Git / GitHub 入門

M2 田中

Git / GitHub 講習

Git / GitHub を最低限使えるようになるための講座

- 座学パート
- 実践パート
- おまけパート

の 3 段構成になっております.

座学パート

- Git とは
- GitHub とは
- ありがたみを感じるケース
- コマンド
  - クローン
  - **コミット**
  - **プッシュ**
  - プル
  - ブランチ
  - チェックアウト
  - マージ

Git とは

バージョン管理システム

お手元の PC にインストールされています.

（ターミナルを立ち上げて「git」と入力してみましょう）

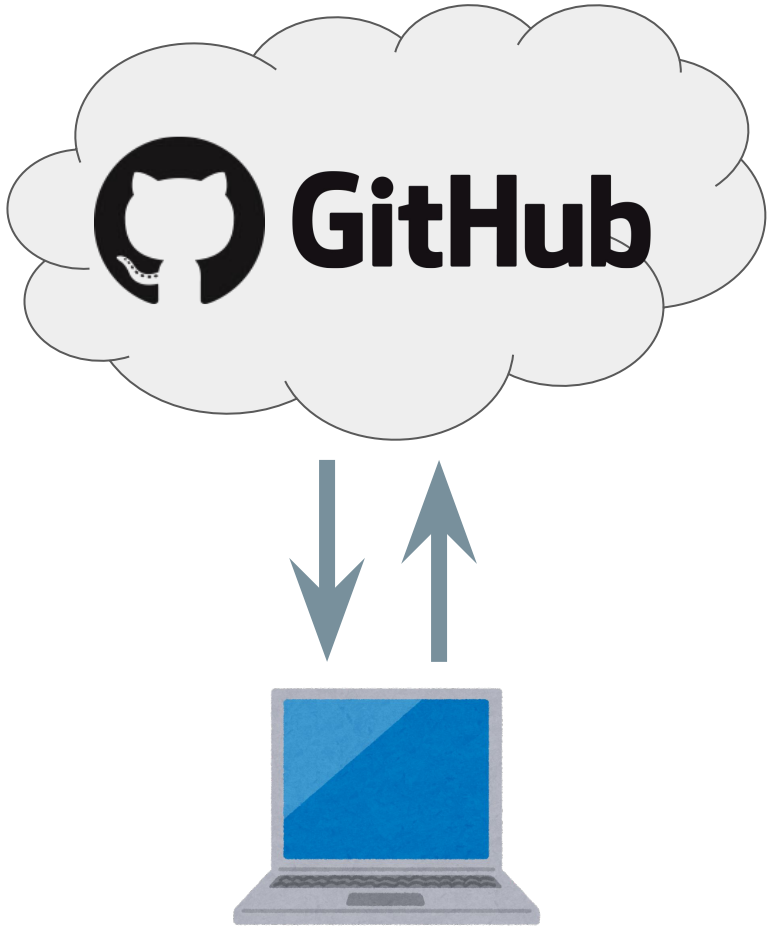
GitHub とは

Git をいい感じに使うための Web サービス（およびその運営会社）

https://github.com でアクセスできます，

アカウントはお持ちでしょうか？？

Git ≠ GitHub



役に立つケース①：もとに戻したい

