

# 大規模データ処理法 BigData Processing

植原 啓介  
Keisuke UEHARA

# 科目概要

## Goal

データサイエンティストに必要なスキルの1つである、大規模解析のためのITスキルを身につけることを目標とする。少量のデータであれば、Excelなどの表計算ソフトでもある程度の解析は可能である。しかし、数TB、数億レコードというデータは表計算ソフトで扱うことはできない。本講義では、大規模データを扱うための処理フレームワークについて学習する。

A purpose of this course is to master an IT skill for the large-scale analysis that is one of the skills necessary for data scientist. If it is small data, analysis can be done by the spreadsheets software such as Excel. However, in case that the size of data is big such as GB or million records, Spreadsheets software, such as Excel, cannot handle the data. In this course, students learn frameworks to process such large-scale data.

# 主題と目標/授業の手法など

## Purpose and Goal / Course style

本授業はデータサイエンティストを目指す学生のために、Excelで扱えない規模のデータの処理方法について学ぶ授業である。統計そのものやプログラミングそのものを扱うのではなく、データの解析に必要な前処理、計算結果を得るのに時間がかかるような大規模データ処理などについて実習を交えて学ぶ。本授業の受講者が、大規模なデータを短時間で処理するのに必要な技術を身につけることを目標とする。

授業は講義と演習によって進める。学生が実際のデータを用いた演習を進めることによって、本当に使えるスキルとして身に付けることを目指す。

This is a course for a student to be a data scientist to learn methods to handle a large-scale data which cannot be processed by Excel. This course don't treat statistics itself nor programming itself. This course treats preprocessing methods for data analysis and processing methods for large-scale data through the training. The purpose of this course is to master the skills processing large-scale data in short time. Class consists of lecture and training. This course aims at acquiring a really usable skill by a student pushing forward the practice using real data.

## 諸注意 Notification

- 学生が準備するソフト・機材  
必ず毎回、ネットワークに接続可能なラップトップコンピュータを持ってくるください。Windows / Mac / Linuxは問いません。
- Software and Machinery student must prepare  
Students have to bring your laptop computers. OS can be one of Windows, Mac and Linux.
- データを取得したり利用したりするために、外部のシステム（e-Stat(<https://www.e-stat.go.jp/>), Google, Twitter, Facebook等）のアカウント等を取得しなければならないことがあります。外部のシステムにアカウントを作りたくない人は履修を辞退してください。
- You might have to make accounts on the external systems (i.e. <https://www.e-stat.go.jp/>, Google, Twitter, Facebook). If you don't want to do it, please decline from this course.

## 諸注意 Notification

- 特別な環境にアカウントを用意する可能性があります。公序良俗に反する使い方はやめてください。
- You will have an account on a special server. Please use it well-mannered.
- 授業中に出される課題および授業後半で予定されている発表で成績をつけます。
- Grading will be done by assignments and presentation.
- ある程度のプログラミング能力や統計ソフトの使い方は前提として進みますので、不得意な人は予習/復習を忘れずに。（プログラミング言語としてはPython/Rubyを、統計ソフトとしてはPythonまたはRを基本とします。）
- You are assumed to aware with programing and statistics software. If you are not familiar with them, please take a time to study more than others. (Programing language "Python"/"Ruby" and statistics software "Python"/"R" will be used mainly)

# シラバス（１） / Syllabus (1)

1. ガイダンスと概要 / Guidance and Introduction [4/9]
  - 講義の内容及び進め方について説明する。  
Abstract and method of this course will be explained.
  - 大規模データ処理の簡単な例を紹介する。  
Simple example of large-scale data processing will be introduced.
2. CLIとシェル[4/16]
  - CLIとShell / CLI and Shell
  - パイプ / pipe
3. Pythonでのプログラミング / Programming in Python [4/23]
  - プログラミング言語であるPythonについて学ぶ。  
To learn the programming language Python.
  - Pythonの実行環境を紹介する。  
Python runtime environment will be introduced.
  - Pythonの基本的な文法を紹介する。  
The basic grammar of Python will be introduced.

