Evaluation de Performances

-

Stats II

Pascal Mérindol

merindol@unistra.fr
http://www-r2.u-strasbg.fr/~merindol

Remarques générales

- **Il ne s'agit pas d'un cours de statistique formel**
- **Il ne s'agit pas de statistiques avancées mais d'outils standards**
- **Aucune preuve analytique dans ce cours**
- **Un certain nombre de raccourcis sont empruntés pour «aider l'intuition»**
- **Le contenu de ce cours est une exploration partielle et subjective d'un certain nombre d'outil statistique utile pour un chercheur réseaux**
  - Illustrations à l'appui !
- **L'objectif est de vous inculquer des réflexes du type :**
  - faire attention aux estimations, approximations et biais !
  - quel outil pour quel problème ?
  - quelle représentation pour quelle distribution ?
  - etc

Plan

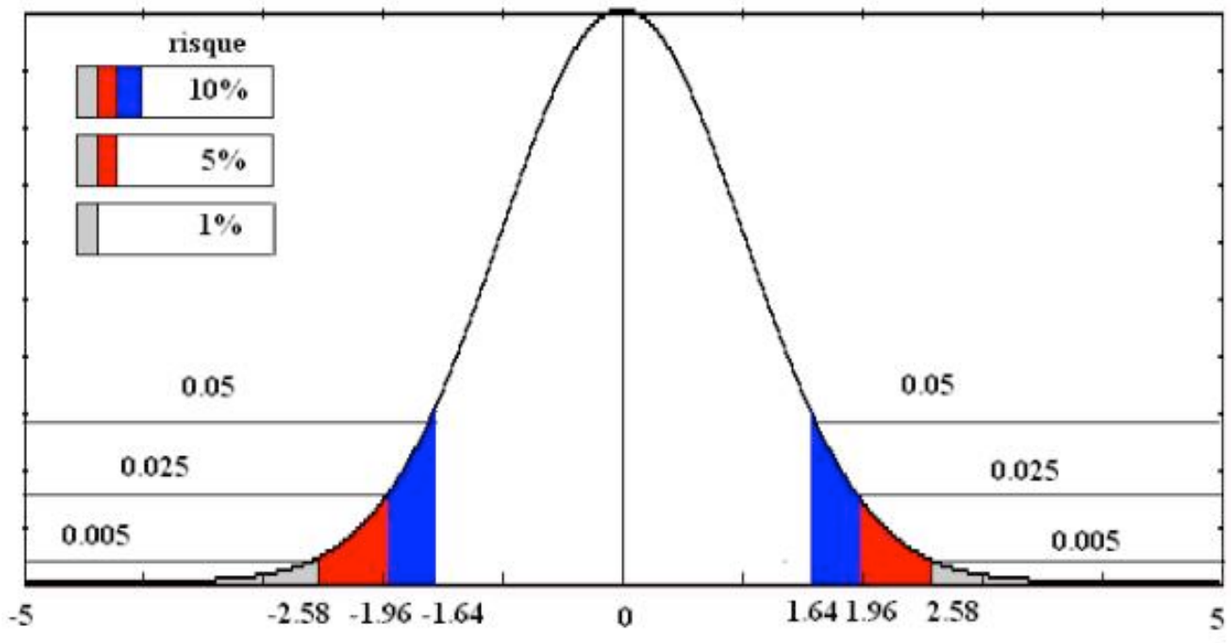
- **Théorème central limite & Moyenne des lots**
  - intervalles/niveaux de confiance en toutes circonstances !
- **Indépendance : corrélation et scatter plot**
  - a t-on le droit de considérer les variables comme indépendantes ?
- **Adéquation, homogénéité & indépendance : test du khi deux**
  - regrouper par classes pour mesurer une différence significative
- **Loi/Test de student**
  - petit échantillon, variance inconnue et test de moyennes (intervalles de confiance)
- **De la régression linéaire (LSF) au maximum de vraisemblance (MLE)**
  - ajustement par inférence des paramètres d'une fonction ou d'une distribution
- **Interpolation Lagrangienne**
  - polynôme passant exactement par une série de points (≠ ajustement)

Plan (suite)

- **Inférence Bayésiennes pour modélisation probabiliste**
  - Des effets (mesures observées) aux causes (avec a priori)
- **Vers du trafic réaliste : le modèle ON/OFF**
  - auto-similarité et long range dependance
  - distribution de Pareto (loi de puissance)
- **Distributions continues vs. discrètes (**Zipf**)**
  - Génération de lois de puissances avec un PRG uniforme
- **Représentations graphiques**
  - error bars, pdf, cdf, ccdf, icdf, boxplots, logscale, loglogscale, roc curve, **...**
- **Outils : quelques exemples de calculs et visualisations**
  - Matplotlib
  - Rstat

Théorème central limite & Moyenne des lots

- **<u>Rappel</u> : la somme d'une suite de variables aléatoires «tend» vers la loi normale**
  - valable pour les moyennes ! => méthode de moyenne des lots (batch mean method)
    - mais n'oublions pas la loi des grands nombres...
- **Réplications (avec/sans warm up ?) ou moyenne des lots**
  - moyenne des lots = découpe de la simulation en plusieurs lots (taille et nombre ?)
  - la distribution d'origine est quelconque (inconnue a priori)
  - la moyenne reste identique mais pas la variance (on se ramène à une loi normale)
  - on peut donc utiliser les intervalles et niveaux de confiance !



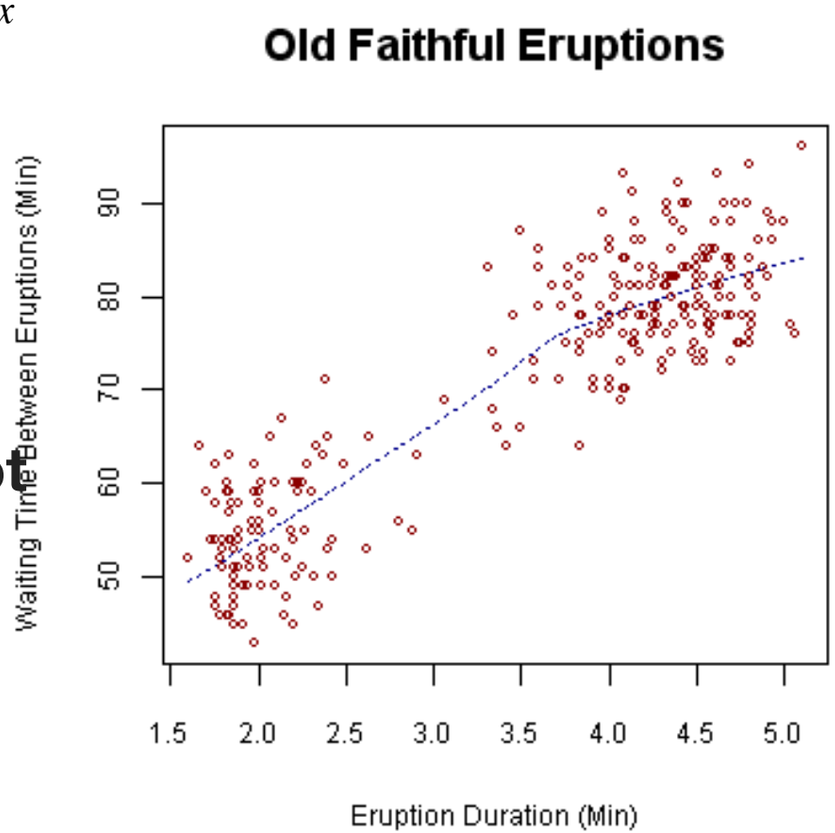
Indépendance (IID) : auto corrélation linéaire et scatter plot

- **Est-ce que les données (lots) sont indépendants (au moins linéairement) ?**
  - permet de vérifier la taille k des lots
    - aspect temporel sur k (~ période ?)
  - coefficient de corrélation linéaire (x,y) => coefficient d'auto-corrélation (x,k) :

$$r_{xy} = \frac{\sum (x_i - \bar{x}).(y_i - \bar{y})}{\sigma_x \sigma_y} \qquad r_x^k = \frac{\sum (x_i - \bar{x}).(x_{i+k} - \bar{x})}{\sigma_x^2}$$

  - toute autre forme de dépendance n'est pas détectée :(
    - pas suffisant dans tous les cas...(autre forme)

- **Corrélation par voie graphique en général : le scatter plot**
  - visualiser toute forme de corrélation
  - pas forcément dépendance (cause commune ?)



Test du khi deux : adéquation, homogénéité & indépendance

- **Soient { x_i } k variables indépendantes suivants des lois normales alors**

$$X = \sum_{i=1}^{k} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2$$ **suit une loi du khi-deux à k degrés de liberté.**

- **Calculer la distance T (pour test) du khi-deux**
  - soit une variable $\mathbf{X_i}$ ayant **I** valeurs possibles
  - soit une variable $\mathbf{Y_j}$ ayant **J** valeurs possibles
  - O_{ij} := valeur observée pour (i,j)
  - E_{ij} la valeur «espérée» := $\dfrac{\sum_{j=1}^{J} O_{ij} \times \sum_{i=1}^{I} O_{ij}}{N}$

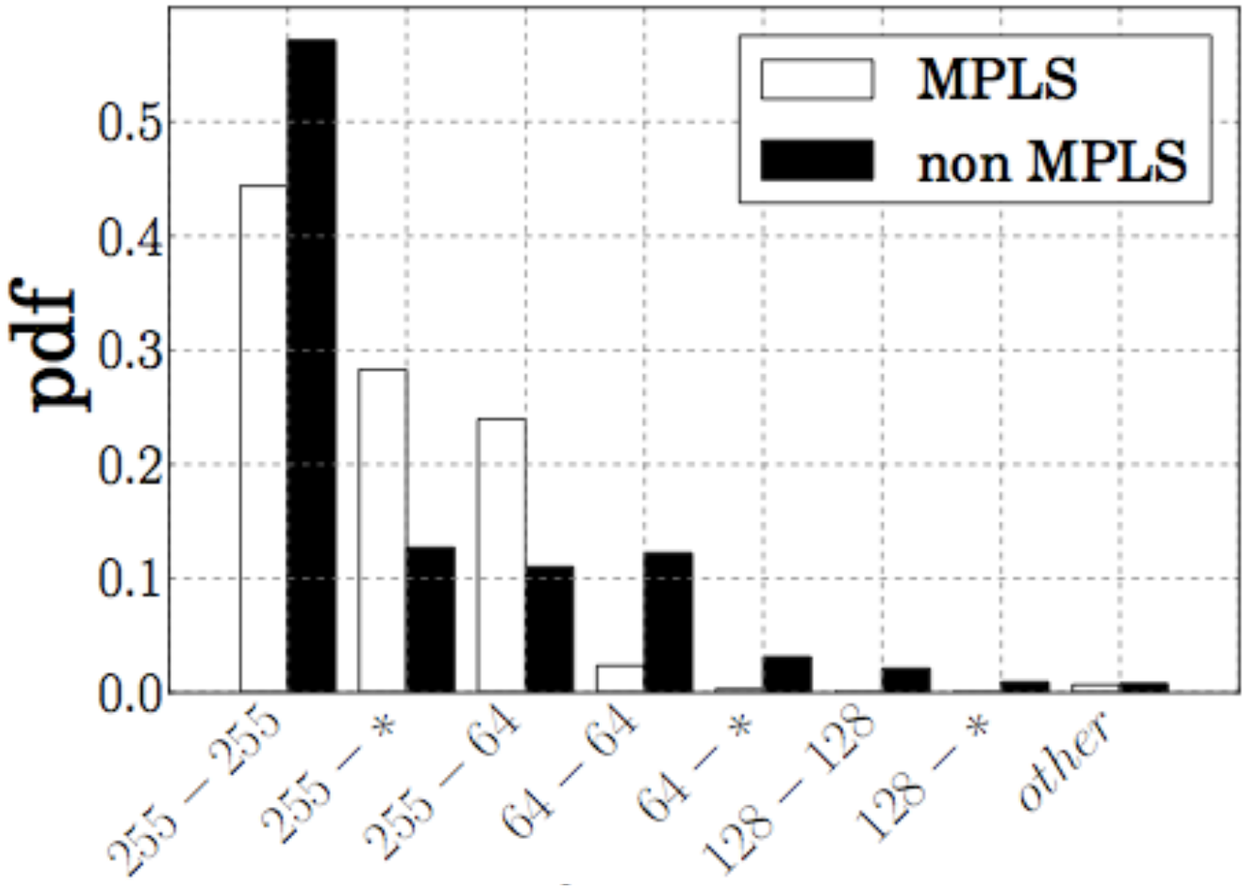
$$T = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