① ちょっと書き加えよう～～

② あれ，上手くいかない．
おかしいぞ～

③ よくわかんなくなってきた
一旦もとに戻そう…

役に立つケース①：もとに戻したい

④　よし，これで多分元通りだ

⑤　あれ，元通りにした
つもりなのに動かない…

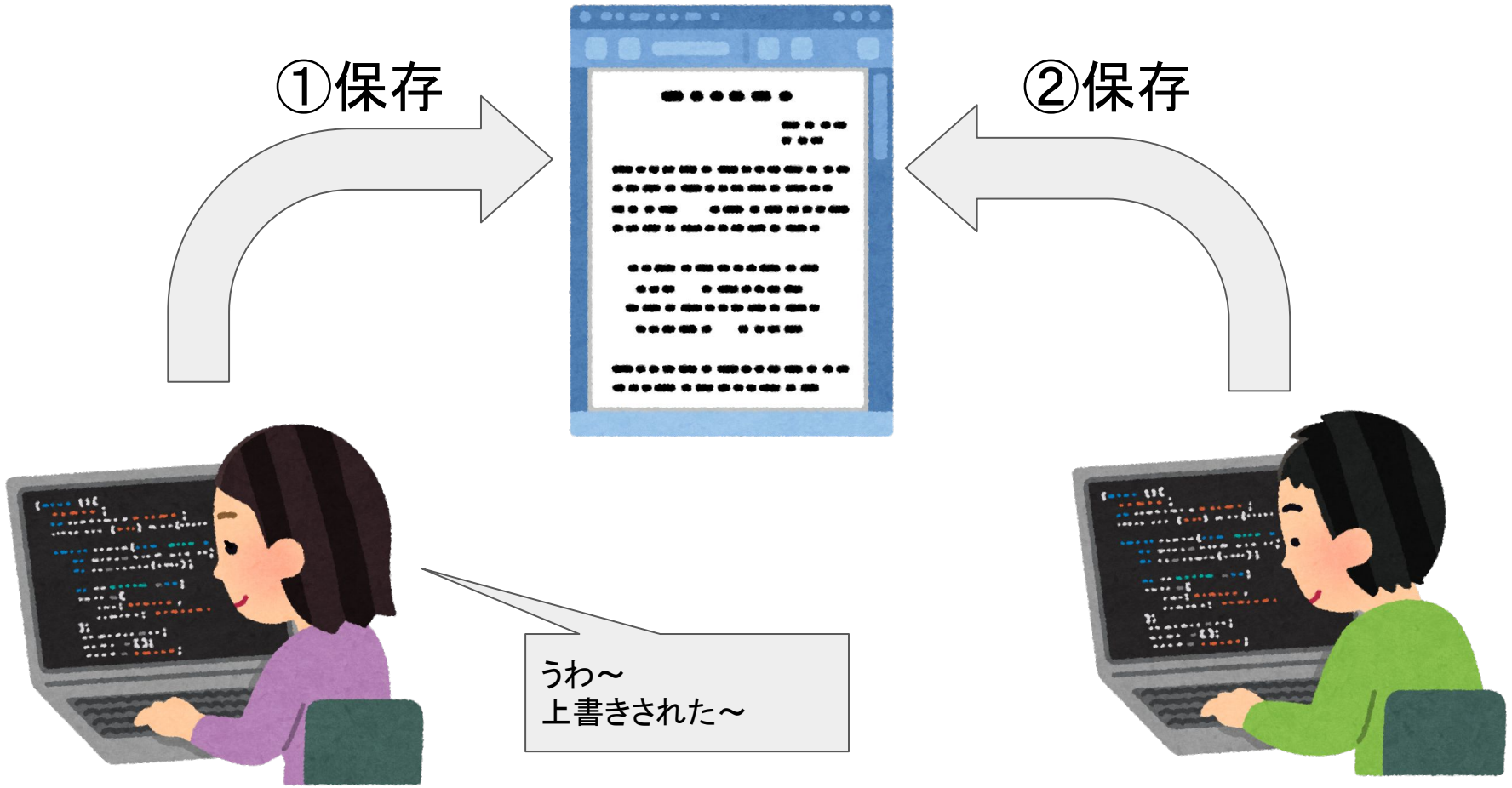
⑥　元通りにするのに
時間かかりそうだ…

役に立つケース①：もとに戻したい

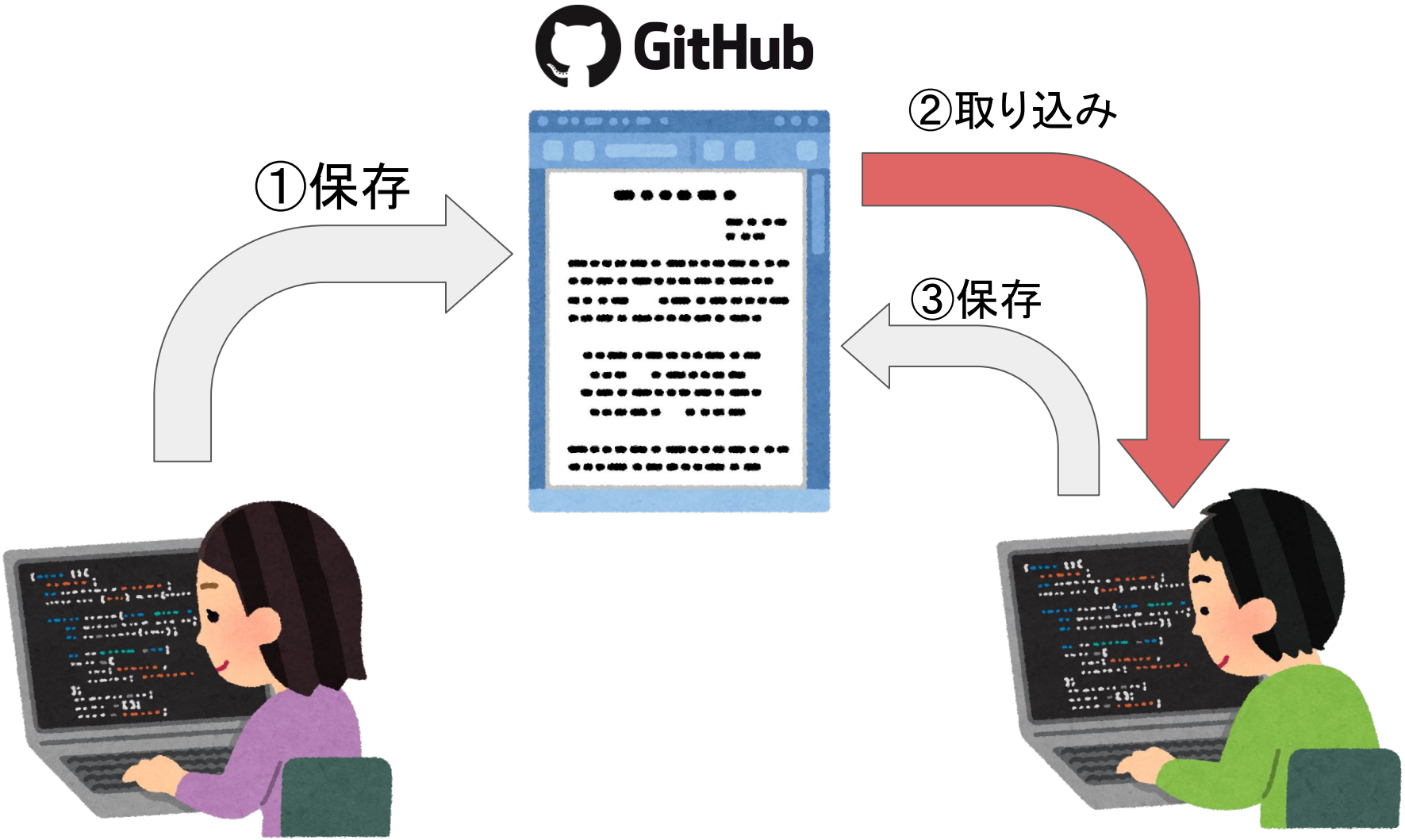
Git を使えば…

コマンド1発で元通り！
気を取り直して考えよう

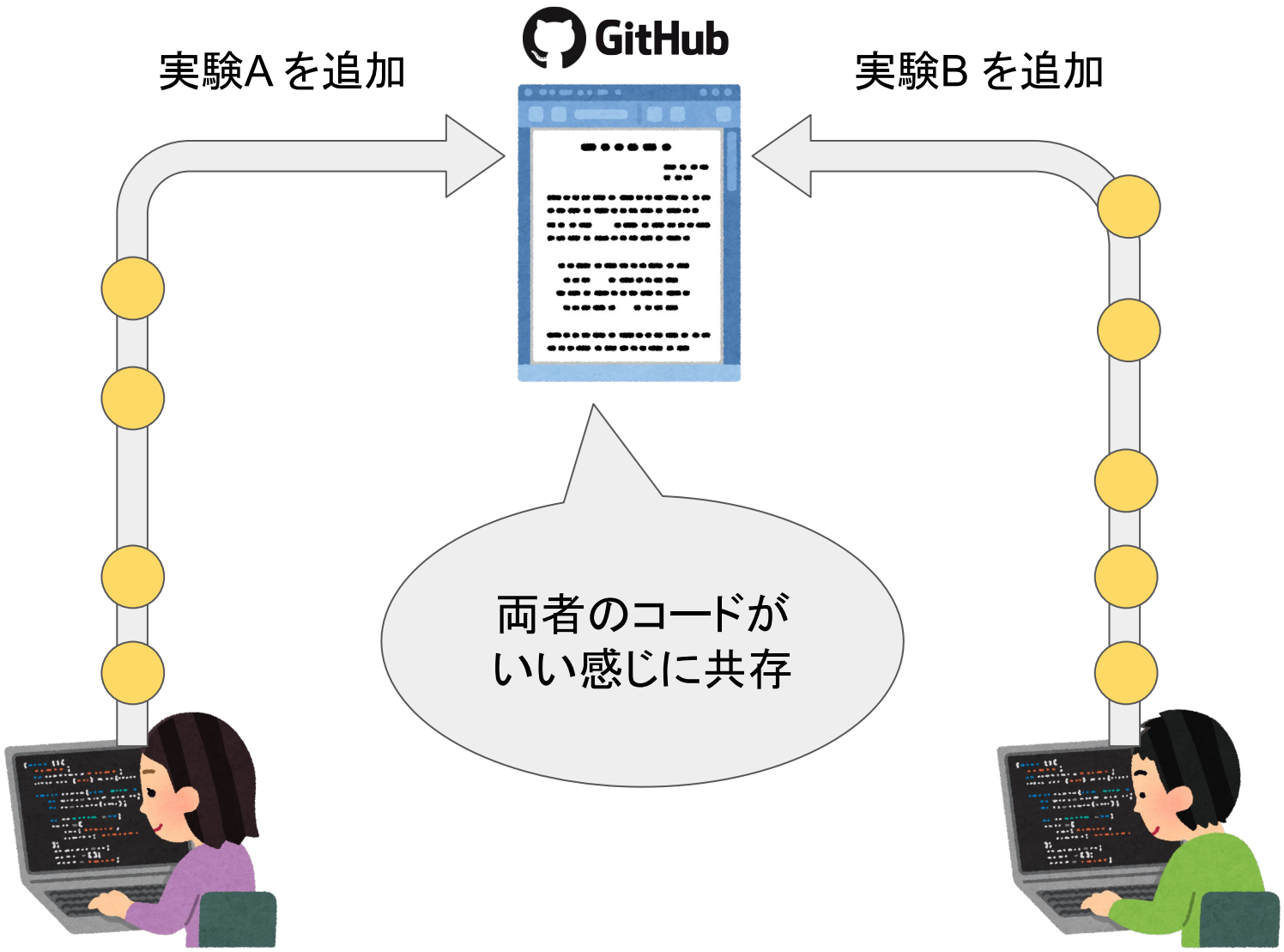
役に立つケース②：共同作業したい



役に立つケース②：共同作業したい



役に立つケース②：共同作業したい



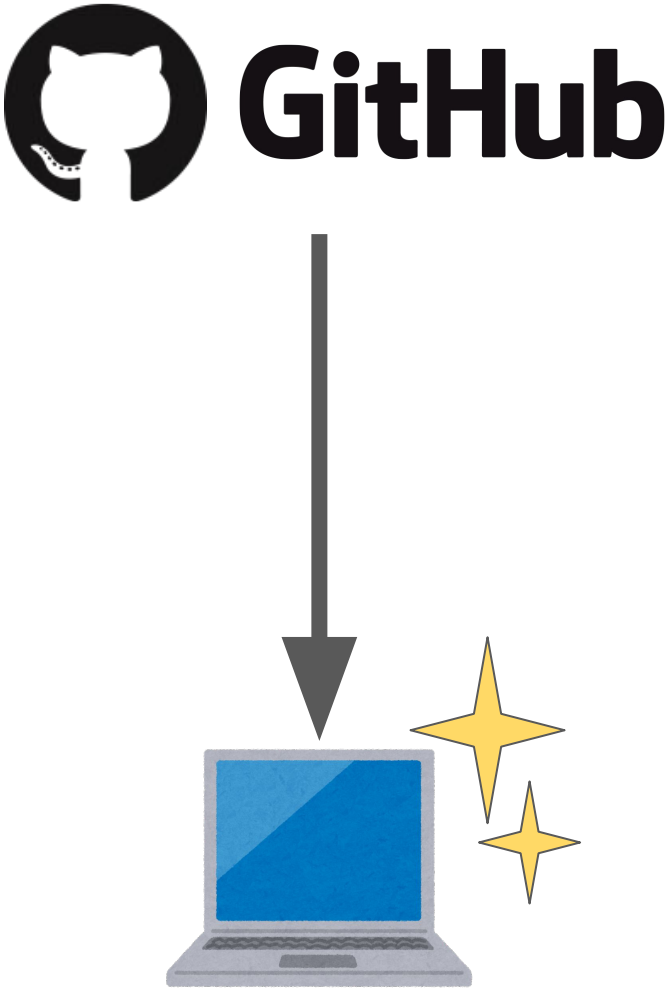
役に立つケース③：PC 壊れた

GitHub 上で管理していれば
並行作業していた内容も完全に復元できます.

