GitHub を用いた開発フローの判別分析

プロジェクトマネジメントコース
ソフトウェア開発管理グループ
矢吹研究室
1242132
若月　純

謝辞

本研究を進めるにあたり，矢吹研究室矢吹太朗准教授には，多くの時間をご指導にさいて頂きました．また，先行研究を行った小野寺航己先輩をはじめ矢吹研究室の皆様には，多くの知識や示唆を頂きました．協力していただいた皆様に感謝の気持ちと御礼を申し上げます．

目次

第 1 章

序論

当研究は，GitHub を用いた開発フローの判別分析を行う．開発フローとは，開発の手順，ルールである．

GitHub を用いた開発フローは 13 種類ある．それぞれの開発フローには，メリット・デメリットがある．そのため，適切な開発フローを選択できる基準が必要である．

先行研究は，プロジェクトをリスクにより分類していた．そのため，選択基準が定性的になっていた．例を 1 つあげる．メンバのスキルが高く，大規模なプロジェクトの場合，フローが自動化されているイストフローが選択される [1].

そこで当研究は，定量的な選択基準を提供することを目指す．

GitHub 上のプロジェクトの性質を調査し，決定木分析を行う．そうすることで，同一の開発フローを選択しているプロジェクトの共通項を見つけることができる．共通項を見つけることで，開発フローを選択する基準を割り出せると考える．

第 2 章

背景

ソフトウェア開発では，複数のメンバが同時に開発する場合がある．そのため，様々な問題が発生する．課題をチーム間で適切に共有できず，進捗が見えにくくなったり，複数の人たちで 1 つの製品のソースコードを編集するため，開発内容が競合したりすることもあります．さらに複数の人たちが関わることによって，コードの質を均一化することも難しくなりますし，製品のコードの全容を把握することも難しくなります [2].

このような問題を解決するため，バージョン管理システムを用いる．バージョン管理システムとは，変更履歴を管理するシステムのことである．

バージョン管理システムを提供するサービスに，GitHub がある．GitHub は，バージョン管理システムに加え，branch，PullRequest といった開発を補助する機能を提供するサービスである．

GitHub を使用する手順を開発フローと呼ぶ．現在わかっている開発フローは 13 種類ある．それぞれ異なったリスクがあるため，最適な開発フローを選択する基準をリスクの面から分類する研究が行われた．研究の結果が表 2.1 である．プロジェクトの特徴から最適な開発フローが選択できるようになった．

表 2.1　プロジェクトと開発フロー出典：[1]p62

プロジェクトの特徴	開発フローの特徴	該当する開発フロー
メンバのスキルが高い 大規模なプロジェクト	フローが自動化されている	フィヨルドフロー イストフロー
参加メンバが多い	複数のリポジトリを使用する	Aming フロー サイボウズフロー 矢吹研フロー① 矢吹研フロー②
アジャイル型のソフトウェア開発	アジャイル開発のような流れが可能	Git フロー キャスレーフロー
GitHub を今まで導入した経験が無いプロジェクト	使用するブランチが少ない	GitHub フロー 日本 CAW フロー はてなブログフロー
ソフトウェアに何を盛り込むのかが明確	使用済みブランチを破棄	ラクスルフロー LINE フロー

しかし，メンバのスキルが高い，大規模なプロジェクト等，定性的な表現が多い．

そこで当研究では，選択する基準を人数が 5 人の場合，15 人の場合等，定量的に分けられるようにすることを目指す．そうすることで，既存の基準より，適切な開発フローを選択できるようになると考える．

第 3 章

目的

GitHub を用いたソフトウェア開発プロジェクトの性質において，適切な開発フローを選択できるようにするための基準を提供する．

第 4 章

プロジェクトマネジメントとの関連

ソフトウェア開発プロジェクトにおいて，プロジェクトマネージャーは，QCD を達成させるためにスケジュール，コスト，品質コントロールを行う．これらのコントロールを，GitHub を用いた開発フローを導入することで，補助することが出来る．

第 5 章

GitHub

GitHub の説明，GitHub 用語の説明を記述する．

5.1 GitHub

GitHub とは，Git リポジトリのホスティング機能をもつ．また，スローガンに SocialCoding をかかげているように，複数人で共同開発を行いやすくするための機能を提供している．機能とは，トラッキングや管理を行うための Issue，ソースコードの差分を論議するための Pull Request 等がある．

GitHub は 2008 年のサービス開始以来、年率 100 %という成長率で登録ユーザー数やレポジトリ数を伸ばしてきて、オープンソースの世界ではデファクトのプラットフォームとなっている．2015 年 6 月現在、GitHub の登録ユーザー数は 970 万、レポジトリ数は 2330 万．最近では Microsoft や Oracle といったトラディショナルな IT 企業も GitHub にレポジトリを用意するようになってきているし、広く知られたメジャーなオープンソース製品の多くが GitHub をプロジェクトのホスト先に選ぶのがここ数年のトレンドだ [3].

5.2 GitHub 用語

5.2.1 リポジトリ

ファイルやディレクトリの状態を記録する場所.