  - degré de liberté k = ***ddl := (I-1)*(J-1)***
  - on se fixe un risque d'erreur **α** et on vérifie que ***T < Khideux[ddl][α]*** sinon rejet !

ddl \ α	0,90	0,50	0,30	0,20	0,10	0,05	0,02	0,01	0,001
1	0,0158	0,4549	1,0742	1,6424	2,7055	3,8415	5,4119	6,6349	10,8274
2	0,2107	1,3863	2,4079	3,2189	4,6052	5,9915	7,8241	9,2104	13,8150
3	0,5844	2,3660	3,6649	4,6416	6,2514	7,8147	9,8374	11,3449	16,2660
4	1,0636	3,3567	4,8784	5,9886	7,7794	9,4877	11,6678	13,2767	18,4662
5	1,6103	4,3515	6,0644	7,2893	9,2363	11,0705	13,3882	15,0863	20,5147
6	2,2041	5,3481	7,2311	8,5581	10,6446	12,5916	15,0332	16,8119	22,4575
7	2,8331	6,3458	8,3834	9,8032	12,0170	14,0671	16,6224	18,4753	24,3213
8	3,4895	7,3441	9,5245	11,0301	13,3616	15,5073	18,1682	20,0902	26,1239
9	4,1682	8,3428	10,6564	12,2421	14,6837	16,9190	19,6790	21,6660	27,8767
10	4,8652	9,3418	11,7807	13,4420	15,9872	18,3070	21,1608	23,2093	29,5879
11	5,5778	10,3410	12,8987	14,6314	17,2750	19,6752	22,6179	24,7250	31,2635
12	6,3038	11,3403	14,0111	15,8120	18,5493	21,0261	24,0539	26,2170	32,9092
13	7,0415	12,3398	15,1187	16,9848	19,8119	22,3620	25,4715	27,6882	34,5274
14	7,7895	13,3393	16,2221	18,1508	21,0641	23,6848	26,8727	29,1412	36,1239
15	8,5468	14,3389	17,3217	19,3107	22,3071	24,9958	28,2595	30,5780	37,6978
16	9,3122	15,3385	18,4179	20,4651	23,5418	26,2962	29,6332	31,9999	39,2518
17	10,0852	16,3382	19,5110	21,6146	24,7690	27,5871	30,9950	33,4087	40,7911
18	10,8649	17,3379	20,6014	22,7595	25,9894	28,8693	32,3462	34,8052	42,3119
19	11,6509	18,3376	21,6891	23,9004	27,2036	30,1435	33,6874	36,1908	43,8194
20	12,4426	19,3374	22,7745	25,0375	28,4120	31,4104	35,0196	37,5663	45,3142
21	13,2396	20,3372	23,8578	26,1711	29,6151	32,6706	36,3434	38,9322	46,7963
22	14,0415	21,3370	24,9390	27,3015	30,8133	33,9245	37,6595	40,2894	48,2676
23	14,8480	22,3369	26,0184	28,4288	32,0069	35,1725	38,9683	41,6383	49,7276
24	15,6587	23,3367	27,0960	29,5533	33,1962	36,4150	40,2703	42,9798	51,1790
25	16,4734	24,3366	28,1719	30,6752	34,3816	37,6525	41,5660	44,3140	52,6187
26	17,2919	25,3365	29,2463	31,7946	35,5632	38,8851	42,8558	45,6416	54,0511
27	18,1139	26,3363	30,3193	32,9117	36,7412	40,1133	44,1399	46,9628	55,4751
28	18,9392	27,3362	31,3909	34,0266	37,9159	41,3372	45,4188	48,2782	56,8918
29	19,7677	28,3361	32,4612	35,1394	39,0875	42,5569	46,6926	49,5878	58,3006
30	20,5992	29,3360	33,5302	36,2502	40,2560	43,7730	47,9618	50,8922	59,7022

Indépendance avec khi deux : illustration

Est-ce que les routeurs MPLS sont «quelconques» ?



Distribution selon une signature fingerprint

$$E_{ij} = \frac{\sum_{j=1}^{J} O_{ij} \times \sum_{i=1}^{I} O_{ij}}{N}$$

O_{ij}									$\sum$
$\sum$									N

Données brutes et (sous-)totaux

=> Deux variables, huit classes : degré de liberté 7

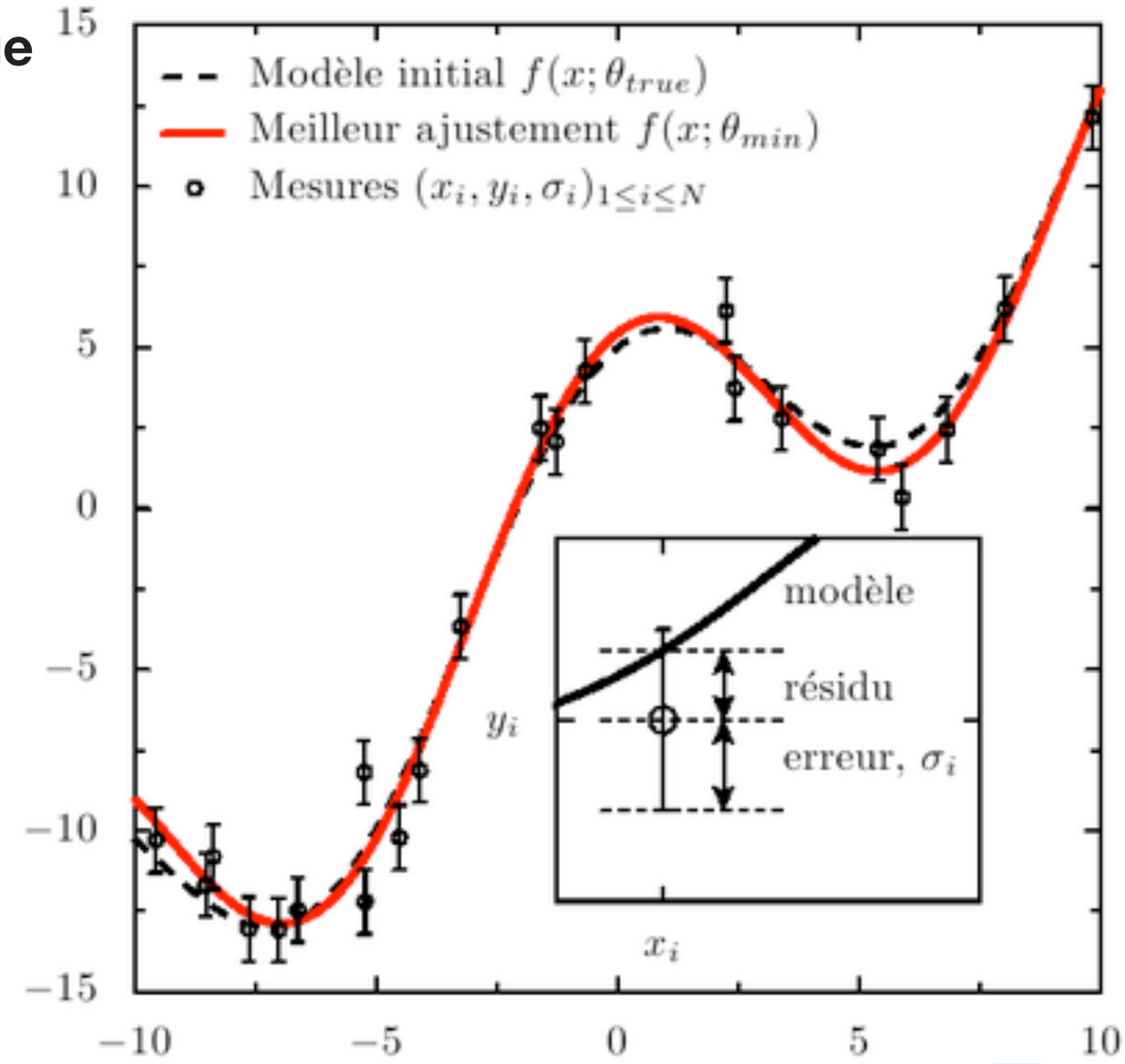
Test de student

- **La variable $T = \frac{Z}{\sqrt{U / k}}$ suit une loi de Student à k degrés de liberté (avec Z loi normale centrée réduite et U loi du khi-deux à k degré de liberté)**

- **Pour k élevé, loi de Student ~ loi normale centrée réduite**
- **Applications possibles :**
  - intervalle de confiance avec k relativement faible (petit échantillon)
  - intervalle de confiance avec variance inconnue
  - comparaison d'une moyenne observée à une moyenne théorique
    - test de student $$t = \frac{\bar{x} - \mu}{s / \sqrt{k}}$$
    - si ***t > Student[ddl][α]*** alors on rejette l'hypothèse

Techniques d'ajustement (fitting methods)

- **Déterminer la classe de fonctions (affines, polynomiales, lois de probabilité)**
- **via une première approximation graphique**
- **... puis techniques statistiques**
  - Technique des moindres carrés
  - Maximum de vraisemblance
  - Estimateurs robustes en général...



