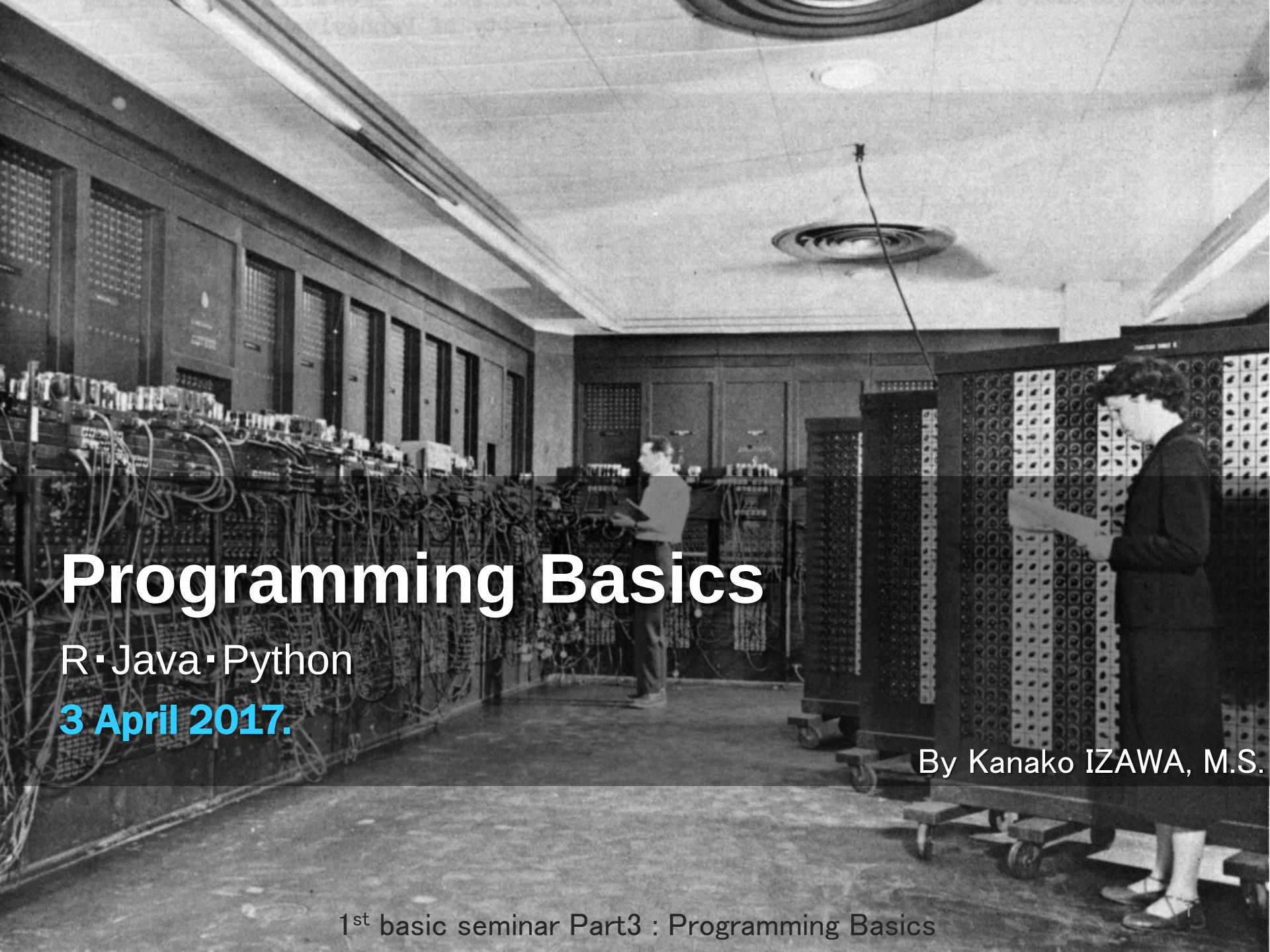


Programming Basics

R・Java・Python

3 April 2017.

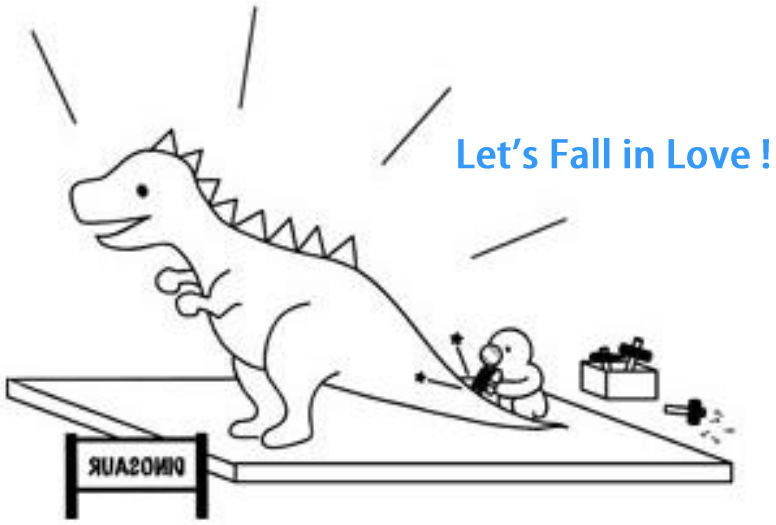
By Kanako IZAWA, M.S.

本日の内容

Part 1. 研究の進め方と都市をつくる仕事について

Part 2. プログラミングの基礎 ; R・Java・Python

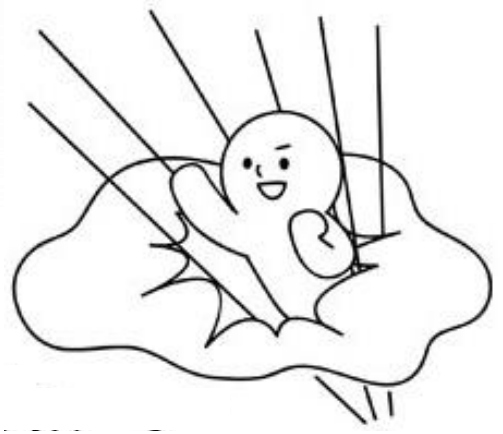
1. プログラミングとは？
2. どんなデータを使うの？
3. プログラミングの基礎
   - Rと推定
   - Javaとマップマッチング
   - Pythonと集計分析
4. まとめ



本日の目標

1. Rを使ったモデル推定の一連の流れを知る.
2. 位置情報データ(Probe person data)から
   経路を特定する方法の1つについて知り,
   Javaを使ったマップマッチングができるよう(な気持ち)になる.
3. Pythonを使ったデータの集計方法について知る.

これから研究に
どんなふうにプログラミングが使えそうか
知るきっかけになればと思います




はじめに

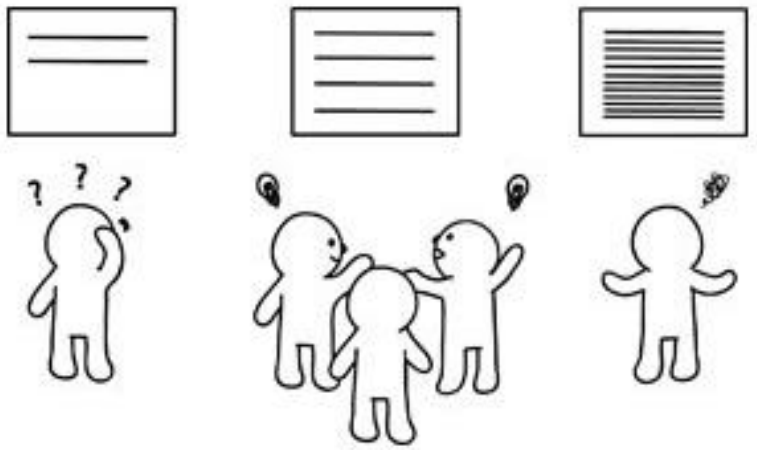
そもそもどうしてプログラミング？

プログラミングを使うことで、人間にできないような複雑な処理をコンピューターに任せられるから

あるいは、

コンピューターに任せないと出来ない複雑な処理をしたいからです。

Ex. 膨大なデータ処理
高度な計算
データの可視化
シミュレーション
正確な同じ処理の繰り返し



複雑な処理ってどんなもの？

羽藤研で分析するデータは主に行動データです。

歴史分析史料の分析をすることもありますが、

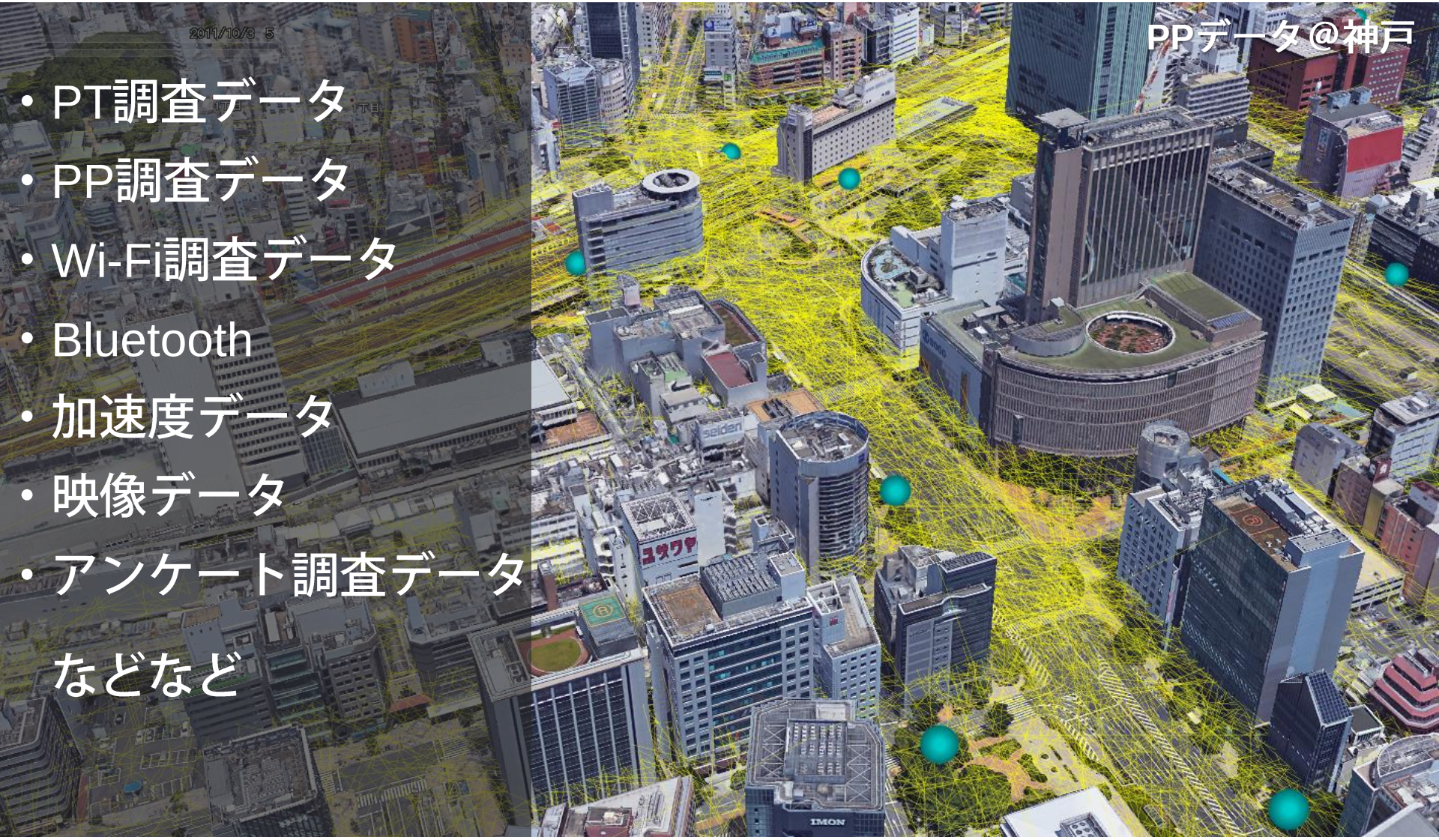
ここでは簡単に、**実際にどんなデータがあって、**

どんなことを知りたいのか

を少しだけ先に確認して、プログラミングへの

モチベーションを高めたいと思います。

分析したい行動データ



- PT調査データ
- PP調査データ
- Wi-Fi調査データ
- Bluetooth
- 加速度データ
- 映像データ
- アンケート調査データ

などなど

プログラミングとは？

- **プログラムを設計すること**です。
- プログラムとは，コンピューターに任せたい処理の順番や内容を**正確に指示する**ためのものです。
- コンピューターは機械語しか分からないし、人間には機械語が難しいので、**人間の言葉に近い言葉 (=プログラミング言語)** でプログラムを作成することが必要です。
- 人間と同じで言語ごとに得手不得手があるので、コンピューターに**任せたい処理の内容によって言語を選ぶ**必要があります。
- では、どういう処理を任せたくなるのか。

分析したい行動データ

プローブパーソン(PP)データ

：被験者(移動体)に検出器を取り付けることで、移動状態を観測・推定・解析するために得られるデータのこと。
高精度かつミクロレベルでのデータを継続的に取得することができ、個人属性情報も同時に得られることが特徴。

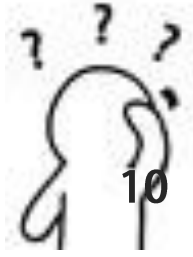
Ex. Location data: 数秒ごとに位置データを記録したもの.