5.2.2 リモートリポジトリ

手元に置いてあるローカルなリポジトリ以外の，ネット上に置かれたリポジトリのこと．

5.2.3 commit

ファイルやディレクトリの変更をリポジトリに記録する機能である．

5.2.4 clone

ネット上にあるリポジトリをローカルにコピーする機能である．

5.2.5 Origin

clone 元のリモートリポジトリのこと．

5.2.6 Push

リモートリポジトリに自分の変更履歴がアップロードされ，リモートリポジトリ内の変更履歴がローカルリポジトリの変更履歴と同じ状態にする機能である．

5.2.7 branch

履歴の流れを分岐して記録していくためのもの．分岐したブランチは，他のブランチの影響を受けないため，同じリポジトリ中で複数の変更を同時に進めていける機能である．

5.2.8 pull

リモートリポジトリから最新の変更履歴をダウンロードしてきて，自分のローカルリポジトリにその内容を取り込む機能である．

5.2.9 Pull Request

相手に対して自分の変更をを pull してもらうように要求する機能である．

5.2.10 Revert

ステージングエリアに追加した変更を取り消す機能である．

5.2.11 **タグ**

コミットを参照しやすくするために，わかりやすい名前を付ける機能である．

5.2.12 Label

自由に作成でき，Issue をフィルタリングできる機能である．

5.2.13 Merge

当該ブランチに対して別のブランチの差分を取り込むことである．

5.2.14 Fork

GitHub のサービスで，相手のリポジトリを自分のリポジトリとしてコピー・保持できる機能ある．

5.2.15 Issue

ソフトウェア開発におけるバグや議論などをトラッキングして管理するために発行する．

5.2.16 **デプロイ**

ソフトウェアの分野で、開発したソフトウェアを利用できるように実際の運用環境に展開する．

5.2.17 **リリース**

プロセスを次の段階に進めることを認める機能である．

5.2.18 Watch

リポジトリに関する情報を Notifications に表示する機能である．

5.2.19 Star

リスト一覧からリポジトリを探すことが出来るようにする機能である．また，注目度を表す指標にもなる．

5.2.20 Fork

GitHub 側にある特定のリポジトリを自分のアカウント以下のリポジトリに複製する機能である．

5.2.21 **人数**

開発人数のことである．ここでは，Origin リポジトリにコミットした人数のことを示す．

5.2.22 MileStone

やるべきタスクの管理に Issue を用いることができるようにする機能である．

5.2.23 Wiki

簡単な記法によってドキュメントを作成，編集するための機能である．

第 6 章

GitHub を用いた開発フロー

GitHub を用いた開発フローを説明する．

6.1 GitHub フロー

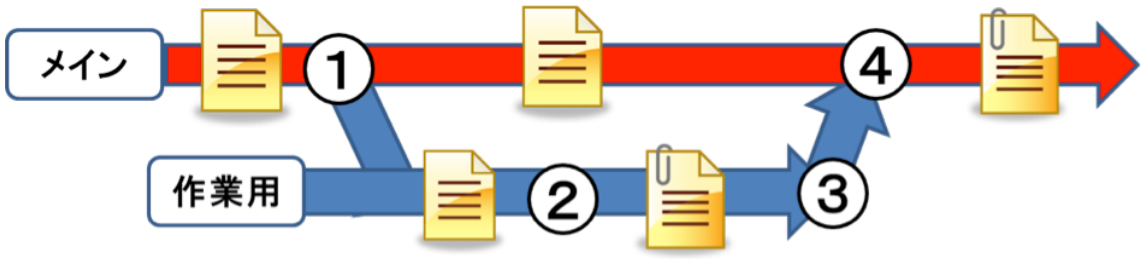


図 6.1 GitHub フロー図 出典：[1]p17

GitHub フローは GitHub 社が実践しているシンプルなワークフローである．基本的には特定の作業をするブランチを作成するだけなので，作業を始めてデプロイするまでの過程がとてもシンプルです．これはワークフローを実施するまでの学習コストを抑えられるという利点があります．さらに大きな利点として，シンプルであるからこと，多くの開発者がすばやく行うことを可能にします．そして，小さな変更などにも柔軟に対処できるようになります [4].

6.2　Git フロー

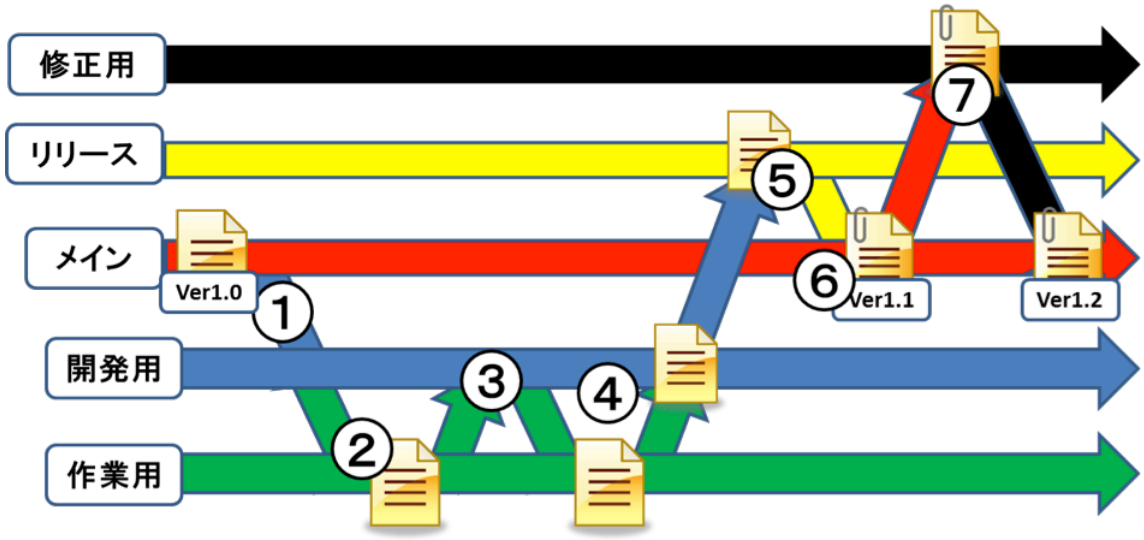


図 6.2　Git フロー図　出典：[1]p19

Git フローはリリース中心の開発スタイルである．この開発フローは，それぞれのブランチがコードの状態を表している．ソフトウェアのリリースを管理するリリースマネージャーなどが存在し，リリースを中心としたソフトウェア開発に向いている．しかし，覚えるブランチの状態が多く，開発フロー全体を事前に学習する必要がある．そのため，git-flow などのツールのサポートを受けることにより，強制的にフローからはずれない工夫が必要である [4].

6.3　Stable フロー

Stable フローは，GitHub フローをベースにしている．

特徴は，branch に stable branch を採用している点である．stable branch とは，安定状態の branch を作ることにより，バグが新たに入り込むリスクを減らす効果がある．stable branch は，できるだけ最新の master ブランチから分岐することで，バグを新たに入り込むリスクを減らせる．

6.4　日本 CAW Flow

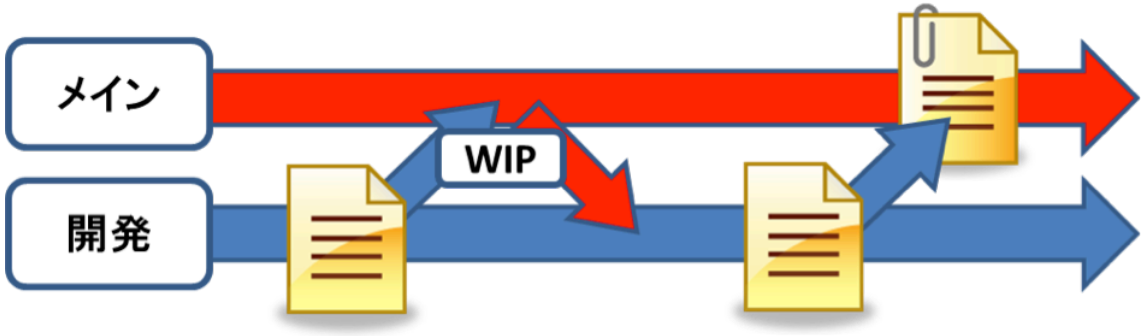


図 6.3　日本 CAW フロー図　出典：[1]p23

日本 CAW フローは，日本 CAW 株式会社が採用しているフローである．このフローは，GitHub Flow をベースにして Pull Request を活用したスタイルを採用している．特徴は，WIP PR を採用している点である．WIP PR(Work In Progress Pull Request) とは，実装の仕方や，コードの設計など，コードに紐づく議論をしやすくするために用いられる．Pull Request 作成時に頭に [WiP] を入れる．[WIP] がついた Pull Request はマージされず，議論や確認のために参照される．作成者は議論や確認が済んだら Close する，といったフローである [5].

