

# **Ruby**によるデータ解析

## **Data Analysis in Ruby**

2016年2月19日

株式会社ネットワーク応用通信研究所 前田 修吾

***NaCl***

# 自己紹介

## ■ 名前

- 前田 修吾

## ■ 所属

- 株式会社ネットワーク応用通信研究所(1999年～)
- 一般財団法人Rubyアソシエーション(2007年～)

## ■ オープンソース活動

- Ruby開発者(1997年～)

# 本日のテーマ

- Rubyによるデータ解析

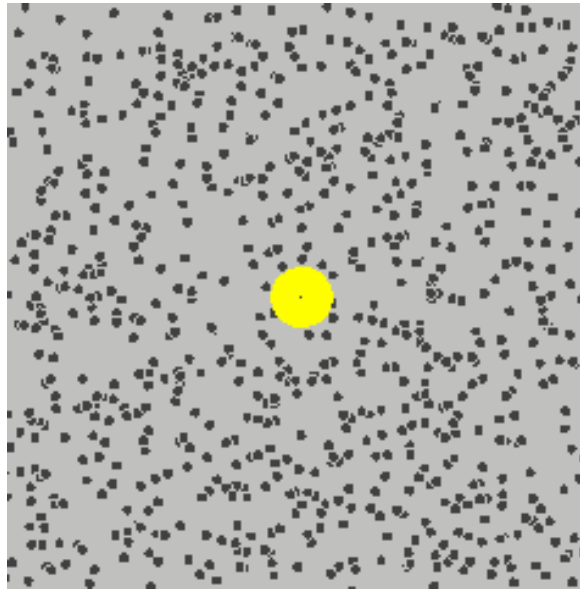
# データ解析の例

# 株式市場のブラウン運動

- M. F. M. Osborne, *Brownian Motion in the Stock Market*, Operations Research, 1959
- 『ウォール街の物理学者』という書籍で紹介
- 株価の対数とブラウン運動における微粒子の座標との類似性
- 統計力学的手法を株価に適用

## ブラウン運動

- 液体のような溶媒中に浮遊する微粒子が不規則に運動する現象



[https://commons.wikimedia.org/wiki/File:Brownian\\_motion\\_large.gif](https://commons.wikimedia.org/wiki/File:Brownian_motion_large.gif) CC-BY-SA 3.0 by Lookang

# 1. 株価の変化

- 株価の変化は離散的(1/8ドル単位)
- 株価の対数も同じ

## 2. 取引数

- 単位時間あたりに有限の取引(あるいは決定)が行われる
  - 一つの株に対して0～1000あるいはそれ以上



### 3. Weber-Fechnerの法則

- 精神物理学の基本法則
- 感覚量Eは刺激量の強度Rの対数に比例する
  - $E = C \log R$
  - 強度100の刺激が200に増加した場合の感覚量と、  
強度200の刺激が400に増加した場合の感覚量は同じ
- 株価という刺激とそれに対するトレーダー・投資家の主観的感覚はこの法則に従うと仮定する

# 統計学的アプローチ

- 金融の知識のない統計学者がNY市場の取引データを解析したら？
  - 集団が均質かどうか
  - 各属性・変数の関連性

# 株価の分布

- 株価の終値を主要な変数と推測
- 1000要素のサンプルの分布をプロット

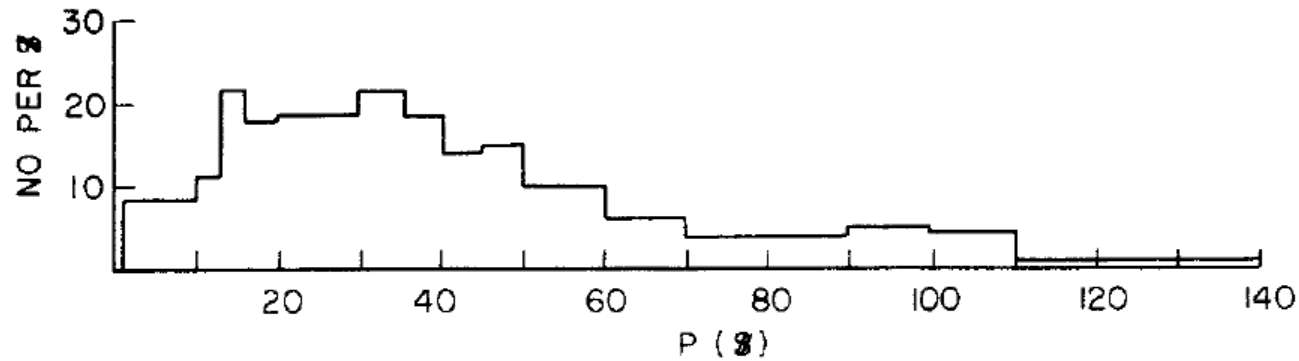


Fig 1 Distribution function of closing prices for July 31, 1956  
(all items, NYSE)