測位ID	ユーザーID	記録日時	緯度	経度	トリップID	移動手段	有効性
18097423	kb009	2015-11-01 20:42:05	34.68976	135.1972	63286	徒歩	1
18100204	kb009	2015-11-01 20:42:09	34.6904	135.1953	63286	徒歩	1
18100205	kb009	2015-11-01 20:42:12	34.69008	135.1954	63286	徒歩	1
18100206	kb009	2015-11-01 20:42:15	34.6901	135.1954	63286	徒歩	1
18100207	kb009	2015-11-01 20:42:19	34.69039	135.1953	63286	徒歩	1
18100208	kb009	2015-11-01 20:42:24	34.69011	135.1949	63286	徒歩	1

分析したい行動データ

とりあえず全部プロットしてみます。

よくわからない…



行動データから知りたいこと

どういう人が多い？通勤のため？買い物？？
どのくらいの時間、街に滞在してるんだろう？

Excelでクロス集計(ピポットテーブル)、GISでプロットしてみる、、

どういうところを皆歩くのかな？
商店街や街路樹があるところって本当に歩いてる？？

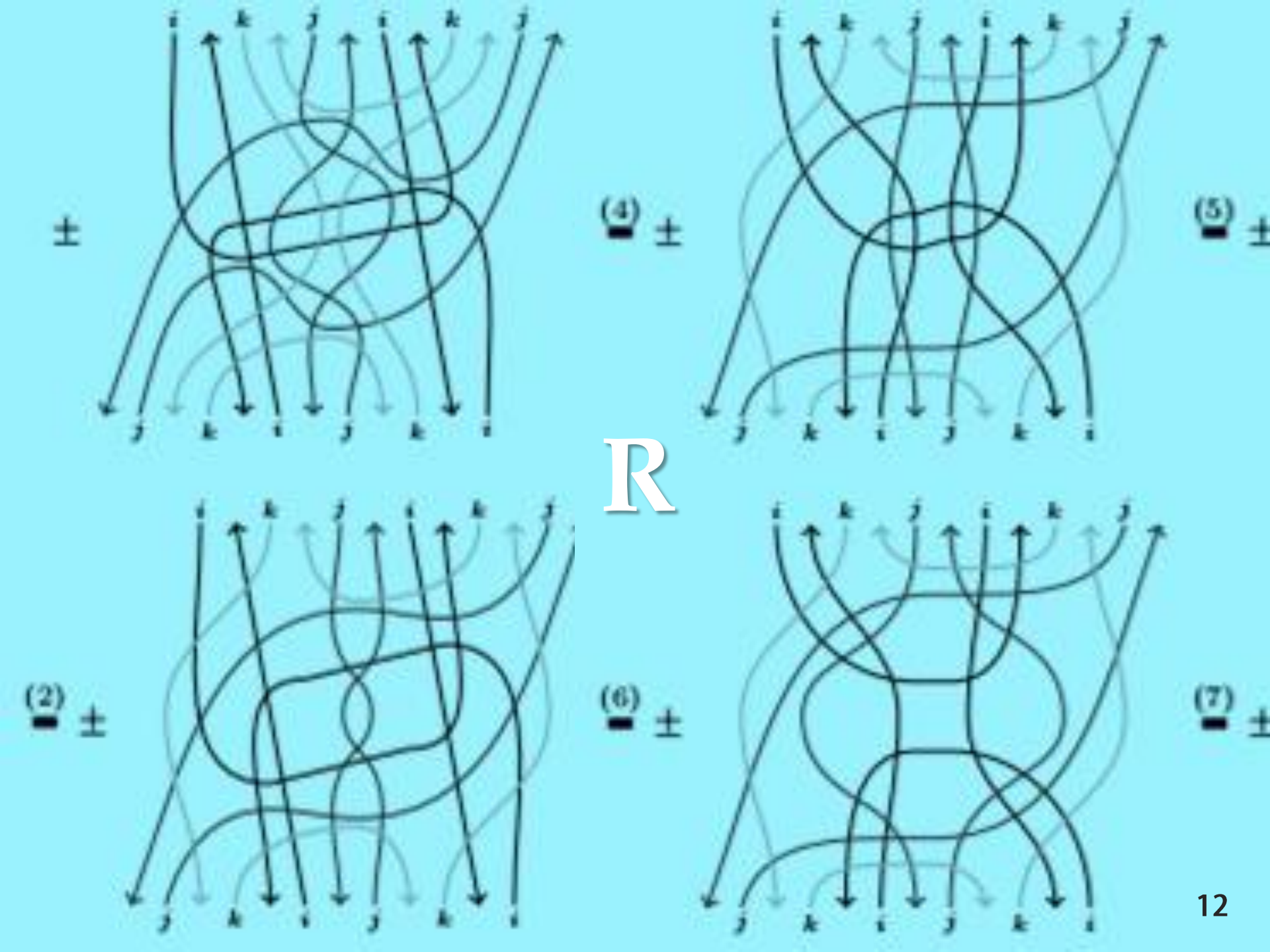
個人の行動/意思決定を分析したい。 → 例えばRで離散選択モデルの推定！

やっぱり雨の日は地下を歩いてる？
感覚的には通勤だったら最短経路をとる気がするけど…

位置情報(ドット)から実空間上のどの経路をとっていたのか特定したい。
→ Javaで処理！

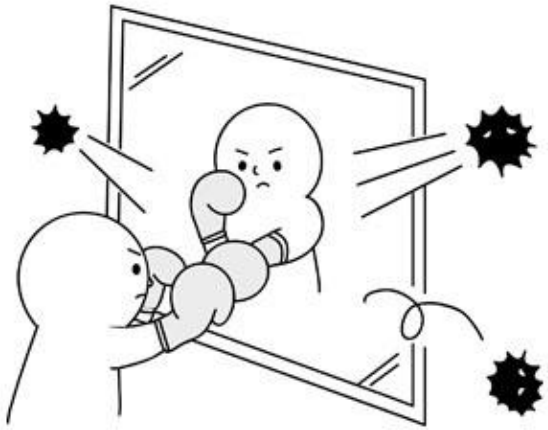
来街頻度ごとに歩きやすい道に違いがあるかも…！

Excelでは分析しにくいこともサクッと集計分析したい。→　Python


±
(4) ±
(5) ±
R
(2) ±
(6) ±
(7) ±

Rの導入

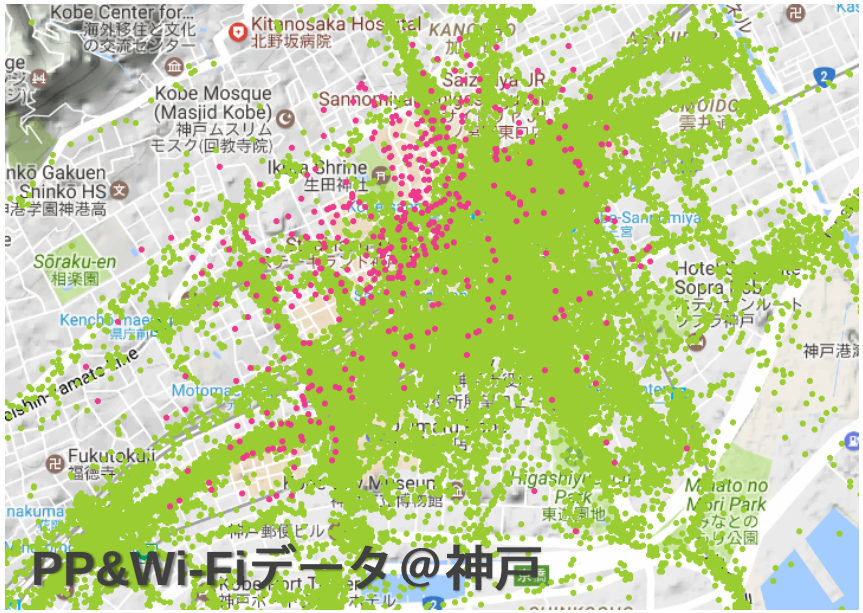
- Rは**統計解析に特化**したプログラミング言語です。
- なので、ちゃんと使いこなすには
  統計・プログラミング・分析したい対象(交通・都市)
  に関する基礎的な知識が必要です。
- が、とりあえずは、
  Rはハイスペック関数電卓
  だと思っていればいいと思います。



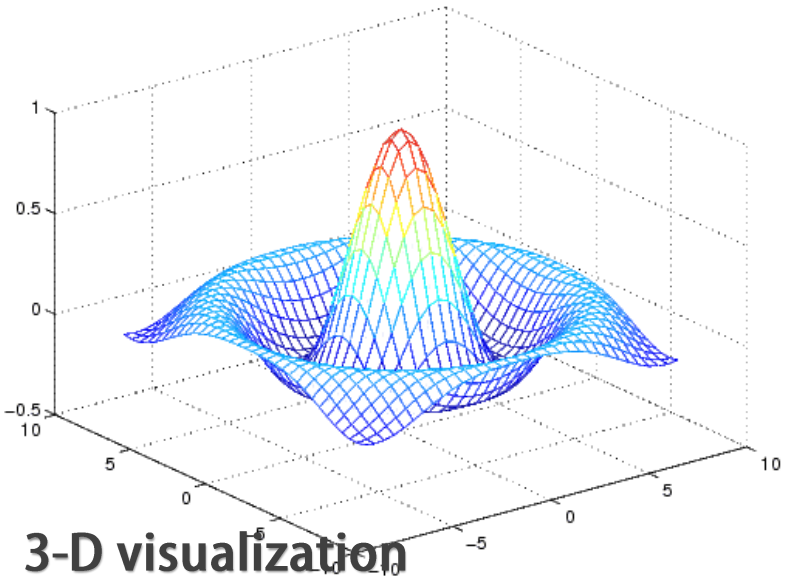
Rにできること

ハイスペックなので統計処理以外にも色々できます。

- ベクトル処理
- 計算も比較的速くできる。
- データの可視化。；バリエーションも豊富。などなど



PP&Wi-Fiデータ@神戸



3-D visualization

Rをはじめる準備①

- 必要なもの

１．R本体

２．Rstudio

; Rを使いやすくするための統合開発環境(IDE)です。
少々日本語入力が苦手ですが、Tinn-Rよりおすすめです。

※裏でRを使っているので公式のR本体のインストールが必要です。

Rをはじめる準備②

- Rのインストール

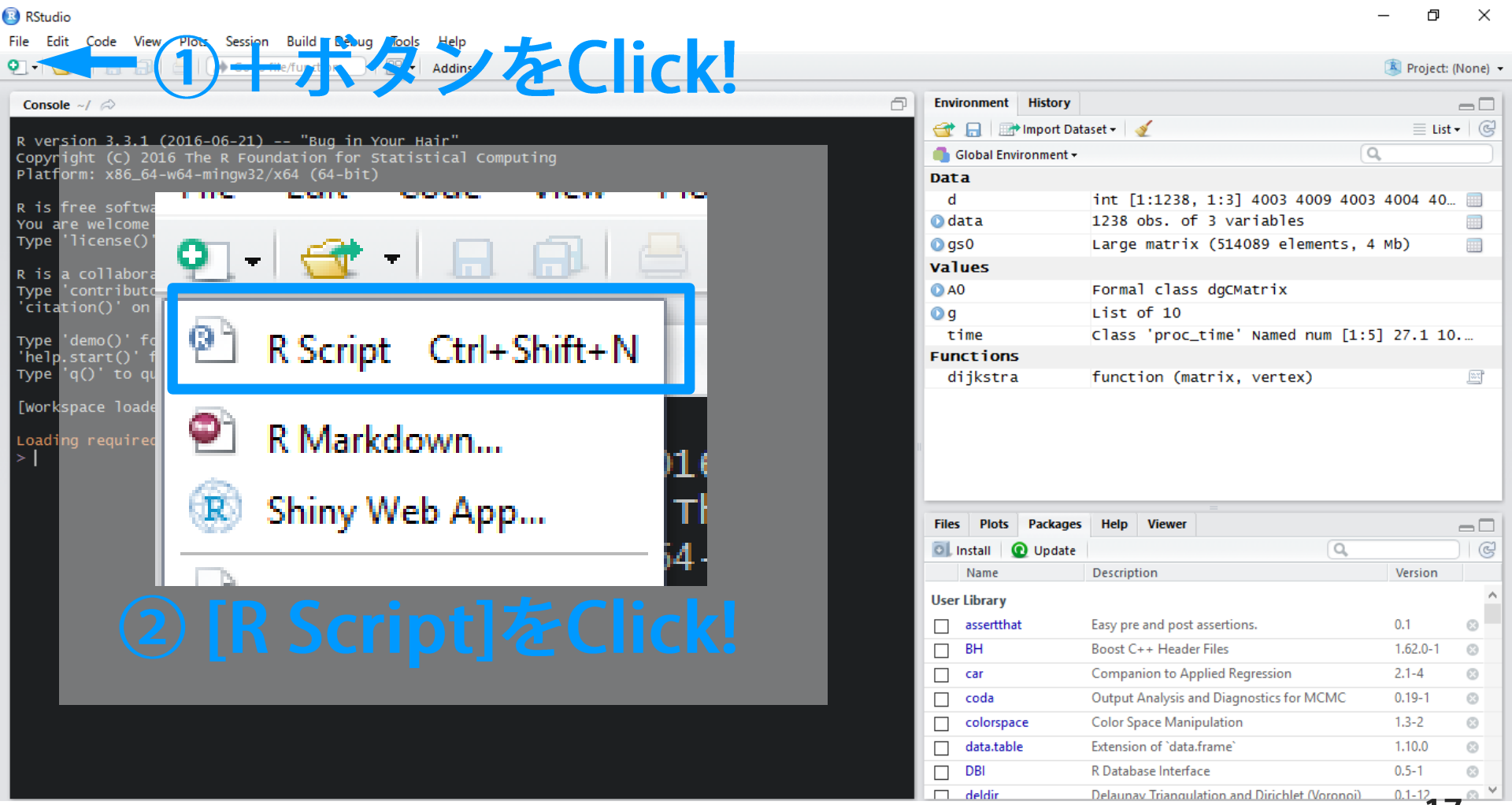
任せました。
下記Webサイト等を参考にしてください。

> RjpWiki ; Rに関する情報を日本語でまとめてあるWiki。
http://www.okadajp.org/RWiki/?RjpWiki

> R-tips ; Rの基本操作についてまとめてある。
http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html

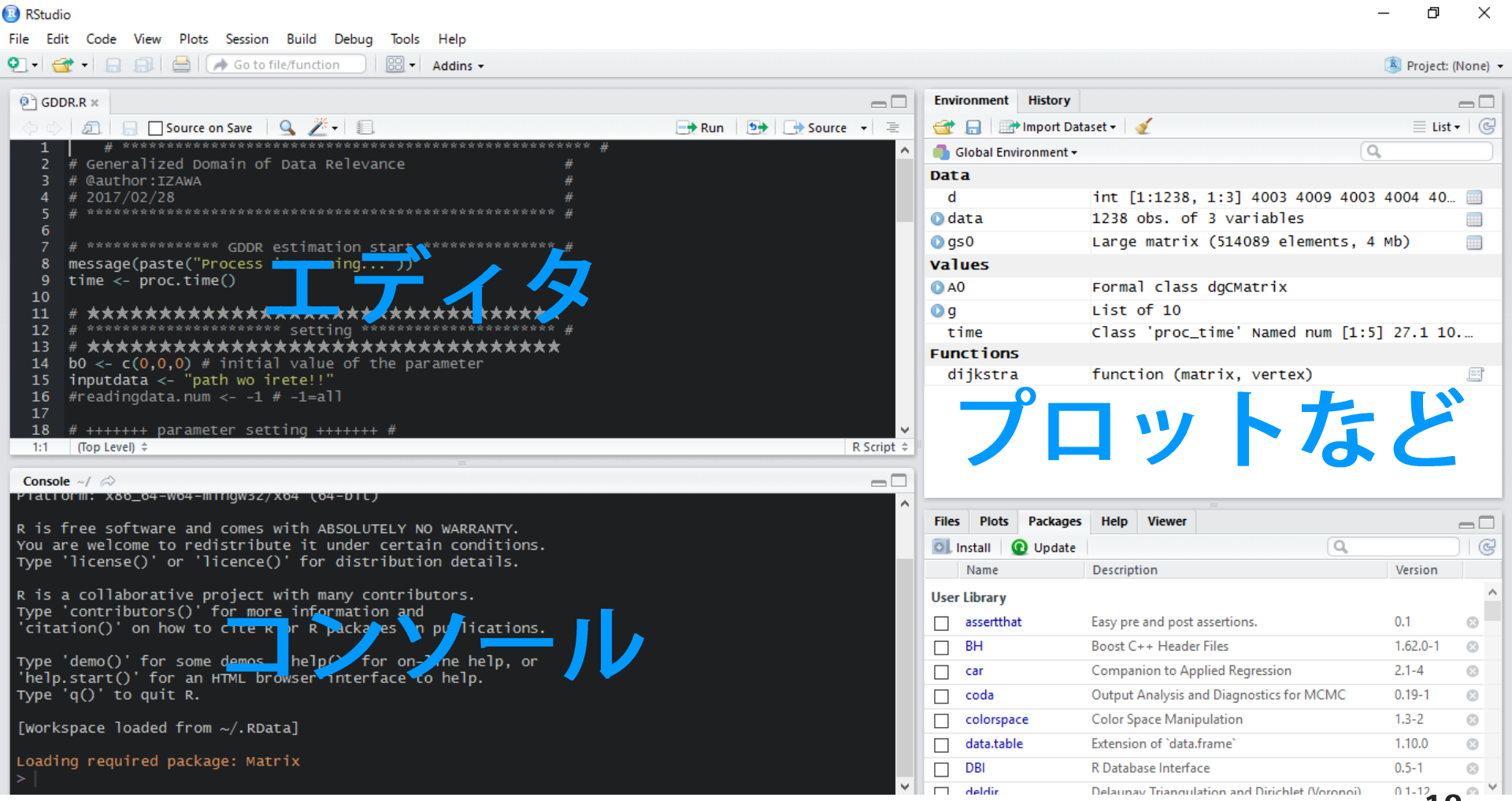
Rを使ってみる

Rstudio起動…!!