6.5 LINE Flow

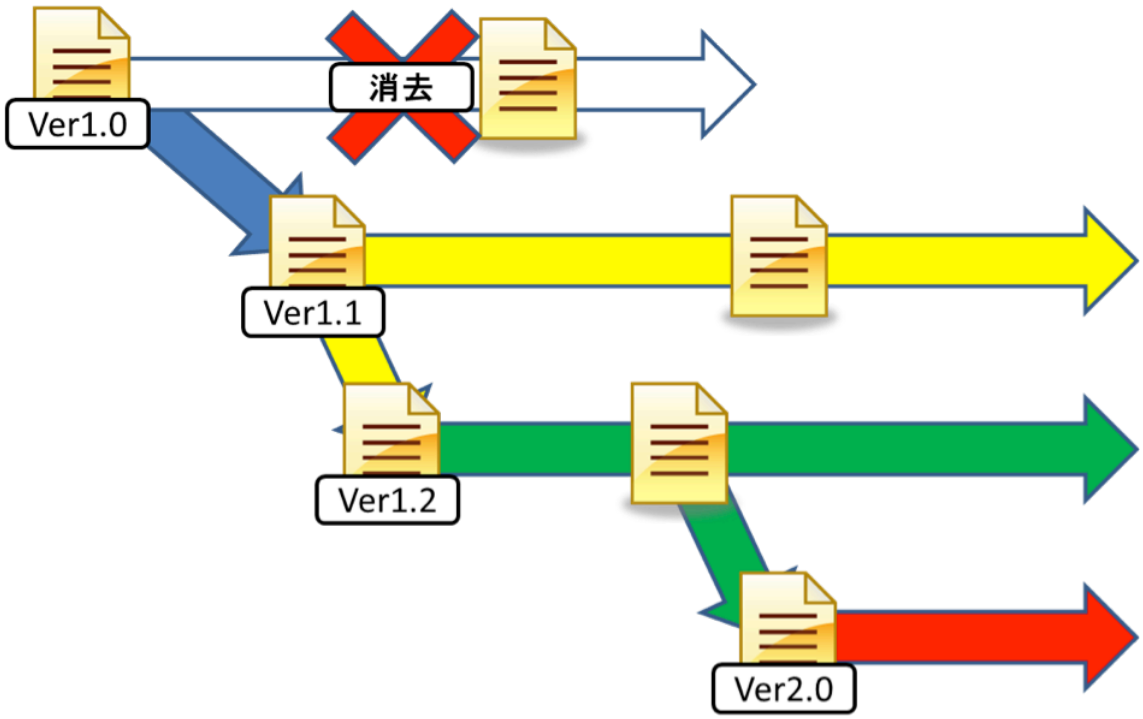


図 6.4 LINE フロー図 出典：[1]p31

LINE フローは Pull Request などを活用して行われる開発フローである．常に deploy 可能なバージョン毎の master を持っているよいう形で運用している．それぞれがリリース可能なブランチで，開発状況によっては下位のバージョンのコミットが上位のバージョンのコミットより新しい場合がある．LINE の IOS アプリではバージョン毎に提供する機能や修正を決めていて，QA 対象の切り分けがしやすいなどのメリットもあり，1 つの master に commit を繋げいく形では運用していない．現在開発中の最も下位のバージョンの修正がリリースされたら，そのブランチを上位のブランチにマージして下位のブランチを消していく流れで行われる [6].

第 7 章

手法

調査対象，調査方法，データの分析方法について記述する．環境は，OS X Yosemite　バージョン 15.15.5 である．

7.1　調査対象

調査対象データと調査対象プロジェクトを記述する．

7.1.1　調査対象データ

調査するデータは，Watch 数，Star 数，Fork 数，Commit 数，branch 数，Release 数，人数，言語，行数，ファイル数，バイト数，OpenIssues 数，ClosedIssues 数，Issues 数，OpenPull Request 数，ClosedPull Request 数，Pull Request 数，Label 数，OpenMileStone 数，ClosedMileStone 数，MileStone 数，Wiki 数，言語，開発フローである．

7.1.2　調査対象プロジェクト

調査するユーザ名とプロジェクト名を記載する．

表 7.1　調査対象プロジェクト

ユーザ名	リポジトリ名
zedapp	zed
LearnBoost	stylus
spine	spine
sirjuddington	SLADE
sass	sass
Polymer	polymer
play	play
ossec	ossec-hids
NTU-CCSP	ntu-ccsp.github.io
neovim	neovim
aol	moloch
rackerlabs	mimic
melonjs	melonJS
nasa	mct
waysome	libreset
knockout	knockout
KnpLabs	FriendlyContexts
freifunkMUC	freifunkmuc.github.io
admc	flex-pilot-x
dfm	emcee
adamhjk	dynect_rest
ashesi-SE	datasaver
pyca	cryptography
karpathy	convnetjs
tlycken	Contour.jl
ellisonleao	clumsy-bird
mozbrick	brick
tyoshii	bms
sampsyo	beets
openstf	adbkit
CyberAgent	android-gpuimage

7.2　調査方法

7.2.1　Watch 数，Star 数，Fork 数，Commit 数，branch 数，Release 数，人数の調査

調査は，手動で行う．

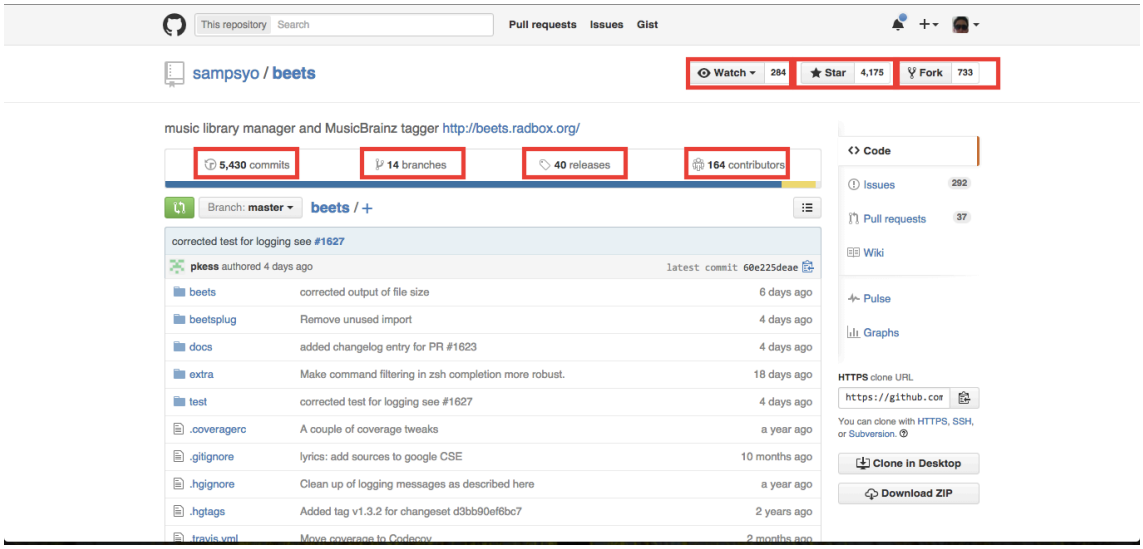


図 7.1　Watch 数，Star 数，Fork 数，Commit 数，branch 数，Release 数，人数の調査

図 7.1 は Watch 数，Star 数，Fork 数，Commit 数，branch 数，Release 数，人数の調査画面である．Commits が Commit 数，branches が branch 数，releases が Release 数，contributors が人数である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