## シラバス（２） / Syllabus (2)

### 4. 時系列データのクレンジング / Data cleansing of time-series data [5/7]

- 前処理の方法について学ぶ。  
To learn method for preprocessing.
- データの外観をしるための方法について学ぶ。  
To learn how to overview the data.

### 5. SNS（１） / SNS(1) [5/14]

- Twitter API / Twitter API
- Twitter情報の取得 / Acquisition of Twitter data
- Twitter情報のネットワーク解析 / Network analysis of Twitter data

### 6. SNS（２） / SNS(2) [5/21]

- Twitter情報のコンテンツ解析 / Content analysis of Twitter data
- 形態素解析 / Text segmentation
- google mapsの活用 / Using google maps

## シラバス（3） / Syllabus (3)

7. 位置情報（1） / Location information (2) [5/28]
  - 軌跡データの統合 / Unification of location tracking data
  - 単一軌跡データの解析 / Analysis of single location tracking data
  - 軌跡データの0次元解析 / 0 dimension analysis of location tracking data
  - 軌跡データの2次元解析 / 2 dimension analysis of location tracking data
8. 位置情報（2） / Location information (3) [6/4]
  - 軌跡データの1次元解析 / 1 dimension analysis of location tracking data
  - 距離-時間グラフによる解析 / Analysis using distance-time graph



# シラバス（４） / Syllabus (4)

9. 大規模データ処理システム（１）  
BigData processing system (1) [6/11]
  - 大規模データ処理システムの概要  
Overview of BigData processing systems
  - Hadoop / Hadoop
10. 大規模データ処理システム（２）  
BigData processing system (2) [6/18]
  - Hadoop streamingを使ったWikipediaアクセスログの解析  
Wikipedia access log analysis using Hadoop streaming
  - データウェアハウス / Data warehouse
11. 大規模データ処理システム（３）  
BigData processing system (3) [6/25]
  - Hadoopを使った解析の実例の紹介  
Introducing real examples of analysis using Hadoop
  - Hadoopを使った軌跡情報の解析  
Analysis of location tracking data using Hadoop

# シラバス（５） / Syllabus (5)

## 12. 機械学習（１） / Machine learning (1) [7/2]

- パターン認識の概要 / Overview of pattern recognition
- 機械学習の概要 / Overview of machine learning

## 13. 機械学習（２） / Machine learning (2) [7/9]

- 機械学習に関する実習 / Training of machine learning

## 14. プレゼンテーション / Presentation [7/16]

- 学生によるミニプロの発表 / Presentation of mini-project by students

多少変更があるかもしれません。  
Contents can be changed.

# 本授業で扱うデータ

## Data to be used in this Course

- 何らかの時系列データ  
Time series data
- 交通流データ  
Vehicle traffic tracking data
- Twitter  
Twitter
- Wikipediaのアクセスログ  
<http://dumps.wikimedia.org/>

