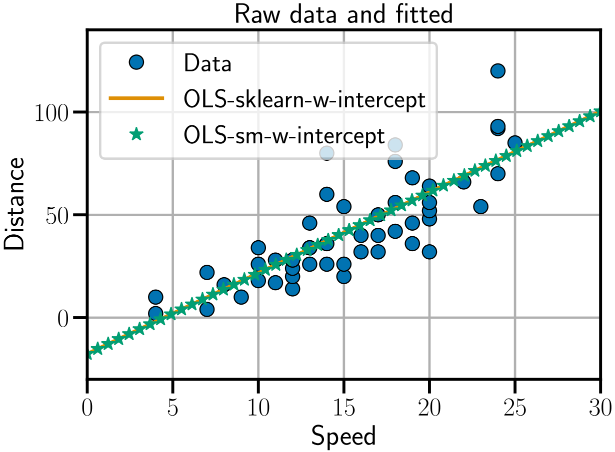
A 2D starting example : with an intercept

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

```python
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)`

Notation interpretation

- $n = 50$
- $p = 1$
- 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

---

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

Modeling I, the 1D case

$$\text{Given a sample :} \quad (y_i, x_i), \text{ for } i = 1, \dots, n$$

Linear model or linear regression hypothesis assume :

$$\boxed{y_i \approx \theta_0^\star + \theta_1^\star x_i}$$

Model

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

Rem: both parameters are unknown from the statistician
Data

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

Modeling II

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

$$
\begin{aligned}
& y_i = \theta_0^\star + \theta_1^\star x_i + \varepsilon_i, \\
& \varepsilon_i \overset{i.i.d}{\sim} \varepsilon, \text{ for } i = 1, \dots, n \\
& \mathbb{E}(\varepsilon) = 0
\end{aligned}
$$

where i.i.d. means "independent and identically distributed"
Interpretation : $\varepsilon_i = y_i - \theta_0^\star - \theta_1^\star x_i$: represent the error between the theoretical model and the observations, represented by random variables ε_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

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

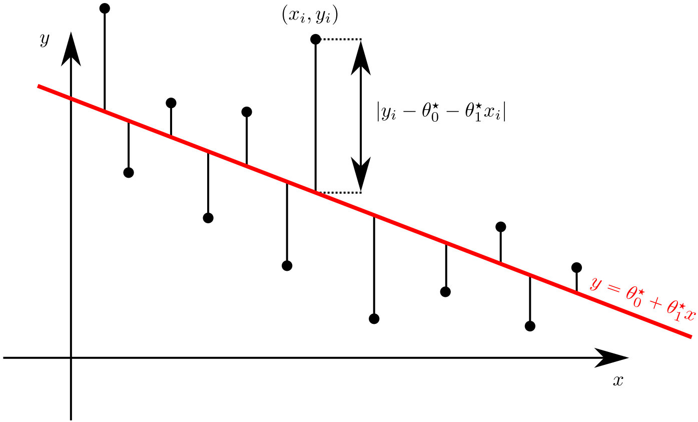
We call

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

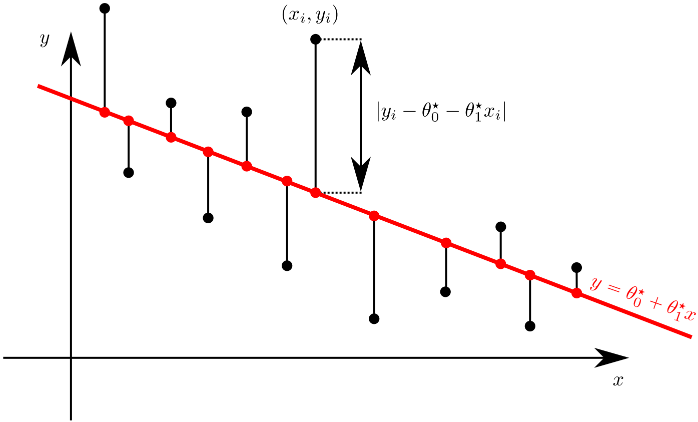
Our **goal** is to estimate $\theta_0^\star$ and $\theta_1^\star$ (unknown) by $\hat{\theta}_0$ and $\hat{\theta}_1$ relying on observations (y_i, x_i) for $i = 1, \ldots, n$

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

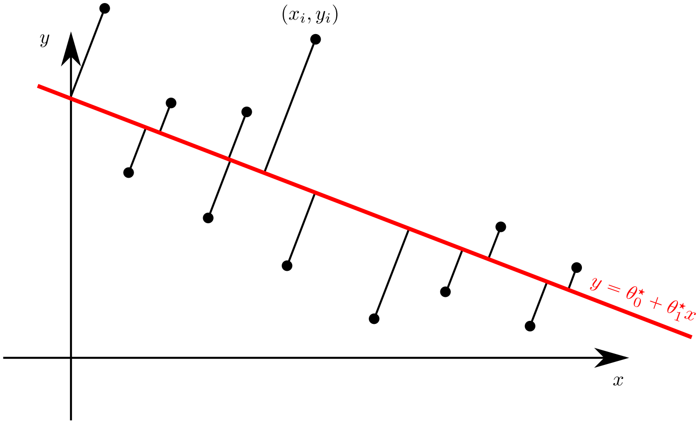
Least squares : visualization



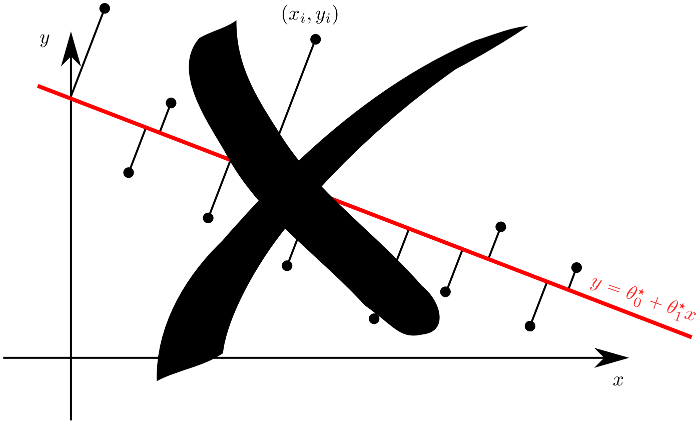
Least squares : visualization



(Total) Least squares : visualization



(Total) Least squares : visualization



Least squares : mathematical formulation

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

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

- 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 : $\lim_{\|x\|\to+\infty} f(x) = +\infty$)

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

Least square authorship (controversial)

FIGURE – Adrien-Marie Legendre and Carl Friedrich Gauss

Historical / robust detour

Definition

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

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

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

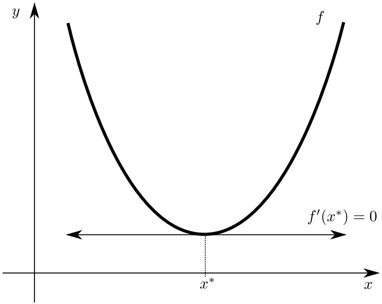
 

FIGURE – Ruđer Josip Bošković and Pierre-Simon de Laplace

Local minimum : first order condition

Theorem : Fermat's rule

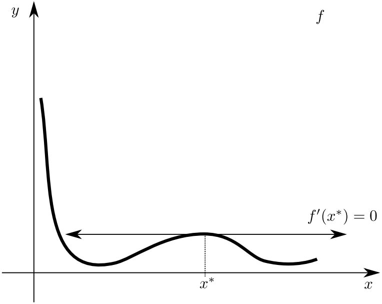
If f is differentiable, then at a local minimum x^* the gradient of f vanishes at x^*, *i.e.*,$\nabla f(x^*) = 0$.



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.*,$\nabla f(x^*) = 0$.

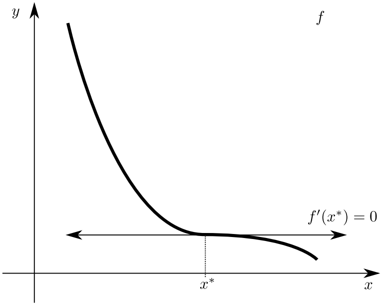


Rem: sufficient condition when f is convex !

