# らくらく電脳 ruby

西澤 誠也 <seiya@kugi.kyoto-u.ac.jp>

http://ruby.gfd-dennou.org/workshop200403/nishizawa/rakuraku/rakuraku.html

# 目次

- 例外（エラー）
  - http://ruby.gfd-dennou.org/tutorial/3-gen/
- 高速化
  - http://ruby.gfd-dennou.org/tips/highspeed.html
- 欠損値処理
  - http://ruby.gfd-dennou.org/tutorial/2-gen/missing.html
  - http://ruby.gfd-dennou.org/products/narray_miss/index-j.html
- SSL2
  - http://www-mete.kugi.kyoto-u.ac.jp/seiya/ruby/workshop/ssl2.html
  - http://ruby.gfd-dennou.org/products/ruby-ssl2/index-j.html


seiya@kugi.kyoto-u.ac.jp

# 例外 -1

期待しない場合が生じると例外が上がる（ようにする）

- 通常はプログラムを終了し、メッセージを出力する

```
%cat exception.rb
require "narray"
def p_4(x)
  p x[4]
end
x = NArray[0,1,2,3]
p_4(x)

%ruby exception.rb
exception.rb:3:in `[]': index out of range (IndexError)
        from exception.rb:3:in `p_4'
        from exception.rb:6
```


seiya@kugi.kyoto-u.ac.jp

# 例外 -2

- 自分で例外が上がるようにする
  - raise

```
%cat raise.rb
def power(x,y)
 if Numeric===x
   return x**y
 else
   raise "x must be a kind of Numeric"
 end
end
p power(4,2)
p power("A",2)
%ruby raise.rb
16
raise.rb:5:in `power': x must be a kind of Numeric (RuntimeError)
        from raise.rb:9
```


seiya@kugi.kyoto-u.ac.jp

# 例外 -3

- 例外が上がった場合プログラムを終了しないようにする
  - rescue, ensure

```
%cat rescue.rb
a = Array.new(2)
begin
 file = File.open("hello.txt")
 a[0] = file.readline
 a[1] = file.readline
rescue
 a[1] = "file is end"
ensure
 file.close if file
end
p a
%cat hello.txt
Hello World!
%ruby rescue.rb
["Hello World!\n", "file is end"]
```


seiya@kugi.kyoto-u.ac.jp

# 高速化 -1

rubyのデメリットである遅さを最小限におさえる

→ループの削減

- NArray のメソッド利用 （内部はCのため高速）
  - sum, mean, stddev, min, max, median, mul_add

```
%cat covariance-1.rb
require "narray"
n = 100000
x = NArray.sfloat(n).indgen
y = NArray.sfloat(n).indgen(10)
p x.mul_add(y,0)/(n-1)
```

1=>0.05s, 2=>0.97s

```
%cat covariance-2.rb
require "narray"
n = 100000
x = NArray.sfloat(n).indgen
y = NArray.sfloat(n).indgen
sum = 0
for i in 0...n
 sum += x[i]*y[i]
end
p sum/(n-1)
```


seiya@kugi.kyoto-u.ac.jp

# 高速化 -2

- 平均からのずれの計算
  - １次元配列の場合

```
xa = x - x.mean
```

  - ２次元配列の場合
    - １次元目平均からのずれ

```
xa = x - x.mean(1)
```

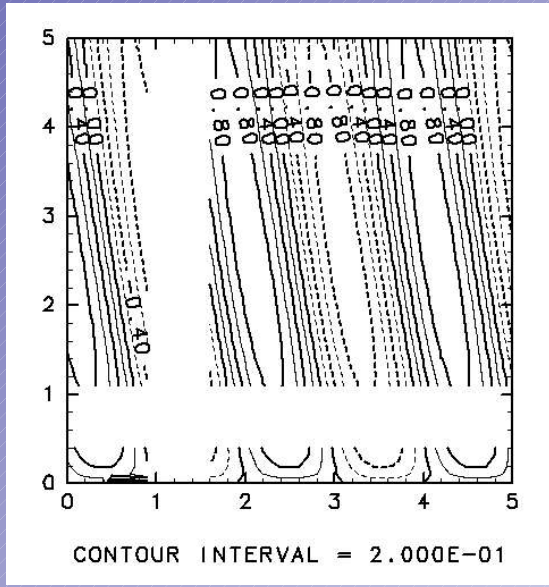
    - ０次元目平均からのずれ

```
xa = x - x.mean(0).reshape!(0,x.shape[1])
```