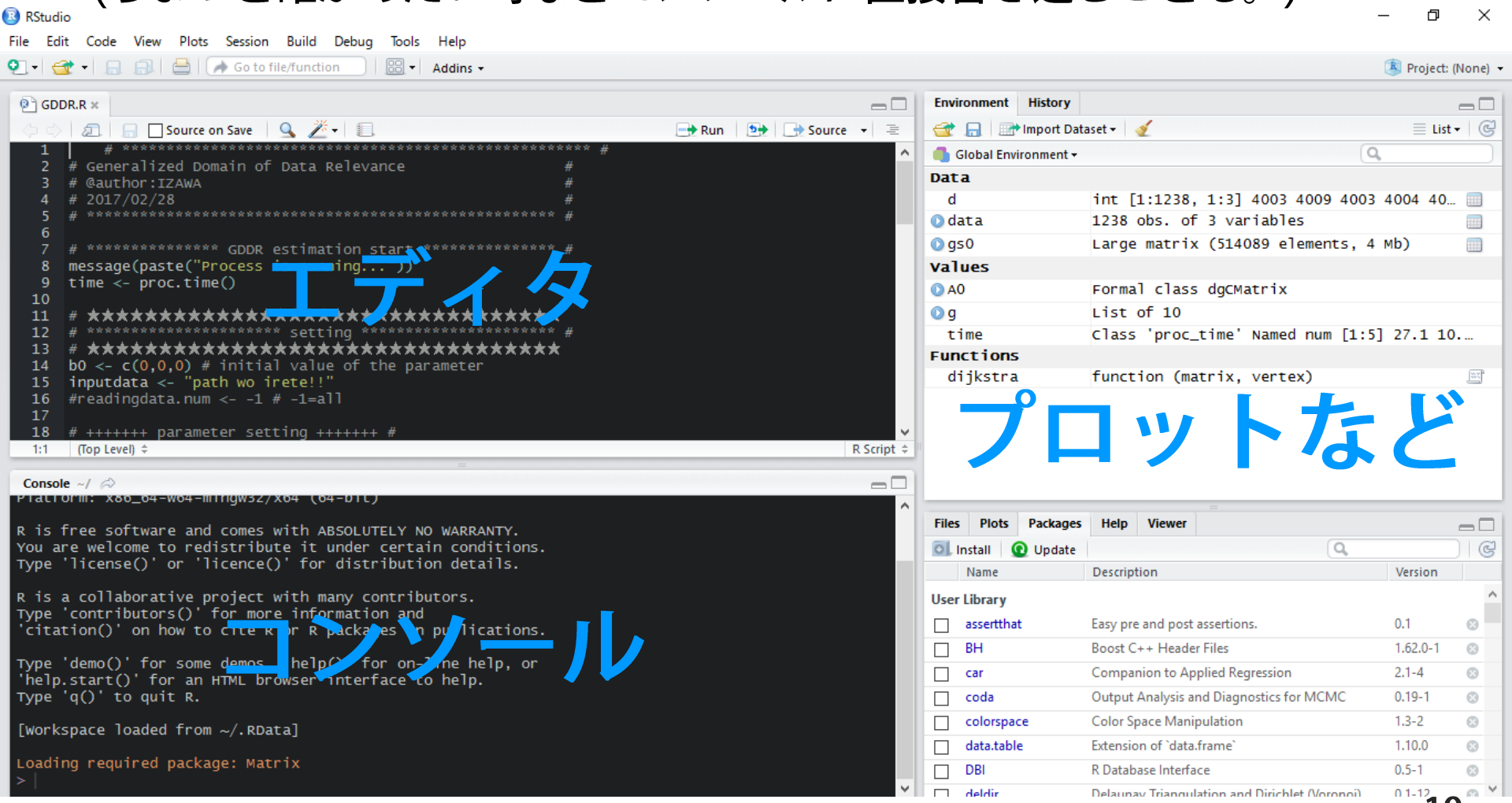
Rを使ってみる

R Script が新規に作成されました。これが基本画面。



基本的な使い方

- スクリプトをエディタに書き込んで実行するだけ！簡単！
  (ちょっと確かめたい時などコンソールに直接書き込むことも。)



基本的な使い方

- **スクリプトをエディタに書き込んで実行するだけ！簡単！**
  (ちょっと確かめたい時などコンソールに直接書き込むことも。)
- スクリプトというのはRへの命令です。
- コンソールに直接入力したときには **[Enter]** を、
  エディタに入力したときには、実行したい部分を選択して
  [ctrl] + [Enter] を押せば実行できます。
  ※Macでは **[command] + [Enter]** です。

全体の注意事項

- 全角と半角，大文字と小文字は区別されます。

  日本語でコメント文を入れていたりすると、{ } を全角
  入力してエラーみたいなことが起こり得るので注意です。

- 当たり前ですが、

  誤字があっても，変なところに空欄があってもだめです。

  正確にコンピューターに命令しましょう。

- こまめにスクリプトは保存しておくと安心です。

- ということで、Rを使いたくなってきました。

基本的なこと

・とっても便利なサイトがあります！！

ここでだいたいは分かると思います。

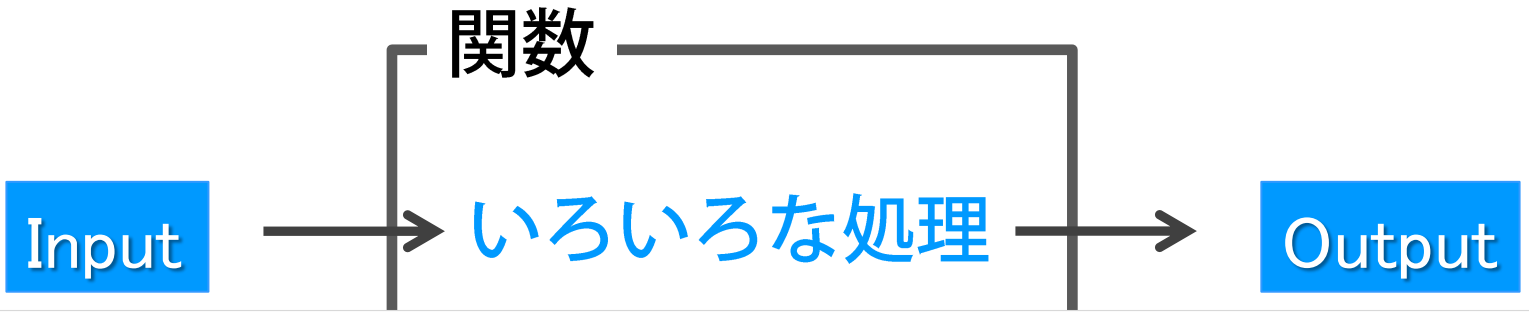
任せました！(信頼)

> [Rtips]
http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html

理解してほしい概念

・演算子
・変数とオブジェクト
・関数

関数



- 関数が使えればもうだいたい大丈夫です。
- なので推定します！！

```
izawa <- function(x, y) {
  diff <- x - y
  return(diff)
}
```

↑関数の名前

↑引数を指定

↖処理の内容

最適化関数（補足）

最適化関数optim()の使い方：

```
optim(par, fn, gr = NULL, method = "BFGS",
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE)
```

par	初期値（必須）
fn	最適化する目的関数（必須）
gr	一階偏微分関数の指定（NULLでデフォルト関数使用
method	最適化手法の指定（5種類から選ぶ）

- Nelder-Mead（デフォルト）
- BFGS
- CG
- L-BFGS-B
- SANN

最適化関数（補足）

最適化関数optim()の使い方：

```
optim(par, fn, gr = NULL, method = "BFGS",
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE)
```

lower	L-BFGS-B法での変数の下限（デフォルトは-Inf）
upper	L-BFGS-B法での変数の上限（デフォルトはInf）
control	制御パラメータ fnscale:関数に与える比例定数 optim関数は最小化を行うので、最大化のときは control=list(fnscale=-1)と指定する
hessian	最適解のヘッセ行列を返すかどうか パラメータ推定では、t値計算にヘッセ行列が必要 なので、hessian＝TRUE

最適化関数を使ってみる（補足）

$f(x, y) = (x - 2)^2 + (y - 3)^2$ の最小値は？

```
f <- function(par){
  (par[1] - 2)^2 + (par[2] - 3)^2
}
optim( c(0,0) , f , method = "BFGS" )
```

- 関数fで関数を定義。
  このとき変数は2変数なのでpar(x,y)の形でベクトル化して
  何番目の要素かを表現します。

- optim関数で初期値(x,y)=(0,0)を与えてfを最小化。
（Rではベクトルはc(要素,要素,…)で定義されます。）

最適化関数を使ってみる（補足）

$f(x, y) = (x - 2)^2 + (y - 3)^2$ の最小値は？

```
> f <- function(par){
+   (par[1] - 2)^2 + (par[2] - 3)^2
+ }
>
> optim(c(0,0), f ,method="BFGS")
$par
[1] 2 3

$value
[1] 3.806254e-29

$counts
function gradient
       9        3

$convergence
[1] 0

$message
NULL
```

結果の解釈

par
：最適解(x,y)=(2,3)

value
：最適値 $3.81 \times 10^{-29} \fallingdotseq 0$

counts
　function
　：BFGS法の繰り返し計算9回
　gradient
　：1階偏微分の計算3回

convergence
：0ならば収束している

message
：その他のメッセージなし

パラメータ推定

なんでパラメータ推定するのか…

そう、個人の行動の原理を知りたかったのでした。

具体的に評価したい内容を明確にしておきましょう。

Ex. 経路選択モデル

歩く距離が短いほど、移動時間が短いほど、
商店街があるところほど、歩道幅員が大きいところほど、
経路が選ばれやすいのではないか？

※こういう仮説をたてるためには基礎分析が重要です。

行動モデルの基礎

詳しくは来週

・たとえば **離散選択モデル**

個人の意思決定や行動は
何らかの「選択」であると捉えることができます。

ex. 新宿までいくとき、
「本郷三丁目」あるいは「東大前」駅を選択する。

・**効用を定式化**

合理的な個人は「効用」が最大になる選択肢を選択する、
と仮定することで、それぞれの選択肢の大小によって
選択が行われるという定式を行うことができます。

効用U_{nk}：個人nが選択肢kに感じている魅力の程度

行動モデルの基礎

詳しくは来週

個人*n*の効用関数（多項式）

説明変数

$$U_{n1} = \beta_1 x_{price,route1,n} + \beta_2 x_{time,\ route1,n} + \beta_3 x_{female,n} + \beta_4 + \varepsilon_{route1}$$

$$U_{n2} = \beta_1 x_{price,\ route2,n} + \beta_2 x_{time,\ route2,n} + \varepsilon_{route2}$$

費用　所要時間　女性ダミー　定数項　誤差項

確定項V

- 選択肢*i*を選ぶ選択確率（ランダム効用モデル）

$$P_{route1} = \frac{\exp(\mu V_{route1})}{\exp(\mu V_{route1}) + \exp(\mu V_{route2})} = \frac{1}{1+\exp(-\mu\,(V_{route1} - V_{route2}))}$$

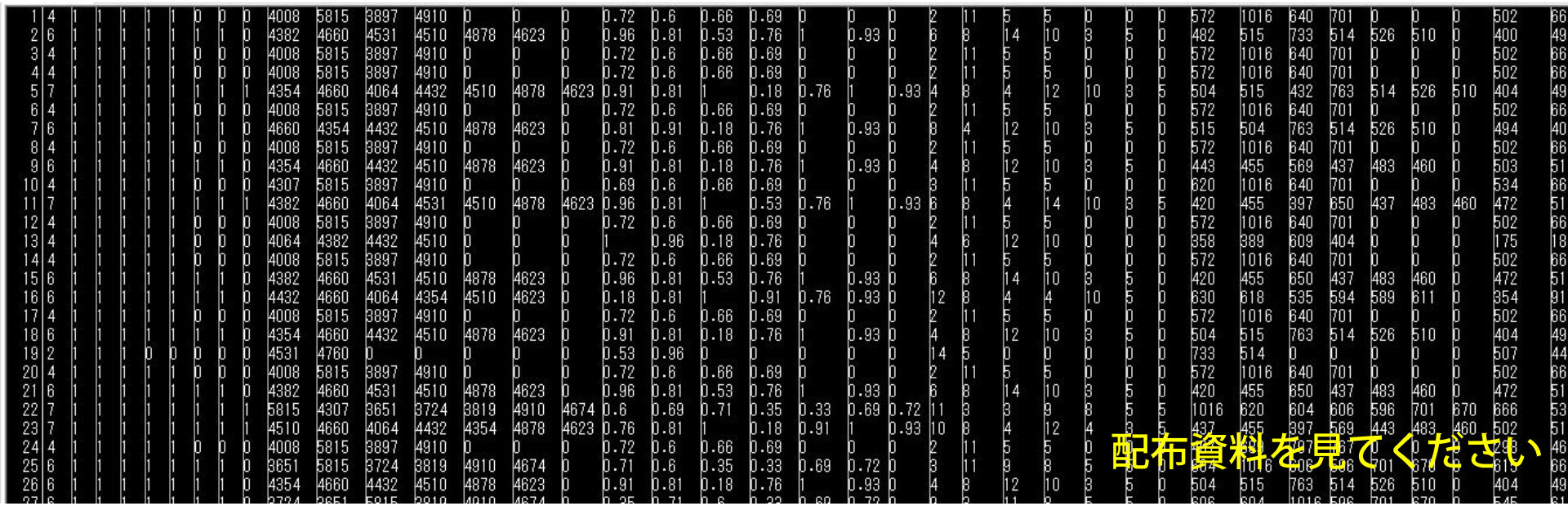
μ：誤差項のばらつきを表すスケールパラメータ
ロジットモデルでは誤差項分布にガンベル分布を仮定

- 選択確率は、選択肢間の確定効用の差で表される

経路選択モデル(MNL)の推定

補足資料

・今回使用するデータはこちら。[route.csv]