Dropbox もバックアップの機能を担っているが
並行作業していた内容は復元できない

（ただし GitHub はサイズの大きいファイルを管理できない）

クローン



プロジェクト（→リポジトリ）を

GitHub から新規ダウンロードする操作

（何度もするような操作ではない）

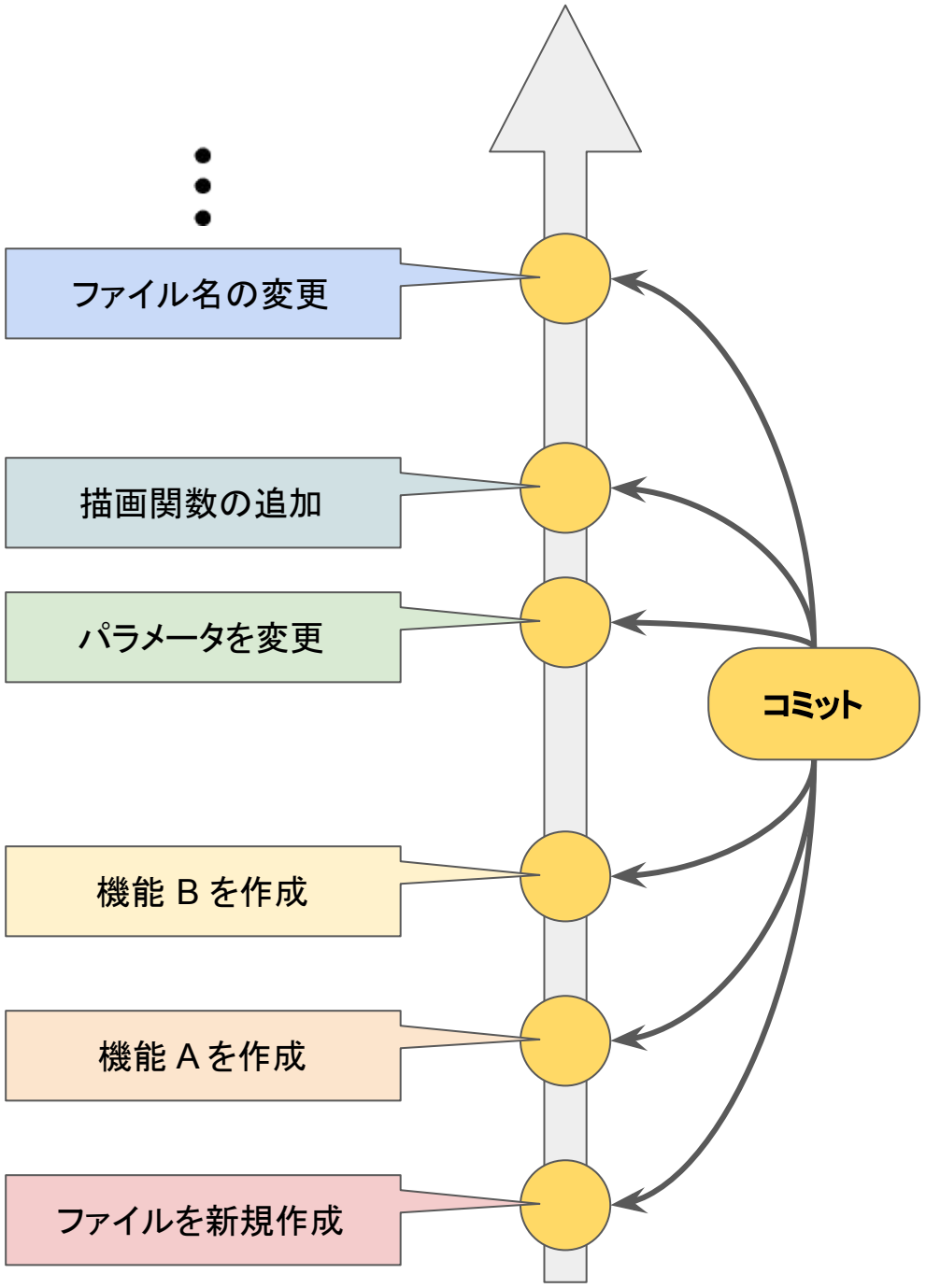
コミット

Git における「**ファイル変更の単位**」を
積み上げる操作

「歴史の積み上げ」とも言うべきか

吹き出しの内容を
「コミットメッセージ」をという

コミットの粒度は人それぞれ
（田中の基準は後述）

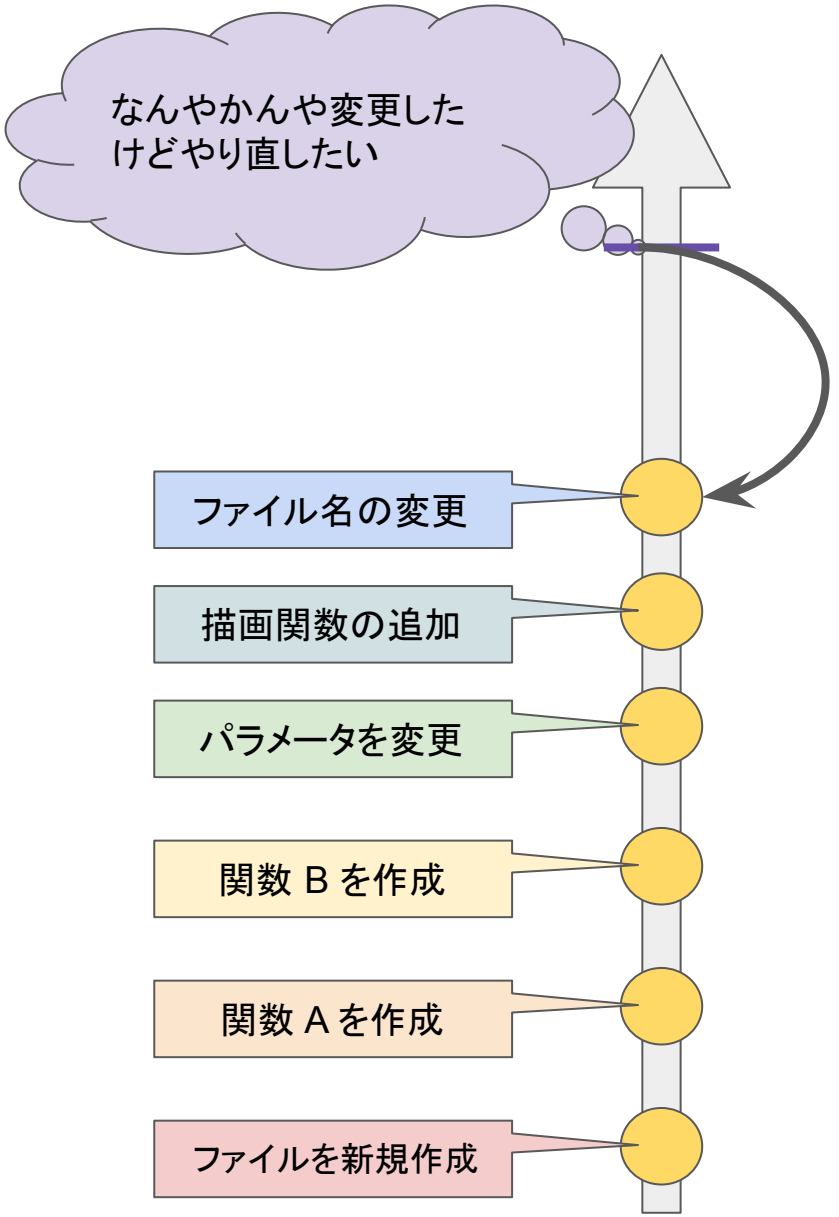


チェックアウト①（リストア）

コミットやブランチ（後述）への移動