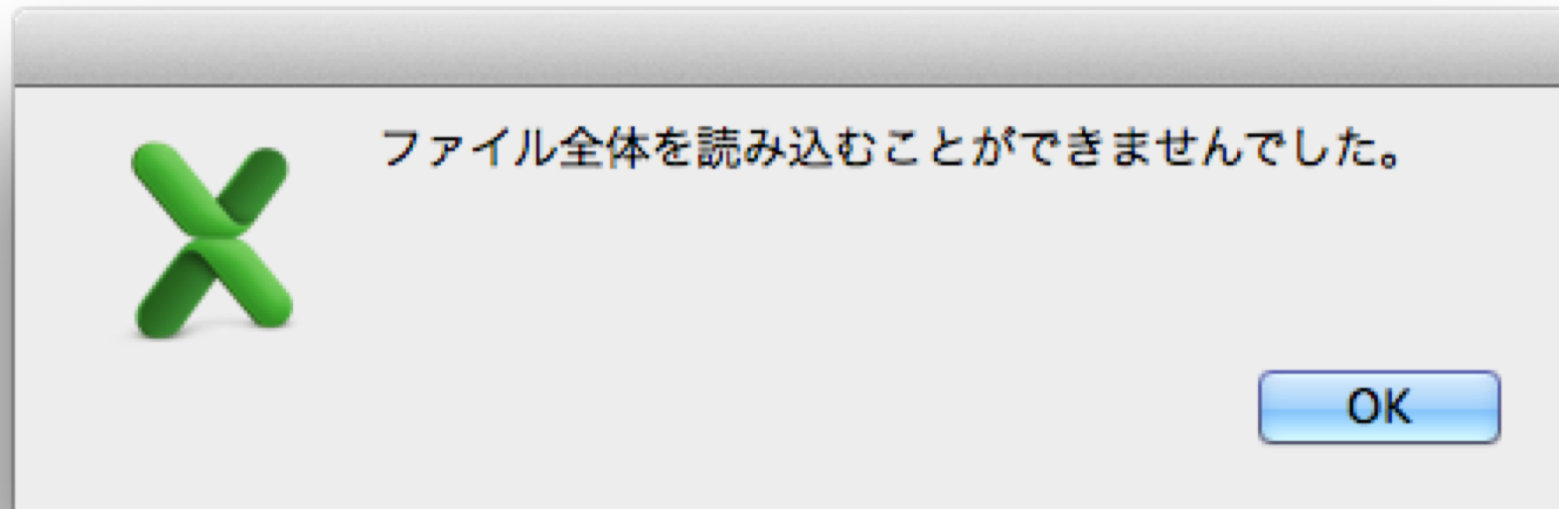
他に希望があればどうぞ。

If you have a request, please let me know.

# 導入

# Introduction

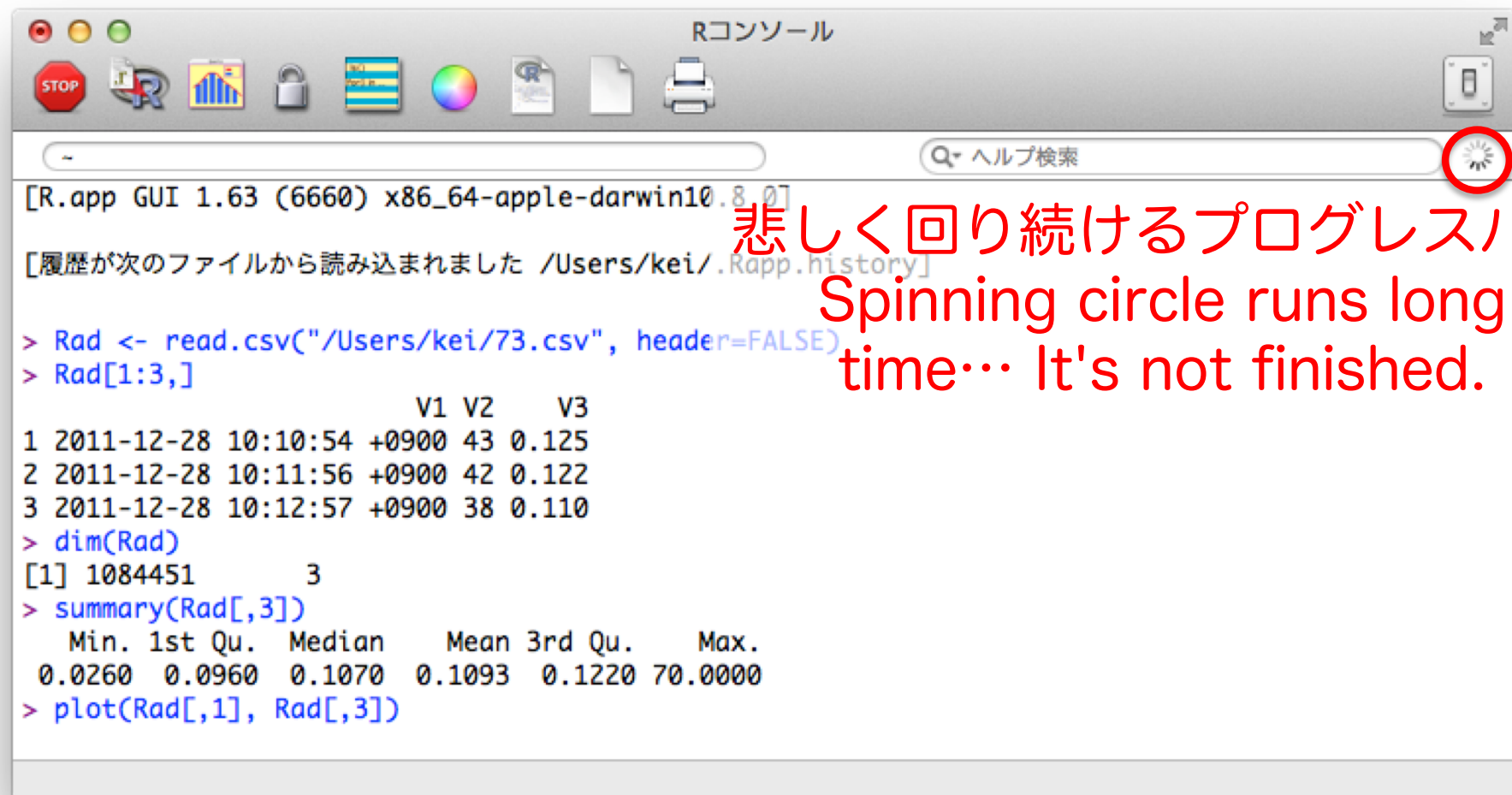
こんな表示に出会ったことはありますか？  
Have you faced this message?