・説明変数には、

経路長、幹線比率、右左折回数、旅行時間、旅行時間標準偏差

を設定しています。

経路選択モデル(MNL)の推定

補足資料

- 来週松岡くんがわかりやすく説明してくれるのでさらっと流れだけ確認しておきます。
- 各選択肢が実際に選択されたとき、それぞれの説明変数がどれくらい選択に影響を与えているのか（パラメータの値）が知りたいです。
- 選択確率が最大であるときに、実際の選択行動が観測されるという事象が生じることが尤もらしいですね。
- なので、尤度関数の最大化問題を解けばいいわけです。

経路選択モデル(MNL)の推定

- **尤度関数**

解きやすくするために対数をとります。

$$\ln L(\theta) = \sum_{n=1}^{N} \sum_{i \in I_n} \delta_{in} \ln(P_{in})$$

$$P_{in} = \frac{\exp(V_{in})}{\sum_{j \in I_n} \exp(V_{jn})}$$

δ_{in}:個人nが選択肢iを実際に選んだ場合1,
そうでなければ0を取る変数
I_n:個人nの選択肢集合
V_{in}:個人nにとっての選択肢iの効用の確定項

尤度関数の定義

1. パラメータの宣言
2. 効用確定項の計算
3. 選択確率の計算
4. 選択結果の判別
5. 対数尤度の計算

経路選択モデル(MNL)の推定

補足資料

・推定コード

ファイル名の上で右クリックでパスを
コピーできます。 ↓

```
###データファイルの読み込み
Data<-read.csv("C:/workspace_r/MNL/route.csv",header=T)

hh<-nrow(Data)##データ数:Dataの行数を数える

#パラメータ数の分だけ初期値を代入した列ベクトル「b0」を作成
b0<-numeric(5)
## Logit model の対数尤度関数の定義
fr <- function(x) {
    ##パラメータ:
    ## 距離
    b1 <- x[1]
    ## 幹線比率
    b2 <- x[2]
    ## 右左折数
    b3 <- x[3]
    ## 旅行時間
    b4 <- x[4]
    ## 旅行時間偏差
    b5 <- x[5]
```

←1.パラメータの宣言

経路選択モデル(MNL)の推定

補足資料

・推定コード

```
    b4 <- x[4]
    ## 旅行時間偏差
    b5 <- x[5]

  ##効用の計算:説明変数にしたい列を入れる
                                          ##経路長          ##幹線比率
      Path <- Data[,3:9]*exp(b1*Data[,10:16]/1000 + b2*Data[,17:23]
      Ut   <- apply(Path,1,sum)
      Prob <- Path[,1]/Ut
      Prob <- (Prob != 0)*Prob + (Prob == 0)
      LL   <- sum(log(Prob))
}

## 対数尤度関数frの最大化
res<-optim(b0,fr, method = "Nelder-Mead", hessian = TRUE, control=l

## estimated parameter
b<-res$par
hhh<-res$hessian
```

↓2. 効用確定項の計算

←3. 選択確率の計算

←4. 選択結果の判別

←5. 対数尤度計算

【重要】最適化関数optim
※各自確認をお願いします。

経路選択モデル(MNL)の推定

補足資料

・推定コード

```
## t値の計算
tval<-b/sqrt(-diag(solve(hhh)))

## 適合度の計算
##初期尤度
Path <- Data[,3:9]*exp(0)
Ut <- apply(Path,1,sum)
Prob <- Path[,1]/Ut
LO<-sum(log(Prob))
print(LO)
##最終尤度
Path <- Data[,3:9]*exp(b[1]*Data[,10:16]/1000 + b[2]*Data[,17:23] + b[3]*Data[,24:3
Ut <- apply(Path,1,sum)
Prob <- Path[,1]/Ut
Prob <- (Prob != 0)*Prob + (Prob == 0)
LL<-sum(log(Prob))
print(LL)

##結果の出力
##ρ^2値
print((LO-LL)/LO)
##修正済ρ^2値
print((LO-(LL-length(b)))/LO)
print(res)
print(tval)
```

optim関数の返り値を使って
t値などの統計結果を表示させています。

※表示に関しては、表示すること自体を
関数にしてもいいかもしれません。
補足資料にありますので見てみて下さい。

推定結果と考察

- 推定結果の書き方（例）

説明変数	パラメータ	t値
所要時間[分/10]	-0.762	-7.41 **
費用[円/100]	-0.044	-0.98
乗車外時間[分/10]	-1.049	-8.07 **
女性ダミー	1.811	5.53 **
定数項（鉄道）	3.168	7.46 **
定数項（自動車）	0.868	2.08 *
定数項（自転車）	0.046	0.13
定数項（徒歩）	2.037	4.81 **
サンプル数	400	
初期尤度	-564.18	
最終尤度	-344.08	
尤度比	0.390	
修正済み尤度比	0.376	

*5%有意 **1%有意

単位、有効数字、t値を掲載などに注意してください。

上手く回らないときの確認事項

- データセットは正しくできているか。
  - 空欄はない？
  - 誤字はない？(数字と文字が混在してない？)
- パラメータの設定はあっているか(個数・宣言)
- ファイルの指定場所は正しいか
- 効用関数の式は正しいか
  - 括弧の閉じ忘れ
  - 説明変数の数
  - 列名の指定

パッケージの紹介

- **data.table** パッケージ

  ；大規模データを高速に処理できるように、
  R組み込みの **data.frame** クラスを拡張したもの。

  fread() 関数で、大規模データを高速に読み込むことができます。**read.table()** などより若干自由度が落ちますが、少しプロットして見たいときなど高速処理ができるので便利です。

- **dplyr** パッケージ
  ；データフレームの操作に特化したパッケージで、
  処理速度も速くて便利です。

発表課題①

補足資料

- 経路選択モデルの推定

  サンプルデータをWikiからダウンロードして
  実際にコードを書いて実行してみてください。

  いきなり経路選択モデルは煩雑なんじゃないか説
  を感じた人は交通手段選択モデルを補足資料に入
  れておいたので、見てみてください。

- 余力があれば、何らかの選択行動を仮想データで
  構わないので推定してみると面白いかもしれません。

おまけ課題例

- 自分で書いて推定してみる(復習)
  - ツイート数の変動とか知りたい.
  - 私の家を出るか否かの二項選択肢モデル(→)

説明変数	推定値	t値
家に出たくない潜在的要素	0.026	2.638 **
雨	0.005	0.381
外が明るい	−0.027	−4.145 **
疲労	0.104	0.509
眠い	0.119	0.287
身だしなみ整えるのが面倒	0.103	0.281
顔面コンディション不良	−0.254	−2.596 **
お腹いっぱい	−0.026	−0.216
尤度比	0.268	
修正済尤度比	0.168	

これで今日のRの説明は以上です。

目標 1. Rを使ったモデル推定の一連の流れを知る Done!!


Java

Javaの導入

- 1991年頃，米国のSun Microsystems社が家電製品用ソフトウェア開発のために作った言語をもとに改良が重ねられ，1995年5月のSunWorldで発表されたのがJavaです。

  ちなみに…Javaという名前はコーヒーに由来しているみたいです。

- Google三大言語のひとつで、コンピューターのOSに依存しないため**汎用性の高い**オブジェクト指向プログラミング言語です。

- コンパイラ言語（コーディング後に実行するためのファイルを作ってから使うもの）なので、Ruby、Pythonのようなスクリプト言語よりも**高速に処理**を行うことができます。

Javaをはじめる準備①

補足資料

- JDKのダウンロード・インストール

Javaプログラムの開発には，
Java開発キット＝JDK(Java Development Kit)が必要です。

Windowsをご使用の皆さんは，
ダウンロード・インストールしないといけません。

※Macの人は搭載済みなはずなのでこの作業は不要です。

任せました。配布した参考資料を参照してください。

Javaをはじめる準備②

補足資料

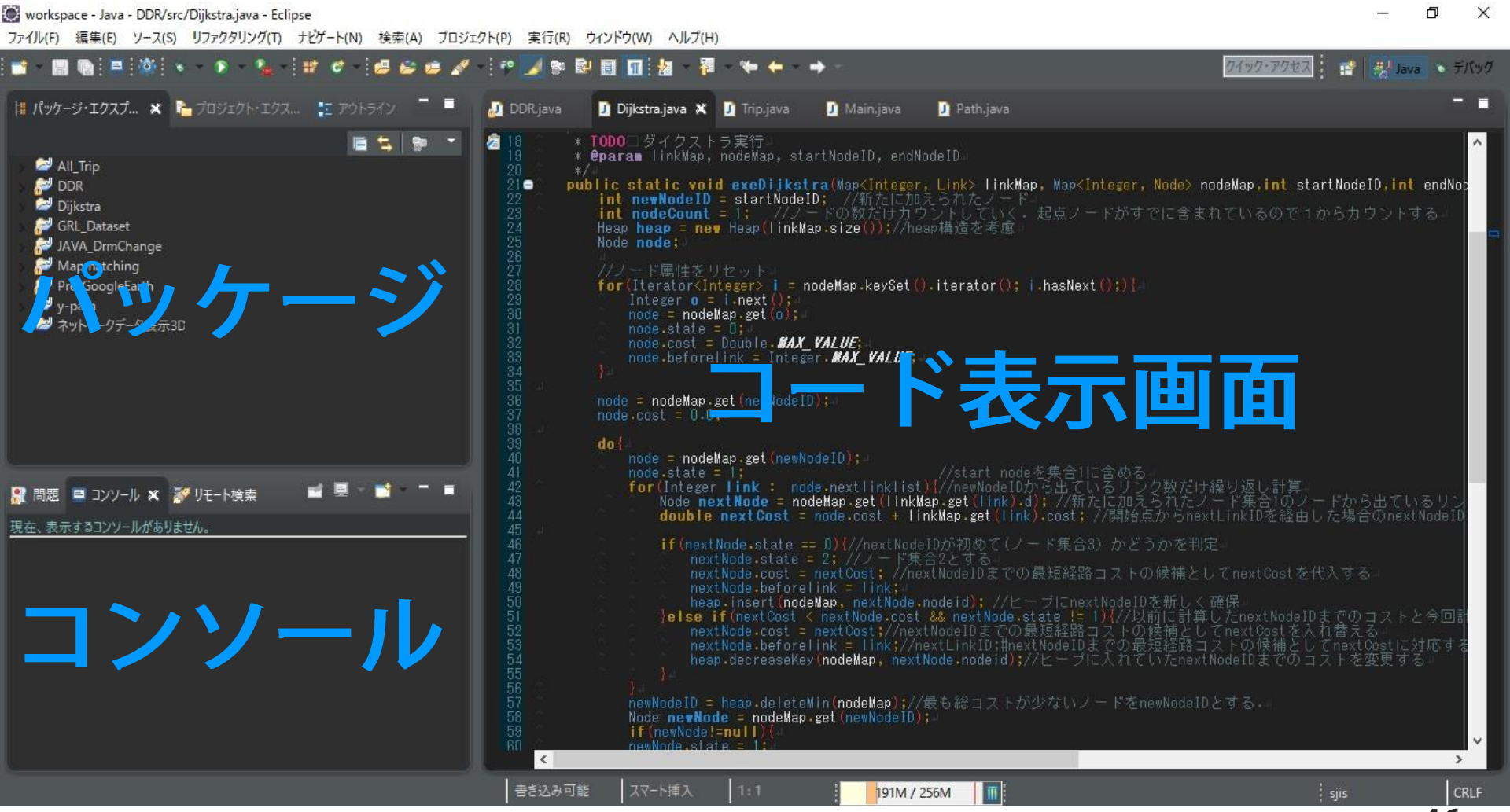
- Eclipse
  ；主にJavaプログラミングに用いられるIDEです。

- Eclipseのインストール

  任せました。
  が、、普通のアプリのように
  「次へ」「はい」とかを押したら完了！！
  みたいな感じではないので気を付けてくださいね！
  配布資料を参考にしてください。