Méthode des moindres carrés

- **Des données expérimentales vers un modèle théorique**
- **Soit $f(x;\theta)$ une famille de fonctions indéxés par un (ou plusieurs) paramètre(s) θ**
- **Objectif : estimation de ce(s) paramètre(s) inconnu(s)**
- **Méthode : minimiser la somme des résidus** $\sum_{i=1}^{N}(y_i - f(x_i;\theta))^2 = \sum_{i=1}^{N} r_i^2(\theta)$

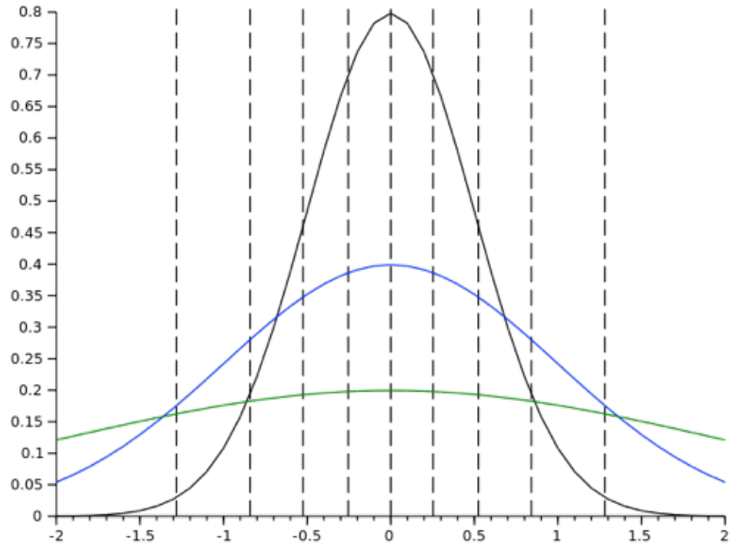
- **Avec estimation de l'écart type du bruit** $\sum_{i=1}^{N}\left(\frac{(y_i - f(x_i;\theta)}{\sigma_i}\right)^2$
  - on reconnaitra la définition de la loi du khi-deux si le bruit est gaussien

- **Applications : régression linéaire ou modèle linéaire quelconque (polynômes)**
  - ex : trouver les paramètres (a,b) tel que les résidus avec ***f(x) = ax + b*** soit minimums
  - optimisation simple et efficace par inversion matricielle

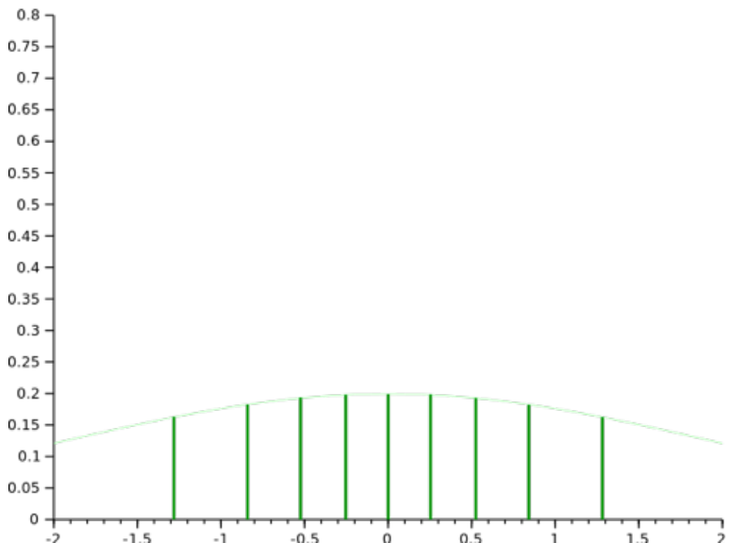
Maximum de vraisemblance

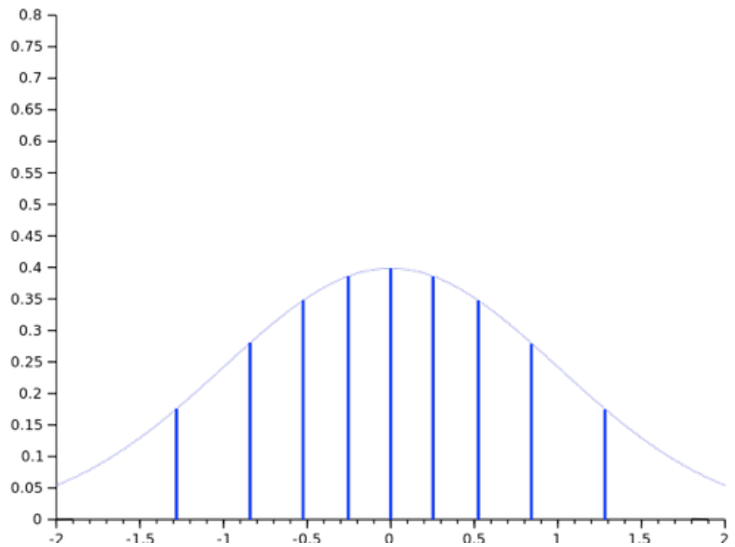
- **Ajustement pour lois lois de probabilités = loi normale, loi de poisson, loi exponentielle, lois de puissance, etc**

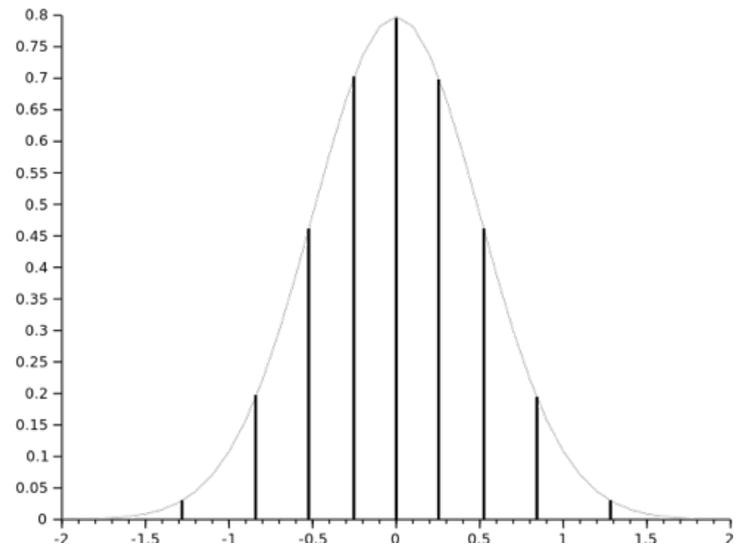
$$\prod_{i=1}^{N} f(x_i;\theta)$$



MAX







Estimateurs

- **Maximum Likelihood Estimator** $\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} f(x_i;\theta)$

- **qui se ré-écrit (pour simplifier l'optimisation par dérivation => formes closes)**

$$\hat{\theta} = \arg\min_{\theta} \left[-\sum_{i=1}^{N} \log f(x_i;\theta) \right]$$

- **En généralisant aux M-estimateurs :**

$$\hat{\theta} = \arg\min_{\theta} \left[\sum_{i=1}^{N} p(x_i;\theta) \right]$$

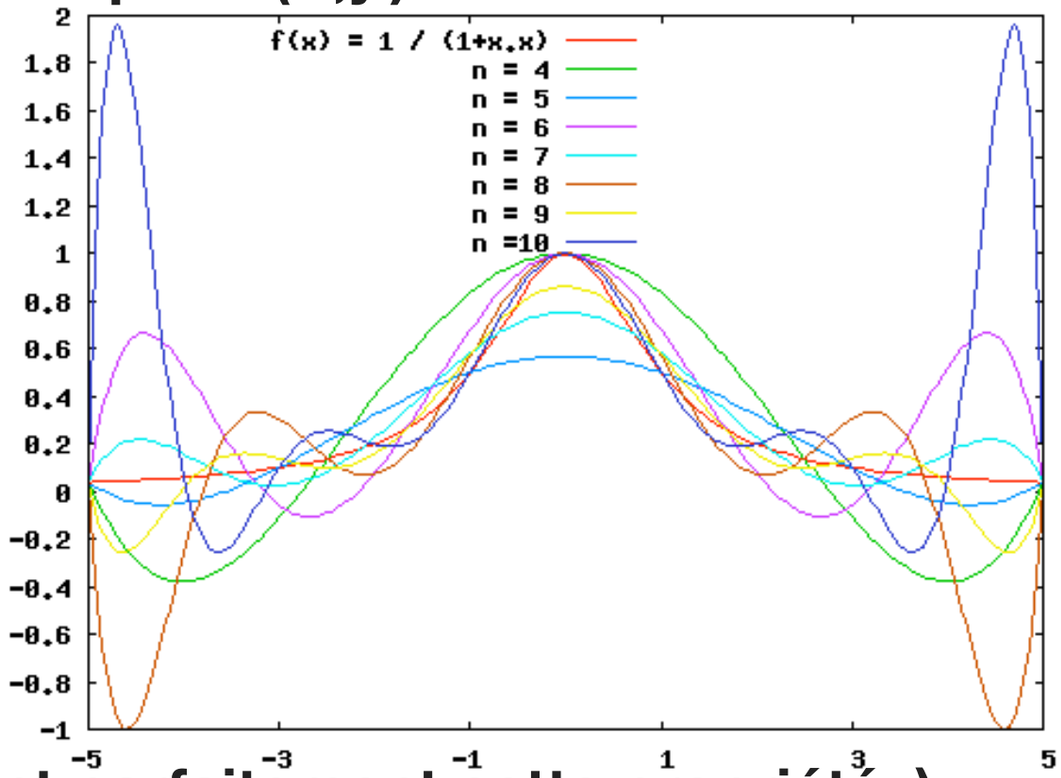
  - avec ***p*** une fonction ayant certaines propriétés de robustesse et de dérivation :
    - moyenne, déviation absolue / médiane, résidus au carrés, etc

- **Pas toujours de formes closes pour l'optimisation => méthodes plus lourdes...**

Interpolation de Lagrange

- **Attention : pas équivalent à une technique d'approximation !**
- **Trouver un polynôme passant <u>exactement</u> par n+1 point (x_i,y_i)**

$$L(X) = \sum_{i=0}^{n} y_i \left(\prod_{j=0, j\neq i}^{n} \frac{X - x_j}{x_i - x_j} \right)$$



- **Le seul polynôme de degré au plus n satisfaisant parfaitement cette propriété :)**
- **<u>mais</u> : contre exemple de Runge => divergence quand n tend vers l'infini**
  - Alternative : Lagrange par morceaux...
  - ou moindres carrés !

Inférence Bayesienne

- **Théorème de Bayes** $$P(A \mid B) = \frac{P(B \mid A).P(A)}{P(B)}$$
- ***P(X/Y)* := probabilité de A sachant Y (Y est un maintenant un fait)**
- **Exemple : T un test quelconque et M la propriété «le sujet est positif»**
  - faux négatif = 1/10 et faux positif = 1/100 et
    - P(T|M)=0.9 et P(T|¬M)=0.01
- On en déduit (plus de P(T)) : $$P(M \mid T) = \frac{0.9.P(M)}{0.89.P(M) + 0.01}$$
- Si P(M) est peu probable alors P(M|T) aussi ! (le test n'est pas si robuste que ça)
  - ex : avant le test si P(M) = 1/1000 alors moins de 9% de chances ...
- En pratique, notons que ce test se révèle aussi utile pour de petits échantillons si estimation «subjective a priori» pour guider (une forme de confiance sur P(M))

Méthode bayesienne : illustration réseau

- **Soit un ensemble de mesures $\mathscr{M}_i$ et une propriété : ici une interface IP i «est MPLS»**
  - Soit ***MPLS$_i$*** une interface IP i appartenant à un routeur MPLS
- **Si on souhaite estimer la probabilité P, la règle de Bayes nous donne :**