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

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

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 :

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

Exercise: Is f convex ? help : a sum of convex functions is convex

---

The solution is unique ($\Leftrightarrow f$ its strictly convex)

F is quadratic $\implies f$ is convex. Moreover, $det(\nabla^2 f(\hat{\boldsymbol{\theta}})) > 0 \Leftrightarrow 0$ is not an eigenvalue of $det(\nabla^2 f(\hat{\boldsymbol{\theta}})) \Leftrightarrow \nabla^2 f(\hat{\boldsymbol{\theta}}) p.s.d. \implies f(\hat{\boldsymbol{\theta}})$ strictly convex $\Leftrightarrow$ unique solution.
For

$$\nabla^2 f(\hat{\boldsymbol{\theta}}) = \begin{bmatrix} 1 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix} \tag{1}$$

we have

$$det(\nabla^2 f(\hat{\boldsymbol{\theta}})) \neq 0 \Leftrightarrow det(\nabla^2 f(\hat{\boldsymbol{\theta}})/n) \neq 0 \Leftrightarrow \frac{\sum x_i^2}{n} - \left(\frac{\sum x_i}{n}\right)^2 \neq 0 \Leftrightarrow$$
$$n^{-1} \sum (x_i - \bar{x})^2 \neq 0 \quad \text{i.e., the variance } \neq 0 \tag{2}$$

Conclussion : uniqueness is guaranteed as long as the variance is different from zero, i.e. the values of x_i are not reduced to a single point.

Calculus continued

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

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

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

