

Pythonで体験するベイズ推論

3.2~3.4

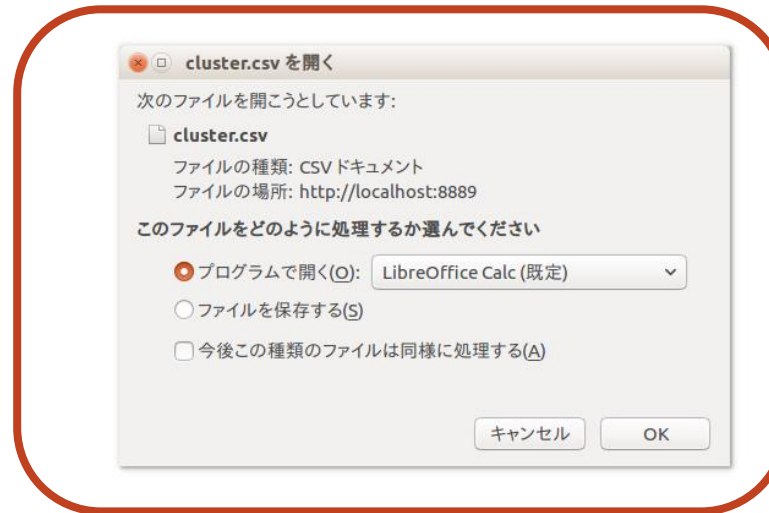
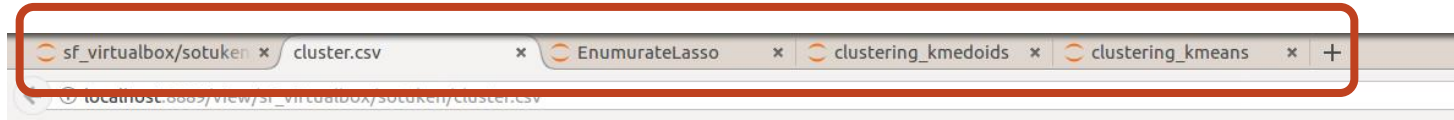
秋山研 B4 多治見 隆志

その前に…

jupyterlabのススメ

jupyter notebookを使うと...

ブラウザのタブがめっちゃ増える...



.csvがjupyterで
開けない...

- 便利だけど細かい所で不便な感じ...

jupyter labなら！

1つのタブで！

複数のコードを編集！

The screenshot displays the JupyterLab Alpha web interface. On the left, a file browser shows a directory structure with files like 'kmedoids', 'clustering_kmeans...', 'clustering_kmedoids...', 'EnumerateLasso.ipynb', and others. The central area contains several open tabs: 'Launcher', 'cluster.csv', 'clustering_kme', 'clustering_kme', and 'EnumerateLass'. The 'cluster.csv' tab is active, showing a table with columns 0 through 13 and rows of numerical data. The right side of the interface shows a terminal or console output area.

簡単にコードを開ける！

.csvの中身も見れる！

インストールもたった 3 手順！

1. `pip install jupyterlab`

2. `jupyter serverextension enable --py
jupyterlab --sys-prefix`

(2. は 1 行, 何してるかよく分かってない)

3. `jupyter lab` (起動)



快適なjupyterlabライフを！

というわけで本題を始めます

自己相関

自己相関 $R(t, k)$ は以下のように定義される

$$R(t, k) = \text{Corr}(x_t, x_{t-k})$$

$$\text{Corr}(X, Y) = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

- 時刻 t におけるデータ列 x_t と時刻 $t - k$ におけるデータ列 x_{t-k} の間の相関

Example

以下の二つのデータ列について考える

$$x_t \sim \text{Normal}(0, 1), x_0 = 0$$
$$y_t \sim \text{Normal}(y_{t-1}, 1), y_0 = 0$$