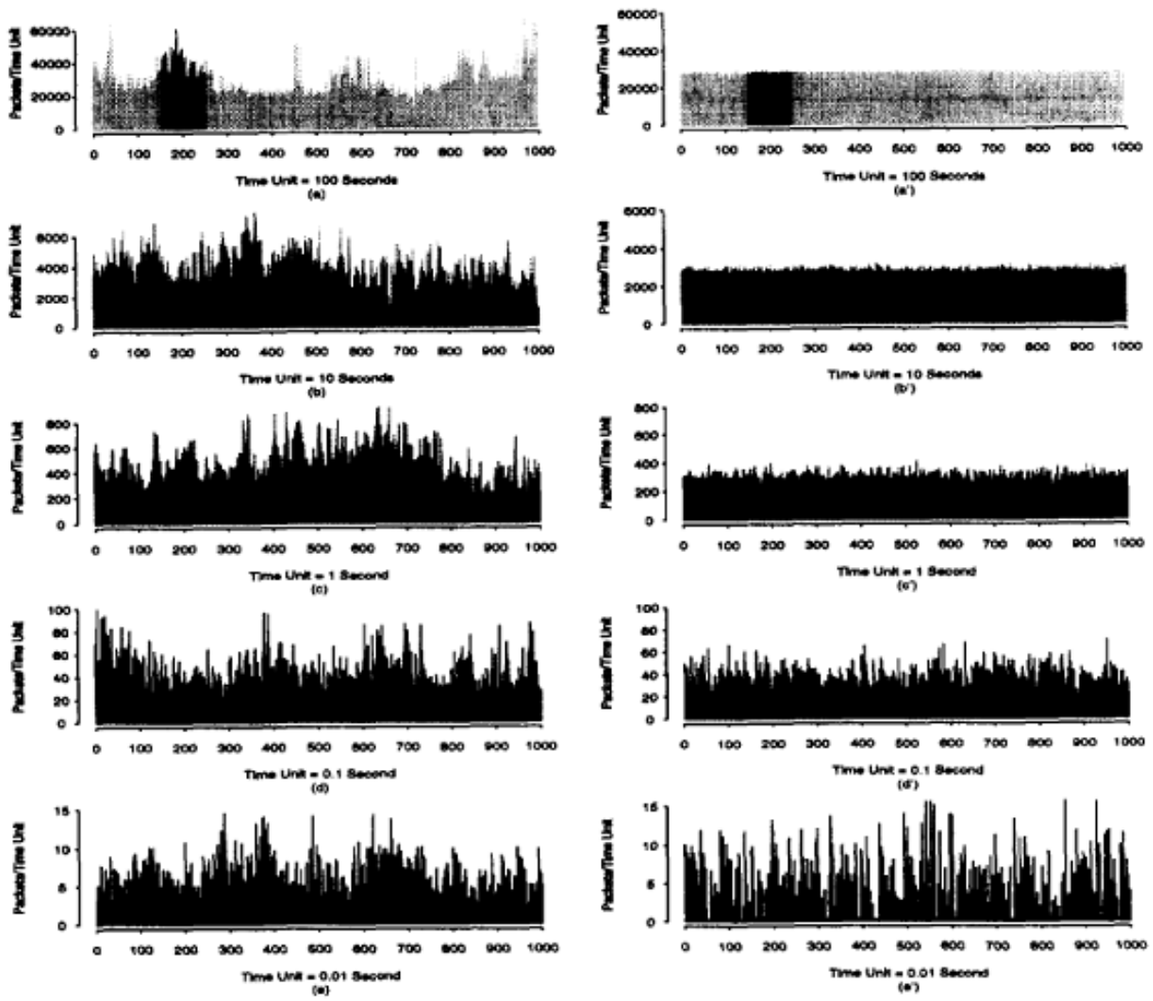
$$P(\mathrm{MPLS}_i) = P(\mathrm{MPLS} \mid \mathscr{M}_i)$$
$$P(\mathrm{MPLS}_i) \propto P(\mathscr{M}_i \mid \mathrm{MPLS})\, P(\mathrm{MPLS})$$

- **Il faut maintenant estimer ces termes pour déterminer si une IP est MPLS ou non**
  - Méthodes d'entrainement
    - Validation croisée initiale
  - Test : P(MPLS$_i$) > P(¬MPLS$_i$) avec une certaine sensibilité
- **Utile dès lors qu'on a une indication a priori**
  - Ici un RFC ICMP MPLS permettant de calibrer ses tests...

Trafic réaliste

- **Le trafic des réseaux IP n'est jamais stable...**
- **...il est auto-similaire (propriété «bursty» fractale à multiple échelles de temps)**
  - pas de distribution de poisson !
  - + dépendant sur de longues périodes...
- **Le modèle source ON/OFF**
  - ON : burst de trafic à une fréquence f
  - OFF : pas de trafic
  - ex : Loi de Pareto selon deux modes
    - taille des paquets min x_{min} et courbure k

$$P(X > x) = \left(\frac{x}{x_{\min}} \right)^{-k}$$



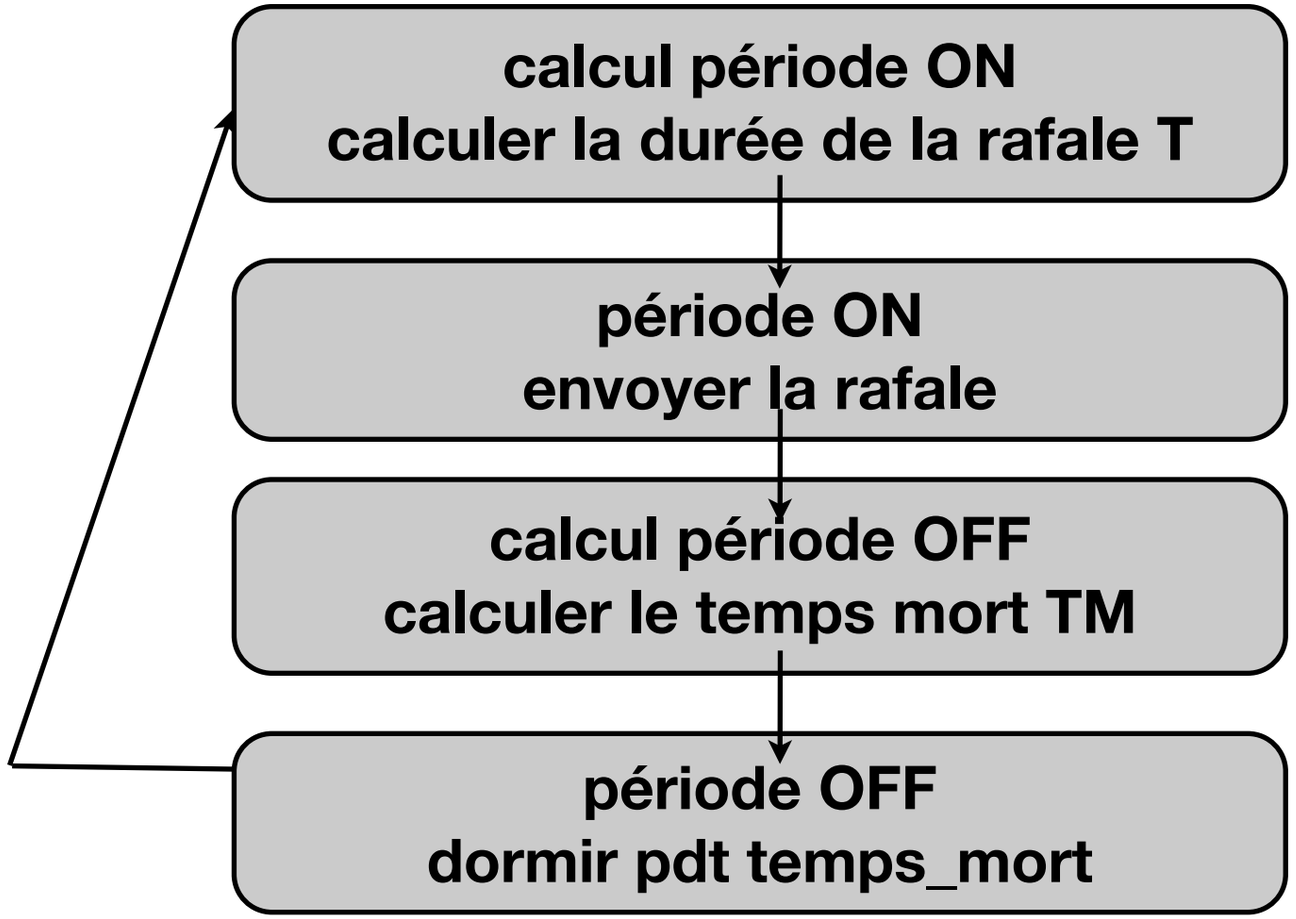
Self Similar **Poisson**

Générateur ON/OFF Pareto dans NS2

- **Utilisation**

```
set p [new Application/Traffic/Pareto]
$p set burst_time_ BT (espérance période ON)
$p set idle_time_ IT (espérance période OFF)
$p set rate_ R (débit dans le burst)
$p set packetSize_ PS (taille fixe en octet pour une appli)
$p set shape_ k (courbure de Pareto)
```

- **Fonctionnement**



T selon une loi de Pareto (BT*(k-1)/k,k)

au moins un paquet
-> ⌈T*R/8PS ⌉

TM selon une loi de Pareto (IT*(k-1)/k,k)

et c'est reparti...

Distributions continues ou discrètes (Pareto ou Zipf)

- **Génération Pareto à partir d'une loi uniforme U dans [0,1]**

$$T = \frac{x_{\min}}{U^{1/k}}$$

- **Pareto borné avec L=x_{min} ≤ T ≤ H**

$$\left(-\frac{UH^k - UL^k - H^k}{H^k L^k} \right)^{-\frac{1}{k}}$$

```
pow((pow(xmin, k) / (U*pow((xmin/H), k) - U + 1)), (1.0/k));
```

- **Zipf (Zeta si H-L = inf) ~ version discrète de Pareto, en mode borné :**

$$P(x) = \frac{x^{-k}}{\sum_{i=L}^{H} i^{-k}}$$

Constante c pour la normalisation →

- **Génération depuis une loi uniforme (ou ZipfR) :**

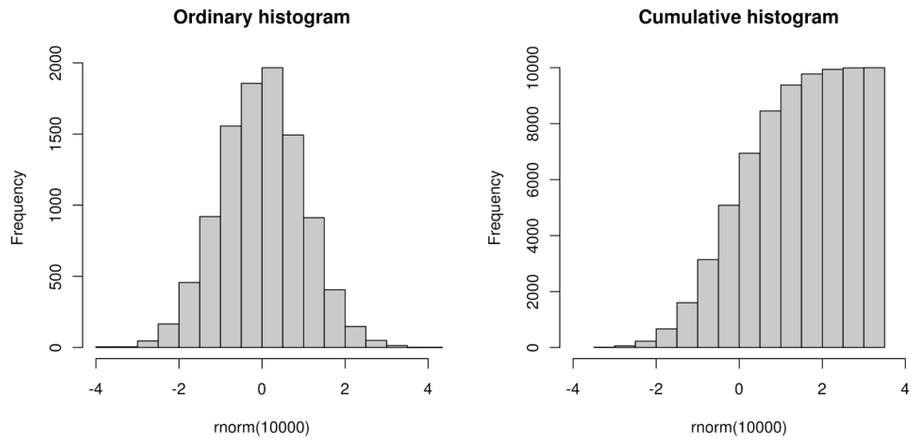
```
// Compute normalization constant on first call only
  for (i=L; i<=H; i++)
    c = c + (1.0 / pow((double) i, k));
  c = 1.0 / c;
// Pull a uniform random number (0 < U < 1)
  U = rand_val(0);
// Map z to the value
sum_prob = 0;
for (i=L; i<=H; i++)
{
  sum_prob = sum_prob + c / pow((double) i, k);
  if (sum_prob >= U)
  { U = i;  break;}
}
```