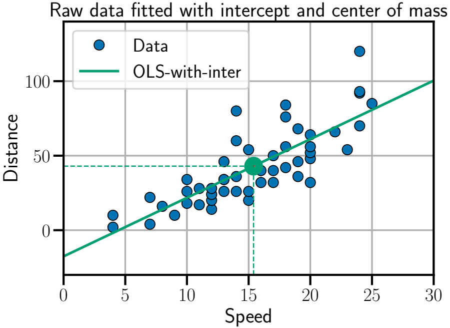
$$\Leftrightarrow$$

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

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

Center of gravity and interpretation

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



- $\overline{speed} = 15.4$
- $\overline{dist} = 42.98$
- $\hat{\theta}_0 = -17.579095$ 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 : $\mathbf{x} = (x_1, \ldots, x_n)^\top$ and $\mathbf{y} = (y_1, \ldots, y_n)^\top$

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

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

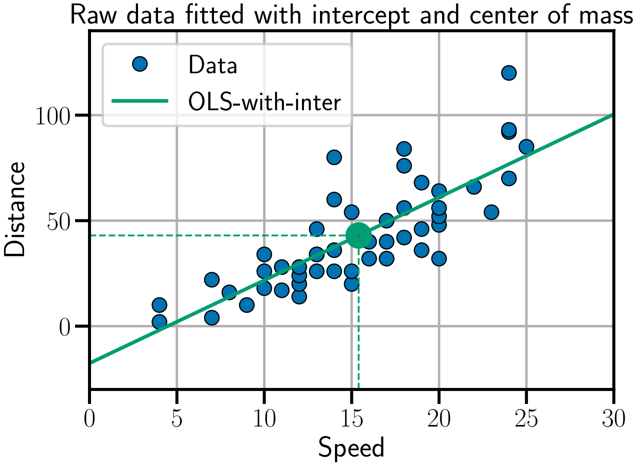
where $$\mathrm{corr}_n(\mathbf{x}, \mathbf{y}) = \frac{\frac{1}{n}\sum_{i=1}^n (x_i - \overline{x}_n)(y_i - \overline{y}_n)}{\sqrt{\mathrm{var}_n(\mathbf{x})}\sqrt{\mathrm{var}_n(\mathbf{y})}}$$

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

respectively **empirical correlation empirical variances**

Back to the *cars* example

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



Centering

Centered model :

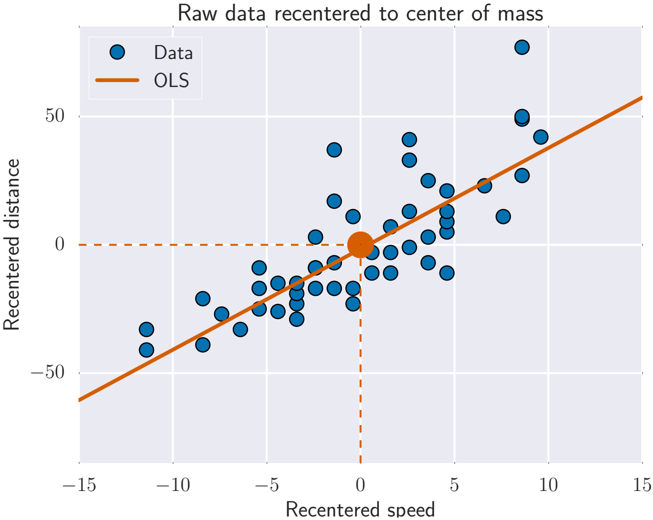
Write for any $i = 1, \ldots, n$: $\begin{cases} x'_i = x_i - \overline{x}_n \\ y'_i = y_i - \overline{y}_n \end{cases} \quad \Leftrightarrow \begin{cases} \mathbf{x}' = \mathbf{x} - \overline{x}_n \mathbf{1}_n \\ \mathbf{y}' = \mathbf{y} - \overline{y}_n \mathbf{1}_n \end{cases}$

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