7.2.2　言語の調査

調査は，手動で行う．

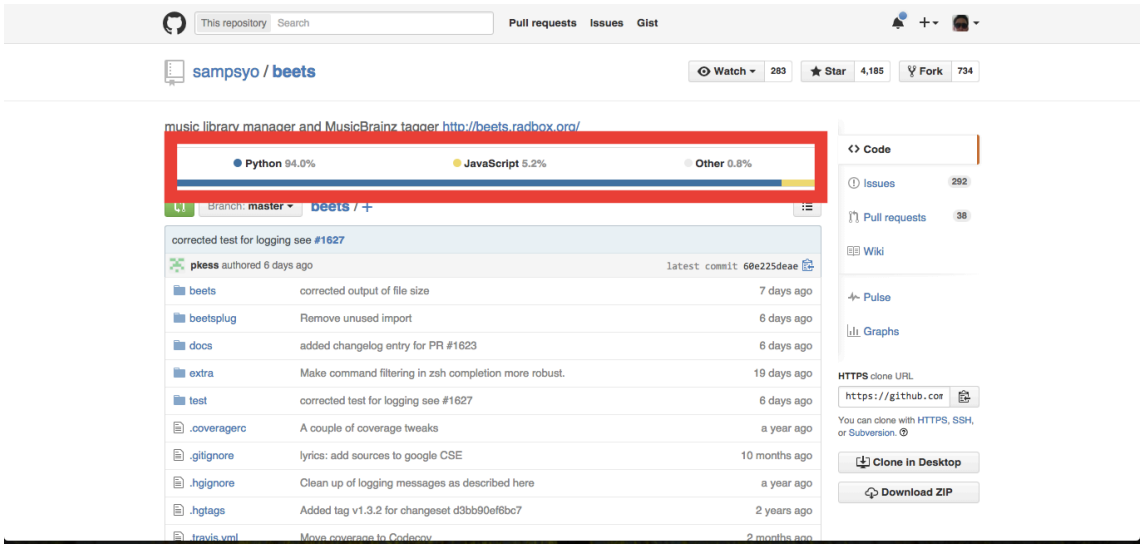


図 7.2　言語の調査

図 7.2 は言語の調査画面である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

7.2.3　OpenIssues 数，ClosedIssues 数，Issue 数の調査

調査は，手動で行う．

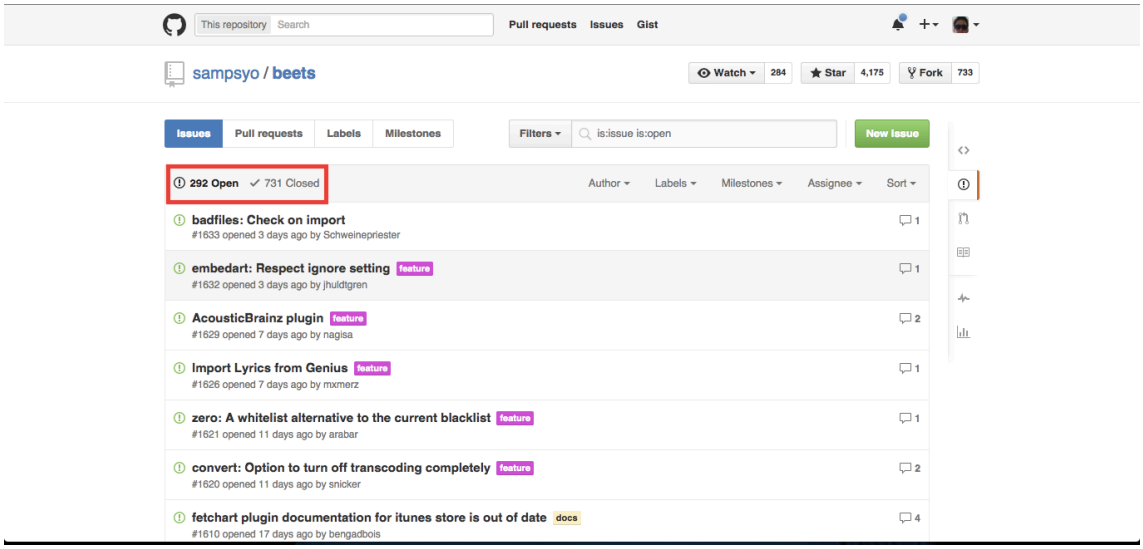


図 7.3　Issues 数の調査

図 7.3 は OpenIssues 数，ClosedIssues 数，Issue 数の調査画面である．Open が OpenIssues 数，Closed が ClosedIssues 数である．Open と Closed の合計が Issue 数である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

7.2.4　OpenPull Request 数，ClosedPull Request 数

調査は，手動で行う．

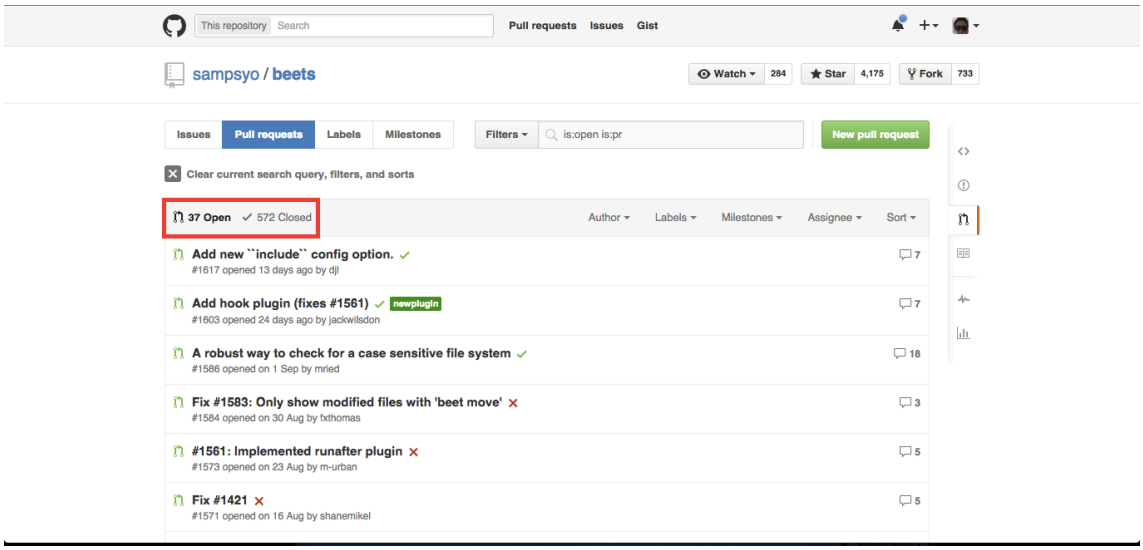


図 7.4　Pull Request 数の調査

図 7.4 は OpenPull Request 数，ClosedPull Request 数の調査画面である．Open が OpenPull Request 数，Closed が ClosedPull Request 数である．Open と Closed の合計が Pull Request 数である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

7.2.5　Label 数

調査は，手動で行う．

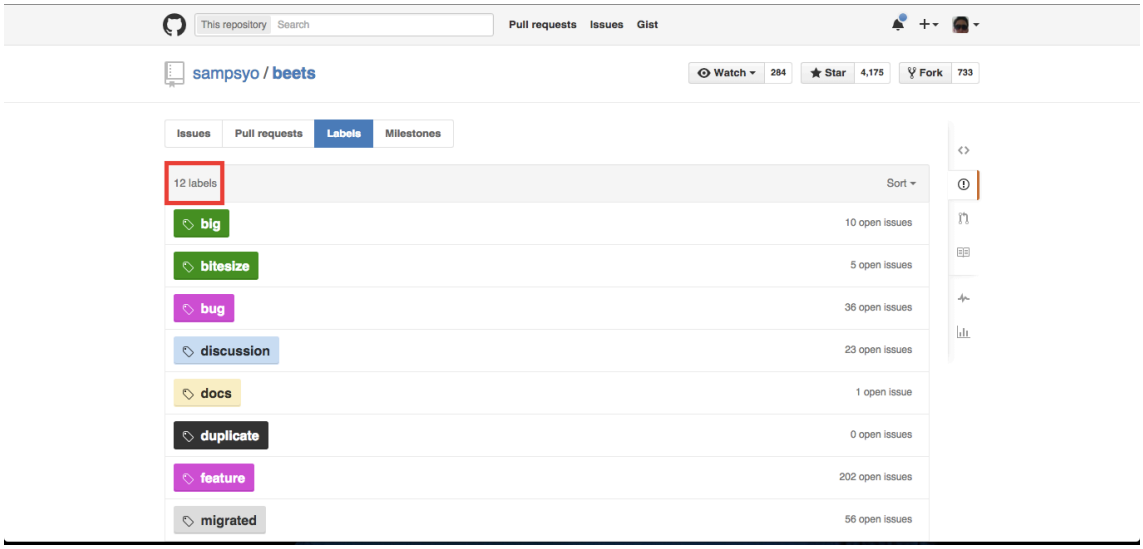


図 7.5　Label 数の調査

図 7.5 は Label 数の調査画面である．labels が Label 数である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