- 株価は正規分布に従わない
- 株価の対数は正規分布に従うかもしれない

# 正規分布

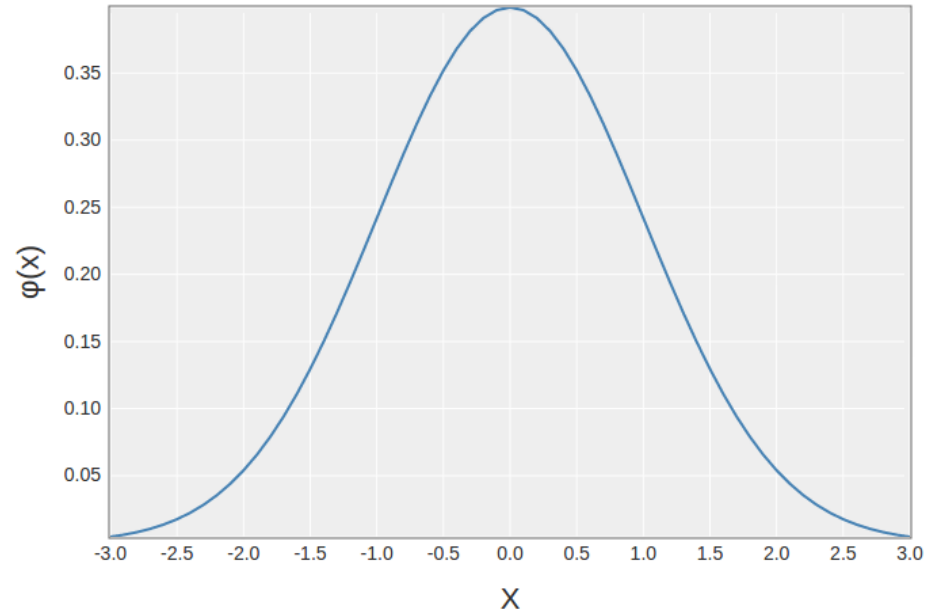
## ■ 以下の確率密度関数を持つ

- $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

## ■ 確率密度関数

- xがある範囲の値となる確率  
= 当該範囲の積分値

## ■ サンプルサイズを大きくすると 標本平均が真の平均に近づく (大数の法則)



# 平均・分散・標準偏差

## ■ 平均

- $\mu = \frac{1}{n} \sum_{i=1}^n x_i$

## ■ 分散

- 平均からのばらつきの指標（正の値にするため自乗誤差を使う）
- $\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$

## ■ 標準偏差

- 分散の平方根（ $x_i$ や $\mu$ の値と比較しやすくするため平方根を取る）
- $\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$

# 株価の対数の分布

- 正規分布ではない
- $\log_e P \approx 45$  周辺の副極大
  - この集団は均質でない
  - 少なくとも二つの下位集団
- 生データの確認
  - $\log_e P \approx 45$  周辺のデータに pfd (preferred) 属性