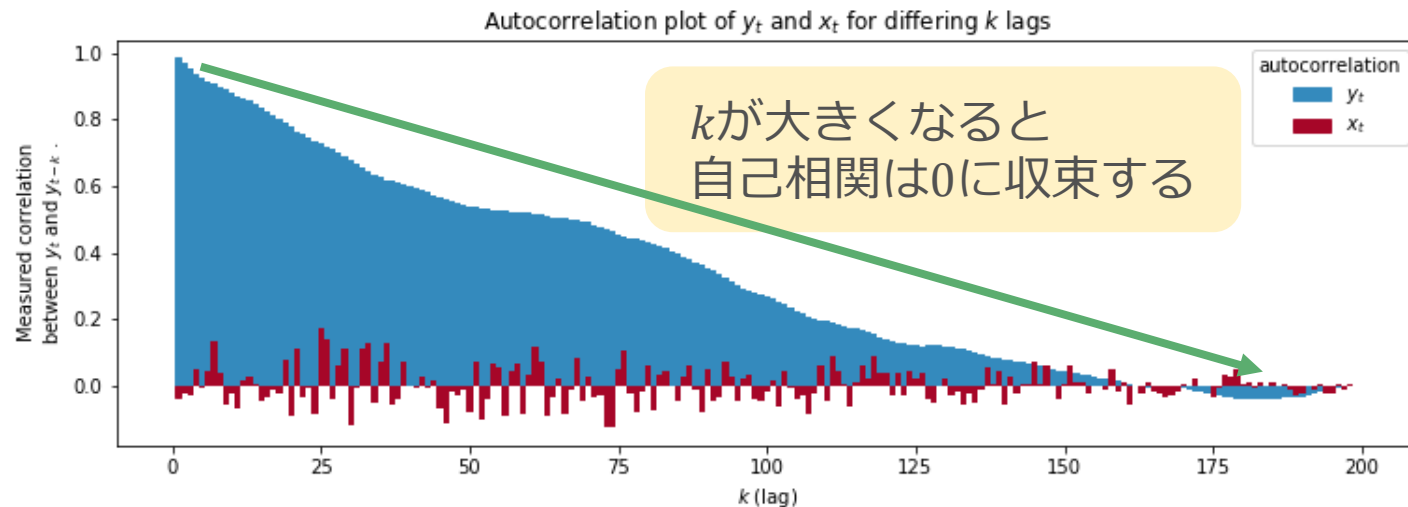
- x_t は x_{t-k} に明らかに相関しない（独立）
- y_t は y_{t-k} に相関してそう（自己相関）

Example



- y_t は前のデータに依存しているように見える
- x_t は前のデータに依存せずランダムな値を取っているように見える

y_t と x_t の自己相関



- y_t は k が小さいほど大きな自己相関を持つことがわかる
- x_t には k に関わらずほぼ自己相関はない

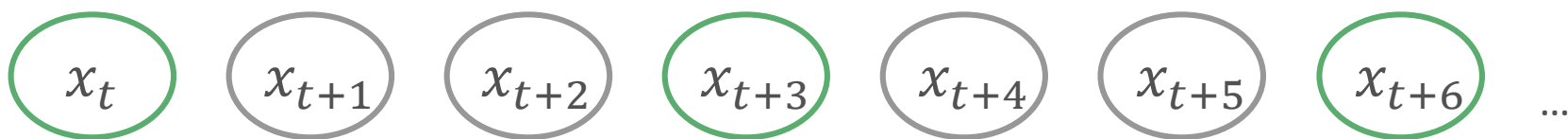
MCMCの収束

- MCMCアルゴリズムの性質上得られるサンプルは自己相関を持つ
- 後処理のアルゴリズムはサンプルに自己相関がないことを仮定している
- 自己相関を避けるために**間引き処理**を行う

間引き処理 (thinning)