$$\begin{cases} \widehat{\theta}'_0 = 0 \\ \widehat{\theta}'_1 = \dfrac{\dfrac{1}{n}\displaystyle\sum_{i=1}^n x'_i y'_i}{\dfrac{1}{n}\displaystyle\sum_{i=1}^n x'^2_i} \end{cases}$$

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

Centering (II)



Centering and interpretation

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

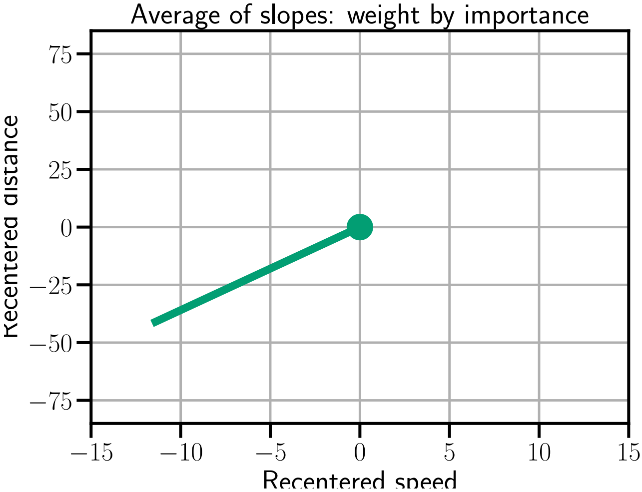
$$\widehat{\theta}'_1 \in \arg\min_{\theta_1} \sum_{i=1}^{n} (y'_i - \theta_1 x'_i)^2 = \arg\min_{\theta_1} \sum_{i=1}^{n} x'^2_i \left(\frac{y'_i}{x'_i} - \theta_1 \right)^2$$

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

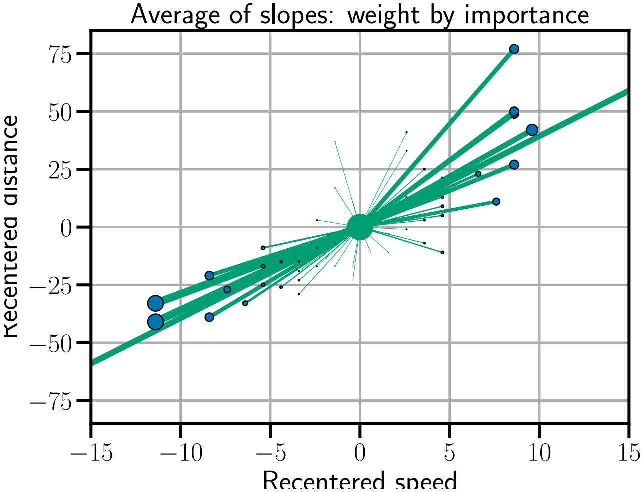
$$\widehat{\theta}'_1 = \frac{\displaystyle\sum_{i=1}^{n} x'^2_i \frac{y'_i}{x'_i}}{\displaystyle\sum_{j=1}^{n} x'^2_j}$$

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

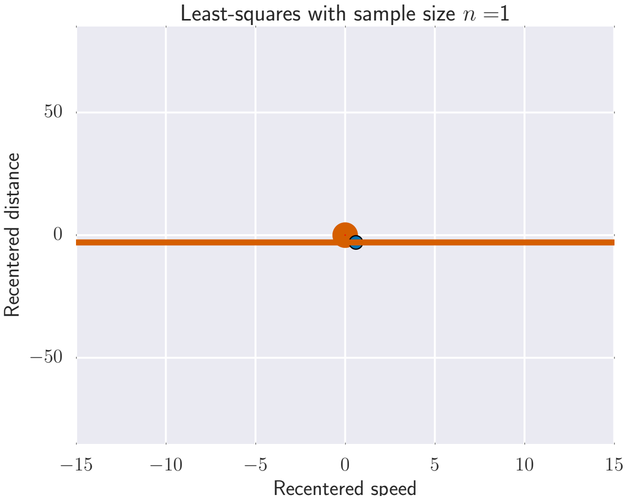
Extreme points – leverage effect



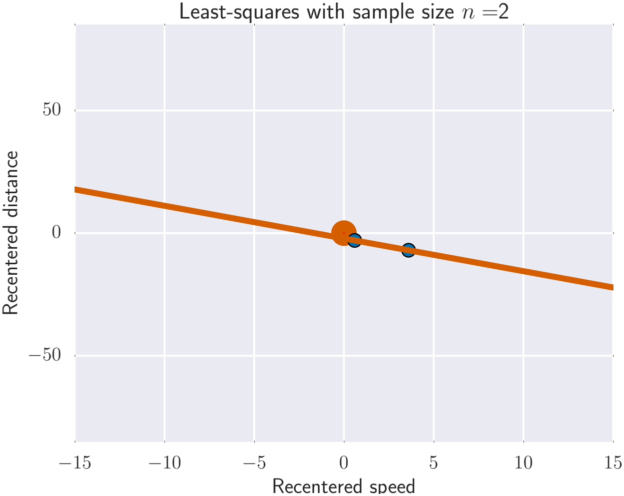
Extreme points – leverage effect



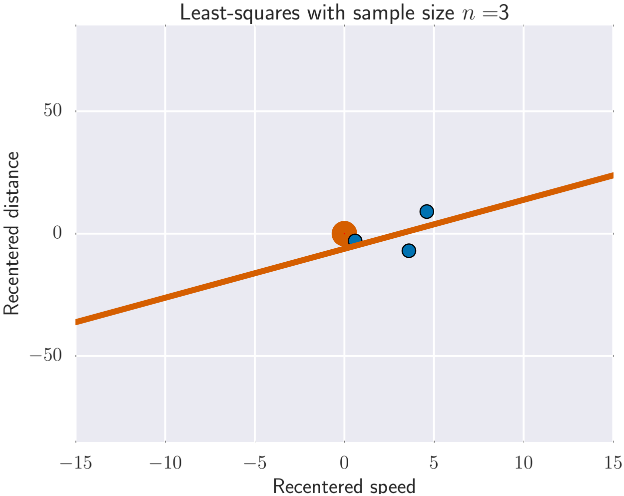
Extreme points – leverage effect (II)



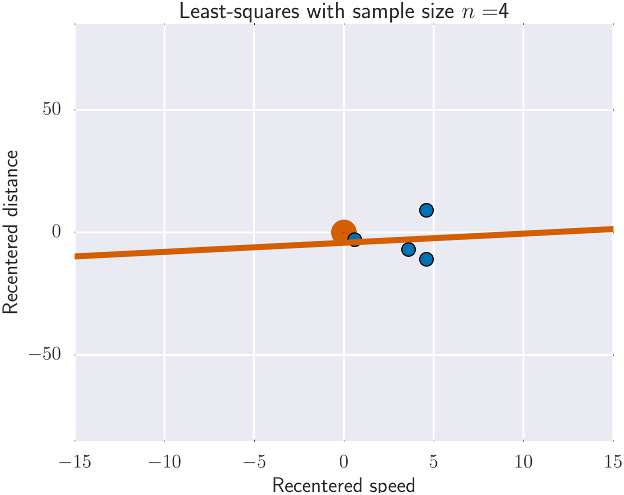
Extreme points – leverage effect (II)



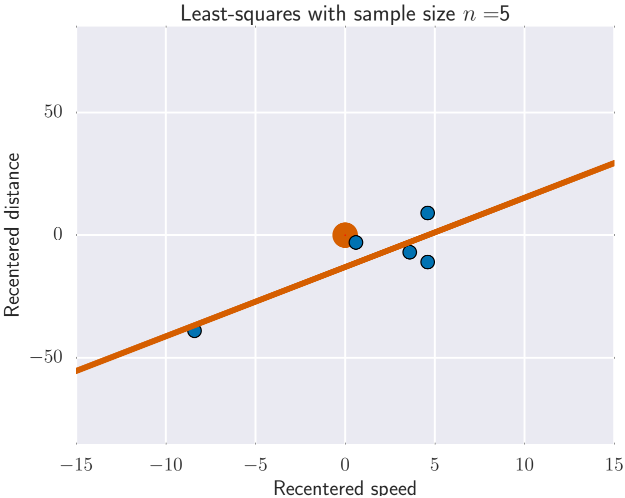
Extreme points – leverage effect (II)



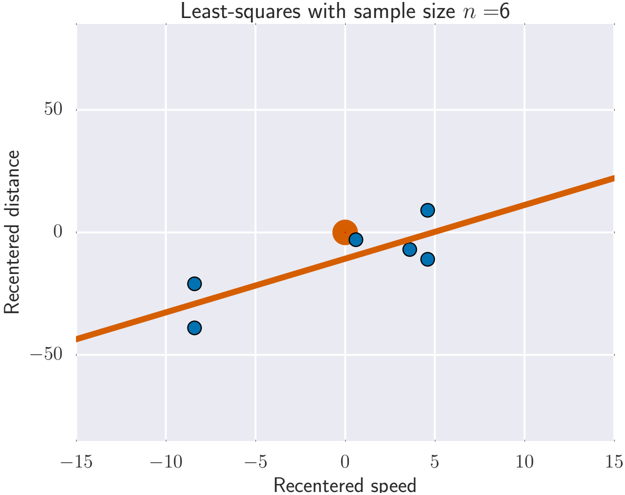
Extreme points – leverage effect (II)



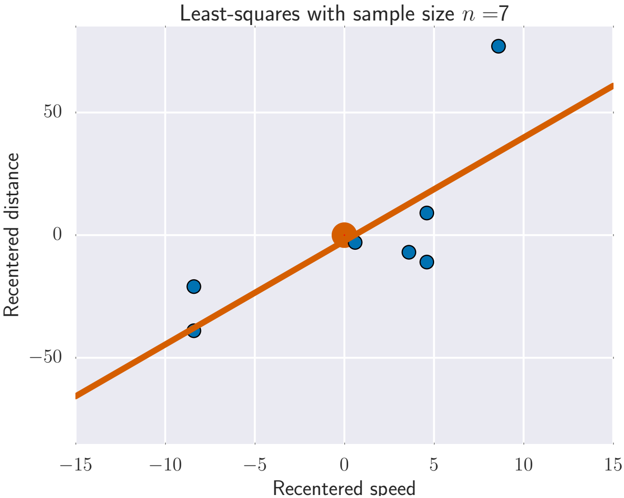
Extreme points – leverage effect (II)



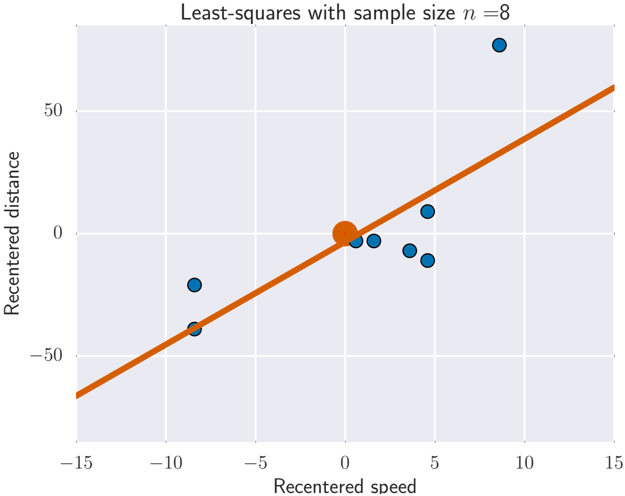