7.2.6　OpenMilestone 数，ClosedMilestone 数，Milestone 数

調査は，手動で行う．

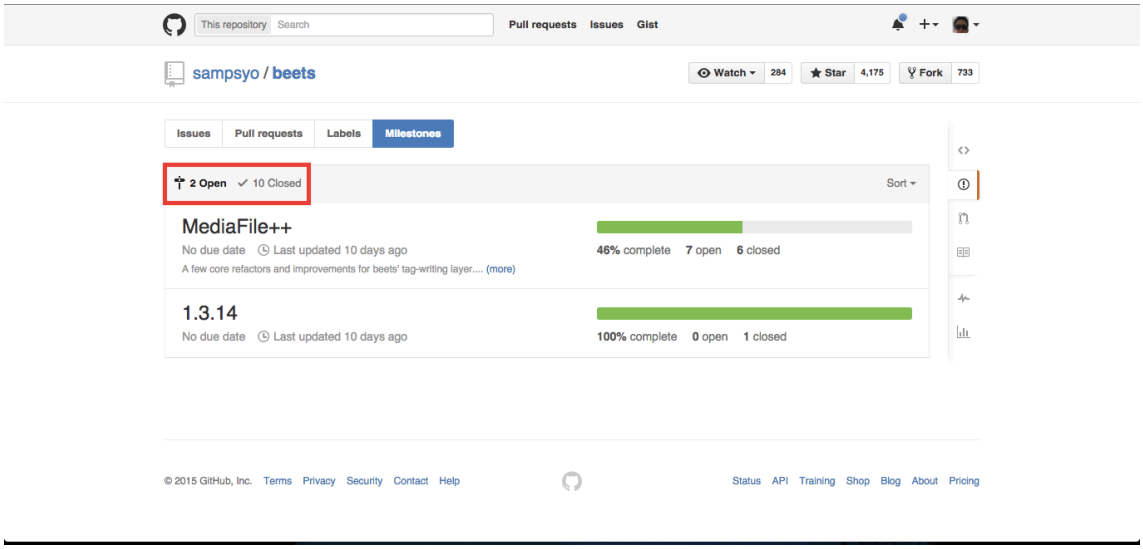


図 7.6　Milestone 数の調査

図 7.6 は OpenMilestone 数，ClosedMilestone 数，Milestone 数の調査画面である．Open が OpenMilestone 数，Closed が ClosedMilestone 数である．Open と Closed の合計が Milestone 数である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

7.2.7　Wiki 数

調査は，手動で行う．

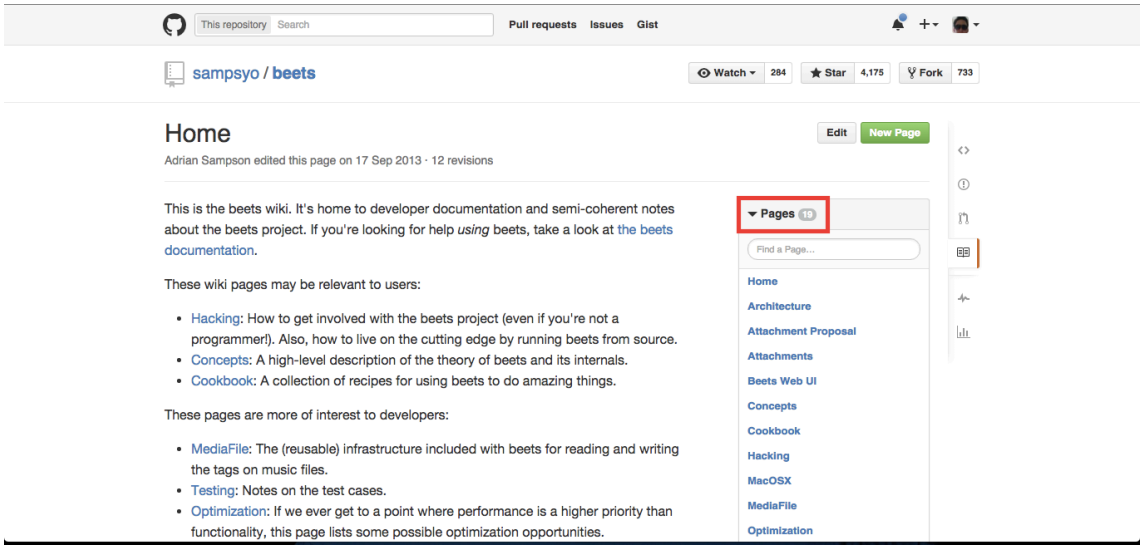


図 7.7　Wiki 数の調査

図 7.7 は Wiki 数の調査画面である．Pages の隣の数値が Wiki 数である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

7.2.8　ファイル数，バイト数

指定のリポジトリを clone する．

```
git clone git@github.com:アカウント名/リポジトリ名
```

clone したリポジトリの情報を確認する．右クリックし，「情報をみる」を選択する．選択すると，バイト数とファイル数が表示される．表示された数値を，"データ一覧.csv"に保存する．