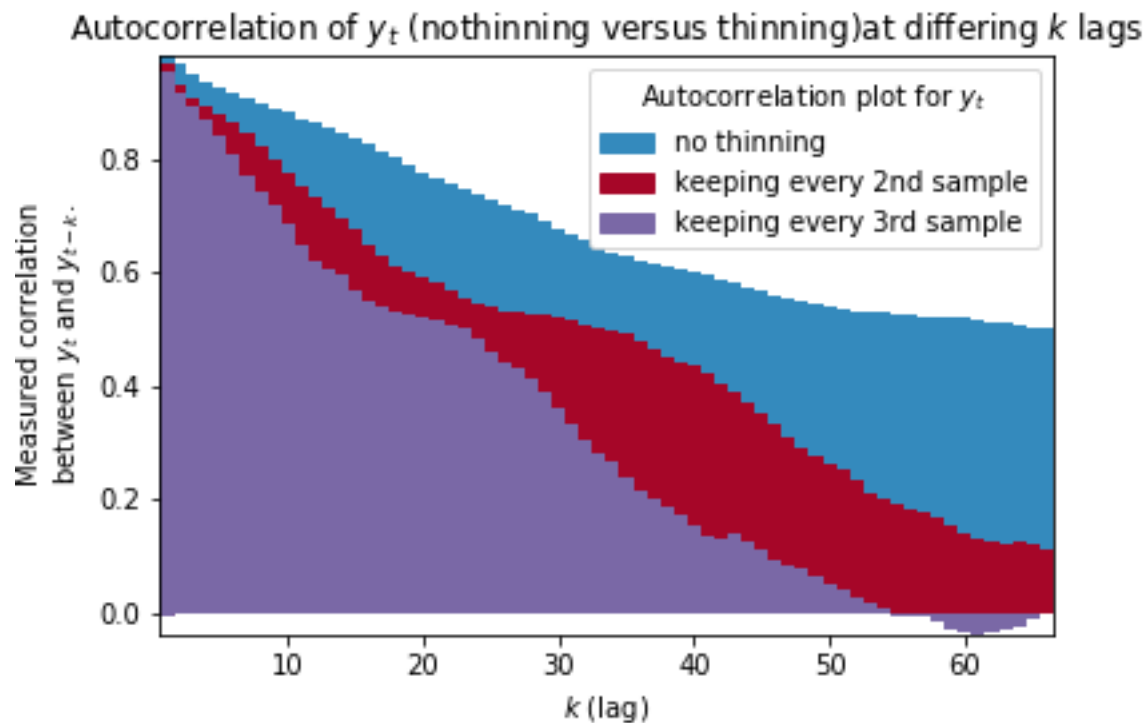
n サンプルごとに間引いた系列を利用することで
自己相関を抑える

(k が大きくなると自己相関が0に収束していくことを思い出したい)



$n = 3$ のときの例

間引き処理の結果



- 間引きを行うことで自己相関の収束が早くなる

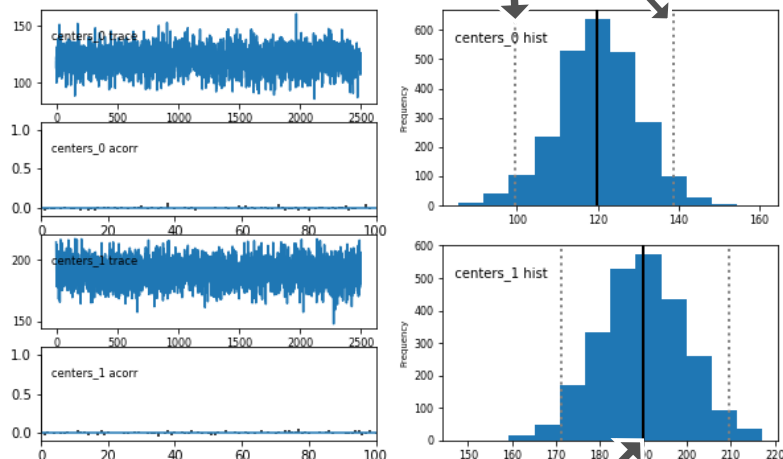
可視化ツール

- pymcでは以下のものを可視化できる
 - ヒストグラム
 - 自己相関
 - 軌跡



可視化の例

事後分布の
95%信用区間



事後分布の平均

これを用いて様々なことが
わかる

- 軌跡から収束を見る
- ヒストグラムから妥当な事後分布が見る
- 自己相関を見る

MCMCの収束判定

教科書に載ってないおまけ

- Raftery and Lewisの診断
- Gelman-Rubin統計量
- Gewekeの診断

+目で見るとか信用できねえ！
って人はこちらでどうぞ

Raftery and Lewisの診断

- 分位点を使った診断
- 一定数以上のMCMC連鎖が必要 (Lower Bound)
 - 正の相関があると連鎖の必要数が増える
 - Dependenceが自己相関の指標
 - 5より大きいと自己相関が強い

```
> raftery.diag(MCMC01)
```

```
Quantile (q) = 0.025  
Accuracy (r) = +/- 0.005  
Probability (s) = 0.95
```

	Burn-in (M)	Total (N)	Lower bound (Nmin)	Dependence factor (I)
(Intercept)	2	3741	3746	0.999
x1	2	3771	3746	1.010
sigma2	2	3865	3746	1.030

Gelman-Rubin統計量

- 2つ以上の連鎖が必要
- 連鎖同士の平均量と個々の連鎖の値を比較
- 統計量が1に近ければ収束しているとみなす

Gewekeの診断

- 連鎖内で適当な分位点同士の差の検定を行う

なんかいっぱいあったけどよくわかんねー



MCMCの＋コツ＋

良い初期値から始める

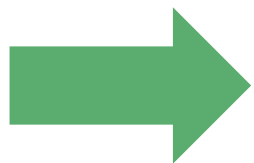
- 悪い初期値を用いると収束しない
- とりあえず困ったらMAP推定値を使おう

今日もMCMCが収束しない...



事前分布の選び方

- 事前分布が悪いと収束しないことがある
 - ー パラメータが真の値に対して確率が全く割り当てられていない事前分布を用いた場合、事後確率は0になる



収束が悪かったり、サンプルが密集している場合は事前分布を選び直したほうがよい

今回のまとめ

- MCMCサンプリングは自己相関が起こるので間引き処理をしよう
- 可視化することで簡単に収束を確認しよう
- 事前分布や初期値は気をつけて選ぼう