Extreme points – leverage effect (II)



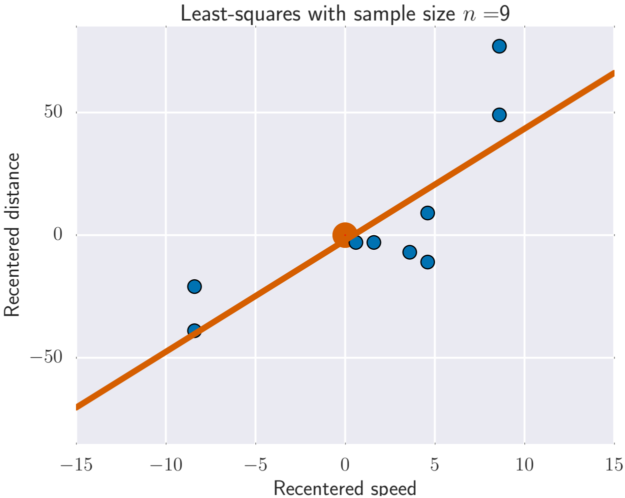
Extreme points – leverage effect (II)



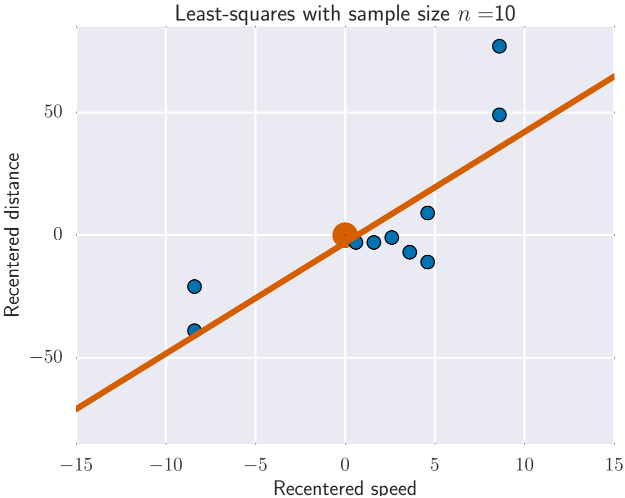
Extreme points – leverage effect (II)



Extreme points – leverage effect (II)

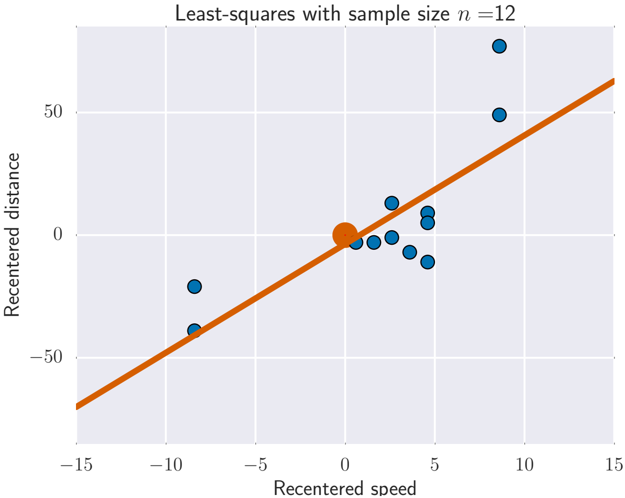


Extreme points – leverage effect (II)



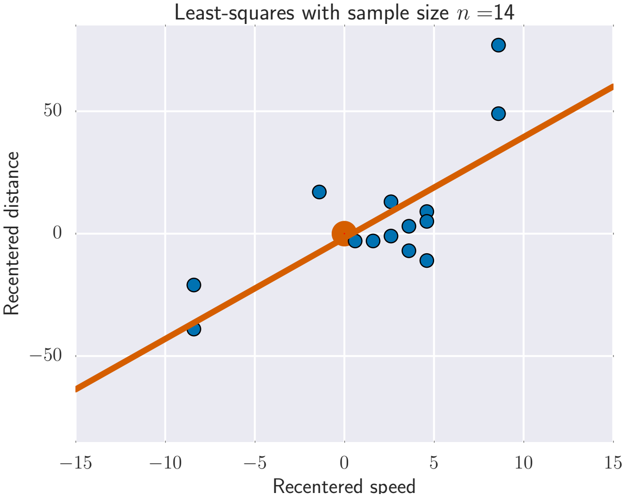
Extreme points – leverage effect (II)

Extreme points – leverage effect (II)

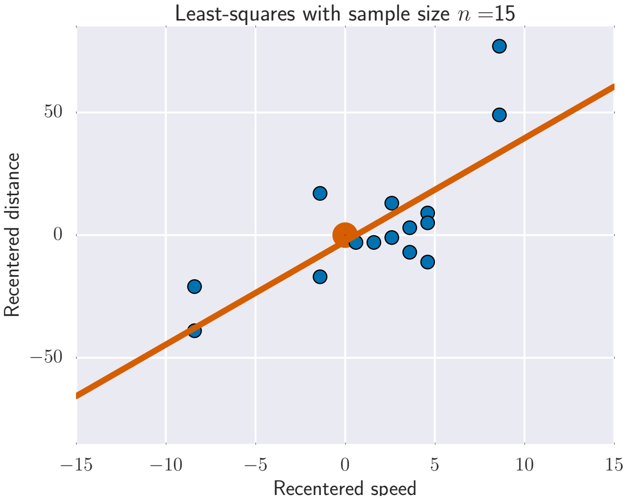


Extreme points – leverage effect (II)

Extreme points – leverage effect (II)

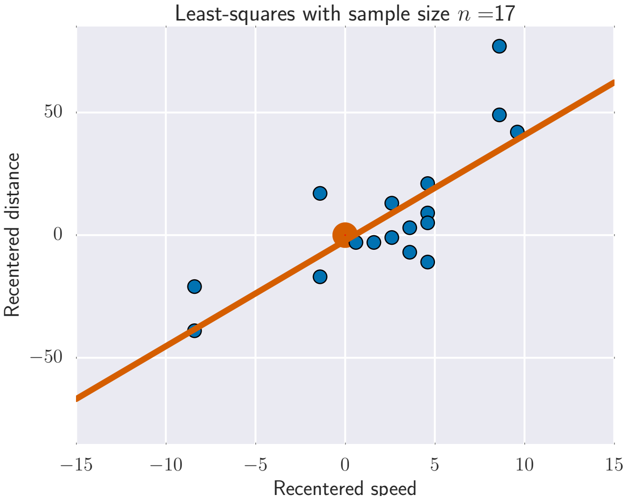


Extreme points – leverage effect (II)

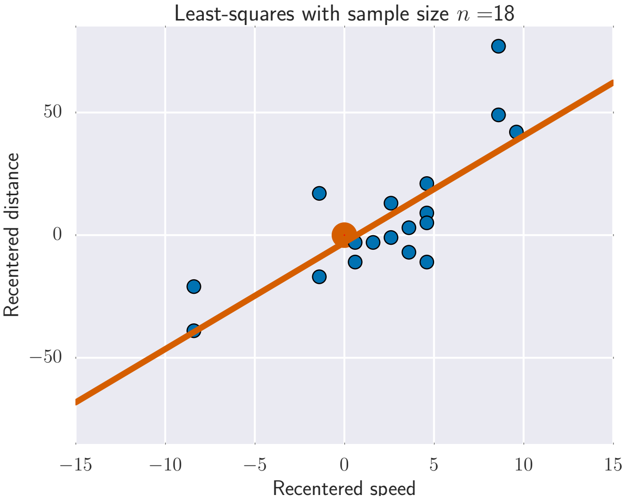


Extreme points – leverage effect (II)

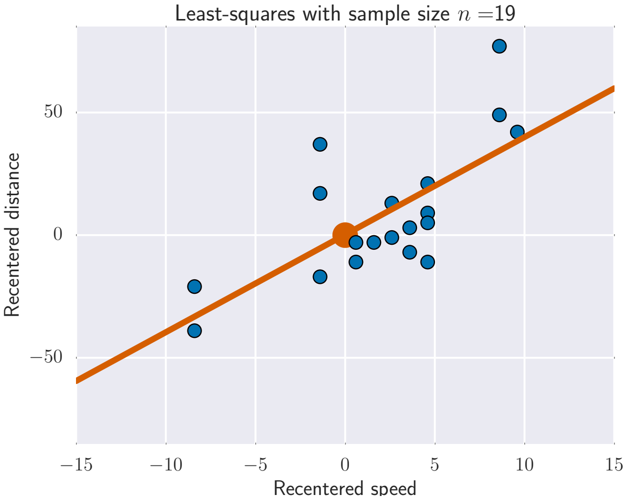
Extreme points – leverage effect (II)



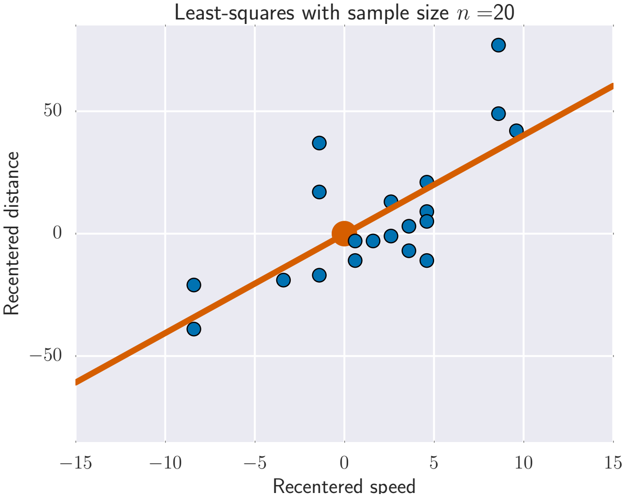
Extreme points – leverage effect (II)



Extreme points – leverage effect (II)

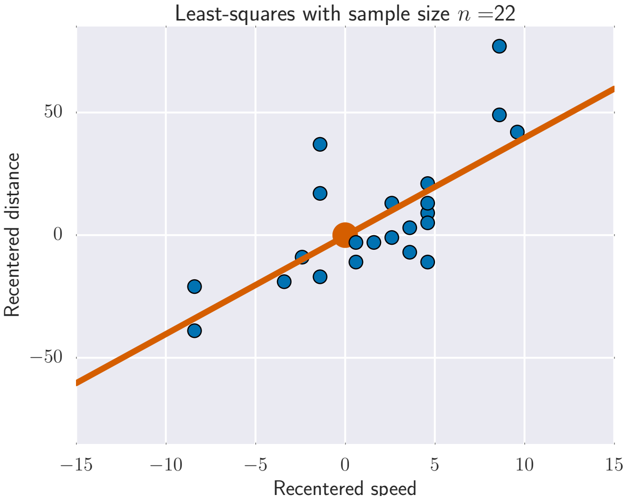


Extreme points – leverage effect (II)

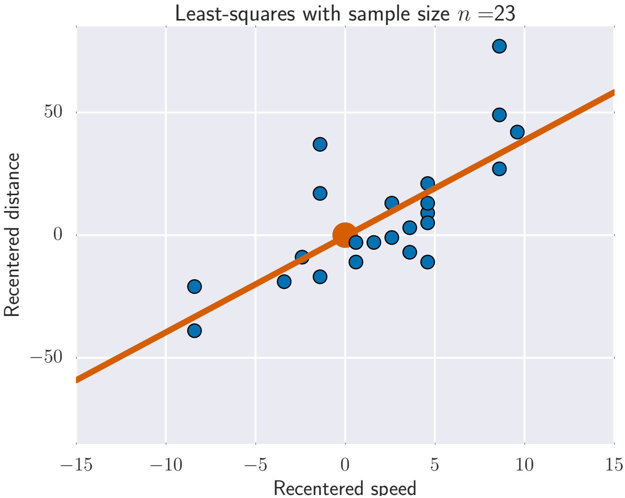


Extreme points – leverage effect (II)

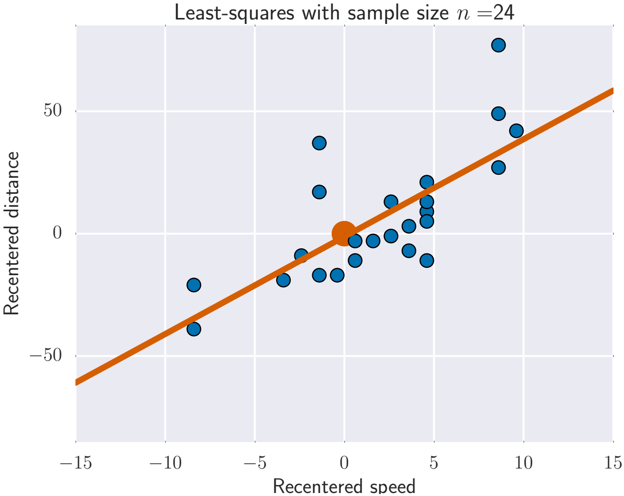
Extreme points – leverage effect (II)



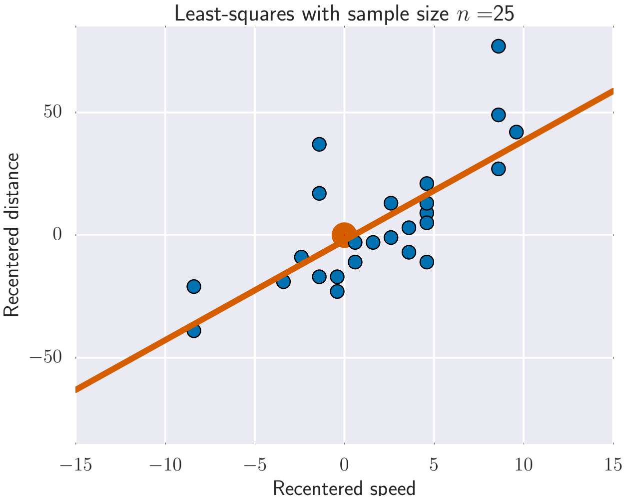
Extreme points – leverage effect (II)



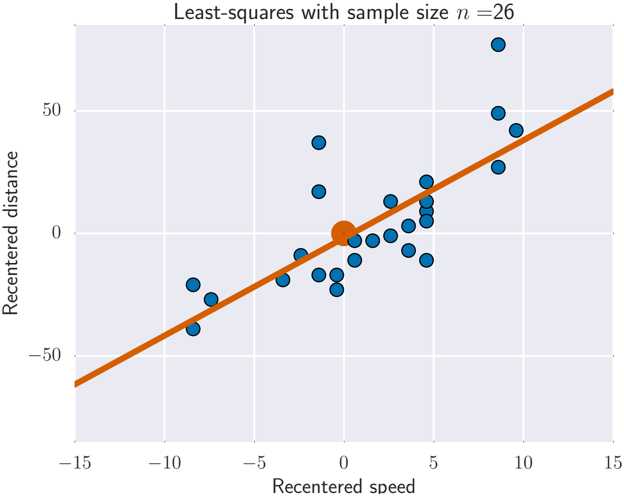
Extreme points – leverage effect (II)



Extreme points – leverage effect (II)



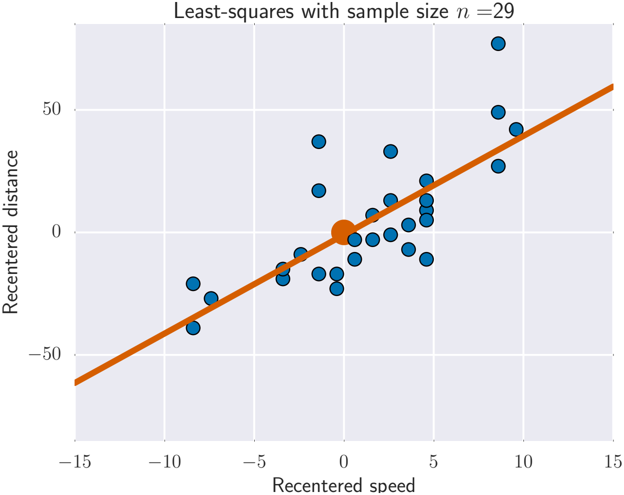
Extreme points – leverage effect (II)