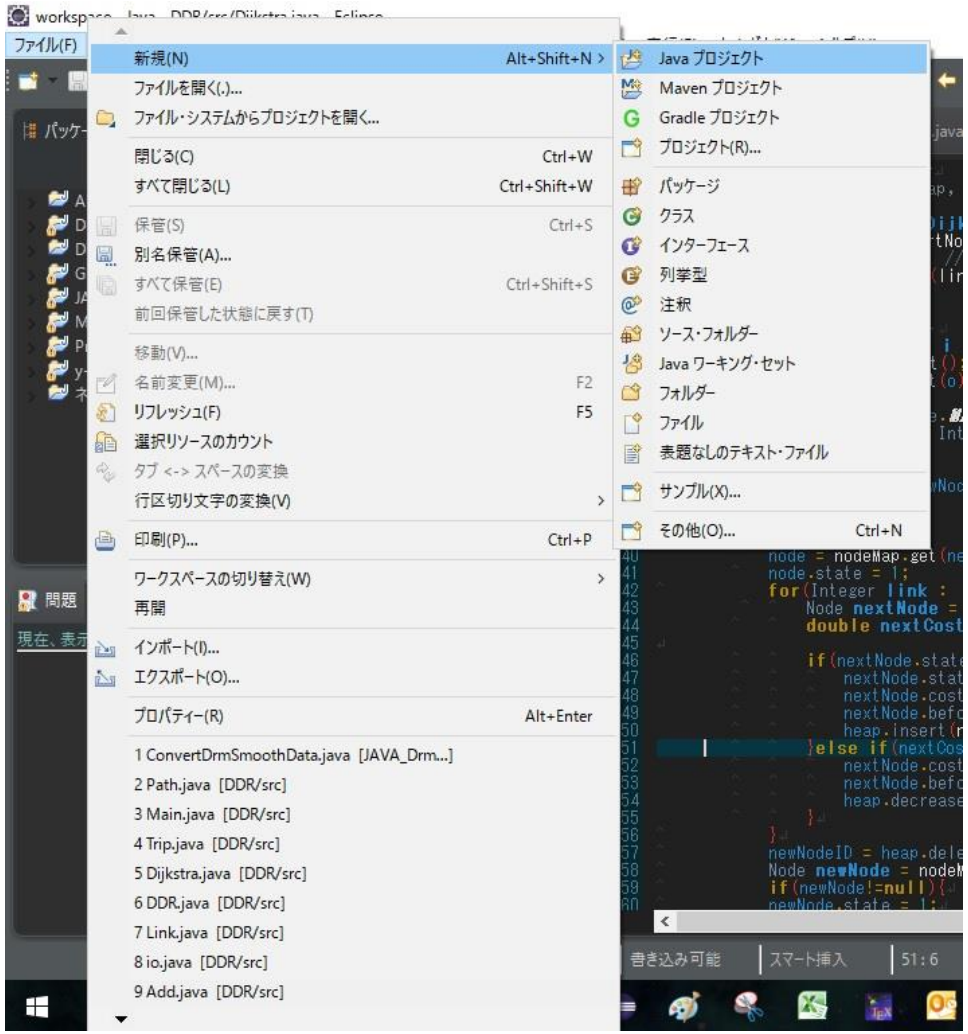
Javaを使ってみる準備①

これがEclipseの基本画面。



Javaを使ってみる準備②

プロジェクトを作成します。

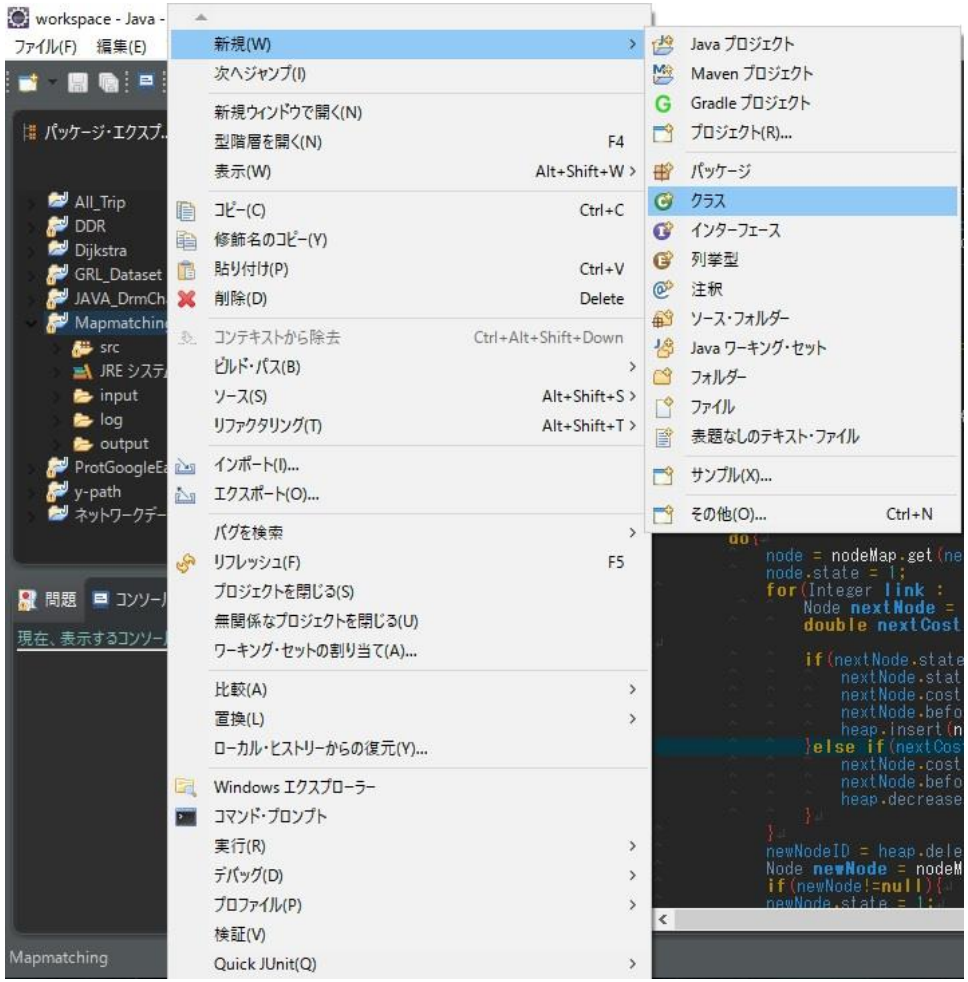


①[ファイル]
↓
②[新規]
↓
③[Javaプロジェクト]
↓
④プロジェクト名を入力
↓
⑤[完了]

ワークスペースにプロジェクトのフォルダが出来ていることを確認すると安心です。

Javaを使ってみる準備③

クラスを作成します。



①プロジェクトの上で
　右クリック
　↓
②[新規]
　↓
③[クラス]
　↓
④クラス名を入力
　※1文字目は大文字で。
　↓
⑤[完了]

public static void main(String[] args)
にチェック

Javaを使ってみるまえに

Javaを使ってコンピューターに命令をする準備ができました！

が、、、

実際にプログラムを書くまえに確認しておいた方がいいことがたくさんあります。

が、、、

たくさんあるので各自確認をお願いします。

確認しておきたい基本的なこと

1. main()メソッドの先頭からプログラムの実行がはじまります。
2. Javaでは1つの小さな処理の単位を**文**と呼んでいて、
   最後に **; (セミコロン)** をつけることが決まりです。
   原則として**先頭から順番に1文ずつ処理**されます。
3. 画面に文字列を表示する方法　→ **System. out. println()** メソッド
   今使っているPCの画面装置に出力する
4. 変数のしくみ（識別子、代入、型：String, int, long, double, …）
5. 式と演算子（演算子の優先順位など）
6. 条件判断文（if ~ else、switch）、関係/論理演算子
7. 繰り返し文（for、while、break、continue）
8. 配列、メソッド、**クラス**について
9. 例外処理（エラーに強いプログラムを作ることができます。）
10. その他に、デバッグ、パッケージ、スレッド、**currentTimeMills()** 等々

確認しておきたい基本的なこと：クラス①

たとえば、

・いま、「ロボット」という物に着目して、それを作るプログラムを作ろうとしているとします。

・このとき「ロボット」に関する情報や性質として、「名前」、「燃料の残量」といったものを挙げることができます。他にも、「ロボットの名前を決める」「ロボットの名前や燃料の残量を表示する」というような機能も考えることができます。

```
// ロボットクラス

class ロボット
{
        名前；
        燃料；
        名前を決める；
        名前や燃料を表示する；
        …
}
```

情報や性質をまとめる

機能をまとめる

確認しておきたい基本的なこと：クラス②

クラスとは、

こういった現実世界の物（オブジェクト）や概念を、
プログラムのなかの変数（オブジェクト）に投影するために
使う概念で、その**状態や性質と機能を定義**します。

メソッドはそのうちの機能を定義するもので、
クラスの中に記述します。
この時、メソッドが記述される順番は関係ありません。

メソッドはRでいうと関数みたいなもので、
よく使うデータの操作などはメソッドにしてしまうと楽です。

練習問題

補足資料

1. コンソールに “Behavior in Networks!” と表示する。
2. 1 ~ 100 までの数字を表示する。 （for文）
3. “1, 2, 3, …, 99, 100”と1行で表示する。 （for文）
4. 1 ~ 612 のうち素数のみを表示する。 （for文）
   (解答例は補足資料を参照)
5. 以下の式を満たす最小の $x,\ y$ を求める。

$$\left(\sum_{i=1}^{x} i\right) > 100 \qquad , \qquad (y!) > 1,000,000$$

基本的なアルゴリズム

アルゴリズム
；問題を解くための処理手順、設計図です。

今日は基本的なアルゴリズムを少し紹介した後、
いきなりですが実際に、
PPデータをネットワーク上の経路情報として
扱えるようにするマップマッチングを
してみたいと思います。

基本的なアルゴリズム①

補足資料

1からnまでの整数の和を求めてみます。

```java
// 1,2,…,n の和を求めます(for文を使った場合)

import java.util.Scanner;

public class Repeat {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ

        Scanner stdIn = new Scanner(System.in);

        System.out.println("1からnまでの和を求めます。");
        System.out.print("nの値:");

        int n = stdIn.nextInt();

        int sum = 0;  //和

        for (int i = 1; i <= n; i++)
            sum += i;                    //sumにiを加えます

        System.out.println("1から" + n +  "までの和は" + sum + "です。");
    }

}
```

【実行結果】
1からnまでの和を求めます。
nの値:612
1から612までの和は187578です。

基本的なデータ構造

配列(array)

；同じ型のデータが集まることによって
実現されるデータ構造(data structure)です。

変数に似ていますが、変数が一つの値を保管するのに対して、配列では複数の値を保管することができます。配列を利用することで、似たような名前の変数を大量に作らなくても済み、配列内の複数の記憶場所に格納された値を順番に処理するといったことが可能になります。

基本的なデータ構造

配列(array)

；必ず [0] から始まるインデックスをもつ。

インデックス（順序）

[0]	Ueda
[1]	Hirose
[2]	Yamaguchi
[3]	Matsuoka

コード例：

```
//4要素の配列を準備します。

String[] name = new String[4];
name[0] = "Ueda";
name[1] = "Hirose";
name[2] = "Yamaguchi";
name[3] = "Matsuoka";
```

取り出し方

Name[1]　→　“Hirose”

基本的なデータ構造

Arryalist（リスト）とHashmap（マップ）

- データを管理するデータ構造
- データの要素数を事前に定義しなくてよい。

Arraylist：順番と要素で管理

0	日本
1	カナダ
2	アメリカ
3	ドイツ
4	中国
:	:

Hashmap：キーと要素で管理

Japan	日本
Canada	カナダ
USA	アメリカ
Germany	ドイツ
China	中国
:	:

取り出し方
(Arryalist名).get(2)　→　“アメリカ”

取り出し方
(Hashmap名).get(“Japan”)　→　“日本”

基本的なアルゴリズム②

補足資料

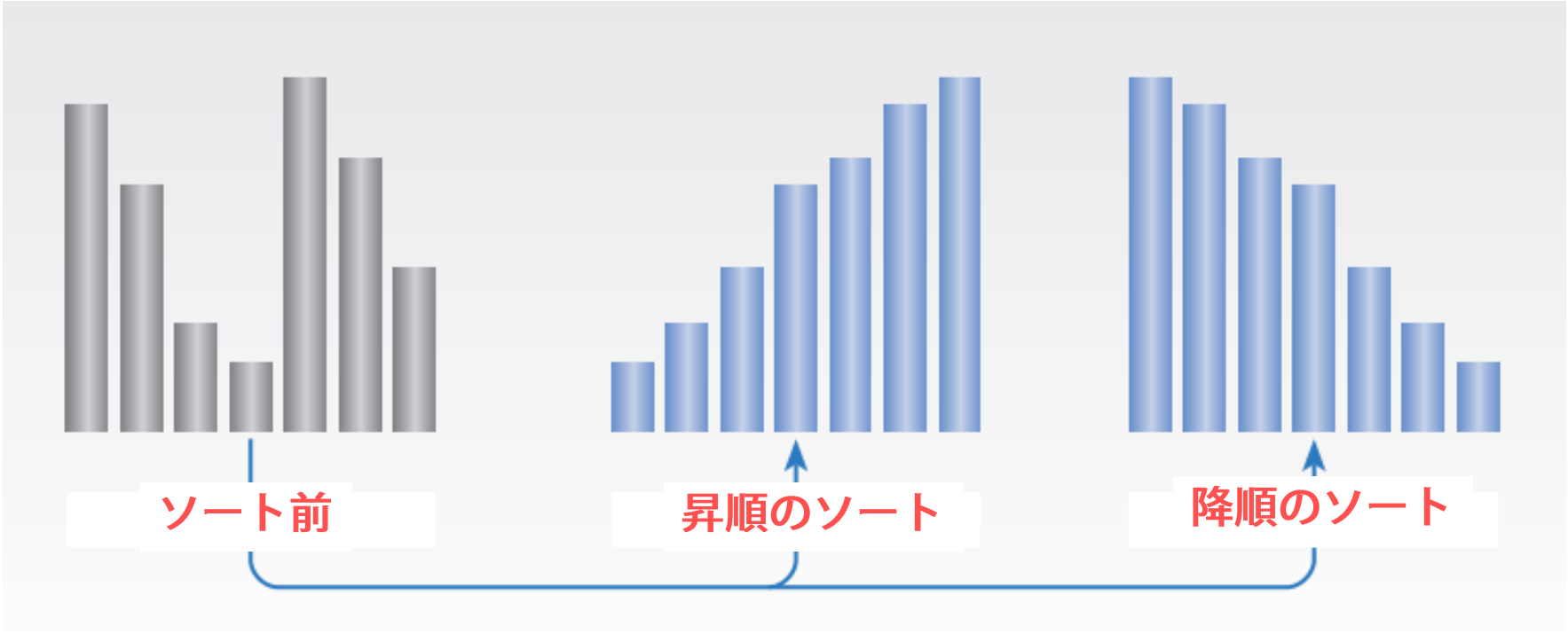
乱数で生成した配列の要素の最大値を表示してみます。

```java
// 配列の要素の最大値を表示します（値は乱数で生成）。

import java.util.Random;
import java.util.Scanner;

public class Max {


    // 配列aの最大値を求めて返します
    static int maxOf(int[] a){
        int max = a[0];
        for (int i =1; i< a.length; i++)
            if (a[i] > max)
                max = a[i];
        return max;
    }

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ

        Random rand = new Random();
        Scanner stdIn = new Scanner(System.in);

        System.out.println("通学にかかる時間の最大値を求めます。");
        System.out.println("日数は：");
        int num = stdIn.nextInt();          // 配列の要素数を読み込みます。

        int[] time = new int[num];      // 要素数numの配列を生成します。

        System.out.println("通学にかかる時間は下記のようであったと想定されます。");
        for (int i = 0; i < num; i++){
            time[i] = 15 + rand.nextInt(20);      //要素の値を乱数で決定します。
            System.out.println("time[" + i + "]:" + time[i]);
        }

        System.out.println("最大値は" + maxOf(time) + "です。");
    }
}
```

【実行結果】
通学にかかる時間の最大値を求めます。
日数は：
7
通学にかかる時間は下記のようであったと想定されます。
time[0]:20
time[1]:15
time[2]:31
time[3]:31
time[4]:29
time[5]:22
time[6]:26
最大値は31です。