7.2.9　行数

行数を出力する

指定のリポジトリを clone する．

```
git clone git@github.com:アカウント名/リポジトリ名
```

clone したリポジトリをターミナルで開く．ターミナルで，以下のコマンドをうつ．

```
grep -rI '' リポジトリ名 | wc -l
```

行数が出力される．
出力された行数を，"データ一覧.csv"に保存する．
コマンドの説明を以下に記述する．

grep

ファイルの中の文字列を検索する．

-r

各ディレクトリ下のすべてのファイルを再帰的に読み取る．

-I

バイナリファイルを無視する．

|

コマンドの標準出力を次のコマンドに渡す処理を行う．

wc

テキスト・ファイルの行数，単語数，バイト数を表示する．

-l

行数のみ集計し表示する．

7.2.10　開発フロー

開発フローの特徴とプロジェクトの特徴を照らし合わせる．

GitHub フロー

master branch から記述的な名前の branch がある場合は，GitHub フローである．

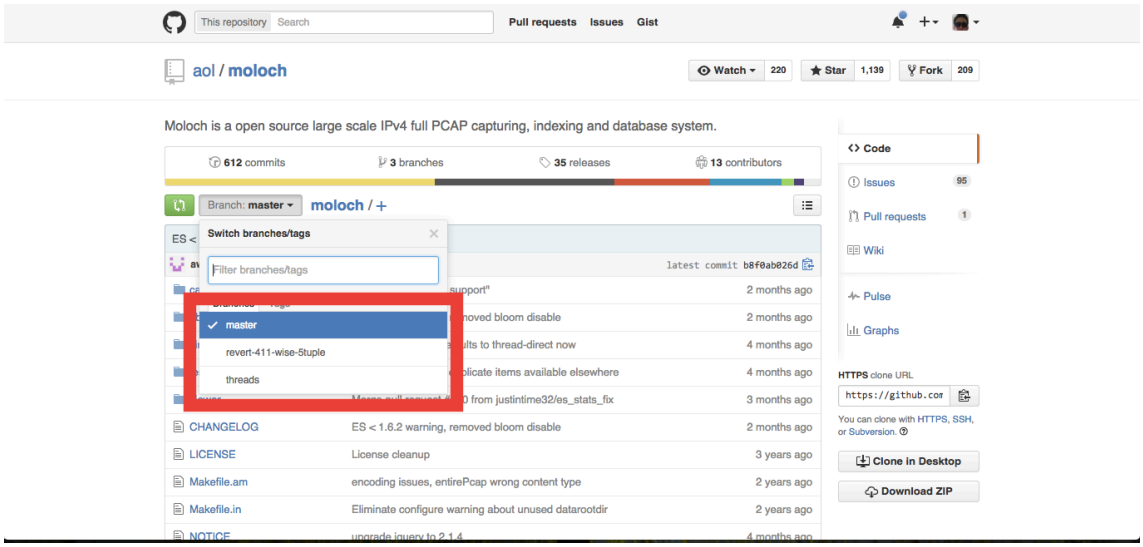


図 7.8　GitHub フローの調査

Git フロー

develop branch と release branch がある場合は，Git フローである．

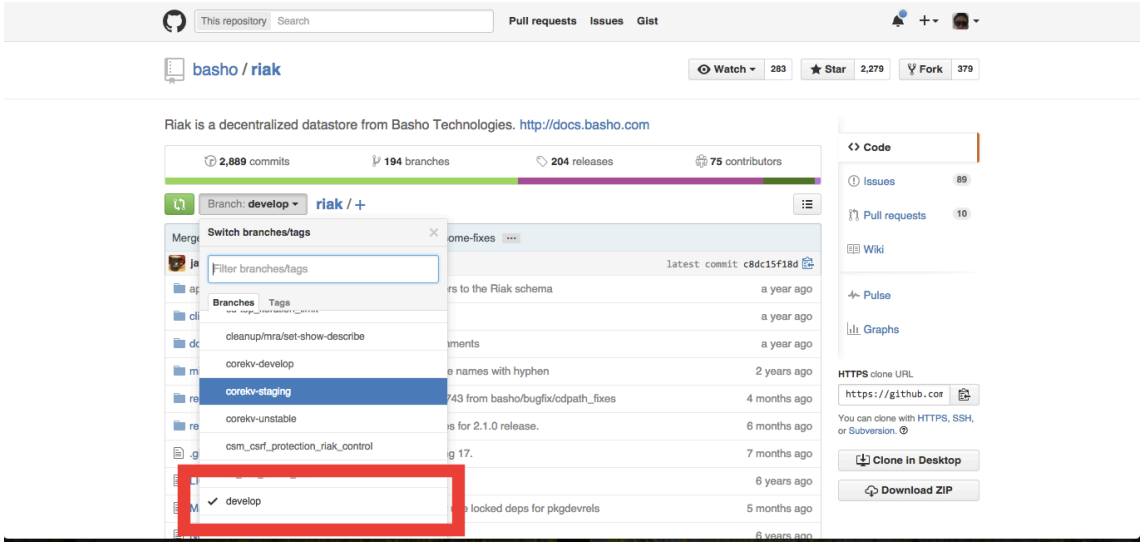


図 7.9　Git フローの調査

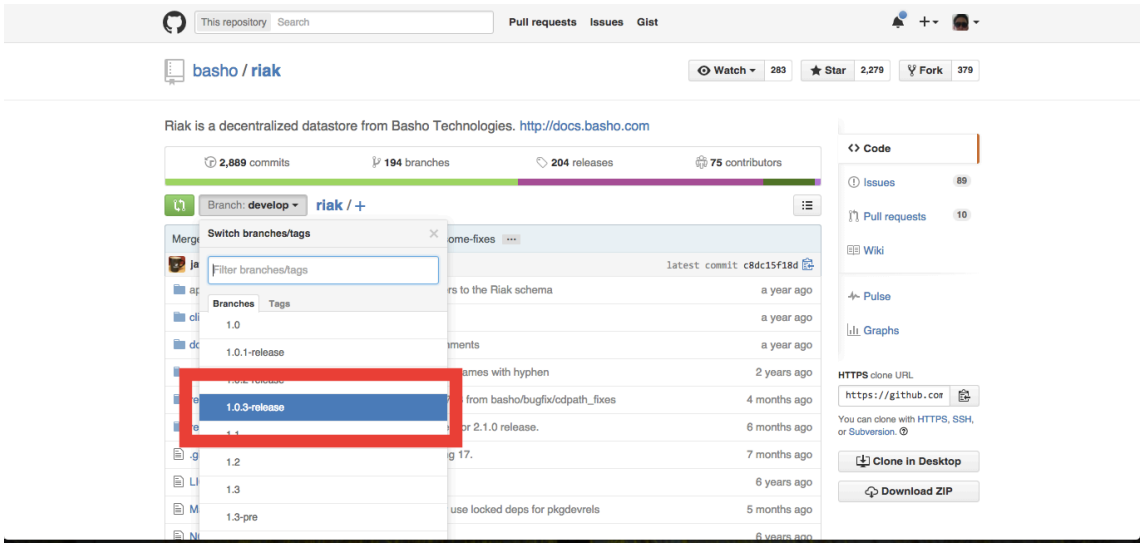


図 7.10　Git フローの調査

LINE フロー

バージョンごとに branch が作られている場合は，LINE フローである．

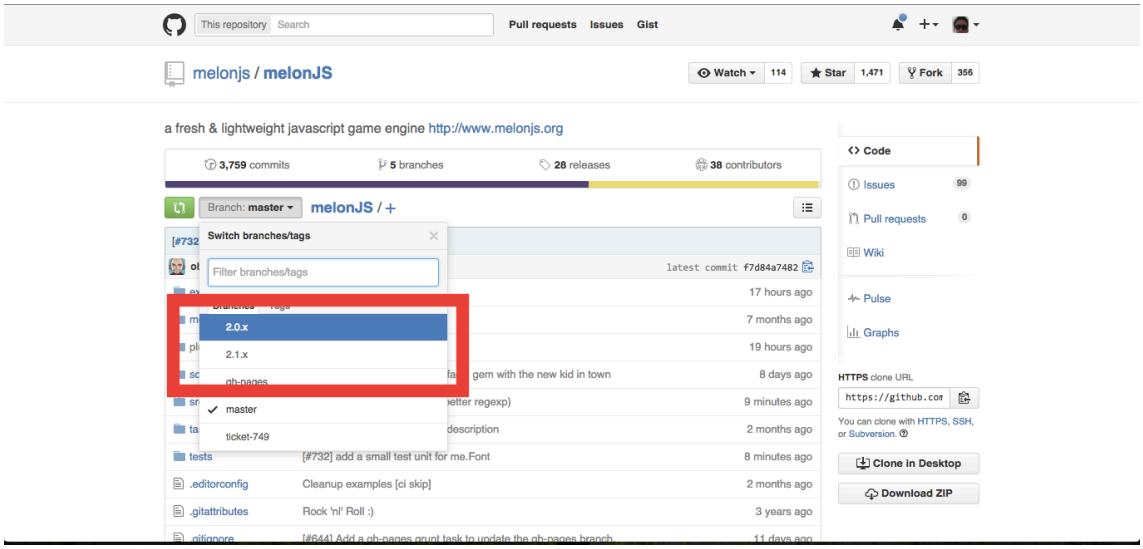


図 7.11　LINE フローの調査

日本 CAW フロー

Pull Request に [WIP] がある場合は，日本 CAW フローである．

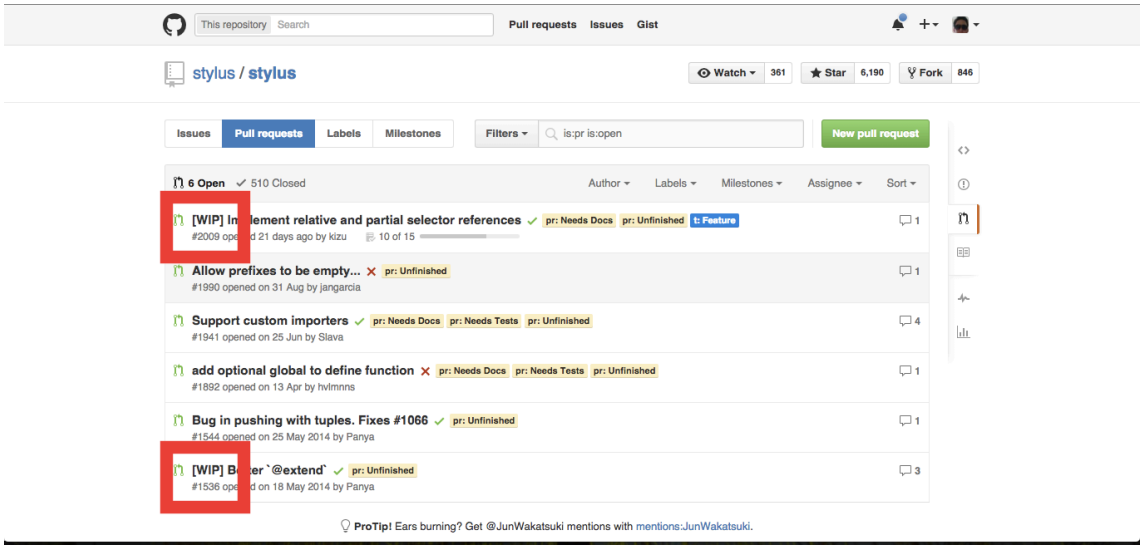


図 7.12　日本 CAW フローの調査

Stable フロー

branch に stable がある場合は，Stable フローである．

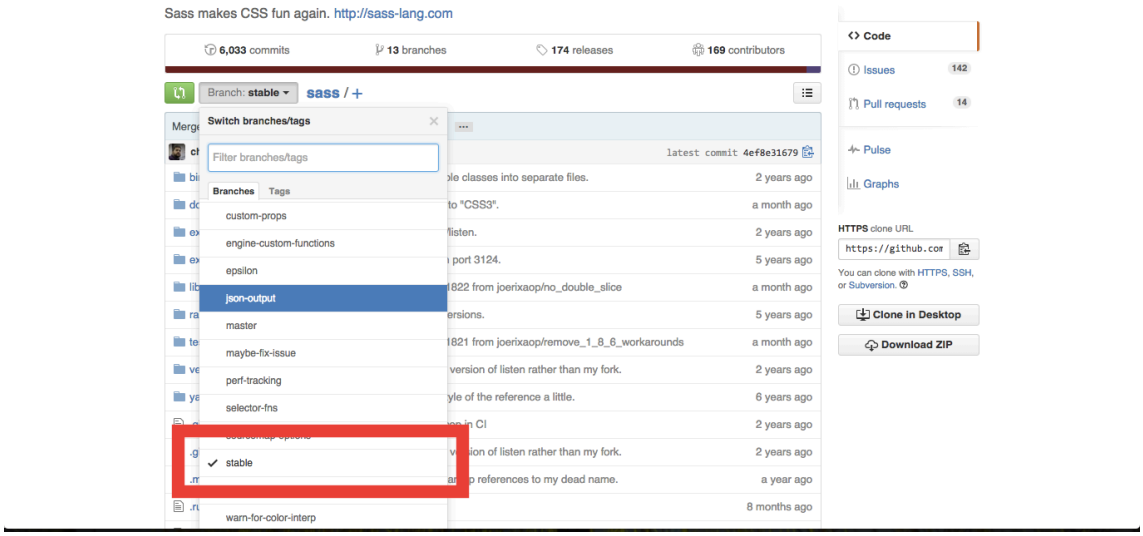