役に立つケース①で使える

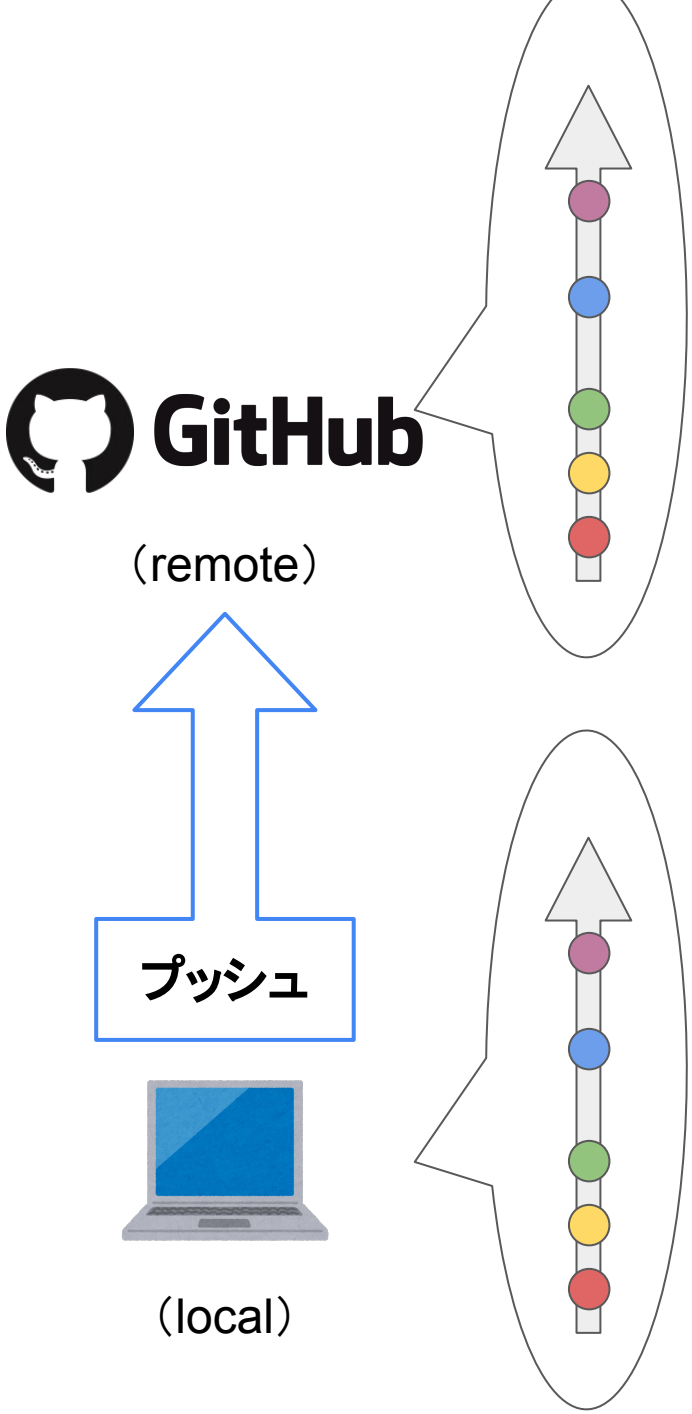
指定したコミットの状態に移動する



プッシュ

手元（= local）の変更内容を

GitHub （= remote）に適応する

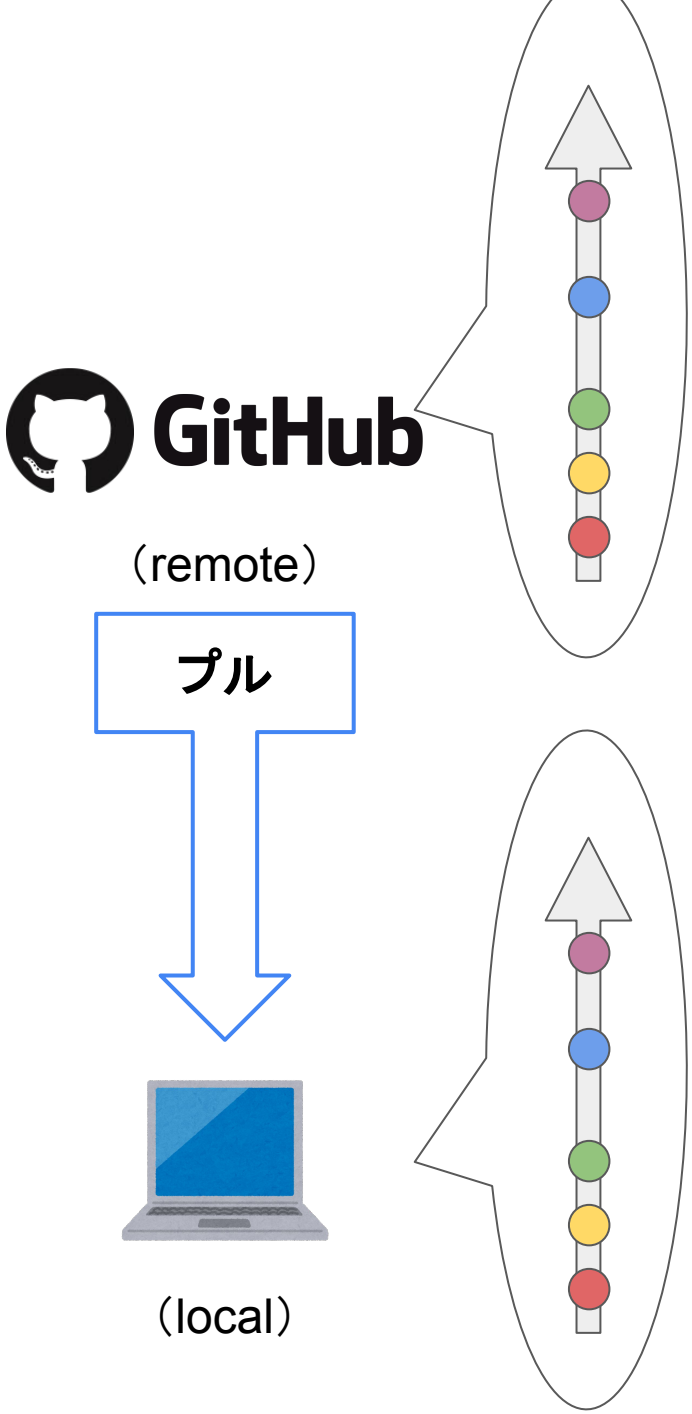


プル

GitHub（= remote）の変更内容を

手元（= local）に適応する

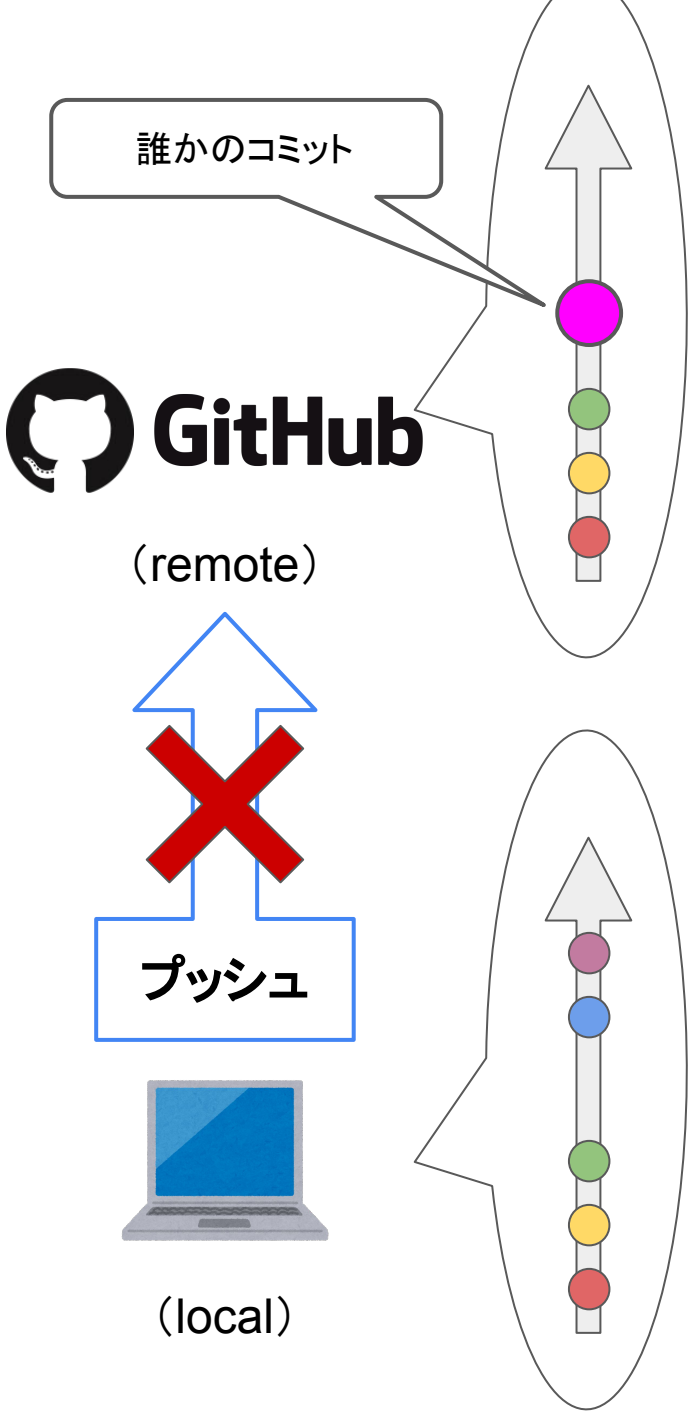
プッシュと真逆の操作



プッシュが拒否されるケース

役に立つケース②のような
上書きが発生する可能性があるとき
プッシュが拒否されることがある

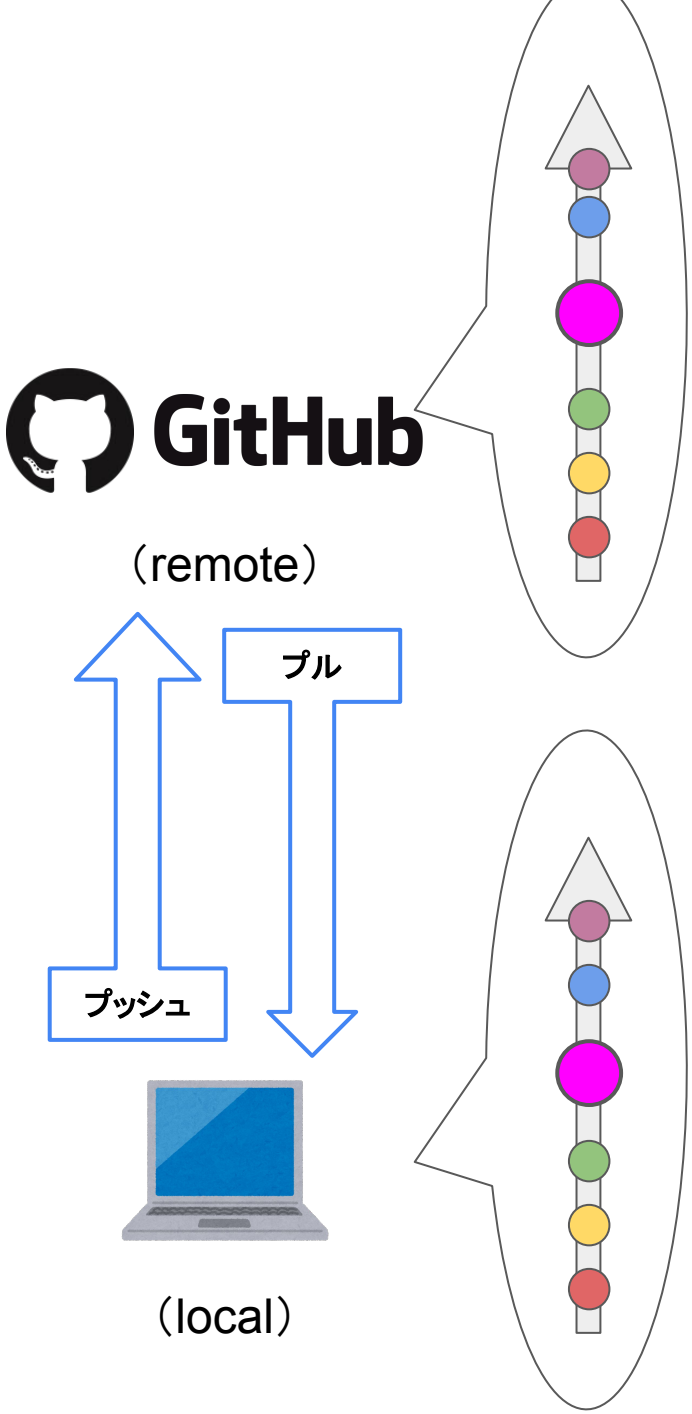
プッシュする前に