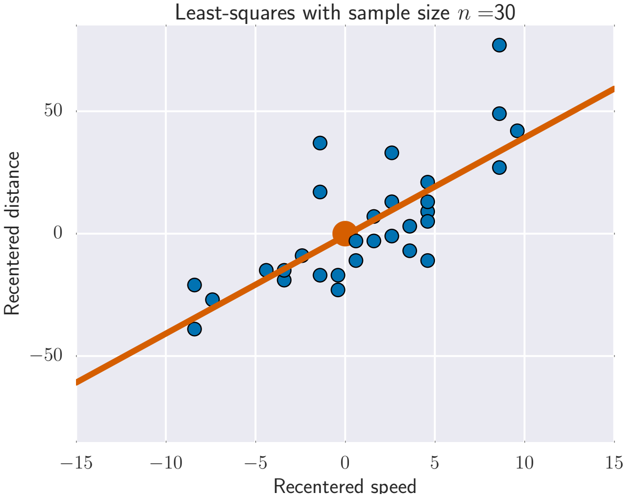
Extreme points – leverage effect (II)

Extreme points – leverage effect (II)

Extreme points – leverage effect (II)

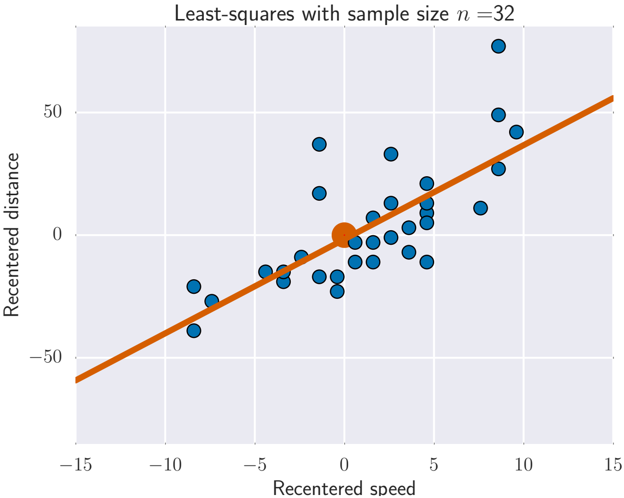


Extreme points – leverage effect (II)



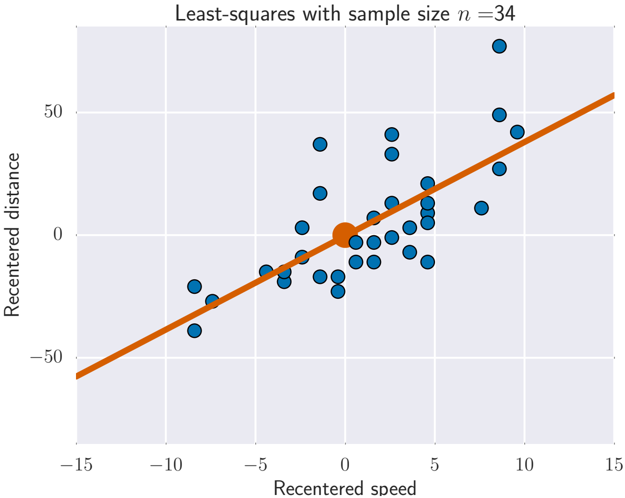
Extreme points – leverage effect (II)

Extreme points – leverage effect (II)

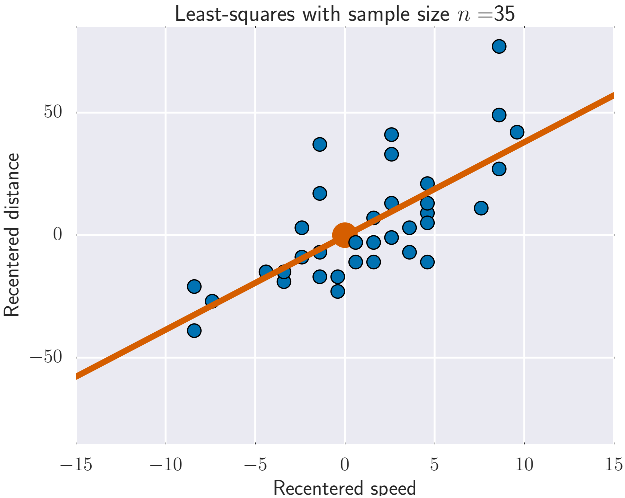


Extreme points – leverage effect (II)

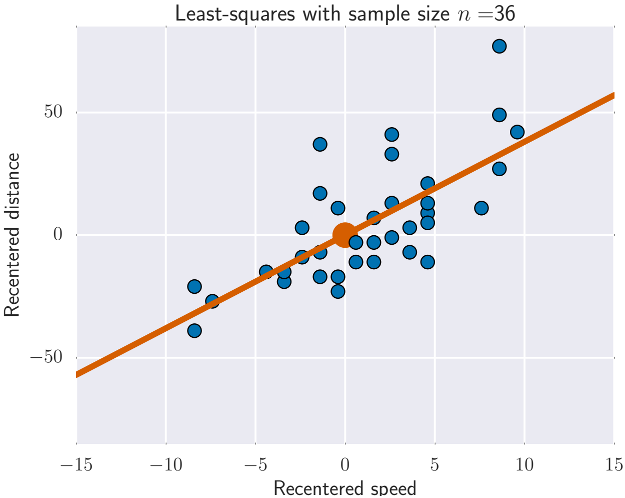
Extreme points – leverage effect (II)



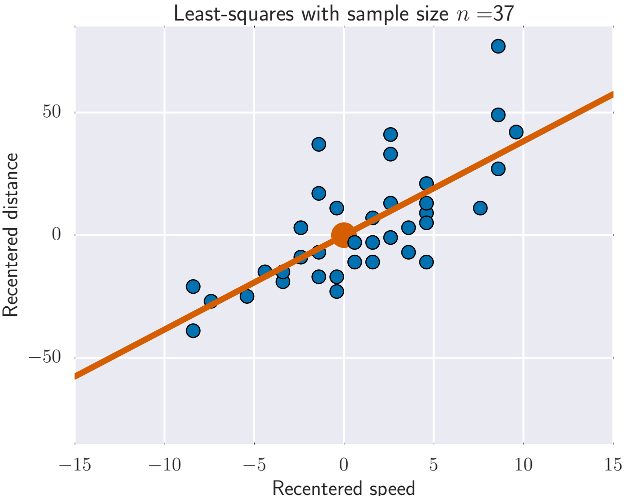
Extreme points – leverage effect (II)



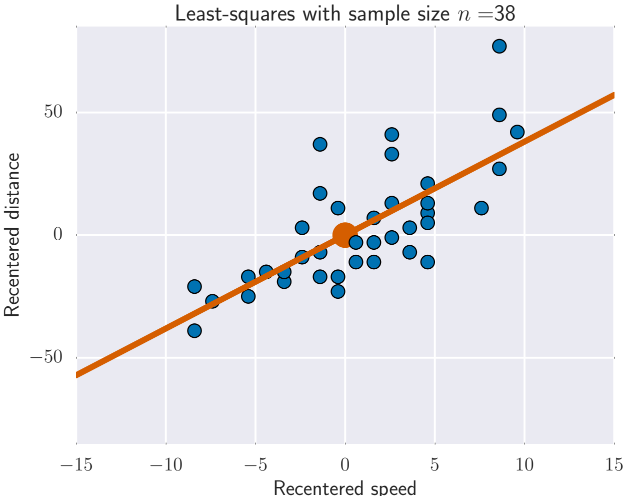
Extreme points – leverage effect (II)



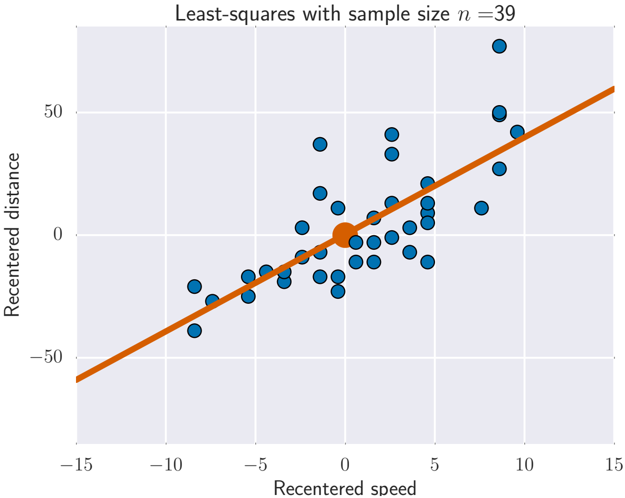
Extreme points – leverage effect (II)



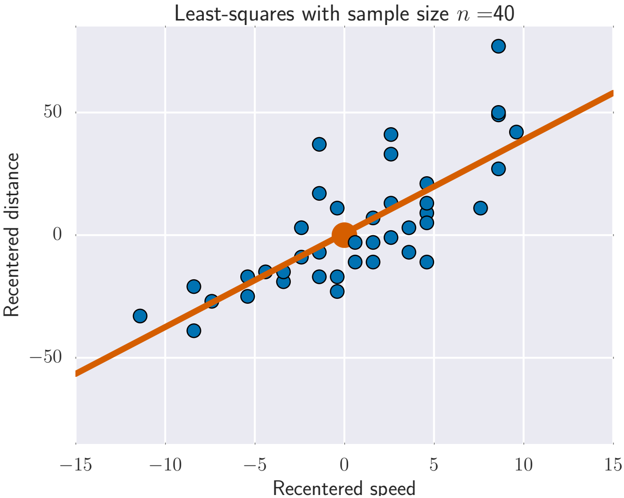
Extreme points – leverage effect (II)



Extreme points – leverage effect (II)

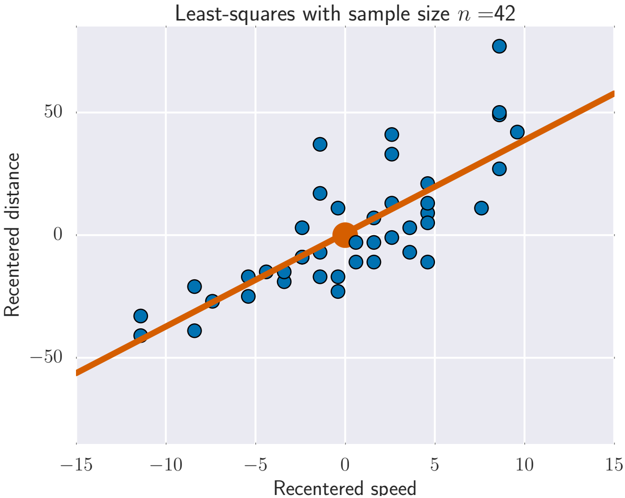


Extreme points – leverage effect (II)

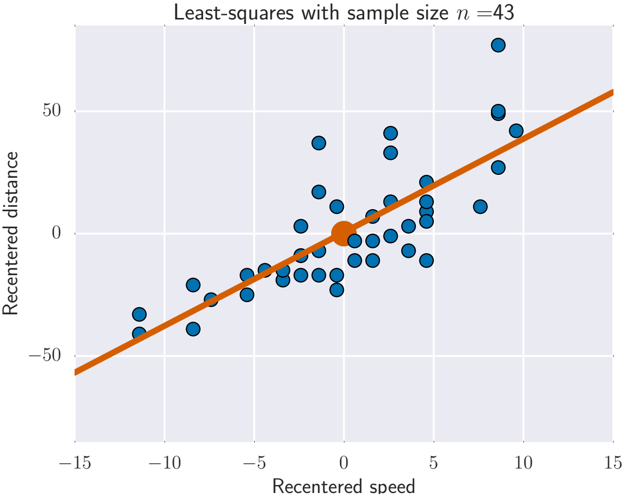


Extreme points – leverage effect (II)

Extreme points – leverage effect (II)

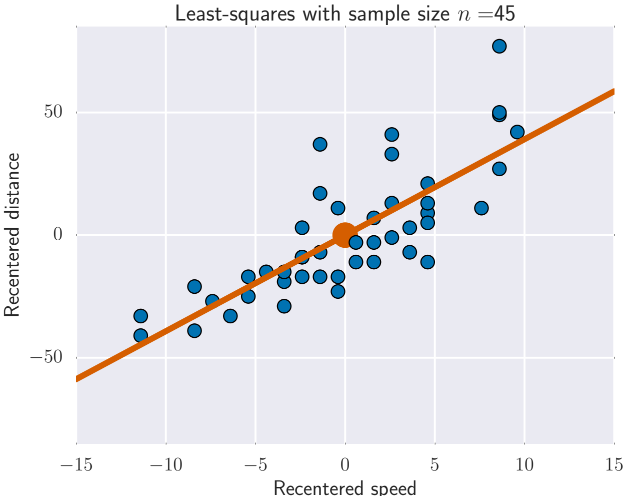


Extreme points – leverage effect (II)

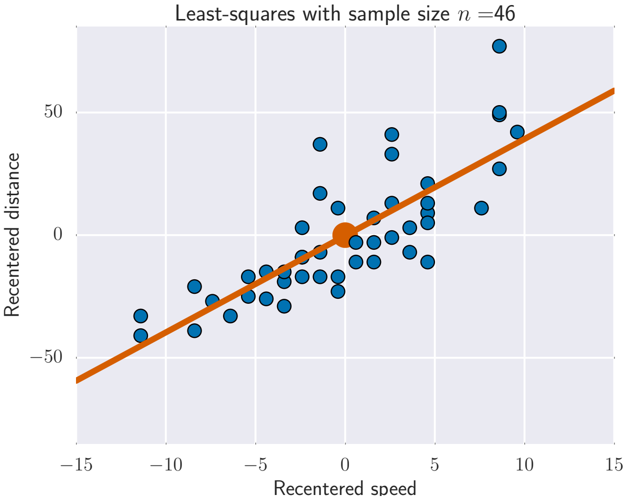


Extreme points – leverage effect (II)

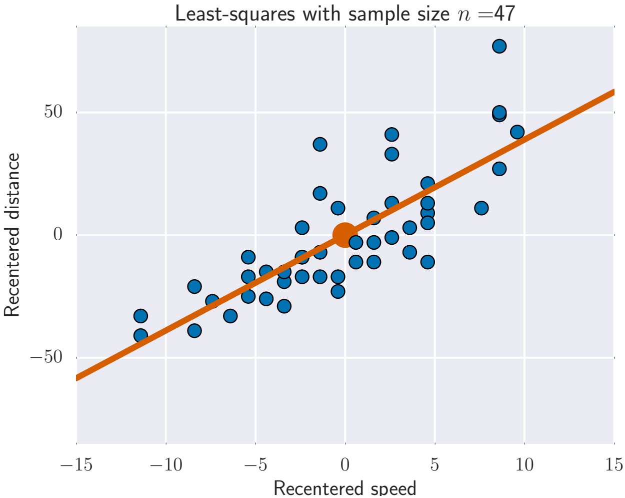
Extreme points – leverage effect (II)



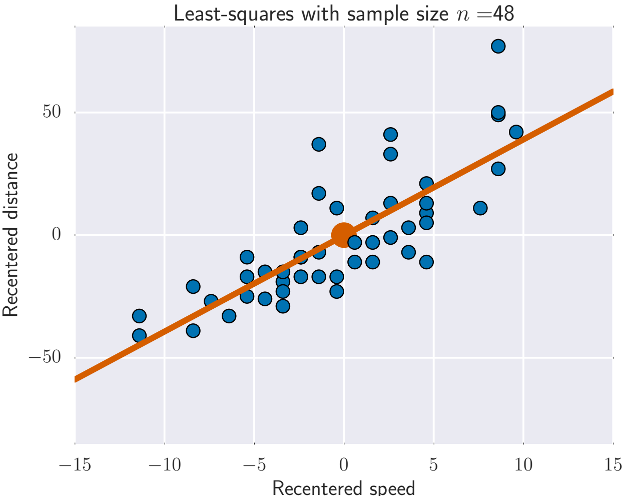
Extreme points – leverage effect (II)