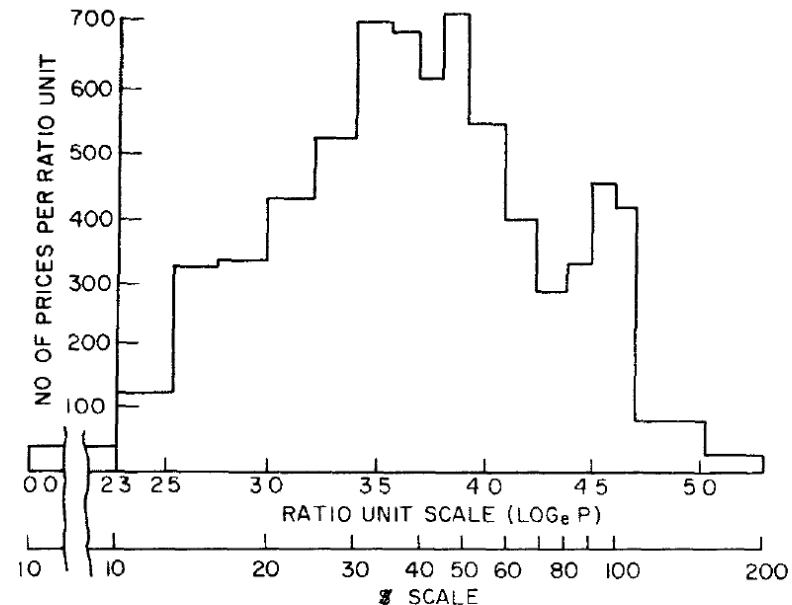
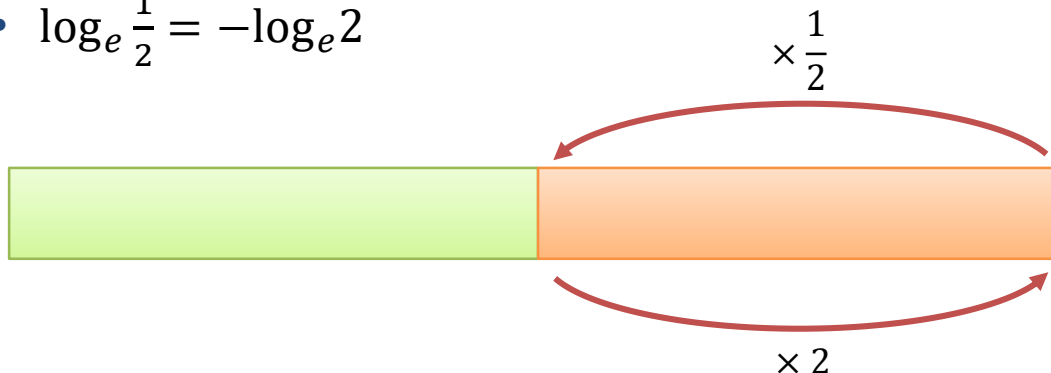


Fig 2 Distribution function for  $\log_e P$  on July 31, 1956 (all items NYSE)

# 対数収益率を使う根拠

- 価格の変化による利益・損失に対する主観的感覚を表す
- \$10から\$11の価格変化と、\$100から\$110の価格変化に対する主観的感覚は同じ
- なぜ収益率  $\frac{P(t+\tau)-P(t)}{P(t)}$  ではなく対数収益率  $\log_e \frac{P(t+\tau)}{P(t)}$  を使うか
  - 対数収益率を使うことで、上昇率と下降率の対称性が得られる
    - 価格が 50% 下降した後に 50% 上昇しても元の価格に戻らない
    - $\log_e \frac{1}{2} = -\log_e 2$



## 4. 論理的決定

### ■ 収益の期待値

- $A$ という一連の行動が $Y_{A1}, Y_{A2}$ という収益を確率 $\varphi(Y_{A1}), \varphi(Y_{A2})$ で生む
- 収益の期待値  $\varepsilon(Y_A) = \sum_i Y_{Ai} \varphi(Y_{Ai})$

### ■ 収益の期待値が高い行動を選択

- $\varepsilon(Y_A)$  と  $\varepsilon(Y_B)$  のどちらが大きい？

### ■ 価格 $P_0(t)$ の株を100株買うかどうか？

- $A =$  将来  $t + \tau$  に株を売るために買う
- $B =$  買わない
- $Y_A(\tau) = \Delta \log_e [100 P(t)] = \log_e [P(t + \tau) / P_0(t)]$
- $Y_B = 0$
- $Y_A(\tau)$ の期待値の見積が正か負かによって論理決定を行う



## 5. 市場の平等性

### ■ $\Delta \log_e P$ を買い手は正、売り手は負と判断する

- $E\varepsilon(\Delta \log_e P)_S + E\varepsilon(\Delta \log_e P)_B = 0$ 
  - ここで  $P$  は1株当たりの価格、 $E\varepsilon$  は期待値の見積