- Excelのワークシートサイズの上限は、1,048,576 行 x 16,384 列
- Excel can load up to 1,048,576 liens x 16,384 rows.
- 読み込めたとしても、計算機のパフォーマンスの問題で使いものにならないことが多い
- If you succeed to load the file, performance of the software might be very bad.

# Rで頑張ってグラフ化を試みても OK, then I'll try to make a graph using R

- そうだ、Rなら $2^{31}-1$ 個のデータまでいける。
- R supports  $2^{31}-1$  data



The screenshot shows an R console window with the following content:

```
[R.app GUI 1.63 (6660) x86_64-apple-darwin10.8.0]
[履歴が次のファイルから読み込まれました /Users/kei/.Rapp.history]

> Rad <- read.csv("/Users/kei/73.csv", header=FALSE)
> Rad[1:3,]
      V1 V2  V3
1 2011-12-28 10:10:54 +0900 43 0.125
2 2011-12-28 10:11:56 +0900 42 0.122
3 2011-12-28 10:12:57 +0900 38 0.110
> dim(Rad)
[1] 1084451      3
> summary(Rad[,3])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0260  0.0960  0.1070  0.1093  0.1220 70.0000
> plot(Rad[,1], Rad[,3])
```

A red circle highlights a spinning progress bar icon in the top right corner of the window.

悲しく回り続けるプログレスバー  
Spinning circle runs long  
time... It's not finished.

そして約90分後、  
And 90 min later,

クラッシュ X\_X  
Crashed X\_X

私の計算機には無理でした。  
My computer has no enough resources.

## この問題を解決するためには？ How to solve these issues?

- 計算機を高性能なものに交換する。 (スケールアップ)
- Change to the high-performance computer (Scale up)
  - これは授業では扱わない予定  
There's no plan to give a lecture about this issue in this course.
- 複数の計算機で分散処理する。 (スケールアウト)
- Use distributed computing environment (Scale out)
  - この授業の後半でやる予定  
Plan to introduce this solution in the later part of this course.
- サンプリングを施し、データ数を減らす。既存の手法で扱えるようにする。  
(スケールダウン!?)
- Decrease the number of data by sampling. You can apply typical solutions. (Scale down?)
  - この授業の前半でやる予定  
Plan to introduce this solution in the early part of this course.



# One-Linerの威力(1)

## Power of One-Liner (1)

- Command Line Interface (CLI)は好きですか?
- Do you like "Command Line Interface (CLI)?"
- そもそもどういうデータなのか
- What is "One Liner" actually?  
`% head 73.csv`
- このデータはいったい何行あるのか
- How many lines this data have?  
`% wc -l 73.csv`

## One-Linerの威力(2) - ruby

### Power of One-Liner (2) - ruby

- 先ほどのデータから10,000個サンプリングしてみよう

- Try to get 10,000 samples from the data

```
% ruby -e "l = STDIN.readlines; 10000.times {print l.sample};" < 73.csv
```