→ **Le rang !**

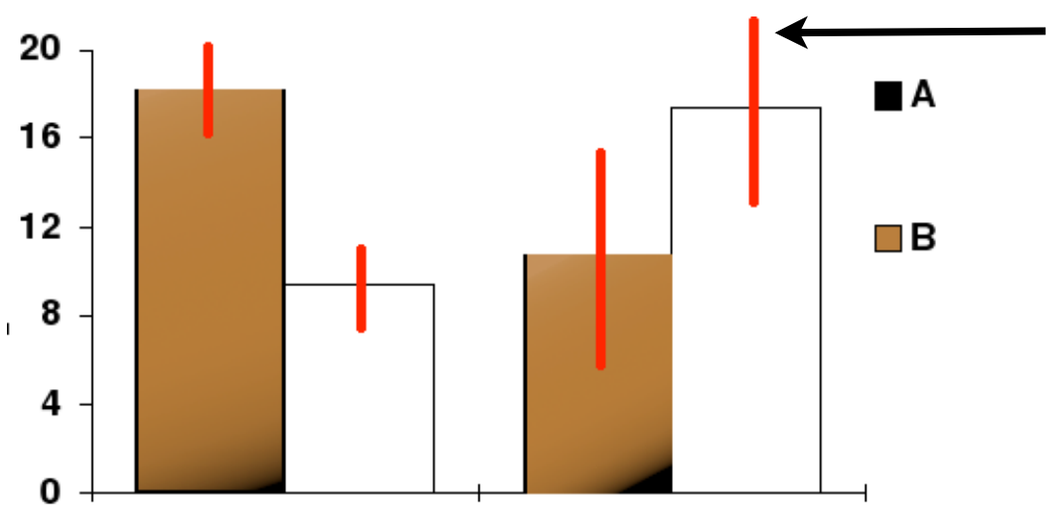
> 0.969579
# 0.611627 0.969579 **1**
# 0.764534 0.969579 **2**
# 0.832492 0.969579 **...**
# 0.870719 0.969579
# 0.895184 0.969579
# 0.912173 0.969579
# 0.924656 0.969579
# 0.934212 0.969579
# 0.941763 0.969579
# 0.947880 0.969579
# 0.952934 0.969579
# 0.957182 0.969579
# 0.960801 0.969579
# 0.963921 0.969579
# 0.966640 0.969579
# 0.969029 0.969579
>>> 0.971145 **16 !**

Représentations graphiques

- **Comment obtenir / représenter une distribution ?**
  - sur un fichier brut (avec valeurs entières en première colonne)
    - `sort -k1n | uniq -c`
    - normaliser en divisant par la somme
  - avec des valeurs flottantes ?
    - regrouper par paquets/bins => histogrammes

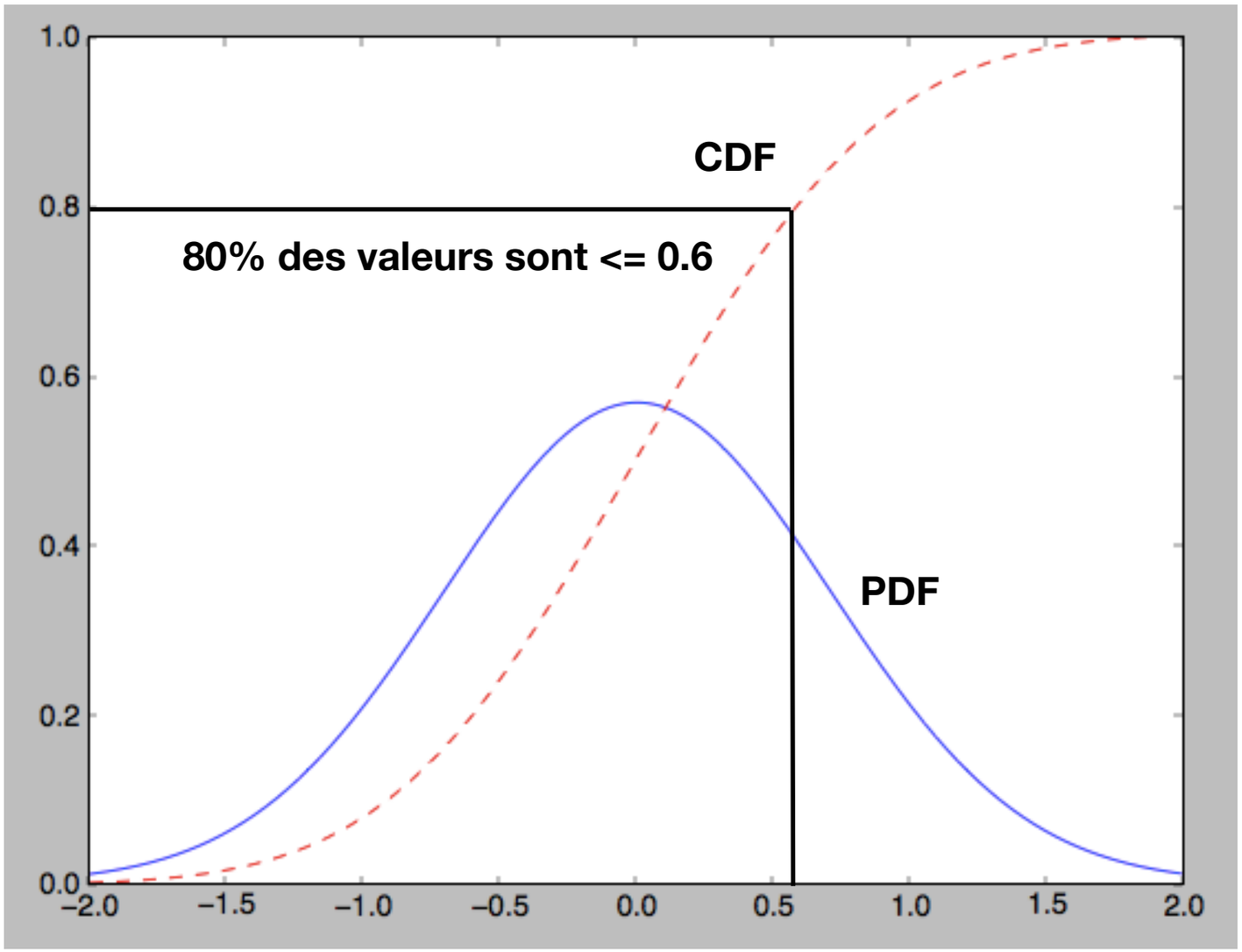


- **Comparaison de deux quantités avec intervalle de confiance**



Notion de barre d'erreur : intervalle de confiance, écart type, erreur type ...

Représentations graphiques




```
import numpy as np
from pylab import *

# Create some test data
dx = .01
X  = np.arange(-2,2,dx)
Y  = exp(-X**2)

# Normalize the data to a proper PDF
Y /= (dx*Y).sum()

# Compute the CDF
CY = np.cumsum(Y*dx)

# Plot both
plot(X,Y)
plot(X,CY,'r--')

show()
```

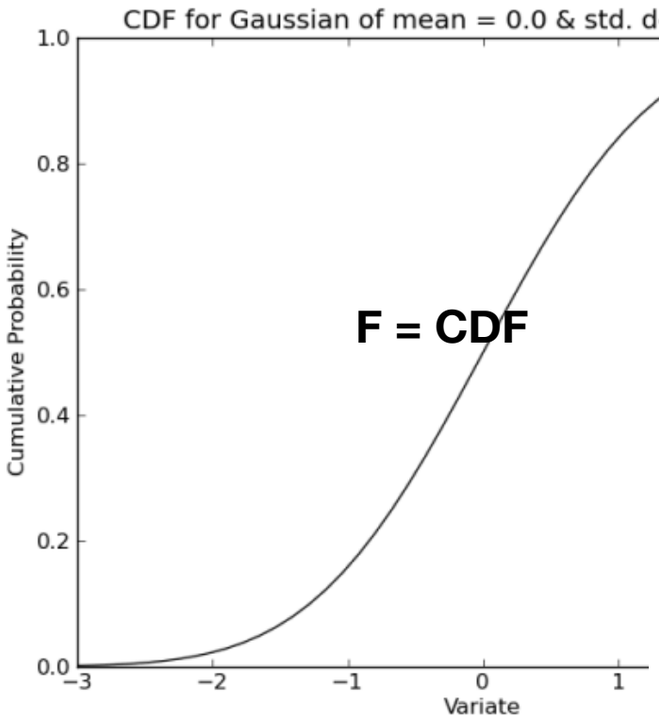
CDF = CUMULATIVE DISTRIBUTION FUNCTION

PDF = PROBABILITY DENSITY FUNCTION

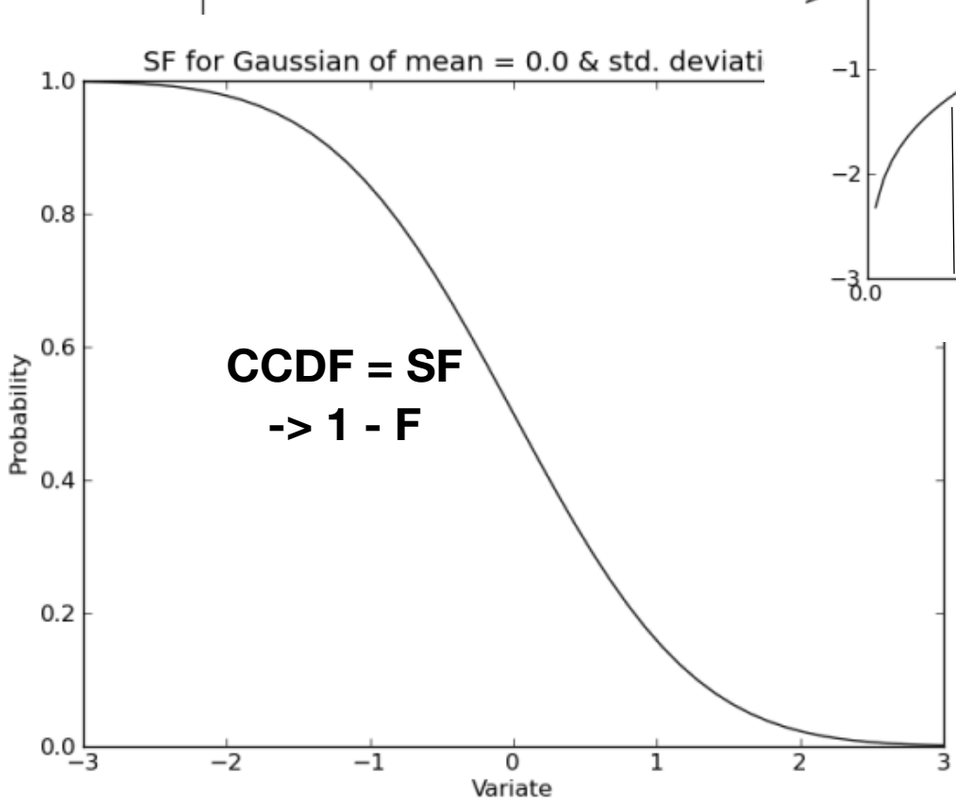
PMF = PROBABILITY MASS FUNCTION (pour variables discretes)

Représentations graphiques

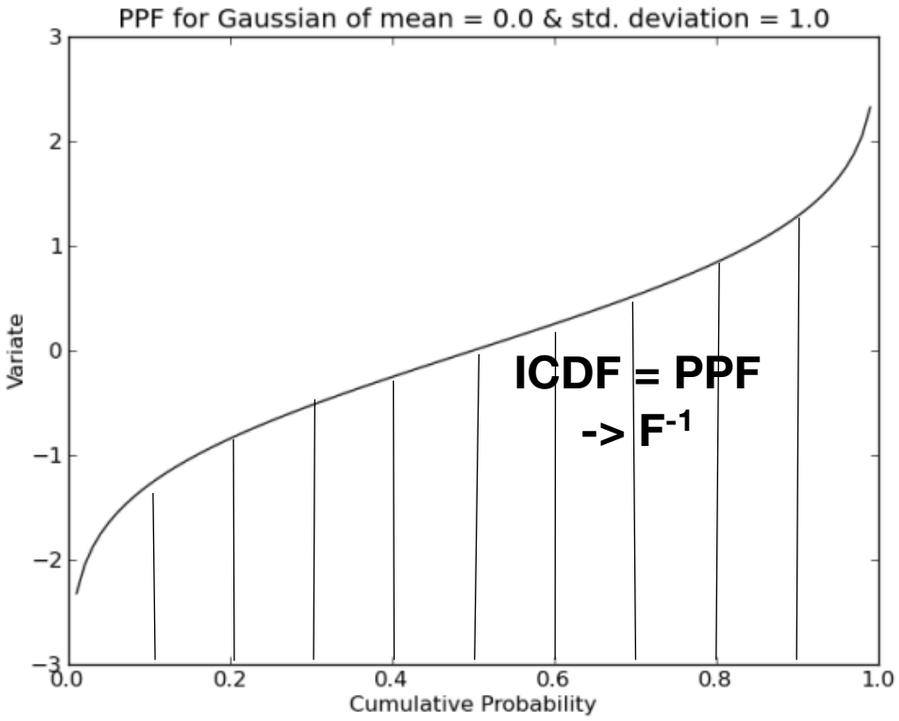
- **Inverse CDF (=PPF percent point F) et Complementary CDF (=Survivor F)**