プルして remote の変更を取り込む



解決方法：先にプルする

役に立つケース②のような
上書きが発生する可能性があるとき
プッシュが拒否されることがある

プッシュする前に
プルして remote の変更を取り込む

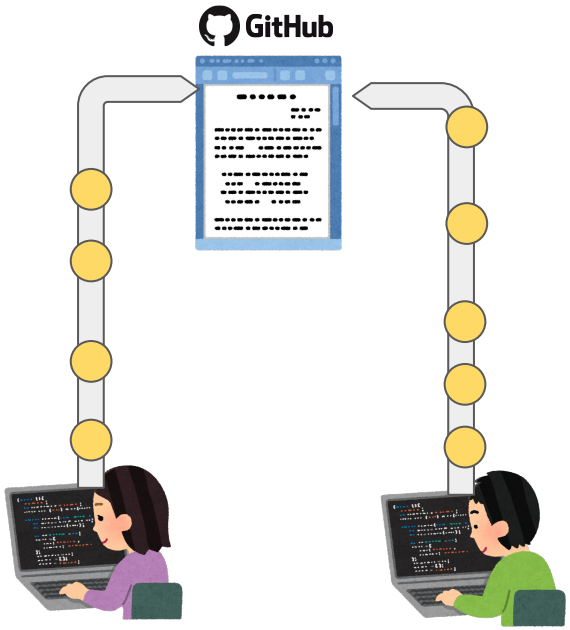


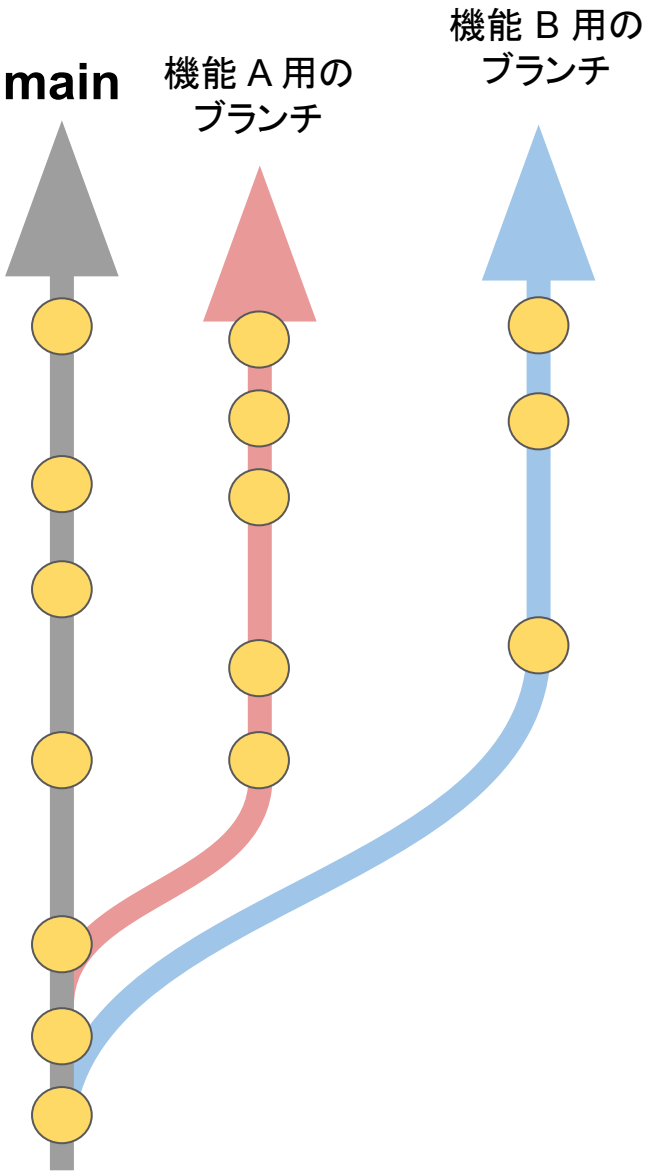
ブランチ

並行作業するのに使用する

開発の「世界線」

作業単位で分岐することが多い
（コミットは変更単位）





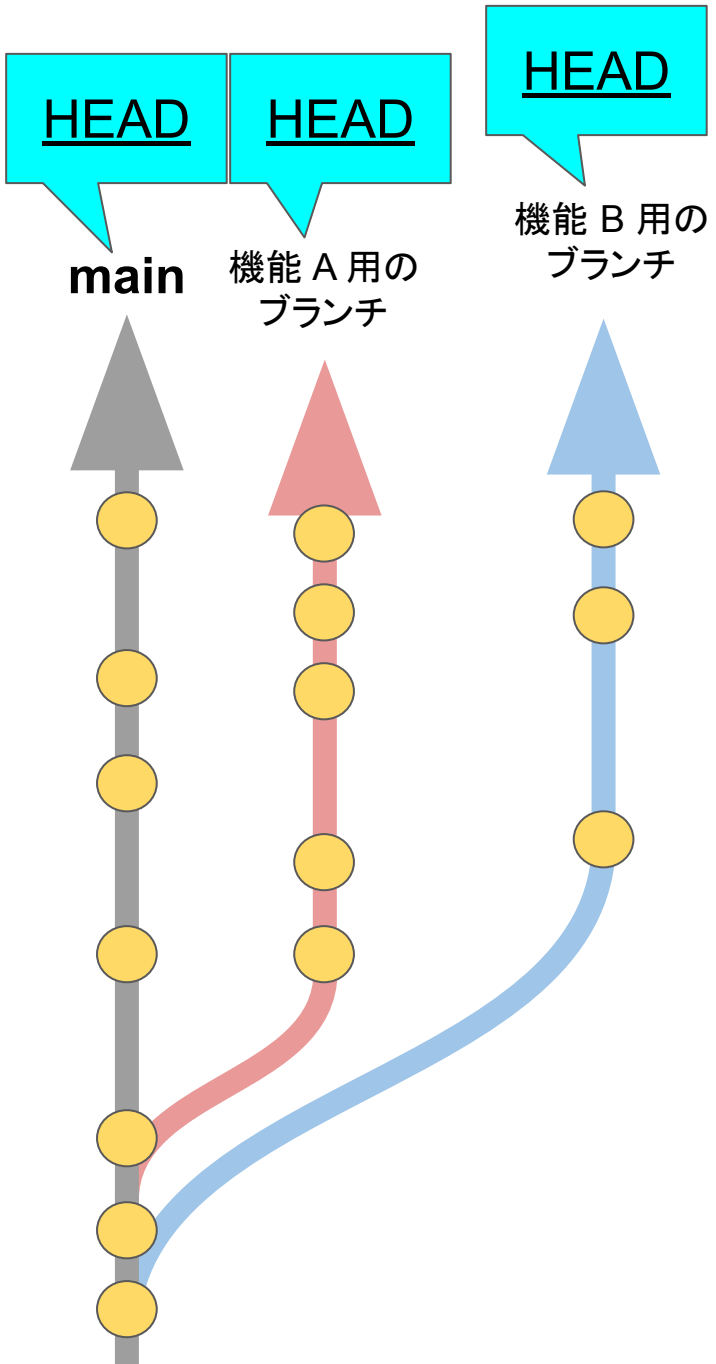
チェックアウト②（スイッチ）

ブランチの移動も

チェックアウトと言います.

（ git の中で）

現在地のことを「HEAD」と呼びます.