### ■ 市場全体では以下の式のような状況

- $E\varepsilon(\Delta \log_e P)_{M=S+B} = 0$
- 上記の式では見積を表す  $E$  はなくてもよいかもしれない

## 6. 株価の収益率の分布

- 以下の  $Y(\tau)$  は、平均 0、標準偏差  $\sigma_{Y(\tau)}$  の正規分布に従うと予測される
  - $Y(\tau) = \log_e [P(t + \tau) / P_0(t)]$
- $\sigma_{Y(\tau)}$  は取引数の平方根に比例する
- 取引数が時間上均一に分布すると考えると
  - $\sigma_{Y(\tau)}$  は時間間隔の平方根に比例する
  - すなわち、 $\sigma_{Y(\tau)}$  は  $\sigma\sqrt{\tau}$  という形式となる

## 7. 数学的表現

### ■ $k$ 個のランダムな独立変数 $y(i) = i, \dots, k$ を仮定する

- $y(i) = \Delta_{i\delta} \log_e P = \log_e [P(t + i\delta) / P(t + \{i - 1\}\delta)]$ 
  - ここで、 $P(t)$  はある銘柄の時間  $t$  における価格、 $\delta$  は取引間の小さな時間間隔
  - $y(i)$  は同じ標準偏差  $\sigma(i) = \sigma'$  を持つと仮定

### ■ $k$ 回の取引、 $\tau = k\delta$ 時間後の $Y(\tau)$ を以下のように定義

- $Y(\tau) = Y(k\delta) = \sum_{i=1}^{i=k} y(i) = \log_e [P(t + \tau) / P(t)] = \Delta_\tau \log_e P(t)$

### ■ $Y$ の標準偏差

- $\sigma_{Y(\tau)} = \sqrt{\varepsilon(Y^2) - [\varepsilon(Y)]^2} = \sqrt{\sum_{i=1}^{i=k} \sigma^2(i)} = \sqrt{k} \sigma' = \sqrt{\tau / \delta} \sigma'$

### ■ 中心極限定理により、 $y(i)$ の分布によらず $k$ が大きくなると $Y(\tau)$ は正規分布に近づく

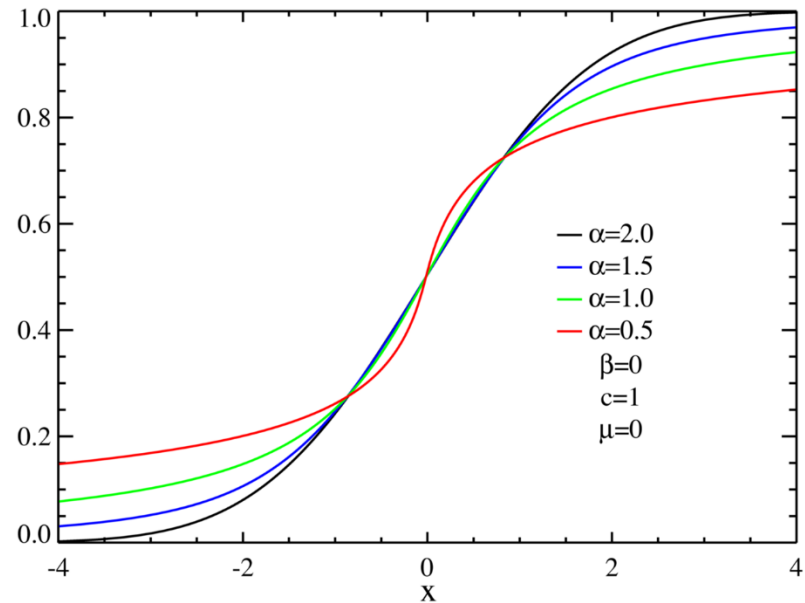
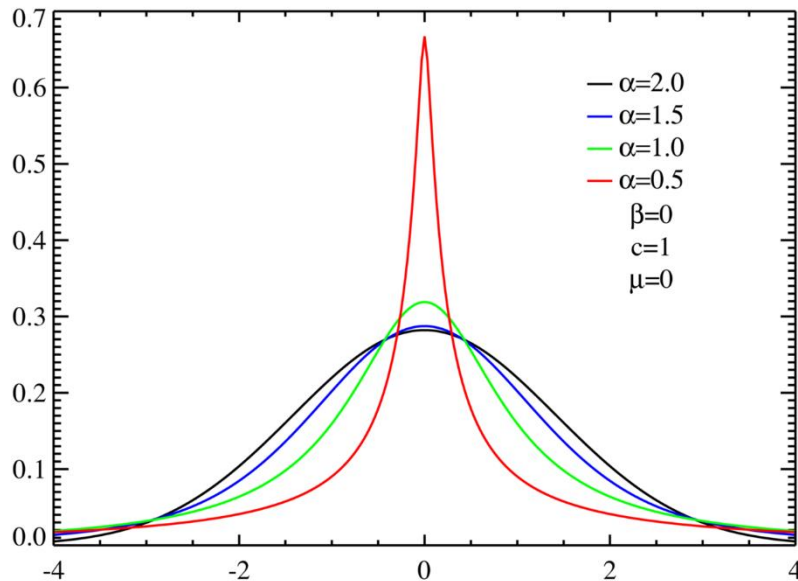
# 観測データとの比較