F = CDF



CCDF = SF
-> 1 - F

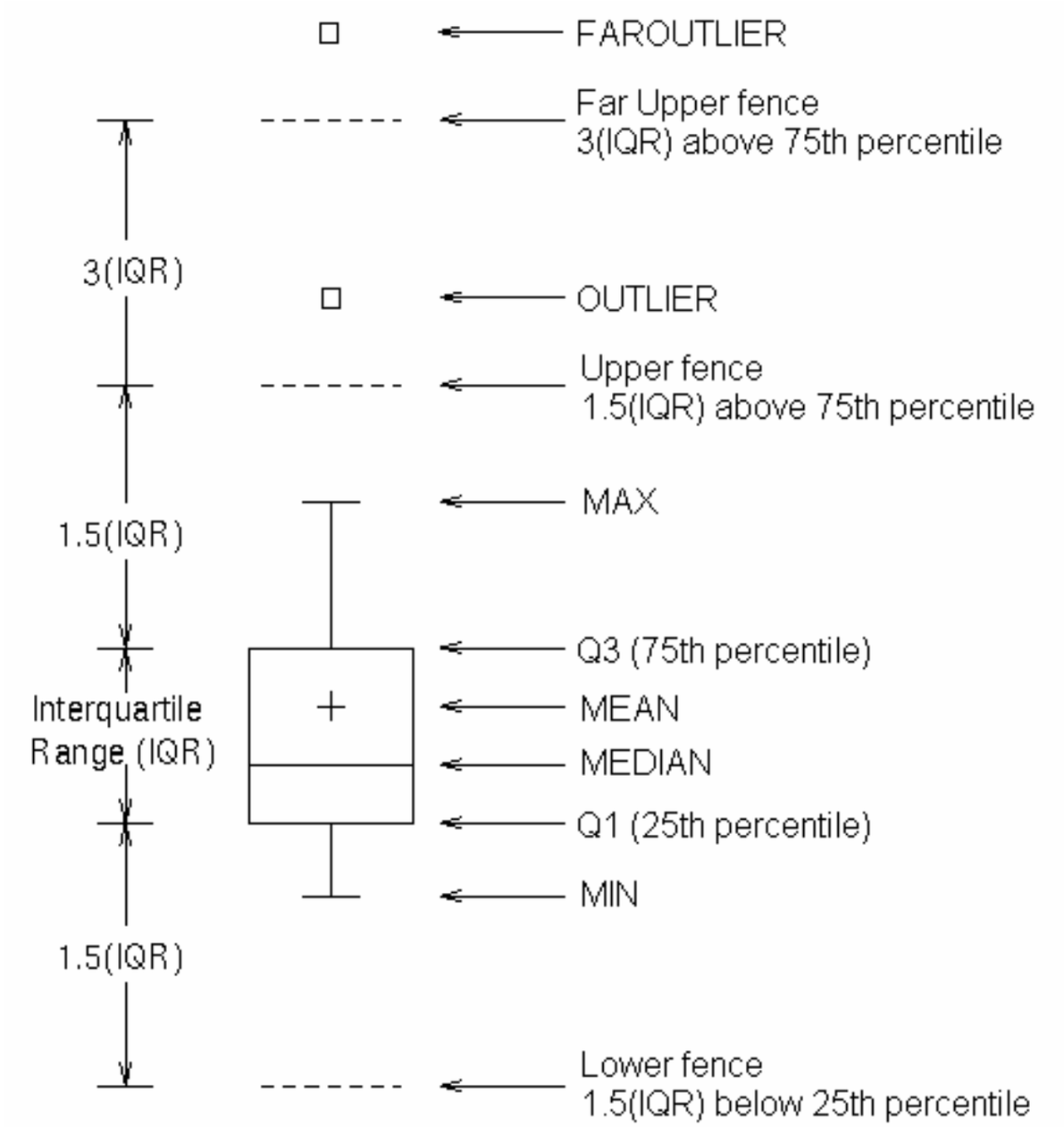


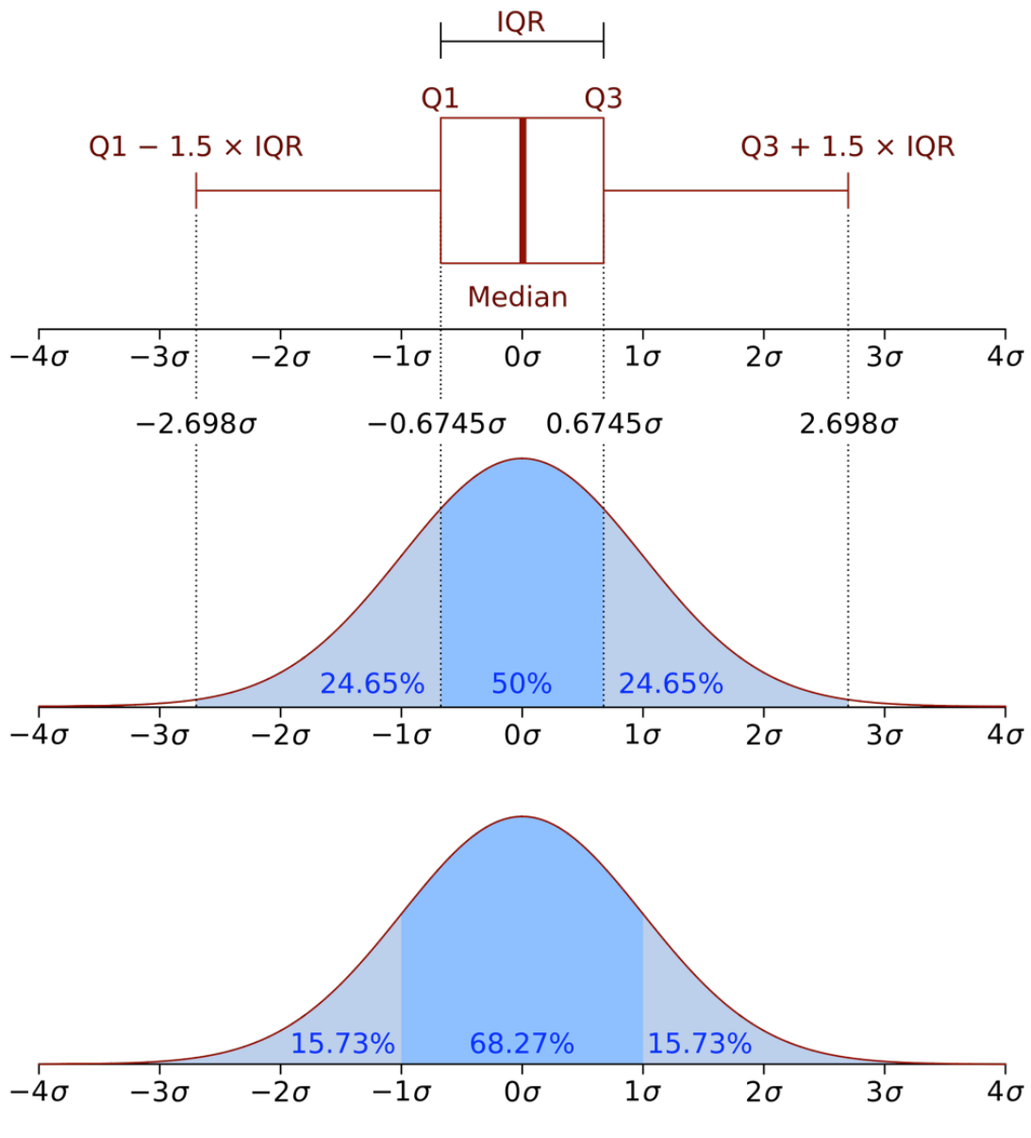
ICDF = PPF
-> F^{-1}

Générer une distribution depuis une loi uniforme

Représentations graphiques

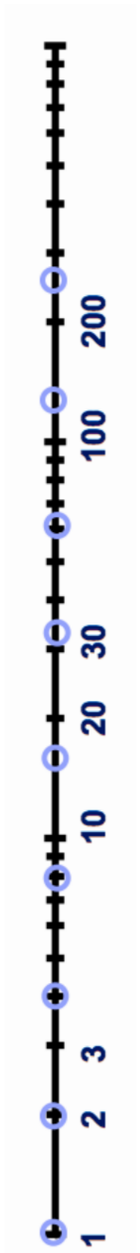
- **Box plots : toutes les stats de base en un coup d'oeil !**