マージ

分岐したブランチを統合する操作

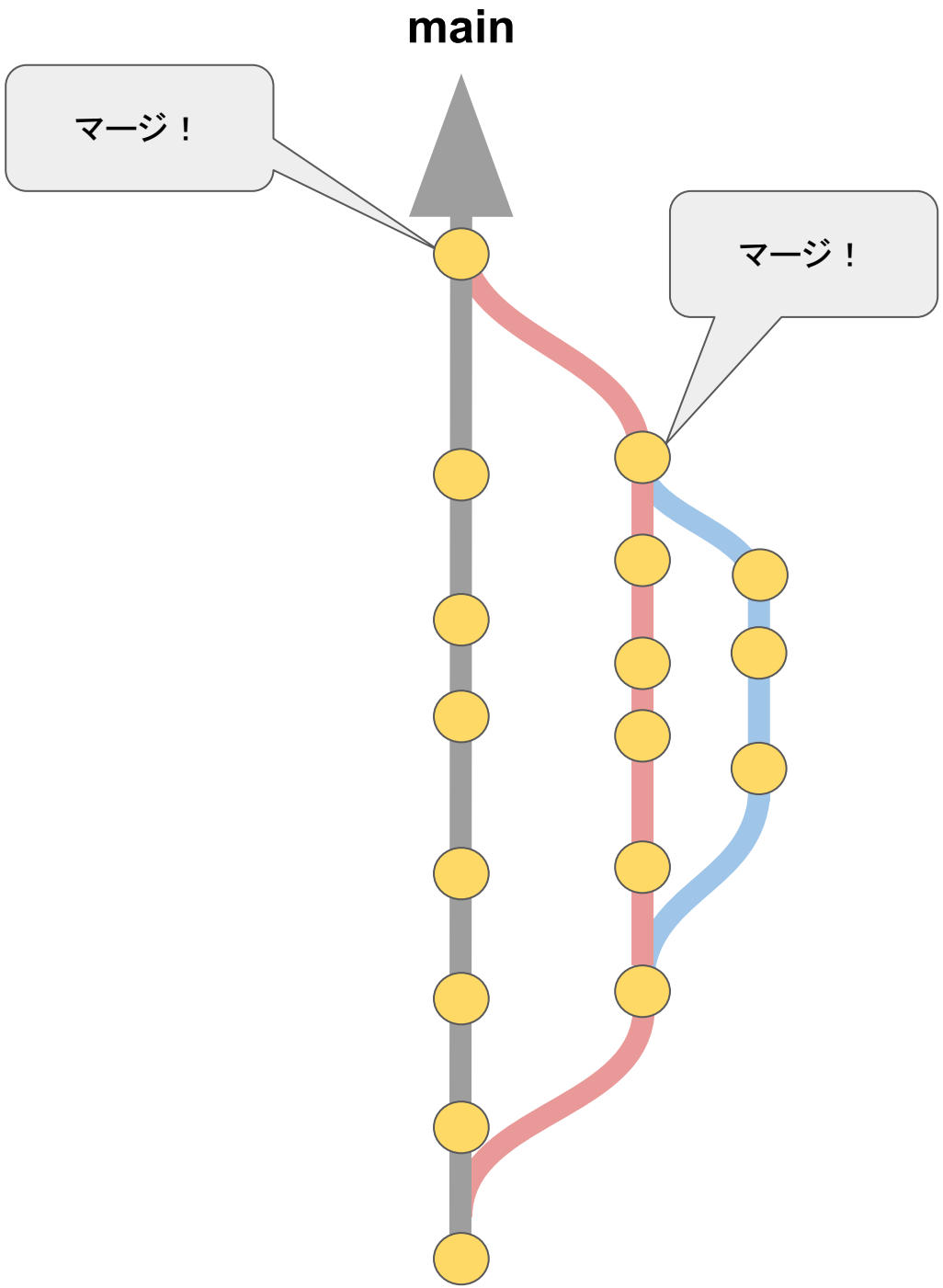
GitHub 上で

「プルリクエスト」として

マージするのが好ましい（後述）

基本的に

生やした元にマージする



まとめ

コマンド	内容
クローン	GitHub から新規ダウンロードする操作
コミット	ファイル変更の単位を積み上げる操作
チェックアウト	コミットやブランチへの移動
プッシュ	手元（= local）の変更内容をGitHub （= remote）に適応する
プル	GitHub （= remote）の変更内容を手元（= local）に適応する
ブランチ	並行作業するのに使用する開発の「世界線」
マージ	分岐したブランチを統合する操作

テスト①

コマンド	内容
クローン	
コミット	
チェックアウト	
プッシュ	
プル	
ブランチ	
マージ	

テスト②

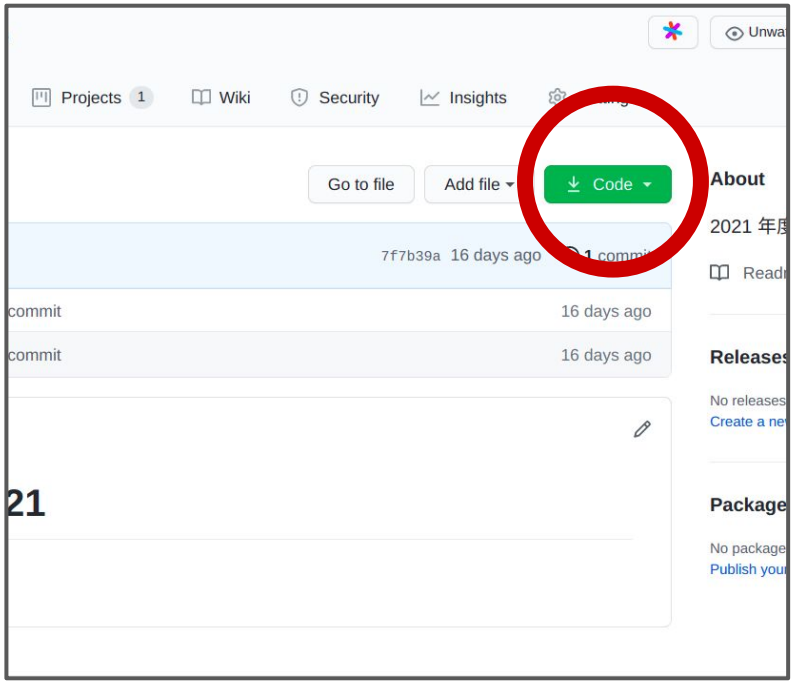
コマンド	内容
	コミットやブランチへの移動
	手元（= local）の変更内容をGitHub （= remote）に適応する
	分岐したブランチを統合する操作
	GitHub （= remote）の変更内容を手元（= local）に適応する
	並行作業するのに使用する開発の「世界線」
	ファイル変更の単位を積み上げる操作
	GitHub から新規ダウンロードする操作