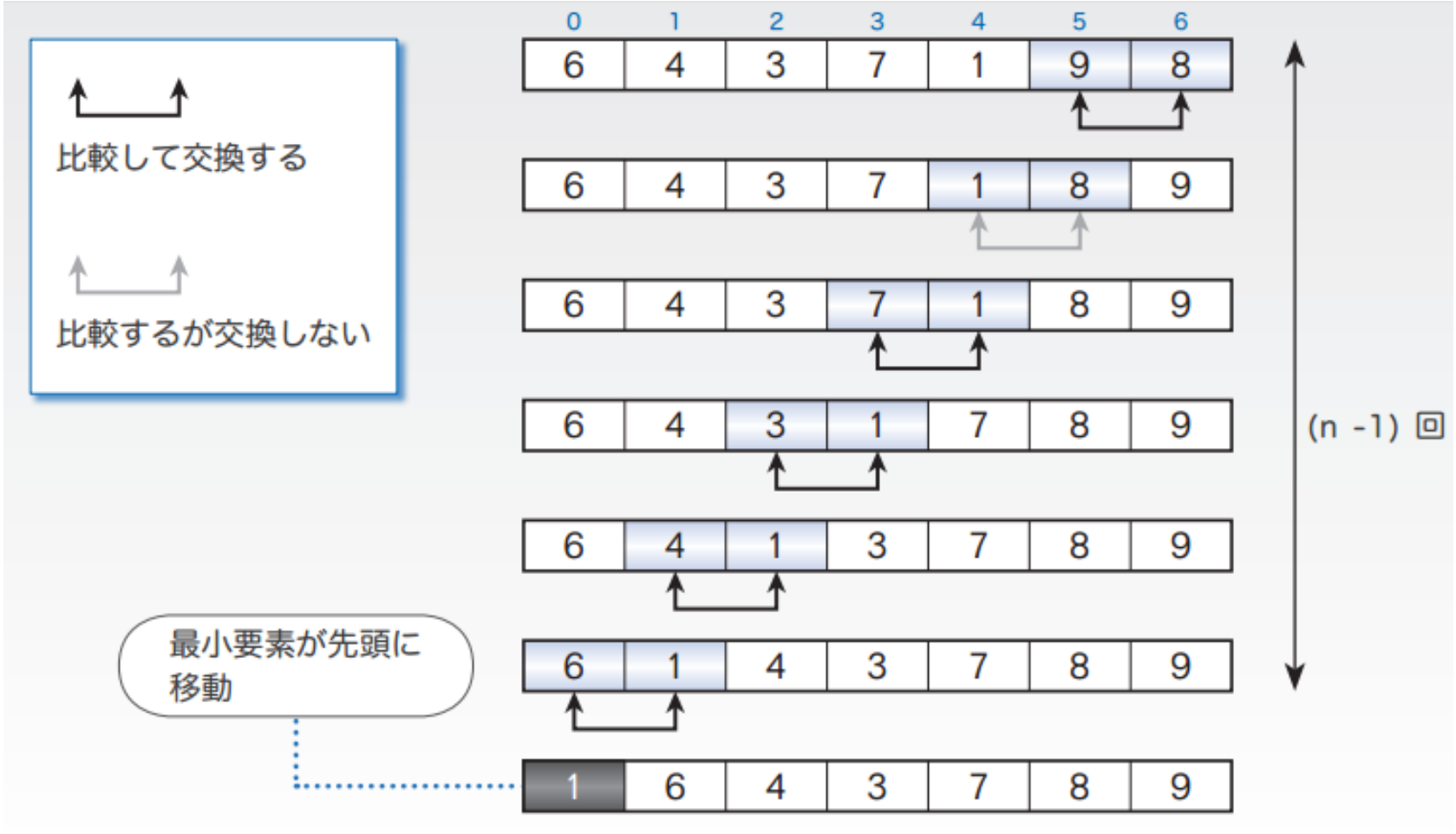
ソート

最も基本的で大切なアルゴリズムです。
ソート(整列)させれば探索が容易なのは自明です。



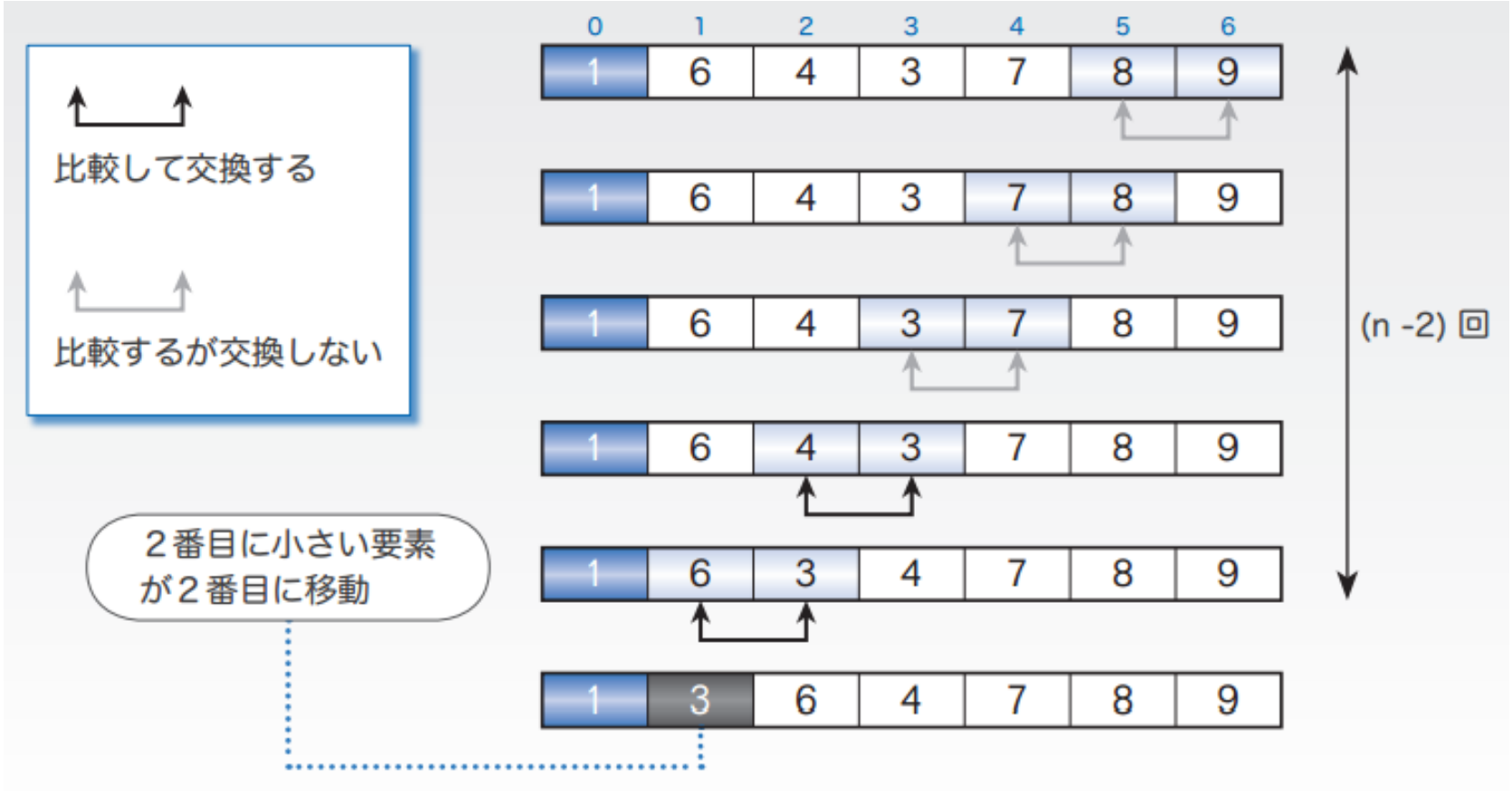
単純交換ソート（バブルソート）

隣り合う二つの要素の大小関係を比べて、
交換を繰り返すアルゴリズム。



単純交換ソート（バブルソート）

隣り合う二つの要素の大小関係を比べて、
交換を繰り返すアルゴリズム。



他のソーティング

補足資料

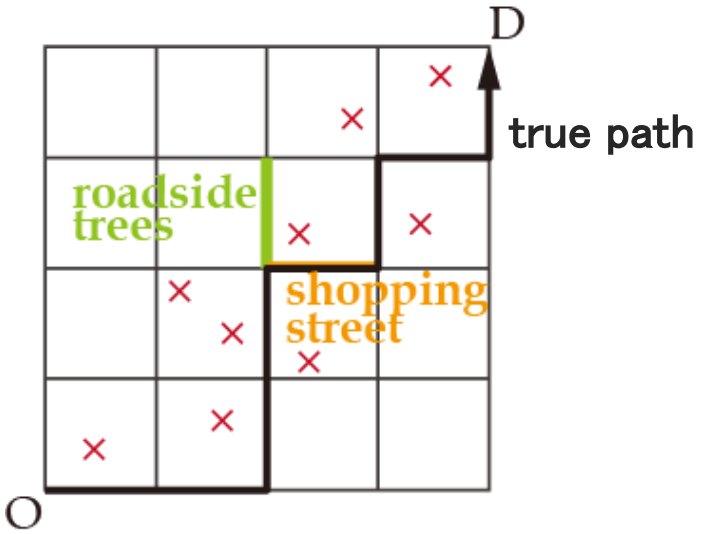
- 単純挿入ソート
- シェルソート
- ヒープソート
- マージソート
- クイックソート　　などなど

いろいろなソートアルゴリズムがあります。

安定なものと安定でないものがあるので、
使うときには注意してくださいね。

発表課題②　Wikiに補足資料とコードをあげます

WikiからPPデータとネットワークデータ(リンクとノード)をダウンロードして、経路情報にしてください。
余力があれば、マップマッチングの課題を解決する工夫をしてみてください。



・ポイント

マップマッチング
ダイクストラ法

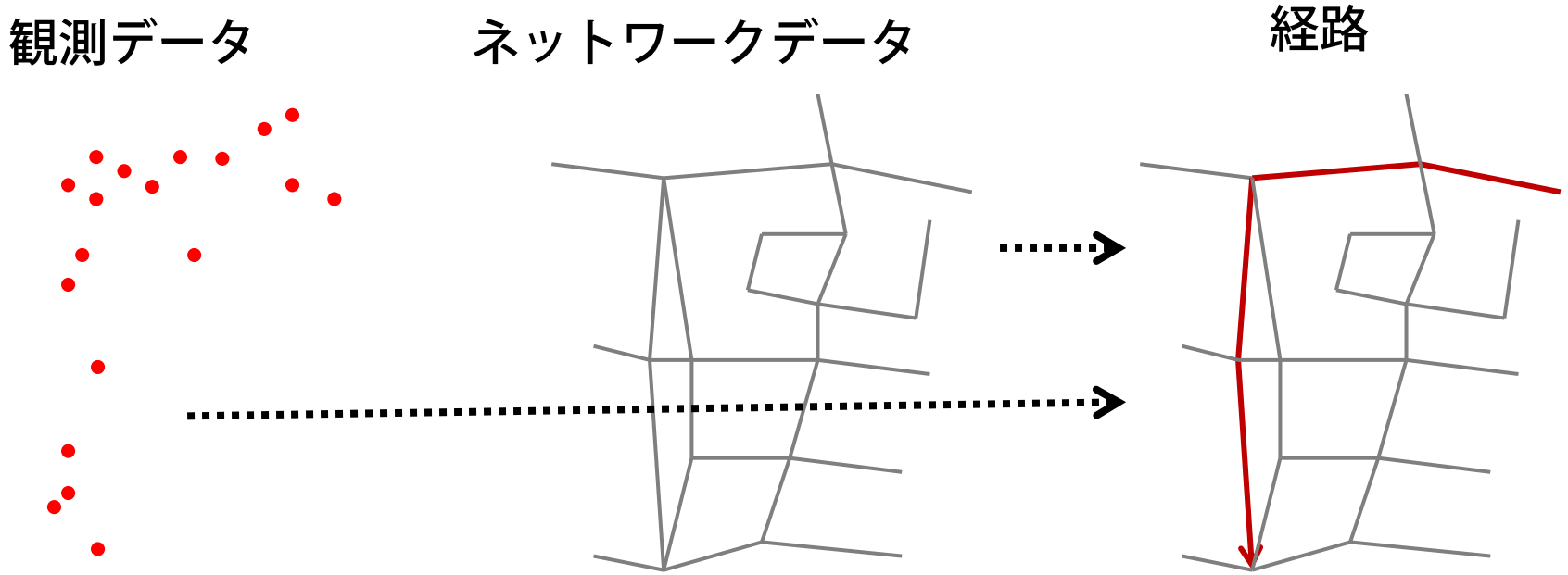
※解答例には三谷(2006)のアルゴリズムを掲載しています。

マップマッチング

詳しくは第3週

3週目に山口さんが詳しく説明してくれますが、

「位置データから経路をネットワーク上に特定することです。」

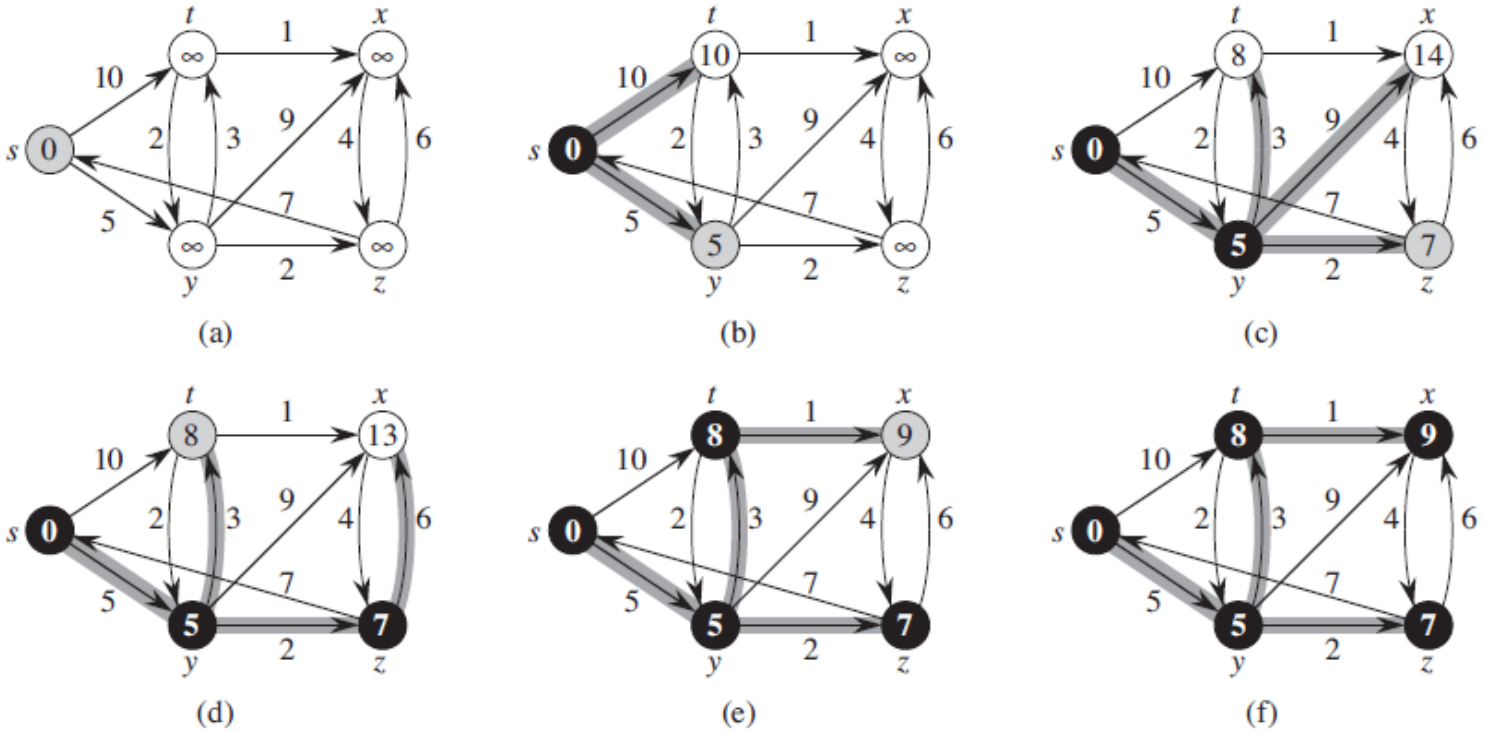


ダイクストラ法

詳しくは第3週

効率的に最短経路を求めるアルゴリズム。

・目的地への最短経路だけでなく、その他の全ての地点への最短経路を同時に求めることができます！！



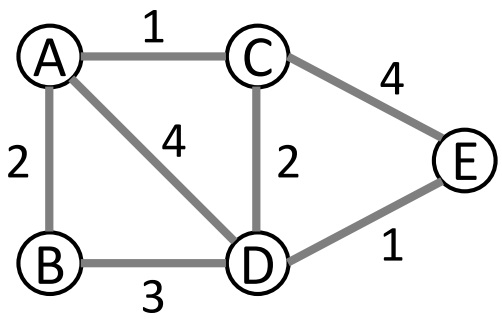
ポイント：分かった最短経路から順番に確定させる。

ダイクストラ法

詳しくは第3週

（補足）グラフのデータ構造

- リストによる構造/行列による構造



リスト

各リンクの起終点ノードで格納
疎なネットワーク向き

起点　終点　コスト

ID

1	A	B	2
2	B	A	2
:	:	:	:
13	D	E	1
14	E	D	1

行列

各OD間コストを格納
密なネットワーク向き

終点ノード

起点ノード

	A	B	C	D	E
A	0	2	1	3	∞
B	2	0	∞	3	∞
C	1	∞	0	2	4
D	3	3	2	0	1
E	∞	∞	4	1	0


Python

Pythonの導入

- 1991年にグイド・ヴァン・ロッサムさん(蘭)が開発したスクリプト言語です。
- 元々シンプルで習得が簡単な言語であることを目標に開発されたのでとってもわかりやすいです。
- しかもGoogle三大言語のひとつで、GoogleAppEngine、YoutubeやDropboxなどのサービスも　Pythonで実装されています！