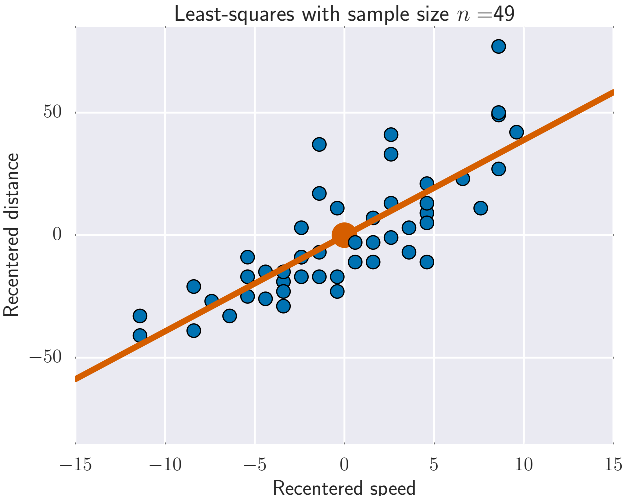
Extreme points – leverage effect (II)



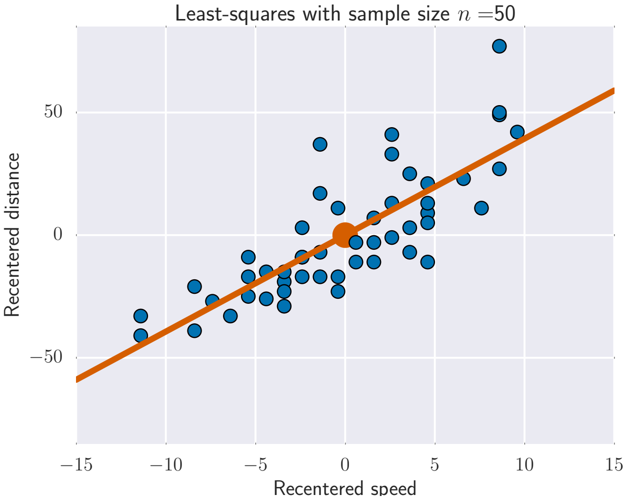
Extreme points – leverage effect (II)



Extreme points – leverage effect (II)



Extreme points – leverage effect (II)



Centering + scaling (standardization)

Centered-scaled model :

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

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

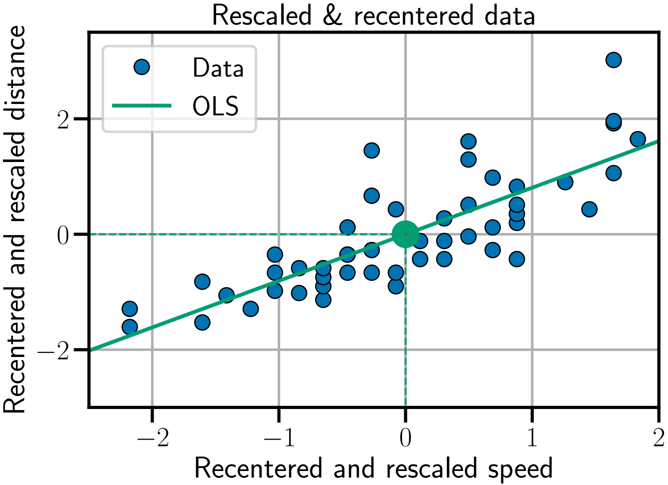
$$\begin{cases} \widehat{\theta}_0'' = 0 \\ \widehat{\theta}_1'' = \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} x_i'' y_i'' \end{cases}$$

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



When/why preprocessing ?

Centering $\mathbf{y}$ or using an intercept (or adding a constant feature) is equivalent

Rem: for sparse (: *creux*) cases centering $\mathbf{y}$ 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, etc.)

Rem: in practice centering/scaling is useful for **estimation** not so much for **prediction** (see next courses)
What happens with the logarithm scaling ?

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

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 :

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

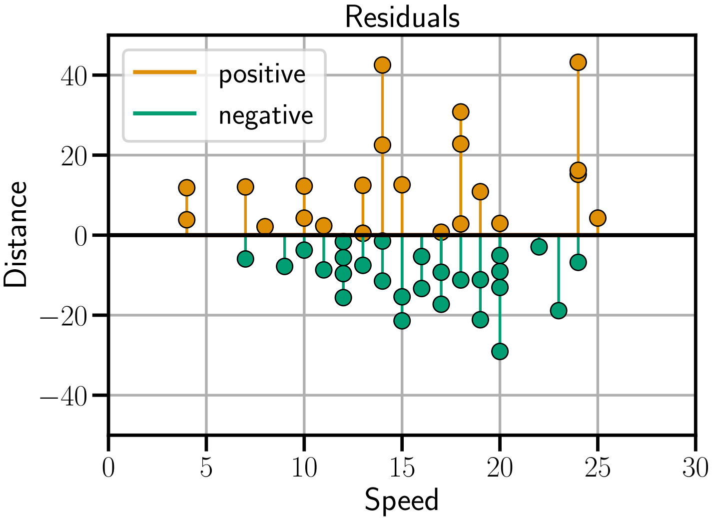
Rem: often written $\hat{y}_{n+1}$ (implicit dependence on x_{n+1}) The

residual : difference between observations and predicted values

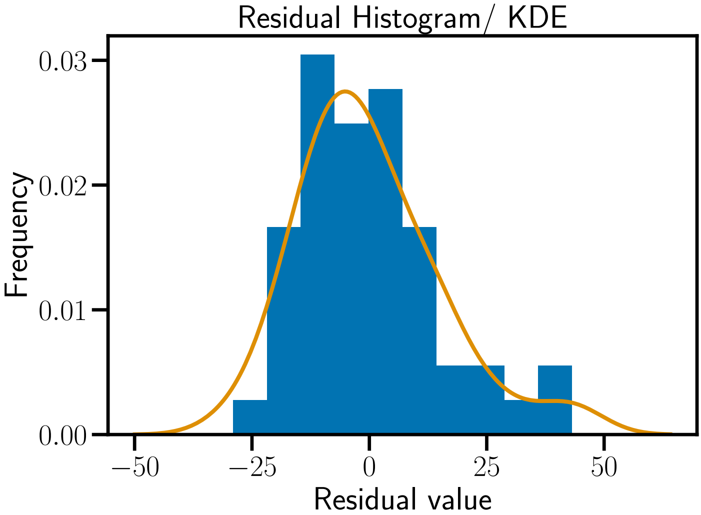