Représentations graphiques

- **Echelles logarithmiques**

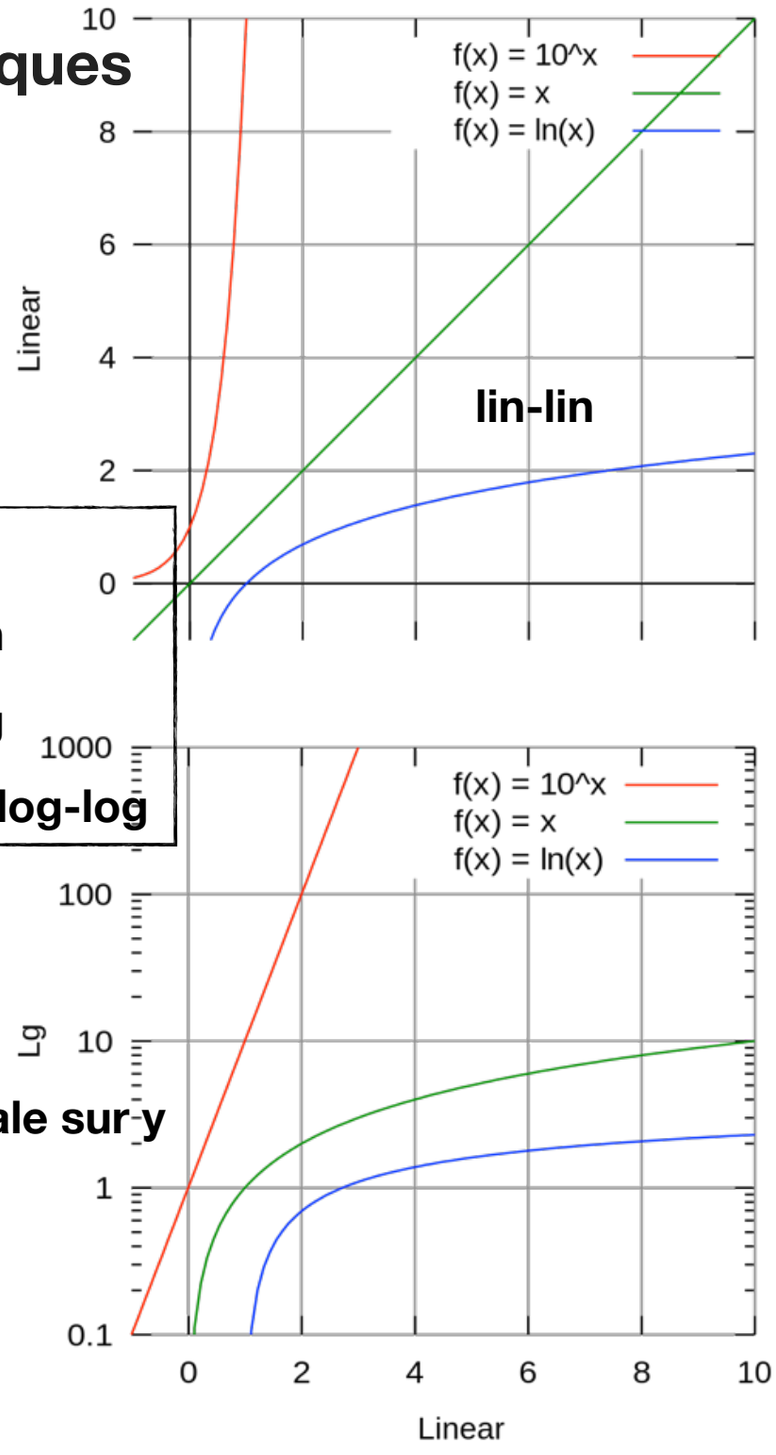


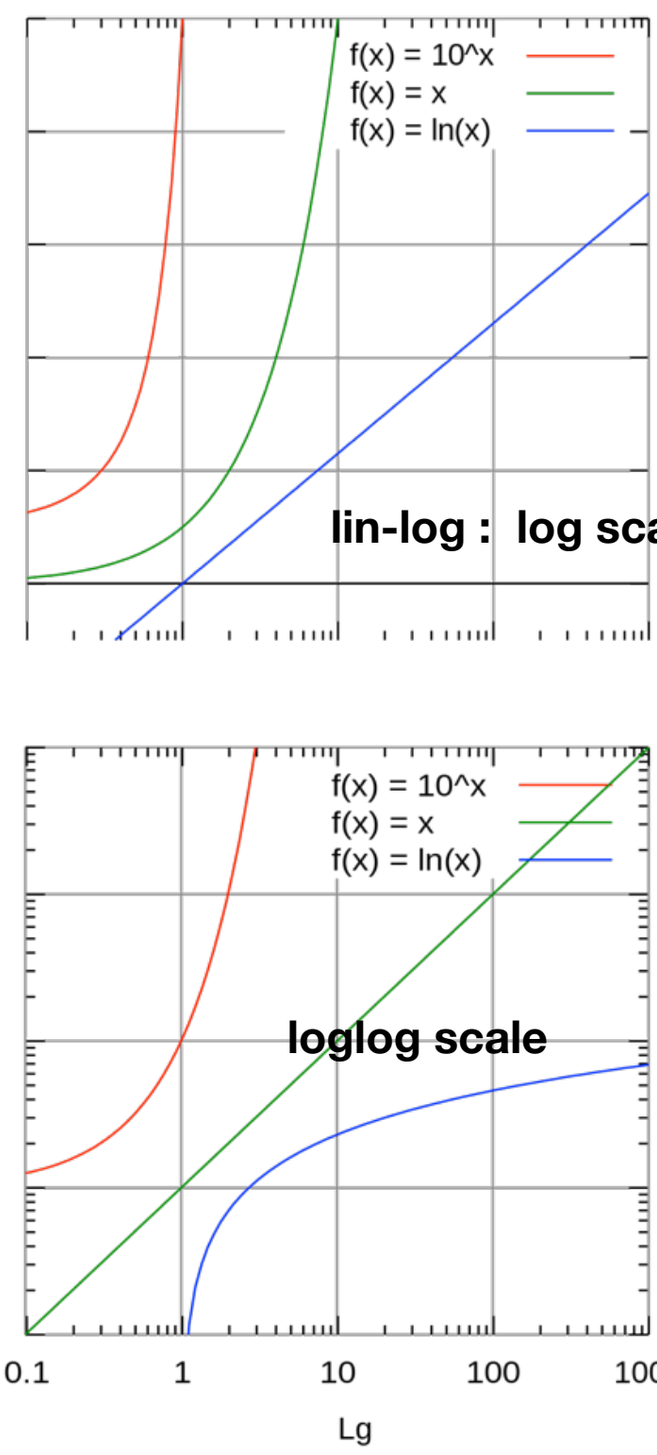
(x; f(x)) lin-lin

(x; $\log_{10}$(f(x))) log-lin

($\log_{10}$(x); f(x)) lin-log

($\log_{10}$(x); $\log_{10}$(f(x))) log-log



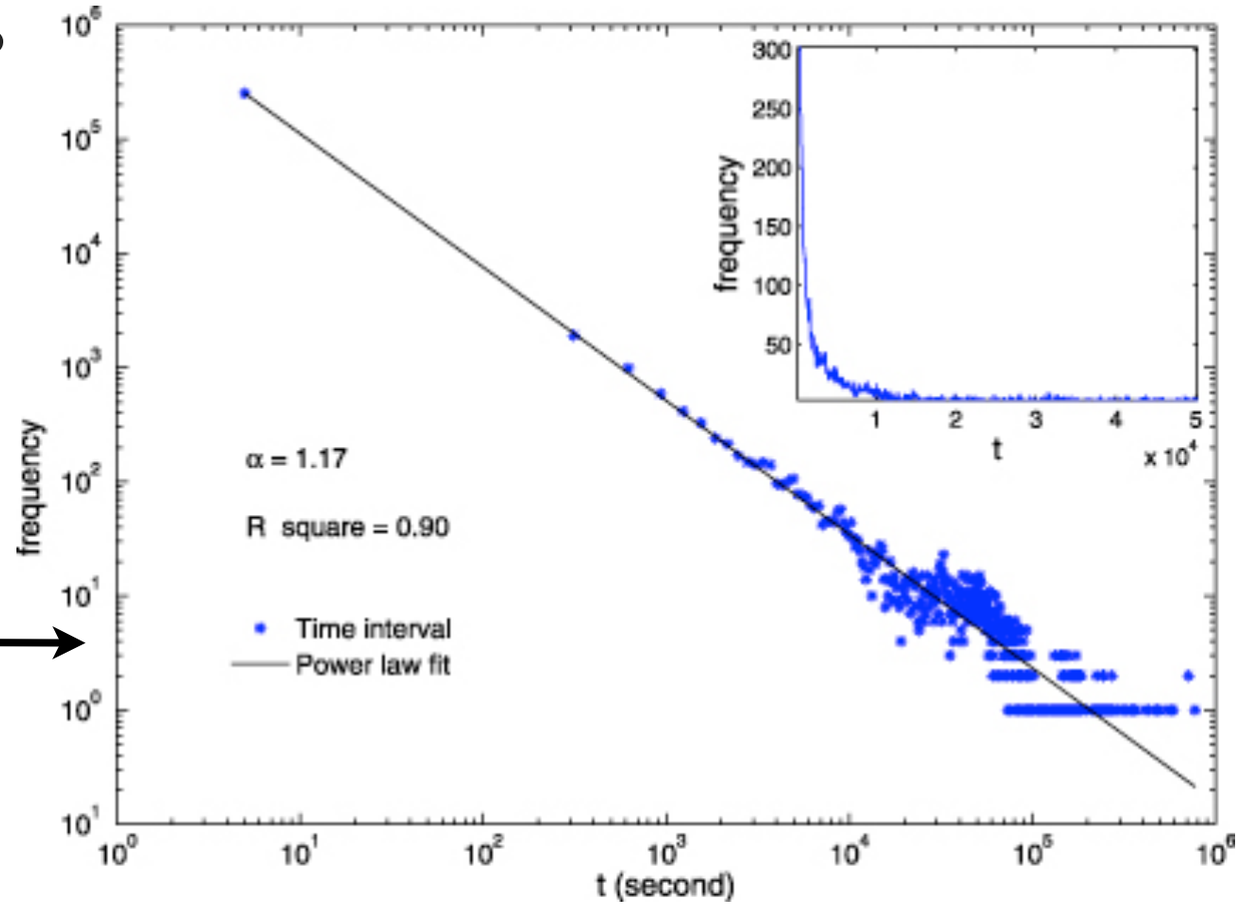


log-lin : log scale sur y

Représentations graphiques

- **Lois de puissance $y = Lx^k$ et loglogscale ?**
  - => apparait comme une droite !
  - ***log(y) = k*log(x) + log(L) ~ Y = k*X + C***

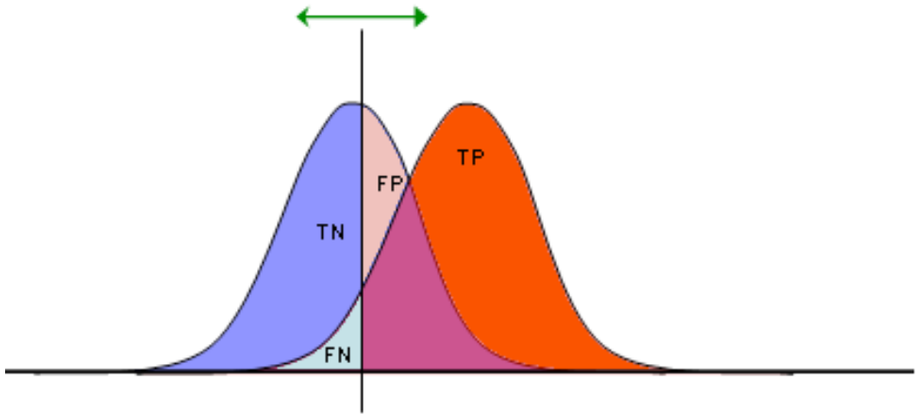
Attention à ce type d'approximation (queue lourde difficile à modéliser pour un bon ajustement) :
- regrouper par bins ou utiliser un ajustement sur la CDF !
- utiliser la distribution la plus adaptée (discrète ? bornée ?)



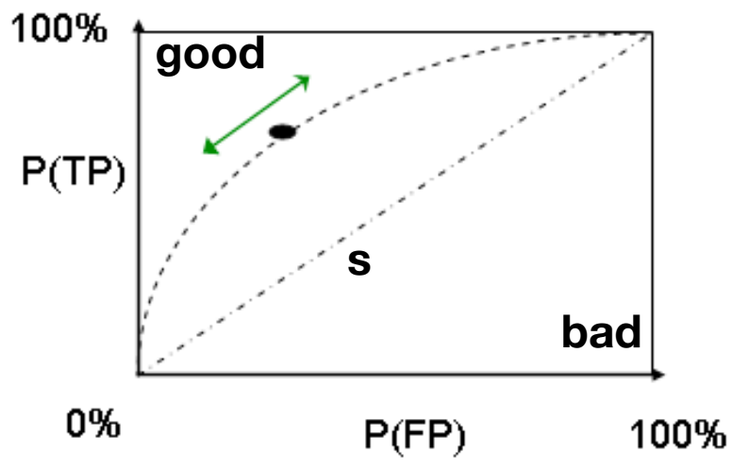
- **Notes : ne pas s'en servir à tort et à travers !**
  - log-lin (semilog y) permet aussi de visualiser $y = k^x$ comme une droite
  - les logscale peuvent être également utiles si les intervalles de valeurs en abscisse et ordonnée diffèrent énormément...

Représentations graphiques

- **Courbe ROC : Receiver Operating Characteristic**
  - Performance d'un test : Faux positifs vs. Faux négatifs en fonction d'un seuil s
    - Exemple du spam ou MPLS (encore !)