図 7.13　Stable フローの調査

7.3　データの分析方法

分析に使うツールは，Excel と R である．R とは，統計計算をしたり，その結果をグラフにまとめるための言語・環境で，ボランティアによって作られた，S-PLUS の主要な関数をほぼ備えた GNU 版（コピーフリー版）です．1991 年，ニュージーランド，オークランド大学統計学科の 2 人の講師，Ross Ihaka と Robert Gentlemant によって開発が始まり，現在も開発が進められています [7].

それぞれで行った作業を以下にまとめる．

7.3.1　Excel にデータをまとめる

Excel の csv ファイルに調査した全てのデータを保存する．実際に保存したデータが以下である．

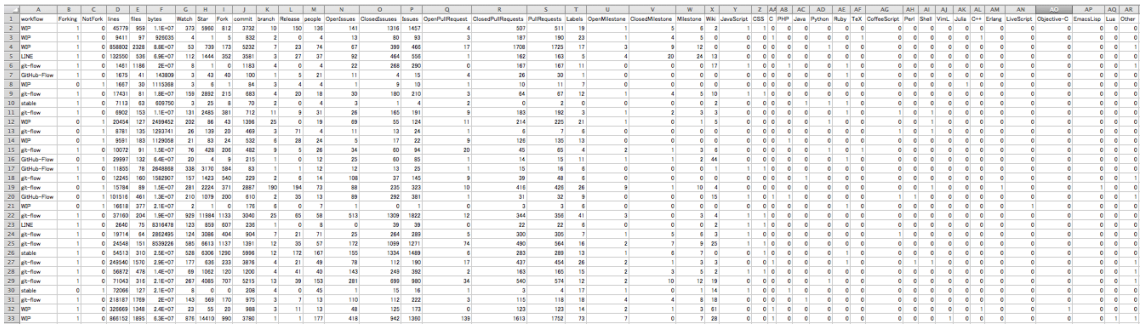


図 7.14　データ一覧

Y 列からは，言語になっている．その言語があった場合は 1，ない場合は 0 を入れている．

7.3.2　R で決定木分析を行う

R で決定木分析を行う方法を記述する．
"データ一覧.csv"のデータを読み込み，データセット"myData"に保存する．

```
myData <- read.csv("データ一覧.csv")
```

データセット"myData"を読み込む.

```
head(myData)
```

library の"mypart"と"maptools"を読み込む

```
library(mypart)
library(maptools)
```

"myData"のデータをデータセット"myTree"に保存する

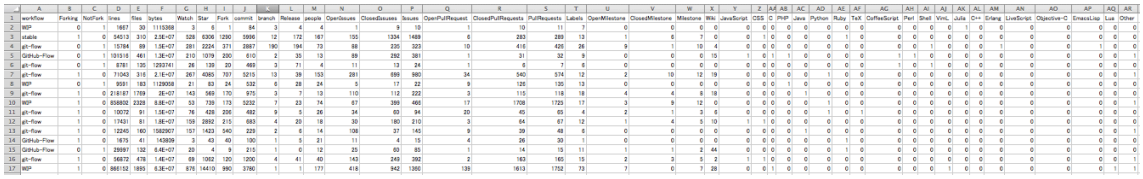
```
myTree <- rpart(workflow~ .,data = myData)
myTree
```

決定木を描画する.

```
plot(myTree)
text(myTree, all=TRUE, use.n=TRUE)
```

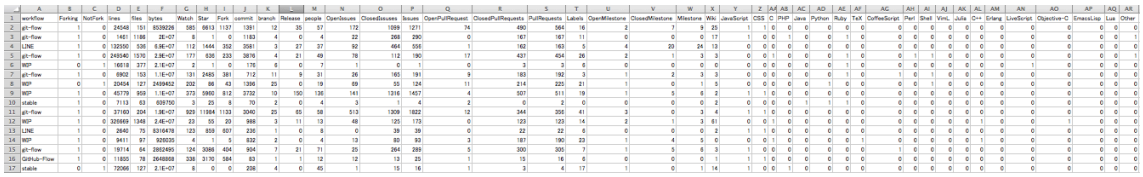
7.3.3　決定木を 2 つにわける

データをランダムに並び替え，2 つにわける．ランダムに分ける方法は，Excel の RAND 関数=RAND() を使用する．わけられたデータで，それぞれ決定木分析を行う．分けられたデータが図 7.15，7.16 である．