- サンプリングして、本当に10,000サンプルあるか数えてみよう

- Are there 10,000 samples actually? Count them.

```
% ruby -e "l = STDIN.readlines; 10000.times {print l.sample};" < 73.csv  
| wc -l
```

- サンプリングして、ソートしてみよう

- Sample and sort.

```
% ruby -e "l = STDIN.readlines; 10000.times {print l.sample};" < 73.csv  
| sort
```

- サンプリングして、ソートして、重複したものを削除してみよう

- Sample, sort and delete duplicated lines.

```
% ruby -e "l = STDIN.readlines; 10000.times {print l.sample};" < 73.csv  
| sort | uniq
```

## One-Linerの威力(3) - ruby

### Power of One-Liner (3) - ruby

- サンプルリングして、ソートして、重複したものを削除した結果のサンプル数を確認してみよう

- Sample, sort, delete duplicated lines, then count lines.

```
% ruby -e "l = STDIN.readlines; 10000.times {print l.sample};" < 73.csv ¥  
  | sort | uniq | wc -l
```

- プロットしてみよう

- Try to plot them.

```
% ruby -e "puts 'set timefmt ¥'%Y-%m-%d %H:%M:%S +0900¥''; puts 'set  
datafile separator ¥',¥''; puts 'set xdata time'; puts 'plot ¥'-¥' using  
1:3'; l = STDIN.readlines; 10000.times {print l.sample}; puts 'end'" <  
73.csv | gnuplot -p
```

# One-Linerの威力(2) - python

## Power of One-Liner (2) - python

- 先ほどのデータから10,000個サンプリングしてみよう

- Try to get 10,000 samples from the data

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df.sample(10000).to_csv(sys.stdout, index=None, header=None)" < 73.csv
```

- サンプリングして、本当に10,000サンプルあるか数えてみよう

- Are there 10,000 samples actually? Count them.

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df.sample(10000).to_csv(sys.stdout, index=None, header=None)" < 73.csv | wc -l
```

- サンプリングして、ソートしてみよう

- Sample and sort.

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df.sample(10000).to_csv(sys.stdout, index=None, header=None)" < 73.csv | sort
```

- サンプリングして、ソートして、重複したものを削除してみよう

- Sample, sort and delete duplicated lines.

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df.sample(10000).to_csv(sys.stdout, index=None, header=None)" < 73.csv | sort | uniq
```

## One-Linerの威力(3) - python

### Power of One-Liner (3) - python

- サンプルングして、ソートして、重複したものを削除した結果のサンプル数を確認してみよう

- Sample, sort, delete duplicated lines, then count lines.

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df.sample(10000).to_csv(sys.stdout, index=None, header=None)" < 73.csv
| sort | uniq | wc -l
```

- プロットしてみよう

- Try to plot them.

```
% python3 -c "import sys; import matplotlib.pyplot as plt;
import pandas as pd; df=pd.read_csv(sys.stdin, header=None);
df[0]=pd.to_datetime(df[0]); df = df.set_index(0); df2=df.sample(10000);
plt.scatter(df2.index, df2[2]); plt.show()" < 73.csv
```

何度かPlotしていたらこんなものが…  
I found this in my trial…



## One-Linerの威力(4) - ruby

### Power of One-Liner (4) - ruby

- 0.3を超えた値はきっと外れ値なので、それがどんなデータか確認してみよう
- The data bigger than 0.3 might be outlier. Check them.  

```
% ruby -e "STDIN.readlines.each do |l| col = l.chomp.split(',');  
print l if (col[2].to_f > 0.3) end" < 73.csv
```
- その数を数えてみよう
- Count the outliers.  

```
% ruby -e "STDIN.readlines.each do |l| col = l.chomp.split(',');  
print l if (col[2].to_f > 0.3) end" < 73.csv | wc -l
```

## One-Linerの威力(4) - python

### Power of One-Liner (4) - python

- 0.3を超えた値はきっと外れ値なので、それがどんなデータか確認してみよう

- The data bigger than 0.3 might be outlier. Check them.

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df[df[2] > 0.3].to_csv(sys.stdout, index=None, header=None);"
< 73.csv
```

- その数を数えてみよう

- Count the outliers.