TP	FP
FN	TN
1	1



Visualisation

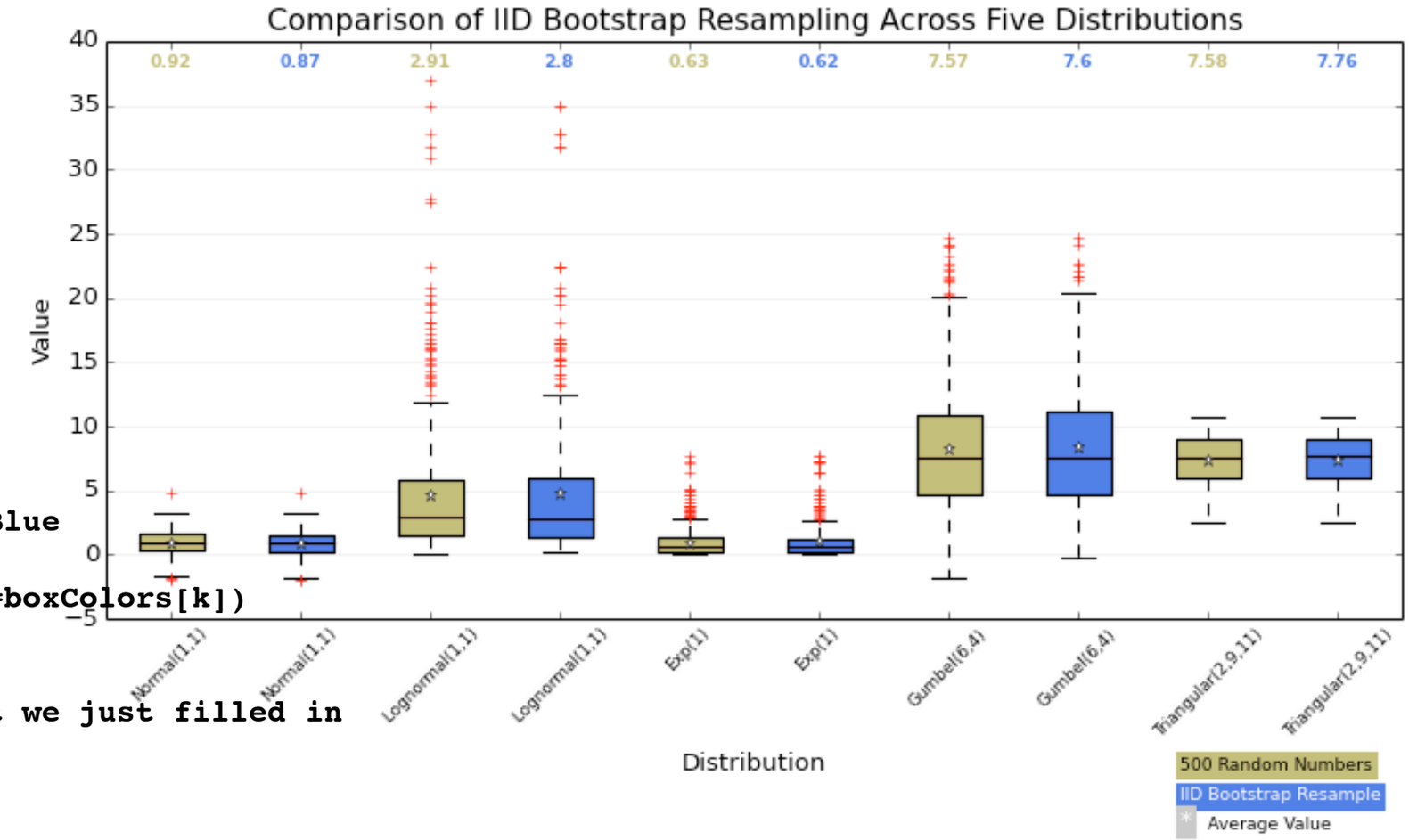
- **Visualisation : gnuplot pour calibrer et matplolib (python) pour de vrai !**

```
... # Now fill the boxes with desired colors
boxColors = ['darkkhaki','royalblue']
numBoxes = numDists*2
medians = range(numBoxes)
for i in range(numBoxes):
  box = bp['boxes'][i]
  boxX = []
  boxY = []
  for j in range(5):
      boxX.append(box.get_xdata()[j])
      boxY.append(box.get_ydata()[j])
boxCoords = zip(boxX,boxY)

# Alternate between Dark Khaki and Royal Blue
k = i % 2
boxPolygon = Polygon(boxCoords, facecolor=boxColors[k])
ax1.add_patch(boxPolygon)

# Now draw the median lines back over what we just filled in
med = bp['medians'][i]
medianX = []
medianY = []
for j in range(2):
    medianX.append(med.get_xdata()[j])
    medianY.append(med.get_ydata()[j])
    plt.plot(medianX, medianY, 'k')
    medians[i] = medianY[0]

# Finally, overplot the sample averages, with horizontal alignment in the center of each box
plt.plot([np.average(med.get_xdata())], [np.average(data[i])],
         color='w', marker='*', markeredgecolor='k') ...
```



Calcul statistique avancée

- **Calcul Rstat**
  - à essayer d'urgence !
  - Un premier exemple de base (moyenne + écart type) :

```
./DT-SPF-LLPLUS $topi 7 tmp > $out;
echo "v1" > slop
cat $out | egrep -v "^[1] " | awk '{print $3}'  >> slop

echo "nom<-read.table(\"slop\",header=T)" > script.r
echo "var<-nom" >> script.r
echo "mean(var)" >> script.r
echo "sd(var)" >> script.r

R --no-save < script.r > r.out

mean2=`cat r.out | tail -5  | grep -v ">" | grep -v "v" | xargs | cut -d' ' -f1 | cut -d'.' -f1`;
sd2=`cat r.out | tail -5  | grep -v ">" | grep -v "v" | xargs | cut -d' ' -f2 | cut -d'.' -f1`;
```

Rstat : percentiles

```
./DT-SPF-LLPLUS $topi 13 tmp > $out;
echo "v1" > slop
cat $out | egrep -v "^[1] " | awk '{print $3}'    >> slop

echo "nom<-read.table(\"slop\",header=T)" > script.r
echo "var<-nom" >> script.r
echo "mean(var)" >> script.r
echo "sd(var)" >> script.r
echo "x<-unlist(var)" >> script.r
echo "quantile(x,probs=c(0.90,0.95))" >> script.r

R --no-save < script.r > r.out

cat r.out | tail -11  | grep -v ">" | grep -v "v" | xargs > plop.tmp

mean=`cat plop.tmp | cut -d' ' -f1 | cut -d'.' -f1`;
sd=`cat plop.tmp | cut -d' ' -f2 | cut -d'.' -f1`;
qTBFH90=`cat plop.tmp | cut -d' ' -f5 | cut -d'.' -f1`;
qTBFH95=`cat plop.tmp | cut -d' ' -f6 | cut -d'.' -f1`;
```

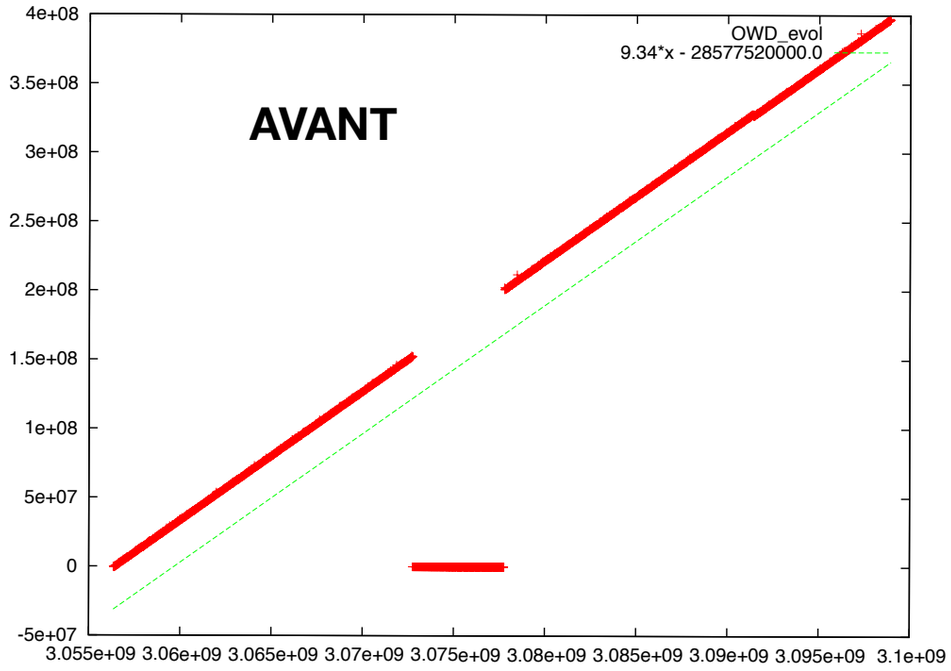
Rstat : les moindres carrés

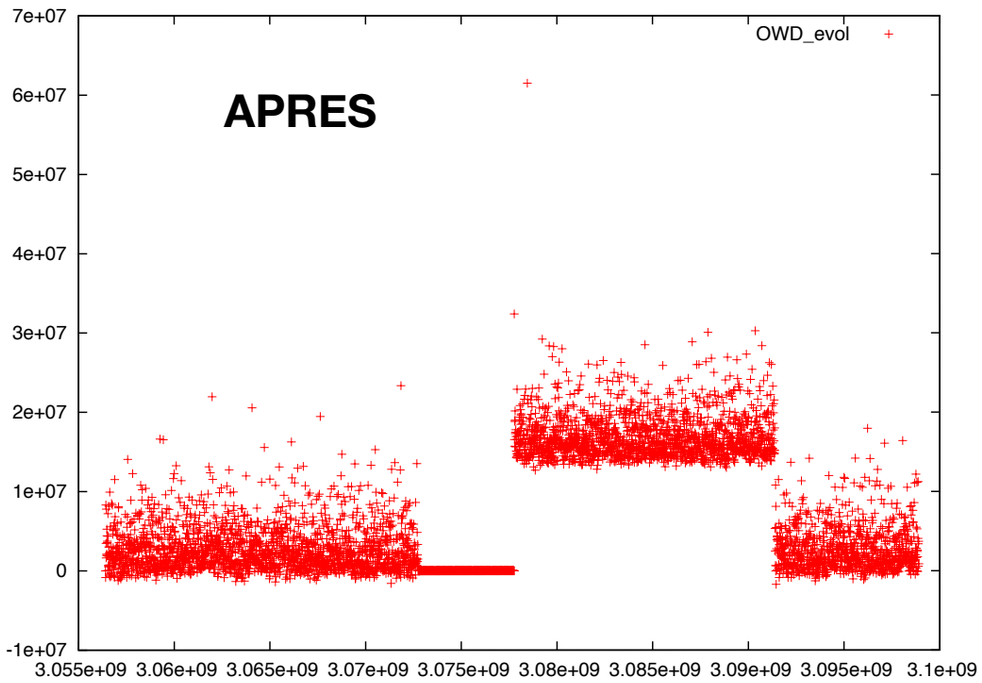
- **Dans le front end R directement :**

```
# nom<-read.table("abs",header=T)
# varx<-nom
# x<-unlist(varx)
# abs<-c(x)

# nom<-read.table("ord",header=T)
# vary<-nom
# y<-unlist(vary)
# ord<-c(y)

# Maf<-lsfit(abs,ord)
# Maf$coefficients
```





Les moindres carrés : illustration