- 後でRubyで確認

# 収益率の分布の非正規性

- Benoit Mandelbrot, *The Variation of Certain Speculative Prices*, Journal of Business, 1963
- Benoit Mandelbrot, *The Variation of Other Speculative Prices*, Journal of Business, 1967
- 収益率の分布は、安定分布だが正規分布ではない
- 安定分布は  $\alpha$  というパラメータで分布の裾の広さが決まる
  - 小さいほど裾が広い(正規分布は  $\alpha = 2$  )
  - $\alpha \leq 1$  の場合、大数の法則に従わない
  - $\alpha < 2$  の場合、中心極限定理は成り立たない
- Mandelbrotは収益率は  $1 < \alpha < 2$  の安定分布に従うと考えた

# 安定分布のPDFとCDF



左: 安定分布の確率密度関数

[https://commons.wikimedia.org/wiki/File:Levy\\_distributionPDF.png](https://commons.wikimedia.org/wiki/File:Levy_distributionPDF.png) public domain by PAR

右: 安定分布の累積分布関数

[https://commons.wikimedia.org/wiki/File:Levy\\_distributionCDF.png](https://commons.wikimedia.org/wiki/File:Levy_distributionCDF.png) CC-BY-SA 3.0 by PAR

# **Ruby**によるデータ解析 の現状

# Why Ruby?

## ■ Pythonと同じ理由 (『Pythonによるデータ分析入門』)

- 「糊(グルー)」としてのRuby
  - C/C++/FORTRANなどで書かれたコードをつなぎ合わせる
- 「2つの言語を利用する」ことの問題を解決する
  - アプリケーション開発とデータ解析で同じ言語を使う

## ■ 多様性は善



# ライブラリ・ツール群

分類	Python	Ruby
ベクトル・行列	NumPy	NArray, NMatrix
データフレーム	pandas	daru
可視化	matplotlib	Nyaplot
対話環境	Jupyter/IPython	Jupyter/IRuby
科学計算全般	SciPy	SciRuby

# NArray

- 多次元配列ライブラリ
- 連続したメモリ領域に要素を配置し、次元毎の要素数を固定することでインデックスアクセスを行う
  - $4 \times 3$ の多次元配列 $a$ のとき、 $a[i, j]$  は  $i + 4 * j$  番目

$a[0, 0]$	$a[1, 0]$	$a[2, 0]$	$a[3, 0]$	$a[0, 1]$	$a[1, 1]$	$a[2, 1]$	.....
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-------

- 次元順序は次期開発版で変更？
- 多次元配列同士の演算、多次元配列とスカラー値の演算が高速
- 最新リリースは2013年2月27日
  - 別リポジトリで次期開発版が開発中

# NMatrix

- SciRubyプロジェクトの行列計算ライブラリ
- NArrayより後発の競合ライブラリ
  - 同時に使えないという意味でも競合...
    - ドキュメントには `require "nmatrix/nmatrix"` で回避せよとあるが、その後で `require "narray"` しているライブラリがあるとエラー...
- 最新リリースは2016年1月18日

# **daru**

- **Data Analysis in R**uby
- 以下のデータ型を提供
  - Daru::Vector
  - Daru::DataFrame

# Daru::Vector

## ■ 1次元ベクトル

```
v1 = Daru::Vector.new([40, 20, 30])  
v2 = Daru::Vector.new([10, 60, 30])  
v1 + v2 ==> Vector[50, 80, 60]  
v1 * 10 ==> Vector[400, 200, 300]  
v1.mean ==> 30.0
```