seiya@kugi.kyoto-u.ac.jp

# 欠損値処理 -1

- NArrayMiss Class
  - 内部で欠損値情報をもつ配列
  - 内部で欠損値処理を行う




```
%cat miss.rb
require "numru/dcl"
require "narray_miss"
include NumRu
rmiss = -999.9
n = 50; min, max = 0.0, 5.0
t = NArray.sfloat(n+1,1).indgen!*(max-min)/n
z = NArray.sfloat(1,n+1).indgen!*(max-min)/n
t[10..15,0] = rmiss; z[0,5..10] = rmiss
t = NarrayMiss.to_nam(t,t.ne(rmiss))
z = NArrayMiss.to_nam(z,z.ne(rmiss))
uz = NMMath::exp(-0.2*z)*(z**0.5)
tz = -2.0*NMMath::exp(-0.1*z)
u = uz*NMMath::sin(3.0*(tz+t))
```

```
DCL::gropn(1)
DCL::gllset("lmiss",true)
DCL::grfrm
DCL::grswnd(min, max, min, max)
DCL::uspfit
DCL::grstrf
DCL::usdaxs
DCL::udcntr(u)
DCL::grcls
```

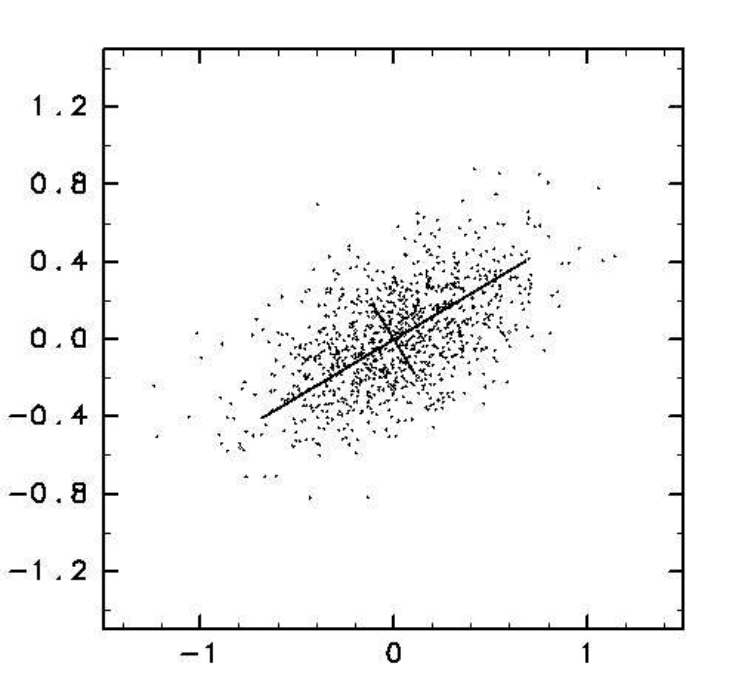

seiya@kugi.kyoto-u.ac.jp

# SSL2

Fujistu 科学用サブルーチンライブラリ（SSL2）が使えます

- 線形計算，固有値固有ベクトル，非線形計算，極値問題，補間・近似，変換数値積分，微分方程式，特殊関数，疑似乱数

Example: EOF (eof.rb)




http://ruby.gfd-dennou.org/workshop200403/nishizawa/rakuraku/eof.rb
http://ruby.gfd-dennou.org/workshop200403/nishizawa/rakuraku/eof_ssl2.rb


seiya@kugi.kyoto-u.ac.jp