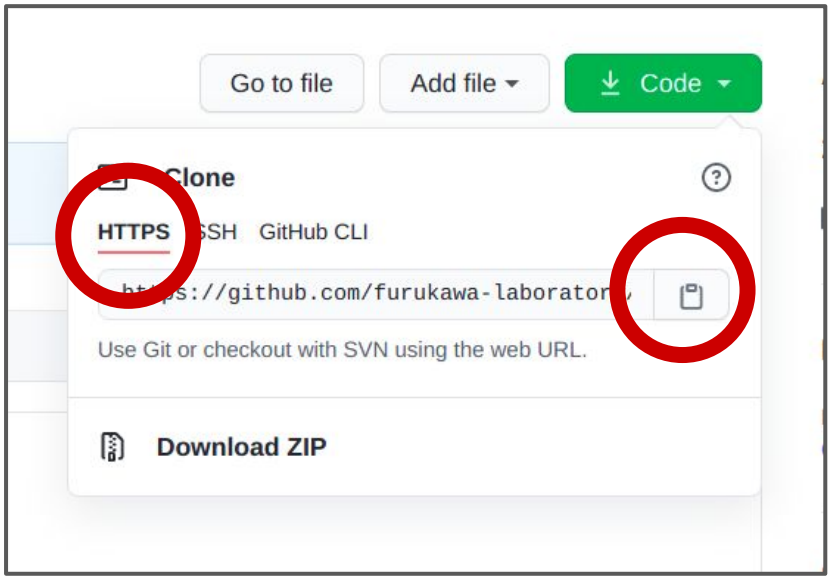
練習パート

- コミット & プッシュ
- ブランチ切って
- コンフリクト解消

準備:クローン

https://github.com/furukawa-laboratory/rookies_workout_2021





3. `git clone <貼付け>`

練習①：別々のファイルで

コミット，プッシュの練習

練習②：同じファイルを編集してみよう

1. 先にプル
2. 先にコミット（コンフリクトの解消）

練習③:ブランチを切って編集しよう

1. ブランチ切る
2. 同じファイルを編集
3. プルリク出す

練習④：Approve →Merge

セルフマージ：自分で Pull Request 出して自分でマージ

Pull Request はコンセンサスを取る場でもあるため
セルフマージはなるべく避ける

ブランチに protect をかけてセルフマージをさせない設定にできる

Approve：「マージしても良いよ！」の合図

③と同じことをやってみよう

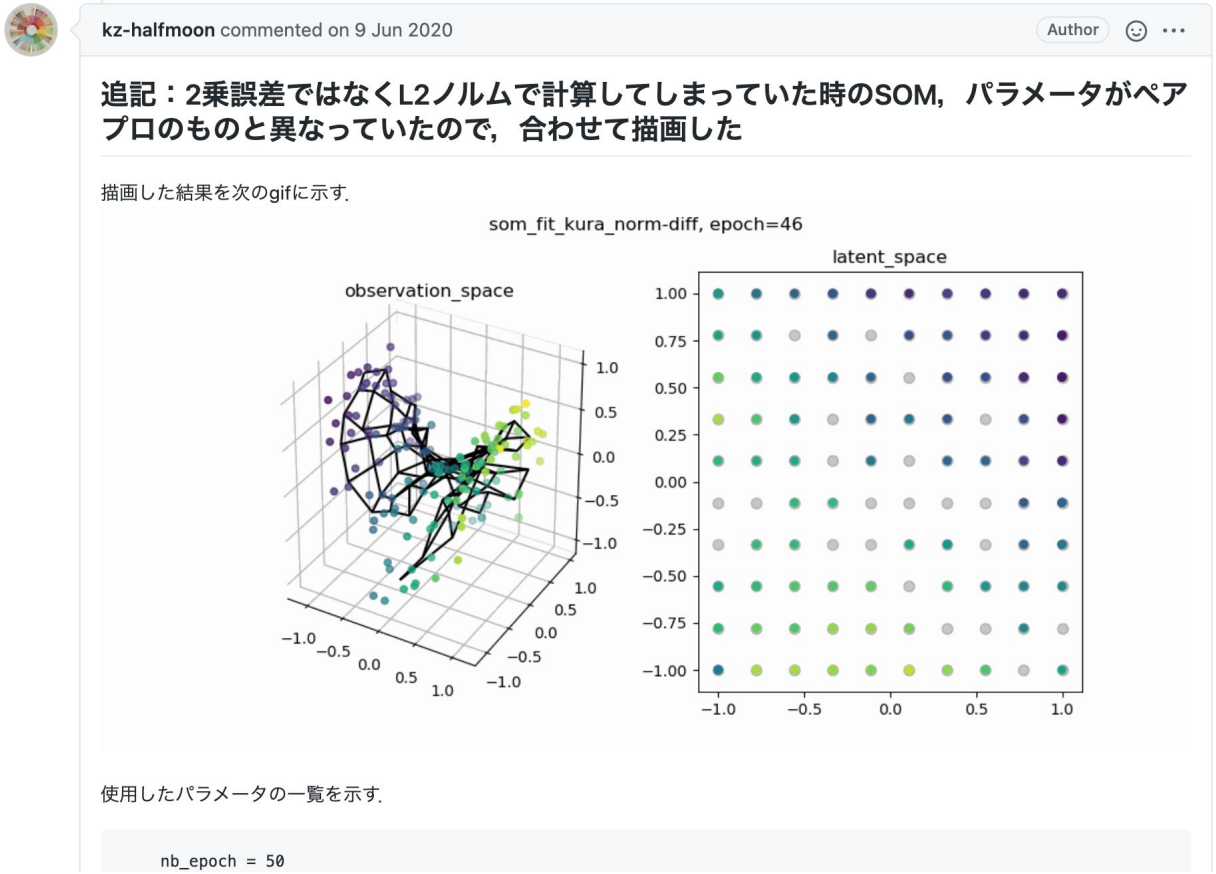
おまけパート

- GitHub の便利機能
- コミットの粒度
- コミットメッセージ
- Git の設定ファイル

issues：論題提起・議論の場

https://github.com/furukawa-laboratory/somf/issues?q=is%3Aissue

図を載せることも可能



kz-halfmoon commented on 9 Jun 2020 Author

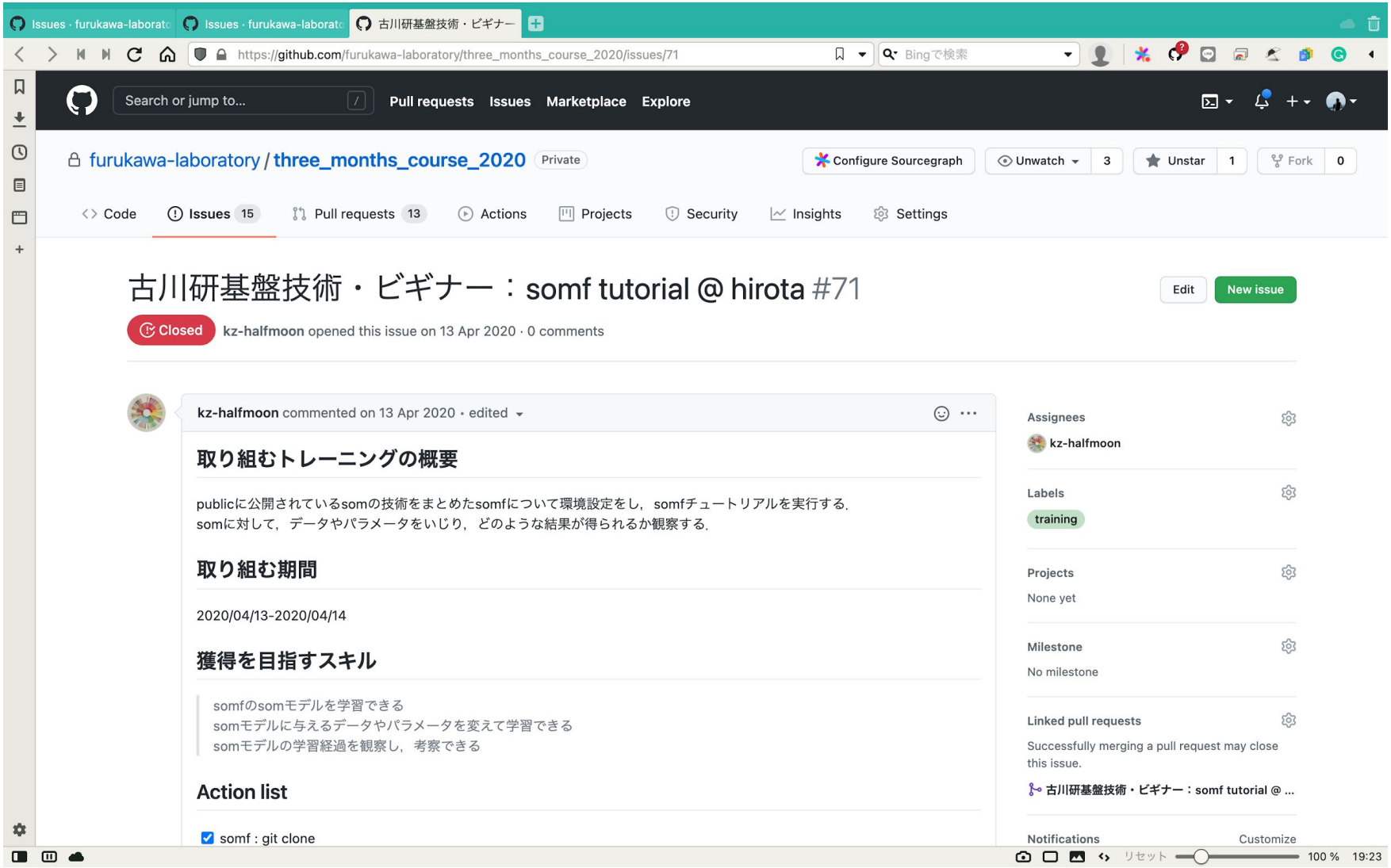
追記：2乗誤差ではなくL2ノルムで計算してしまっていた時のSOM，パラメータがペアプロのものと異なっていたので，合わせて描画した

描画した結果を次のgifに示す.

使用したパラメータの一覧を示す.

```
nb_epoch = 50
```

GitHub の便利機能



ae14watanabe requested changes on 4 Jun 2020 View changes

ae14watanabe left a comment

@kz-halfmoon 手元で動かしてみたら間違ってるところ見つけました。多分これでまともに動くんやないかな。PyCharmのデバッグ機能使うとデバッグしやすいのでおすすめ。

🚀 1

Lecture_Core_technic_flab/Lecture_SOM/hirota/ordinary/som_hirota.py Outdated

```
57  +            #alg(3)Rn(t=epoch)の計算
58  +            for n in range(self.N):
59  +                for k in range(self.K):
60  +                    Rkn[n,k] = np.exp(-(np.linalg.norm(self.Z[n] - self.Zeta[k]))/2*sigma**2)
```