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df[df[2] > 0.3].to_csv(sys.stdout, index=None, header=None);"
< 73.csv | wc -l
```



## One-Linerの威力(5) - ruby

### Power of One-Liner (5) - ruby

- おかしなデータは何時頃のもの？

- When the outliers are observed?

```
% ruby -e "STDIN.readlines.each do |l| col = l.chomp.split(','); print l  
if (col[2].to_f > 0.3) end" < 73.csv | head -1
```

```
% ruby -e "STDIN.readlines.each do |l| col = l.chomp.split(','); print l  
if (col[2].to_f > 0.3) end" < 73.csv | tail -1
```

- その間のデータはいくつある？（それは元データの何行目？）

- How many data are there between outliers? (Which line?)

```
% grep -n '2012-08-01 06:09:48 +0900, 3.000,36.000' 73.csv
```

```
% grep -n '2012-11-15 15:12:38 +0900,434.000,1.259' 73.csv
```

# One-Linerの威力(5) - python

## Power of One-Liner (5) - python

- おかしなデータは何時頃のもの？

- When the outliers are observed?

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df[df[2] > 0.3].to_csv(sys.stdout, index=None, header=None);"
< 73.csv | head -1
```

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df[df[2] > 0.3].to_csv(sys.stdout, index=None, header=None);"
< 73.csv | tail -1
```

- その間のデータはいくつある？（それは元データの何行目？）

- How many data are there between outliers? (Which line?)

```
% grep -n '2012-08-01 06:09:48 +0900, 3.000,36.000' 73.csv
```

```
% grep -n '2012-11-15 15:12:38 +0900,434.000,1.259' 73.csv
```

## One-Linerの威力(6) - ruby

### Power of One-Liner (6) - ruby

- 外れ値のみを外したほうが良さそうだな
- It might be good idea to remove outliers.

```
% ruby -e "STDIN.readlines.each do |l| col = l.chomp.split(','); print l  
if (col[2].to_f <= 0.3) end" < 73.csv > new.csv
```

- 新しいデータセットは何行?
- How many lines are in new data set?

```
% wc -l new.csv
```

- 削除されたデータの数は?
- How many lines are removed?

```
% echo `wc -l 73.csv` | awk '{print $1}' `-' `wc -l new.csv` | awk '{print  
$1}' | bc
```

## One-Linerの威力(6) - python

### Power of One-Liner (6) - python

- 外れ値のみを外したほうが良さそうだな
- It might be good idea to remove outliers.

```
% python3 -c "import sys; import pandas as pd;
df=pd.read_csv(sys.stdin, header=None);
df[df[2] <= 0.3].to_csv(sys.stdout, index=None, header=None);"
< 73.csv> new.csv
```

- 新しいデータセットは何行?
- How many lines are in new data set?

```
% wc -l new.csv
```

- 削除されたデータの数は?
- How many lines are removed?

```
% echo `wc -l 73.csv` | awk '{print $1}' `-' `wc -l new.csv` | awk '{print $1}' `| bc
```

# One-Linerの威力(7)

## Power of One-Liner (7)

- 実際にデータサイエンティストとして働いている人に聞いてみると、データ解析時間の70%～80%くらいはこの手の作業に費やしているそうです。
- Data scientists spend 70% to 80% of there time to do this type of operation.
- Command Line Interface (CLI)は好きですか？
- Do you like "Command Line Interface (CLI)" ?
- Command Line Interface (CLI)を学びたくなってきましたか？
- Now do you want to master "Command Line Interface (CLI)" ?

# 宿題

## Homework

- 来週の授業までに、  
wc/sed/uniq/sort/grep/awk/ruby/python3/wget/curl/gnuplot/Rが実行できる環境を整えてくること。（Windowsの場合はcygwinやPowerShell等を使うと良い）
- Install  
wc/sed/uniq/sort/grep/awk/ruby/python3/wget/curl/gnuplot/R on your laptop (if you are using Windows, please consider to use cygwin or PowerShell)
  - リモート環境でも構いませんが、教室で実行できるようにしておいてください。
  - Remote environment is acceptable. But you must be able to use them in the classroom.
- 提出の必要はありませんが、環境を構築しておかないと次回以降の授業に支障が出ます。
- You don't need to submit them. But if you don't do this, you will have a trouble on the class.