ちなみに…
Pythonという名前は，BBCが製作したコメディ「空飛ぶモンティ・パイソン」から来ています（笑）

Pythonの特徴

- インデントでブロック構造を定義しているので，
  プログラムの構造を統一することができて
  誰が書いても常に読みやすいコードになります！

- インタープリタ言語なので、
  ソースコードのコンパイルが必要なくて、実行
  するコードを入力しながら逐次的に実行できます。

- 他の言語と結合させやすい！
  グルー(glue)言語とか言われています。

Python強い！

Pythonをはじめる準備①

- Pythonのインストール

  任せました。

- Anacondaを利用してインストールするととっても楽です。

  Anaconda：Python 本体に加え、科学技術、数学、エンジニアリング、
  データ分析など、よく利用される Pythonパッケージを一括で
  インストール可能にしたパッケージ。

- 2.x 系と 3.x 系のバージョンがありますが、

  絶対に3.x 系のバージョンを入れたほうがいい！

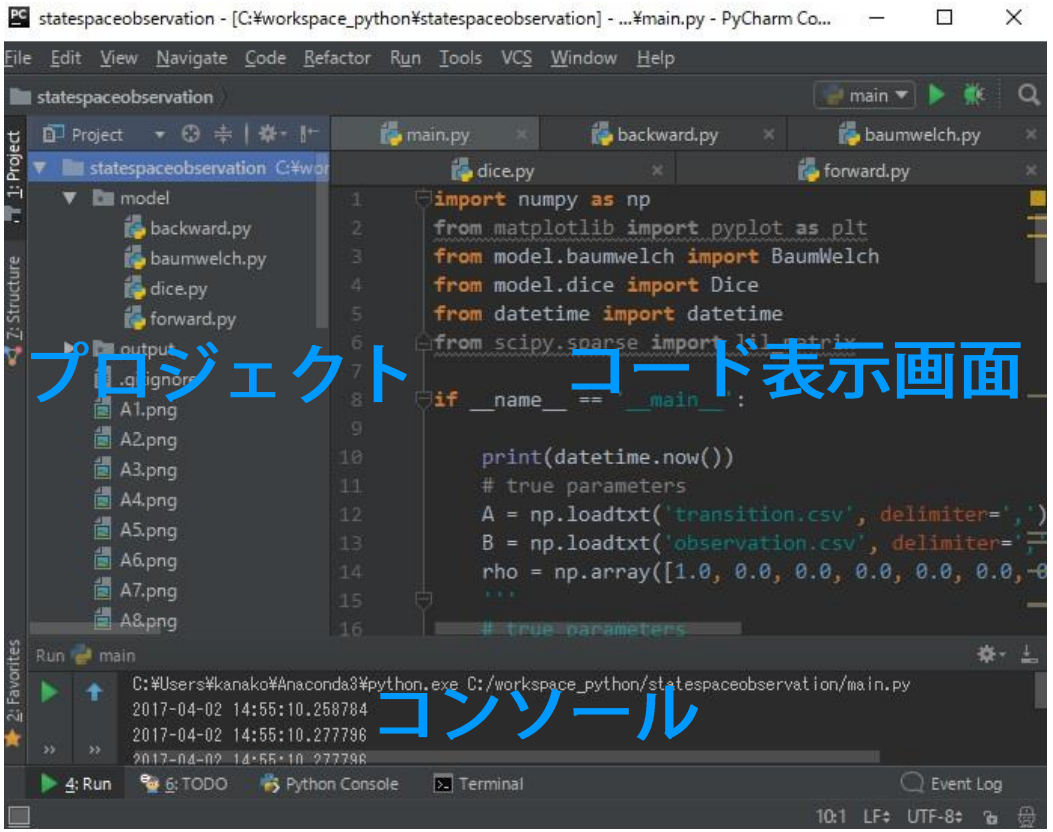
  2.x 系はレガシーです。

Pythonをはじめる準備②

- PythonのIDE

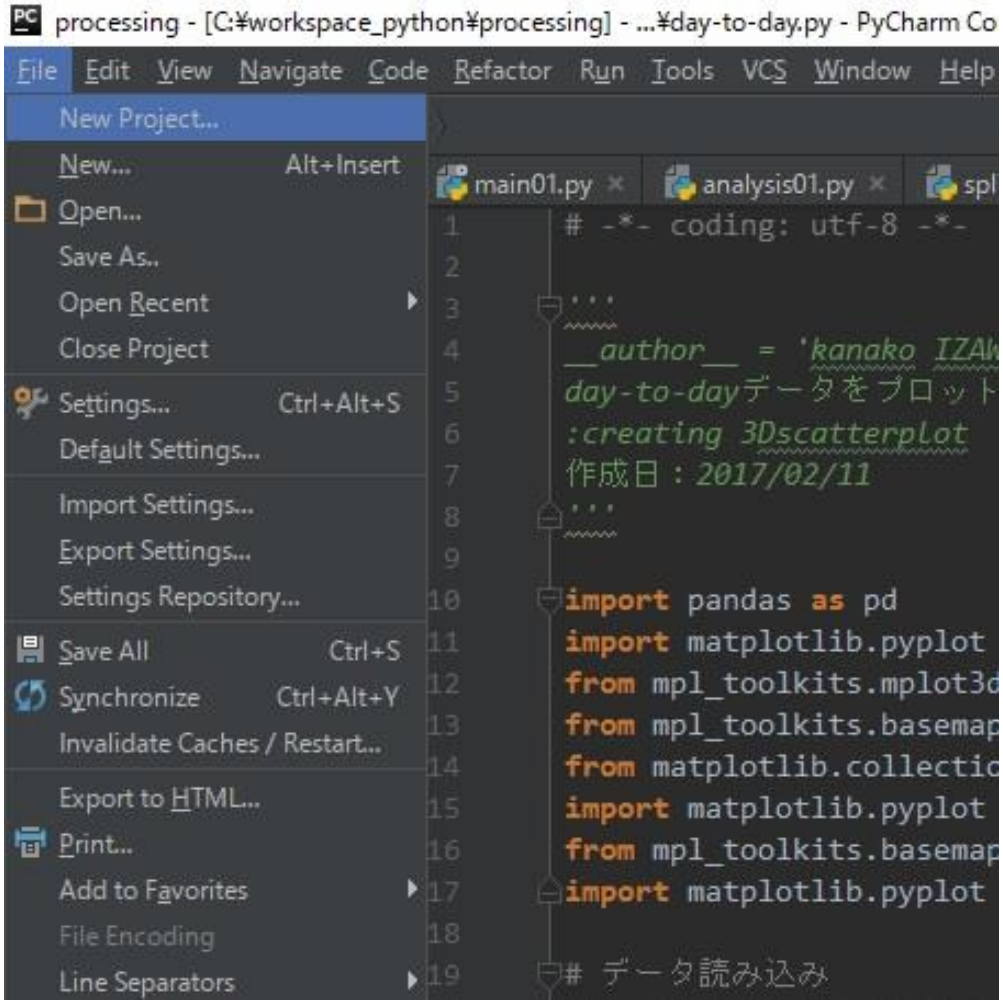
atomとかVisual Studio Codeとかいろいろありますが、
個人的にはPyCharmがおすすめです。

ちなみに
PyCharmの
インタフェースは
こんな感じ。→



Pythonを使ってみる準備①　by PyCharm

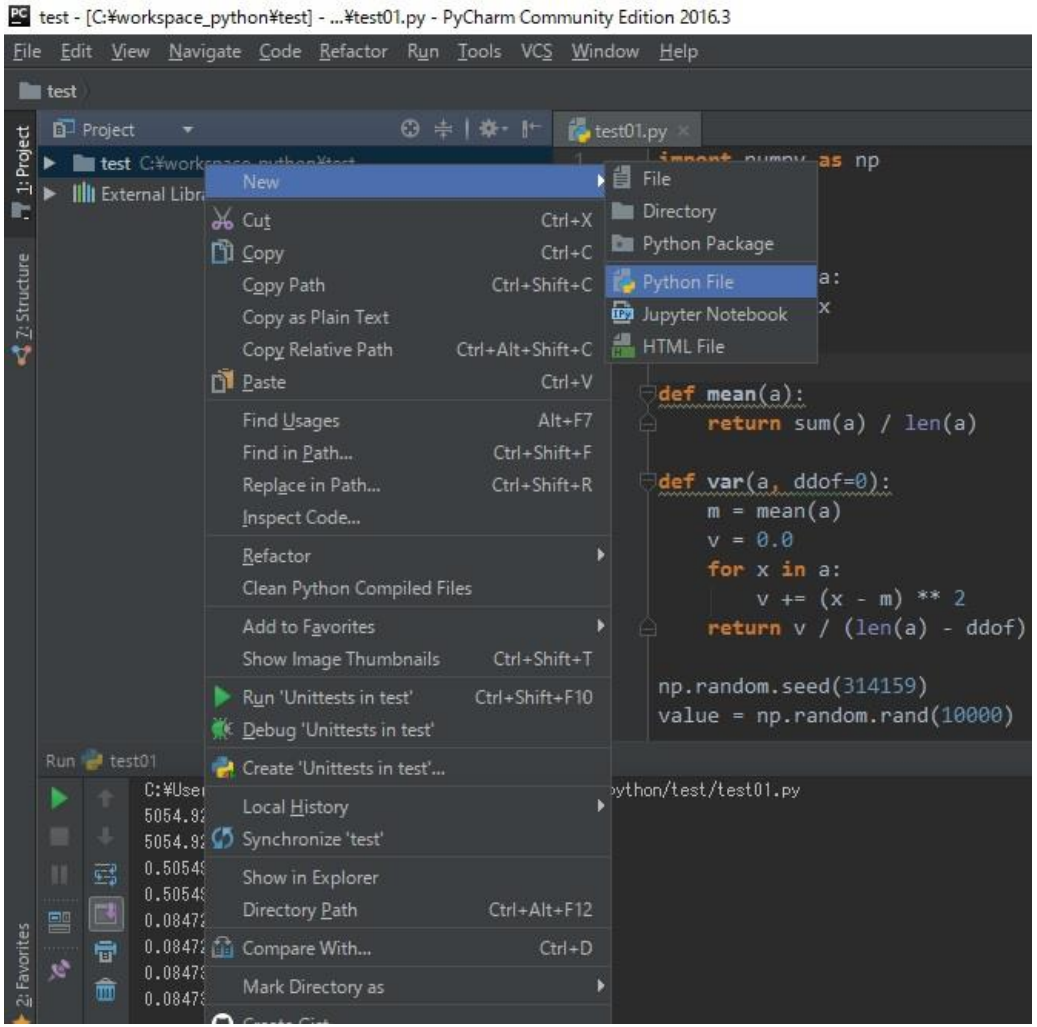
プロジェクトを作成します。



①[File]
↓
②[New Project]
↓
③プロジェクト名を入力
↓
⑤[完了]

Pythonを使ってみる準備②　by PyCharm

ファイルを作成します。



①プロジェクト名を
　右クリック
　↓
②[New]
　↓
③[Python File]
　↓
⑤[完了]

Pythonの基本的なこと①

1．記述スタイル

Pythonは記述スタイルに関してルールが厳しい言語です．
書き方を統一しておくと運用上なにかと便利なので，はじめに基本的なルール (PEP8) について確認しておきます。

A. エンコーディング

；・3.x系のデフォルトエンコーディングはUTF-8です。
・3.x系でUTF-8を使用しているファイルには、
・エンコーディング宣言は必要ありません。
・ファイル全体のエンコーディングと一致していないと，
日本語での処理がエラーになるので気を付けましょう。

B. インデント

C. コメント

Pythonの基本的なこと①

1．記述スタイル

Pythonは記述スタイルに関してルールが厳しい言語です．
書き方を統一しておくと運用上なにかと便利なので，はじめに基本的なルール (PEP8) について確認しておきます。

A. エンコーディング

B. インデント

；・Pythonではインデントが文法的に意味を持ちます。
・インデントには4文字分のスペースを使うのが一般的。
・3.x系ではタブとスペースの混在は禁止です。
・「\(バックスラッシュ)」を使えば次のインデントレベルは無視されます。

C. コメント

Pythonの基本的なこと①

1．記述スタイル

Pythonは記述スタイルに関してルールが厳しい言語です．
書き方を統一しておくと運用上なにかと便利なので，はじめに基本的なルール (PEP8) について確認しておきます。

A. エンコーディング

B. インデント

C. コメント

；・1行のコメントは「#」を先頭につけて記述します。
・複数行のコメントは「”””」or「”’」でコメントを囲みます。

↓↓こんな感じ

Pythonの基本的なこと①

1．記述スタイル

Pythonは記述スタイルに関してルールが厳しい言語です.
書き方を統一しておくと運用上なにかと便利なので，はじめに基本的なルール (PEP8) について確認しておきます。

D. その他

記述するときに関するものにも、

- カンマの後ろは必ず半角スペース
- 変数名はすべて小文字とアンダーバー
- クラス名とエラー名のみ大文字を使う

などなどあるので確認してみてください。

Pythonの基本的なこと②

2．オブジェクトと型

- Pythonもオブジェクト指向プログラミング言語なので、あらゆるものがオブジェクトです。

  ※オブジェクト：データとそのデータに対して規定されたメソッド

- 基本的は考え方はJavaと同じですが、名称が違ったり、Pythonにしかない組み込み型がある(逆もまた然り。)ので、各自確認しておいてください。

確認事項

- 識別子（3.x系では日本語も使えます。）
- データ型；整数型、浮動小数点型、文字列型、ブール型、リスト、タプル、辞書型など

Pythonの基本的なこと③

3．演算子

・これまたRやJavaと似ています。

・各自確認しておいてください。

特に算術演算は少し違うところがあるので注意です。

確認事項

- ブール値と演算； True / False

  例. not x：xが偽なら真、xが真なら偽

- 比較演算子
- 数値型の算術演算

Pythonの基本的なこと④

4．条件分岐

・これまたRやJavaと似ています。

・各自確認しておいてください。

確認事項

・if 文、for 文、while 文、try 文

```
21  ph = 8
22  if ph > 7:
23      print("アルカリ性です。")
24  elif a == 7:
25      print("中性です。")
26  else:
27      print("酸性です。")
```

[結果]