ae14watanabe on 4 Jun 2020

Suggested change

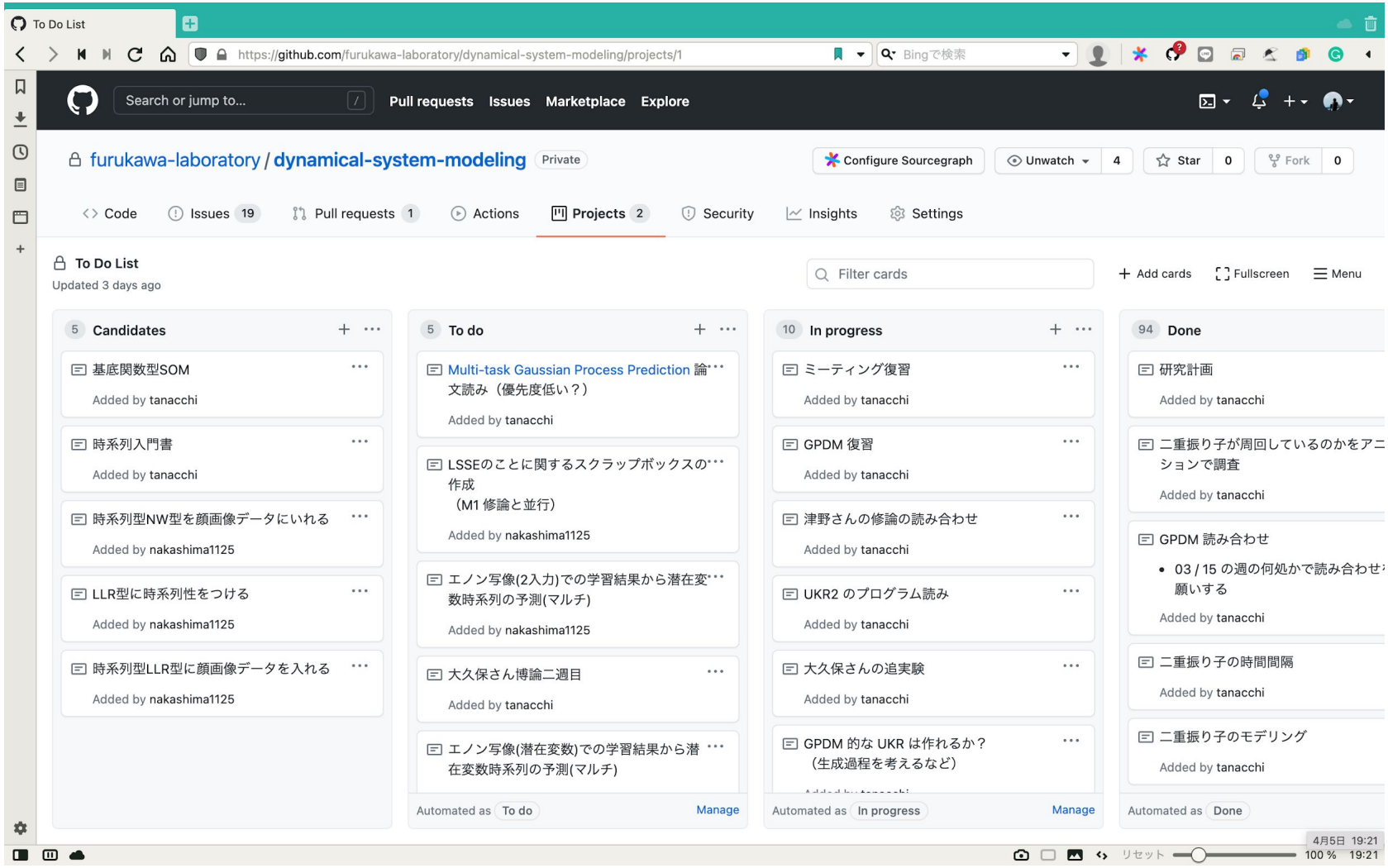
```
-                   Rkn[n,k] = np.exp(-(np.linalg.norm(self.Z[n] - self.Zeta[k]))/2*sigma**2)
+                   Rkn[n,k] = np.exp(-(np.linalg.norm(self.Z[n] - self.Zeta[k])) / ( 2 * (sigma**2) ) )
```

Commit suggestion

pythonの計算の順序的にこうやね
https://www.javadrive.jp/python/num/index3.html

ae14watanabe on 4 Jun 2020

```
1 / 2 * 3 ** 2
= 1 / 2 * 9   # 累乗が先
= 0.5 * 9     # /と*は優先順位が同じなので左の方から評価
= 4.5
```


To Do List
https://github.com/furukawa-laboratory/dynamical-system-modeling/projects/1
Search or jump to...
Pull requests
Issues
Marketplace
Explore
furukawa-laboratory / dynamical-system-modeling
Private
Configure Sourcegraph
Unwatch 4
Star 0
Fork 0
Code
Issues 19
Pull requests 1
Actions
Projects 2
Security
Insights
Settings
To Do List
Updated 3 days ago
Filter cards
Add cards
Fullscreen
Menu
5 Candidates
基底関数型SOM
Added by tanacchi
時系列入門書
Added by tanacchi
時系列型NW型を顔画像データにいれる
Added by nakashima1125
LLR型に時系列性をつける
Added by nakashima1125
時系列型LLR型に顔画像データを入れる
Added by nakashima1125
5 To do
Multi-task Gaussian Process Prediction 論文読み（優先度低い？）
Added by tanacchi
LSSEのことに関するスクラップボックスの作成
（M1 修論と並行）
Added by nakashima1125
エノン写像(2入力)での学習結果から潜在変数時系列の予測(マルチ)
Added by nakashima1125
大久保さん博論二週目
Added by tanacchi
エノン写像(潜在変数)での学習結果から潜在変数時系列の予測(マルチ)
Automated as To do
Manage
10 In progress
ミーティング復習
Added by tanacchi
GPDM 復習
Added by tanacchi
津野さんの修論の読み合わせ
Added by tanacchi
UKR2 のプログラム読み
Added by tanacchi
大久保さんの追実験
Added by tanacchi
GPDM 的な UKR は作れるか？
（生成過程を考えるなど）
Automated as In progress
Manage
94 Done
研究計画
Added by tanacchi
二重振り子が周回しているのかをアニションで調査
Added by tanacchi
GPDM 読み合わせ
03 / 15 の週の何処かで読み合わせ願いする
Added by tanacchi
二重振り子の時間間隔
Added by tanacchi
二重振り子のモデリング
Added by tanacchi
Automated as Done
4月5日 19:21
リセット
100 %
19:21

コミットの粒度

ブランチ：作業単位

- UKR を実装する

コミット：変更単位

- ファイルを新規作成，UKR クラスを追加，学習結果を描画，…

意味をもたせるのが大事

コミットメッセージ

移動，やりなおし，チェックなどの際に

目印になります．

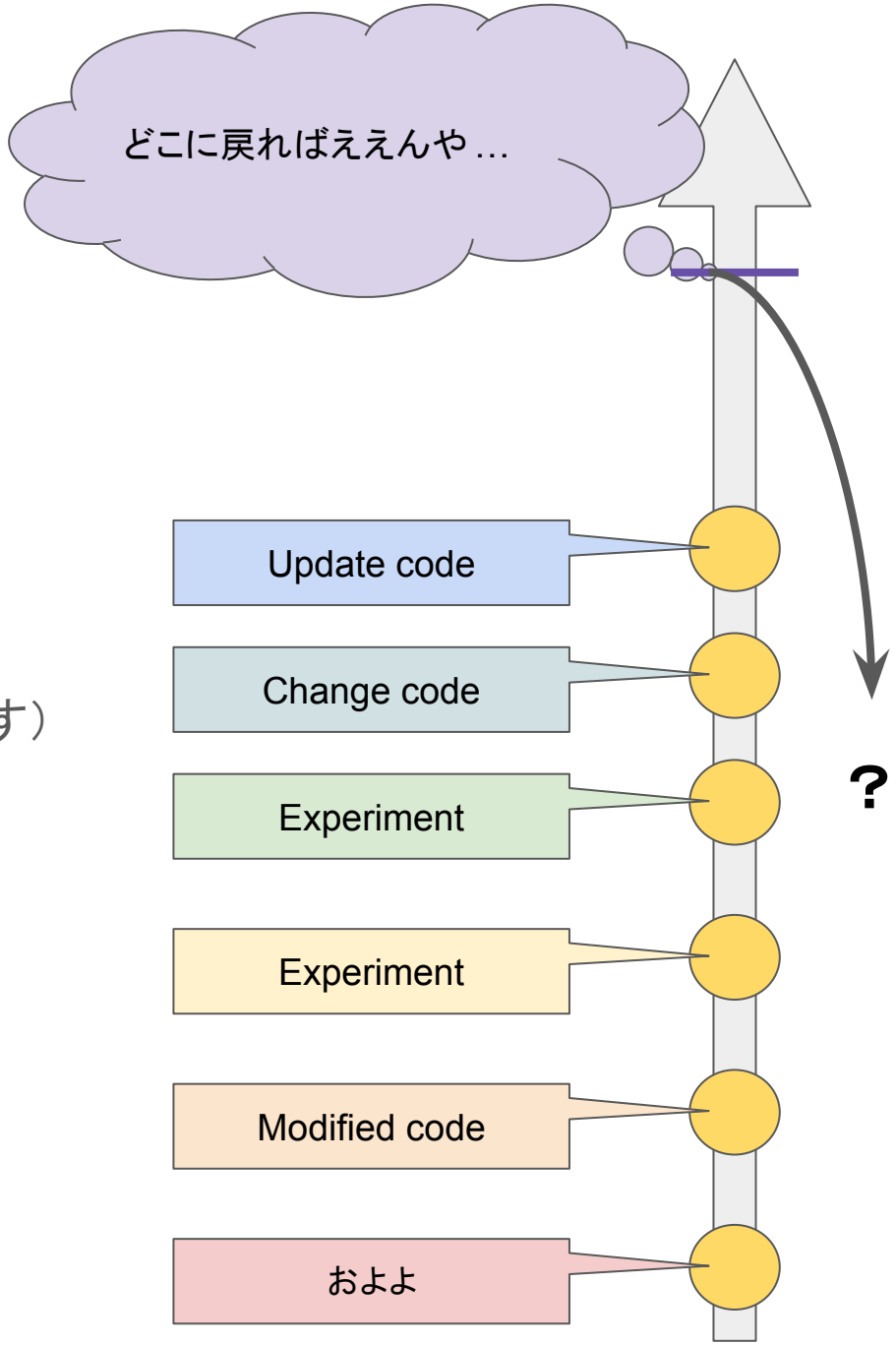
英語 / 日本語は個人の自由です．

（英語 & 先頭大文字のところが多そうです）

半年後の自分が読んでも

何をしたのかフワッとわかる程度に

書くと良さそう



Git の設定ファイル

ホームディレクトリ直下に

「.gitconfig」というファイルを置く

git status → git s

git branch → git br

git log --all --decorate --graph --oneline → git ga

ssh キーを設定しておくと
毎回パスワードを聞かれることがなくなって楽