# Daru::DataFrame

## ■ 2次元のスプレッドシート風データ構造

```
df = Daru::DataFrame.new(x: [1,2,3], y: [4,5,6])
```

## ■ 各列がDaru::Vectorによって表現される

## ■ 列単位の処理は速いが、行単位の処理は遅い

- とくに遅い例

```
df.filter_rows {|row| row[:x] > 1}
```

- 高速化例

```
df.where(df[:x].gt(1))
```

# Nyaplot

- プロットライブラリ
- Jupyter notebook上で動作
- WebGLを利用した3Dプロットも可能
- 散布図のプロット例

```
plot = Nyaplot::Plot.new  
sc = plot.add(:scatter, [0,1,2,3,4], [-1,2,-3,4,-5])  
plot.show
```

# Jupyter/IRuby

## ■ Jupyter

- Webアプリケーションによる対話環境
- Mathematica風のノートブック
  - プログラムの対話的実行
  - グラフの描画
- プログラミング言語非依存
  - 各言語の実行環境をカーネルとして提供
  - プロセス間通信

## ■ IRuby

- Jupyter用のRubyカーネル



# SciRuby

- Ruby用の科学計算ライブラリの開発プロジェクト
- NMatrix, daru, IRuby, NyaplotもSciRubyの一部
- Statsample
  - 統計用ライブラリ
- Distribution
  - 確率分布用ライブラリ

# デモ

## ■ 株価データの解析

- <https://github.com/shugo/DCW2016/blob/master/Stock.ipynb>

## ■ PSDSから取得したデータの解析

- <https://github.com/shugo/DCW2016/blob/master/PSDS.ipynb>
- 相関行列
- 回帰分析
- マハラノビス距離による外れ値の検出

## 今後の課題

# 利用促進

- 現状は利用者が少ない
- 今後の方策
  - ドキュメントの充実
  - 利用事例の発信

# 機能追加・機能改善

## ■ Python/R等に比べ機能的に見劣りする

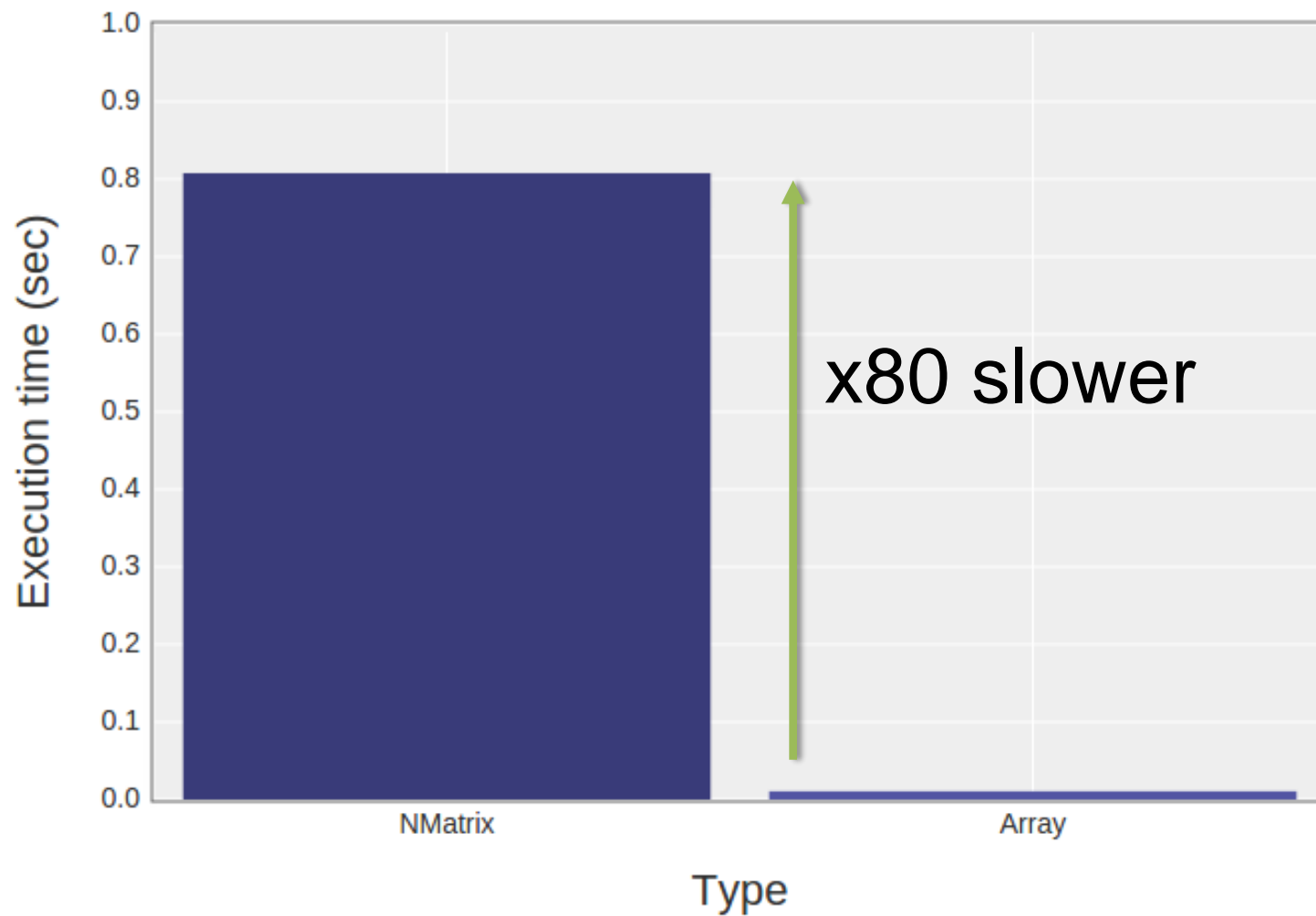
- 欠損値のチェック・穴埋め
- 時系列データの再サンプリング
- 変化率の計算
- 金融関係の機能
- 経験累積分布
- Q-Qプロット
- マハラノビス距離
- ...