アルカリ性です。

Pythonの基本的なこと⑤

5．関数

・関数を定義するには、

def の後に 関数名 と 引数 を指定します。

```
def 関数名(引数1，引数2，…，引数n)：
        """ 関数の説明 """
        statement1
        …
        …
        return something
```

←Docstring

ここに
関数の処理を
かきます。

※lambda式記法を使えば無名関数を作成することもできます。

Pythonの基本的なこと⑥

6．モジュールとパッケージ

- モジュールとは、
  関数やクラスなどが書かれたファイルのことです。

- 大規模なプログラムを書くときには、
  それらを機能ごとに分割しておくと便利です。
  importして使うことができます。

  この仕組みを使うことでPythonでも高速計算が可能です！

- パッケージとは、
  モジュールを束ねて管理するための仕組みです。

ライブラリ

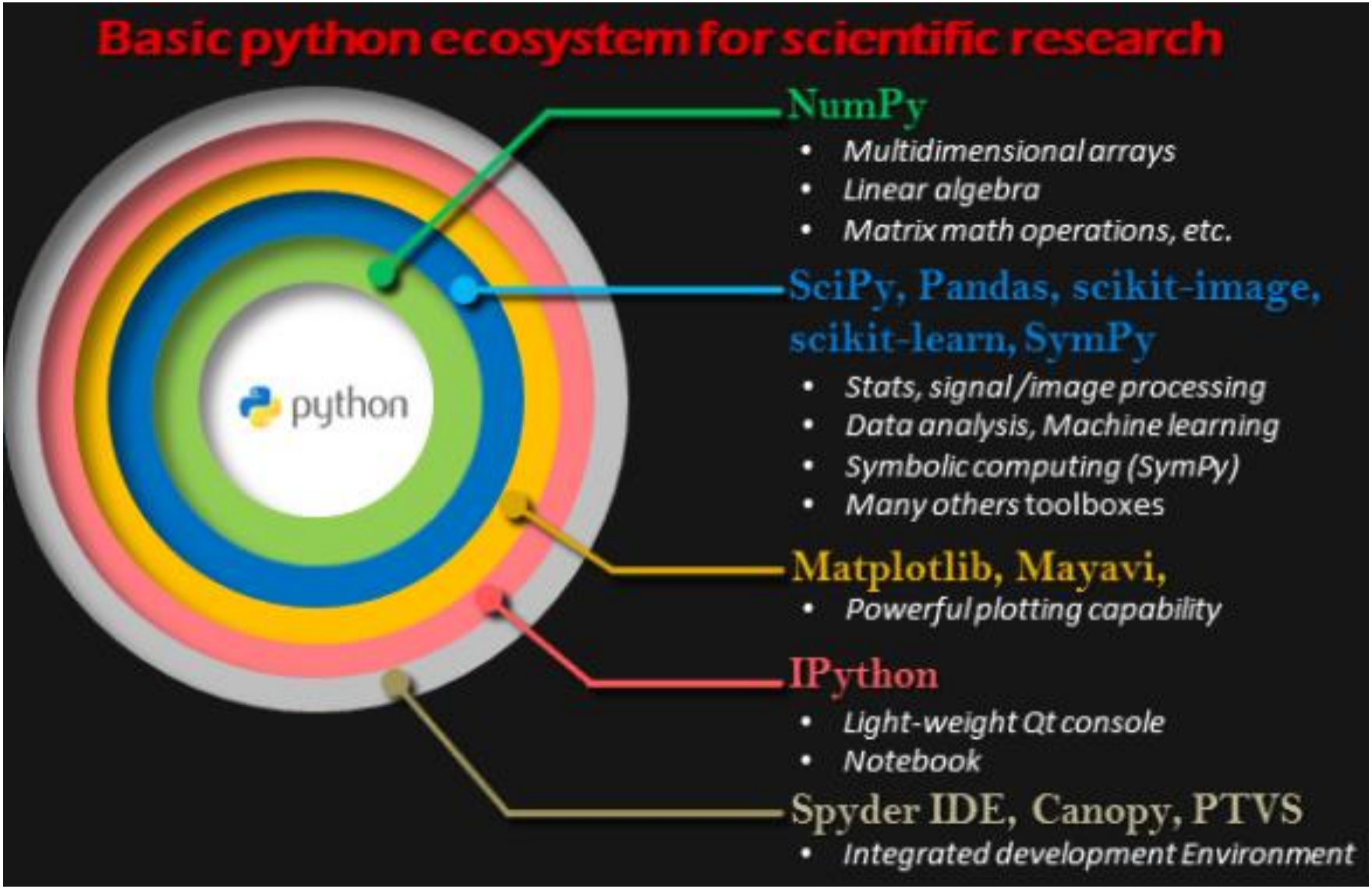
- ライブラリとは、

  あらかじめ特定の機能を実現するために
  作成されたプログラム群のことです。

- このライブラリのなかに
  先ほどのモジュールとパッケージがあります。

- 実はPythonは、
  Python本体と標準ライブラリだけだと
  高度な計算は苦手なんです。

Pythonのエコシステムと基礎ライブラリ



https://indranilsinharoy.com/2013/01/06/python-for-scientific-computing-a-collection-of-resources/

モジュールの使い方

- 使いたいモジュールの名前の前に **import** を付けて最初に宣言します。

test01.py

```
1  # 必要なものをimport
2  import numpy as np
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  import seaborn
6
```

- 他にもいろいろな**import**の方法があります。

from [モジュール名] import [メンバ名1]
from [モジュール名] import *　　などなど

基礎ライブラリ：Numpy

- 配列処理に特化したライブラリで
  多くのライブラリの基礎になっています。
- なのでだいたい **Numpy** の **import** が必要です。
- 組み込みデータ型とは別に **Numpy** 独自の
  細分化されたデータ型があります。
  不必要なメモリの消費を回避し、効率のよい
  処理を行うためです。
- その他 **ndarray** 等々各自確認しておいてください。

基礎ライブラリ：Scipy

- **Numpy** の機能の上に構築された、
  いろいろな計算アルゴリズムを提供する
  パッケージ。
- なので **Scipy** を **import** すれば、
  Numpy の **import** は不要です。

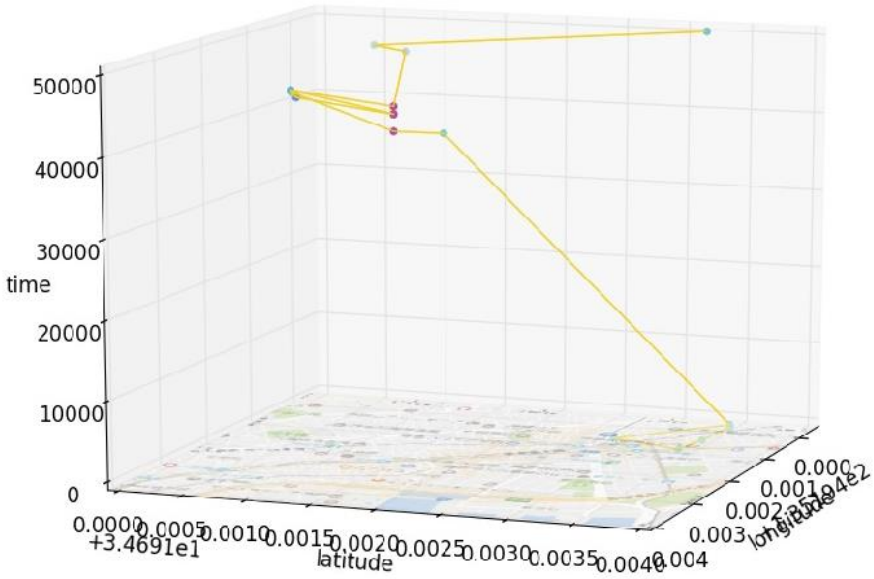
- とても多くのサブパッケージから構成されて
  います。
  少しずつ勉強していきましょう 。

基礎ライブラリ：Matplotlib

- 数値シミュレーションの結果を可視化したり、
  生データを図に落としてみると、
  新たに分かる事実を発見したりできますよね。

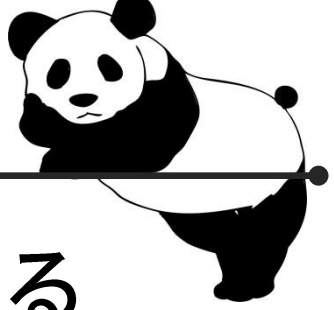
- **Matplotlib** は、

  データや分析結果を
  把握するために必要な
  プロット作成機能を
  たくさん持っています。



↑たとえばこんなのとか…

基礎ライブラリ：Pandas



- データ解析を容易にする機能を提供するライブラリ。
- とてもハイスペックです。

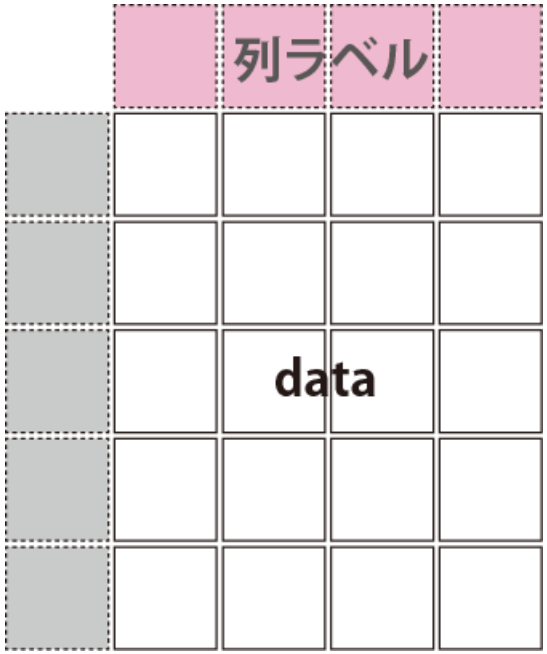
；データの入出力 (CSV, Excelなど)
データの欠損値処理
データの一部取り出しや結合
データのピボット処理
グループ演算　　などなど

- 少しずつ勉強していきましょう。

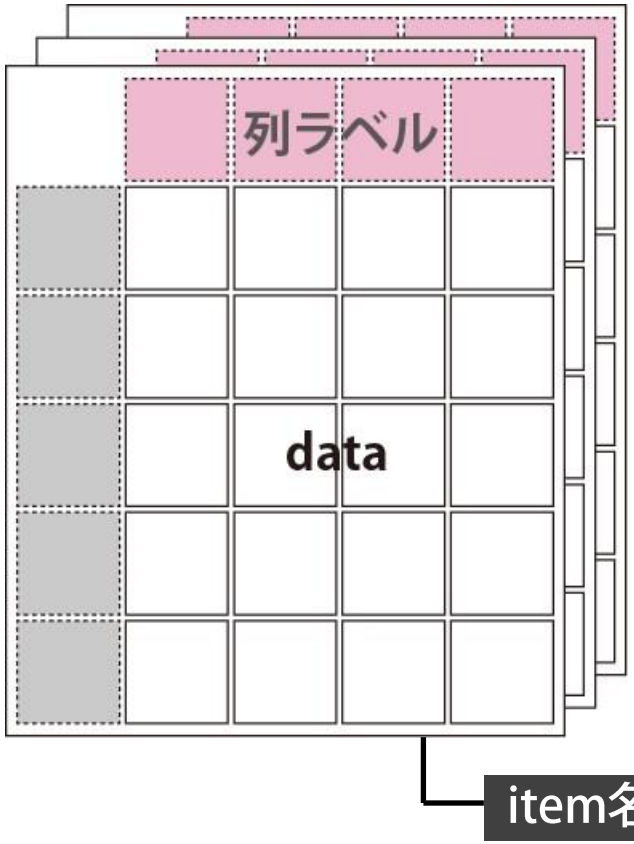
Pandas のデータ型

・シリーズ　・データフレーム　・パネル

1次元データ



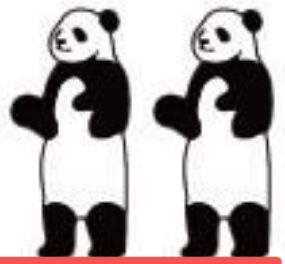
2次元データ(ラベル付き)



3次元データ
(データフレームを重ねた構造)

Pandas のデータ処理

- 基本的な処理を紹介しておきます。
  各自確認してみてください。
  補足資料を参照してください。



確認事項

- 部分データを取り出す。；インデックスの参照。
- 演算規則。；記号による方法とメソッドを使う方法。
- 比較演算
- 統計関数。；any、diff、sum、cumsum などなど
- NaNの処理。；fillna、dropna

頻出文法例

- pd.read_csv：csvファイルの読み込み.
  With open('hogehoge.csv', 'r')よりおススメ.
- df.to_csv：csvファイルへの書き込み.
- df.dropna：削除
- df.filna：定数(0)などで穴埋め.
- pd.datetime：時間を扱うとき.
- df.groupby：columの名前ごとにグルーピングとかできて便利.
- df.pivot_table：Excelのピボットテーブルみたいな.
- pd.concat：データの結合による列の追加とか柔軟な連結ができます．便利.
- df.append：シンプルな縦方向の連結ならこちらで.
- pd.merge：列の名前による結合．便利.
- df.join：indexをキーに結合したいときはこちらで．　などなどなど

Python のエラー処理

- どういうエラー？

エラー発生個所の直前に行われた処理が
比較的わかりやすく書いてあります。

よく出てくるエラー例

- AttributeError : 属性の参照や代入のエラー。
- IndexError : リストの長さを超えているとか、配列のインデックスに関するエラー。
- RuntimeError : 無限ループが発生したときのエラー。
- SyntaxError : 構文エラー。だいたいケアレスミス。
- IndentationError : インデントが間違っているときのエラー。
- TypeError : 正しくない型に対して演算処理などが行われたときのエラー。
- ValueError : 型は正しいがその値は処理できないときのエラー。

その他　今日の内容は…

補足資料

- 計算するための導入部分について、
  ほんの少しだけお話しました。

- クラス、メソッド、日付・時間の扱い方、
  新しいデータ型の作り方など
  各自確認してみてください。

- ネットにいっぱいあるので色々見てみると
  面白いと思います。

発表課題③

- ある人の１カ月間のWi-Fiデータを
  Wikiからダウンロードして
  基礎分析してみてください。

- Excelでも構いません。

- ピボットテーブルとか、
  集計してグラフにしてみるとか、
  楽しいと思います。


さいごに

RとJavaとPythonと

	R	Java	Python
良いところ	・統計計算が得意。 ・統計結果の図示も得意。	・実行速度が速い。 ・頑強なシステムを構築できる。	・コンパイル不要なのですぐ実行できる。 ・多少のエラーがあってもまわる。 ・読みやすい。
悪いところ	・統計計算以外は苦手。 ・メモリを食いやすい。（メモリ上にファイルを全て置いておく仕組みなので。）	・規模によってはコンパイルに時間がかかる。 ・文字列の操作が若干苦手。 ・エラーがあるとコンパイルできない。	・CやJavaと比べると遅い。

あるファイルをまるごと読み込む。

１行ごとにファイルを読み込む。

さいごに

- 何はともあれやってみるのが１番！
- プログラミングを勉強するというよりも、

  何かしたいことに対しての手段

  という意識でやってみるといいと思います。
- 一緒に楽しく研究しましょう！！