$$r_i = y_i - \mathrm{pred}(x_i) = y_i - \hat{y}_i = y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_i)$$

Rem:observable estimate of the unobservable statistical error

Residuals (on cars)



Residual histograms



Least squares motivation

- Computing advantage : computationally heavy methods avoided before computers (*e.g.*,iterative methods)
- Theoretical advantage : least square analysis easy under simple hypothesis
- Interpretability : how much does the regressor increase with the features

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 $(\theta_0^\star, \theta_1^\star)$

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

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

Gaussian likelihood

Rem: 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 : $y_i \sim \mathcal{N}(\theta_0^\star + \theta_1^\star x_i, \sigma^2)$, *i.e.*,$\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, then the most **likely** couple (θ_0, θ_1) based on the observations is maximizing the pdf of $(y_1, \ldots, y_n)$

Under an independence hypothesis, this is achieved by solving :

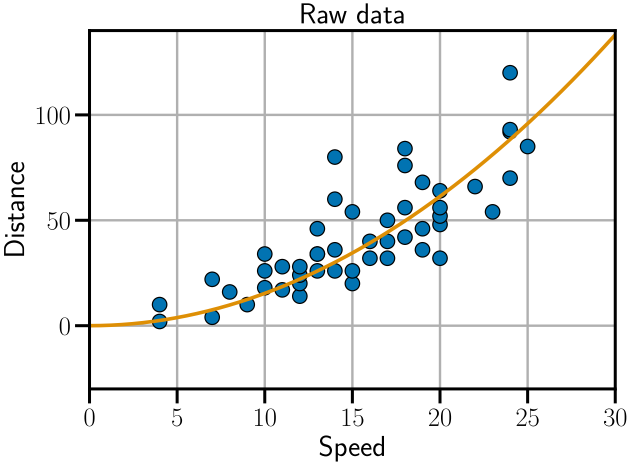
$$(\hat{\theta}_0, \hat{\theta}_1) \in \mathop{\arg\max}_{(\theta_0,\theta_1)\in\mathbb{R}^2} \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \theta_0 - \theta_1 x_i)^2}{2\sigma^2}\right)\right)$$

Exercise: 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.
- 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.