# 性能改善

- NMatrixは遅い
- ベンチマークプログラム

```
m = NMatrix.dindgen([100000])  
a = 0.0.step(99999.0, 1.0).to_a  
  
Benchmark.bmbm do |x|  
  x.report("NMatrix") do  
    m.mean  
  end  
  
  x.report("Array") do  
    a.sum.quo(a.size)  
  end  
end
```

# ベンチマーク結果



# 他言語との差別化

## ■ 内部DSLの活用

- ブロックの効果的利用

## ■ Rubyの動的性質の活用

- スキーマレスなデータとの親和性



# 内部DSLの活用提案

## ■ Daru::DataFrame#where

- 現在の記法

```
df.where(df[:x].gt(1))  
df.where(df[:x].gt(1) & df[:y].lteq(10))
```

- 拡張案

```
df.where { :x > 1 }  
df.where { (:x > 1) & (:y <= 10) }
```

# Rubyの言語拡張案

## ■ ブロック内でのみRefinementsを有効にする

- [https://github.com/shugo/ruby/tree/eval\\_using](https://github.com/shugo/ruby/tree/eval_using)

```
module FixnumDivExt
  refine Fixnum do
    def /(other)
      quo(other)
    end
  end
end

p 1 / 2
instance_eval(using: FixnumDivExt) do
  p 1 / 2
end
p 1 / 2
```

# 拡張の仕組み

```
df.where { (:x > 1) & (:y <= 10) }
```

Symbol#>などの置き換え

抽象構文木

旧形式に変換

```
df.where(df[:x].gt(1) & df[:y].lteq(10))
```

## デモ2

### ■ **Daru::DataFrame#where**の拡張

- <https://github.com/shugo/DCW2016/blob/master/DaruRefinements.ipynb>

# References

- ジェイムズ・オーウェン・ウェザーオール, *ウォール街の物理学者*, 早川書房, 2015
- M. F. M. Osborne, *Brownian Motion in the Stock Market*, Operations Research, 1959
- Benoit Mandelbrot, *The Variation of Certain Speculative Prices*, Journal of Business, 1963
- Benoit Mandelbrot, *The Variation of Other Speculative Prices*, Journal of Business, 1967
- 平岡和幸・堀玄, *プログラミングのための確率統計*, オーム社, 2009
- Wes McKinney, *Pythonによるデータ分析入門*, オライリー・ジャパン, 2013