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR
1	workflow	Forking	NotFork	lines	files	bytes	Watch	Star	Fork	commit	branch	Release	people	OpenIssues	ClosedIssues	Issues	OpenPullRequest	ClosedPullRequests	PullRequests	Labels	OpenMilestone	ClosedMilestone	Milestone	Wiki	JavaScript	CSS	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Other
2	WP	0	1	1667	30	1115368	3	6	1	84	3	4	4	1	9	10	1	10	11	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
3	stable	1	0	54013	310	2.0E-07	528	6306	1290	5996	12	172	167	155	1334	1489	6	283	289	13	1	6	7	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	git-flow	0	1	15786	89	1.5E-07	281	2224	371	2887	190	194	73	88	230	323	10	416	426	26	9	1	10	4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0
5	GitHub-Flow	0	1	101318	461	1.3E-07	210	1079	200	610	2	33	13	89	292	381	1	31	32	9	0	0	0	15	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
6	git-flow	0	1	8781	135	1293741	26	135	20	469	3	71	4	11	13	24	1	6	7	6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
7	git-flow	1	0	71042	315	2.1E-07	261	4383	701	5213	13	39	153	281	699	980	34	540	574	12	2	10	12	19	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	WP	0	1	9091	183	1129008	21	83	24	532	6	28	24	5	17	22	9	126	135	13	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	git-flow	1	0	218187	1769	2E-07	143	569	170	975	3	7	13	110	112	222	3	115	118	18	4	4	8	18	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	WP	1	0	856602	2328	8.8E-07	53	739	173	5232	7	23	14	87	399	466	17	1708	1723	17	3	9	12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	git-flow	1	0	10072	91	1.5E-07	76	428	226	482	9	5	26	34	60	94	20	45	65	4	2	1	3	6	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	git-flow	1	0	17431	81	1.8E-07	159	2892	215	683	4	20	18	30	180	210	3	64	67	12	1	4	5	10	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	git-flow	1	0	12240	160	1562907	157	1423	540	229	2	6	14	106	37	143	9	39	48	6	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	GitHub-Flow	1	0	1875	41	143809	3	43	40	130	1	5	21	11	4	15	4	26	30	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	GitHub-Flow	0	1	29997	132	6.4E-07	20	4	9	215	1	0	12	25	60	85	1	14	15	11	1	1	1	2	44	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	git-flow	1	0	56872	478	1.4E-07	69	1062	120	1200	4	41	40	143	249	392	2	163	165	15	2	3	5	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
17	WP	1	0	866152	1893	6.3E-07	676	14410	990	3790	1	1	177	418	942	1360	139	1613	1752	73	7	0	7	28	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

図 7.15　データ一覧 1



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR
1	workflow	Forking	NotFork	lines	files	bytes	Watch	Star	Fork	commit	branch	Release	people	OpenIssues	ClosedIssues	Issues	OpenPullRequest	ClosedPullRequests	PullRequests	Labels	OpenMilestone	ClosedMilestone	Milestone	Wiki	JavaScript	CSS	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Other
2	git-flow	1	0	24548	151	8539228	585	6613	1137	1391	12	35	57	172	1099	1211	74	490	564	16	2	7	9	23	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	git-flow	1	0	1461	1188	2E-07	8	1	0	1183	4	0	4	22	268	290	0	167	167	11	0	0	0	17	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	LINE	1	0	132550	536	6.9E-07	112	1444	352	3581	3	27	37	92	464	556	1	162	163	5	4	20	24	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	git-flow	1	0	243540	1370	2.3E-07	177	636	233	3816	4	21	49	78	112	190	17	437	454	26	2	1	3	3	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1
6	WP	0	1	16818	377	2.1E-07	2	1	0	176	6	0	7	1	0	1	0	3	3	6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
7	git-flow	1	0	6902	153	1.1E-07	131	2485	381	712	11	9	31	26	165	191	9	183	192	3	1	2	3	3	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0
8	WP	0	1	20404	127	2409402	202	88	43	1396	23	0	19	69	33	124	11	214	225	21	1	0	1	3	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
9	WP	1	0	45779	959	1.1E-07	373	5960	812	3732	10	190	126	140	1316	1457	4	507	511	19	1	5	6	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	stable	1	0	7113	63	609750	3	25	8	70	2	0	4	3	1	4	2	0	2	0	0	0	0	2	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	git-flow	1	0	37160	204	1.8E-07	929	11984	1133	3040	25	65	58	513	1309	1822	12	344	356	41	3	0	3	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	WP	1	0	326669	1348	2.4E-07	23	55	20	988	3	11	13	48	125	173	0	123	123	14	2	1	3	81	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	LINE	1	0	2640	75	8316478	123	859	607	235	1	0	8	0	39	39	0	22	22	6	0	0	0	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	WP	1	0	9411	97	929035	4	1	1	932	2	0	4	13	80	93	3	187	190	23	1	4	5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	git-flow	1	0	19714	64	2982495	124	3086	404	904	7	21	71	25	264	289	5	300	305	7	1	5	6	3	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
16	GitHub-Flow	1	0	11805	78	2648868	338	3170	584	83	1	1	12	12	13	25	1	15	16	6	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	stable	0	1	72066	127	2.1E-07	8	0	0	208	4	0	45	1	15	16	1	3	4	17	1	0	1	14	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図 7.16　データ一覧 2

第 8 章

結果

3 種類のデータを用いて，R で決定決定木分析を行った．すべてのデータで行った場合，データをランダムに並び替え，2 つに分けた場合である．それぞれの結果を記述する．

8.1 すべてのデータで行った場合

図 8.1 は，すべてのデータで分析を行った結果である．

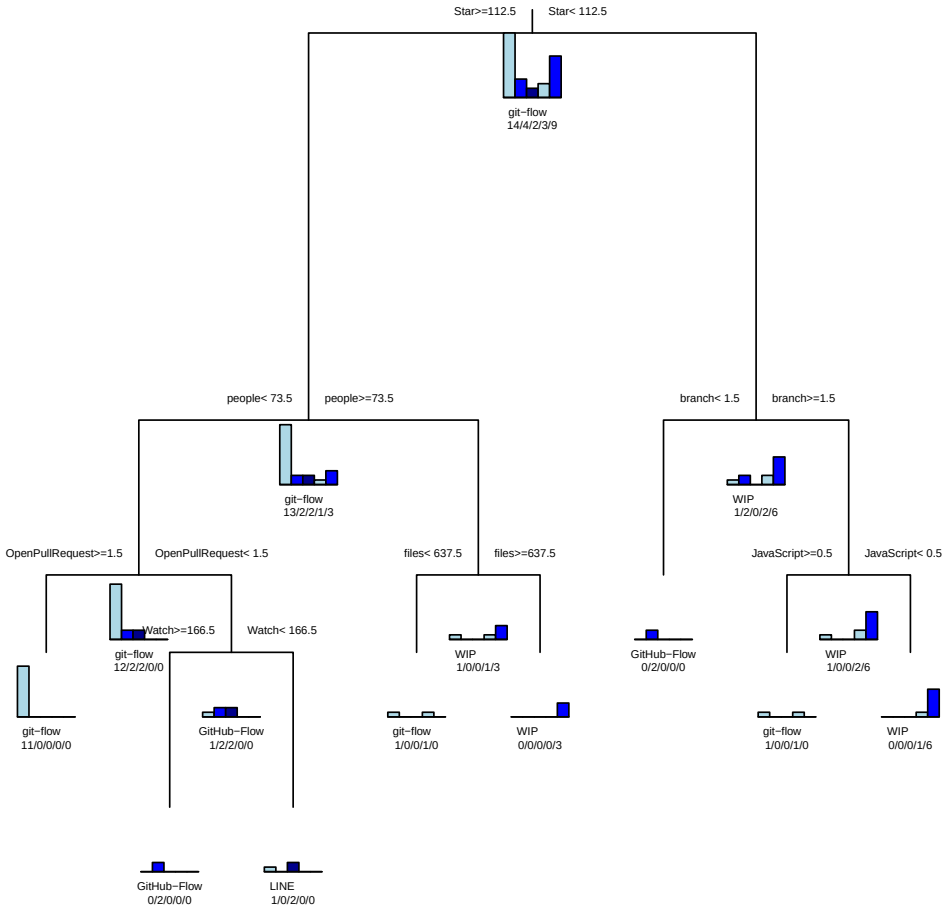


図 8.1　決定木

すべてのデータで分析した場合，Star 数，人数，Open Pull Request 数，Watch 数，ファイル数，branch 数，JavaScript で別れた．

8.2　データをランダムに並び替え，2 つに分けた場合

図 8.2 は図 7.15 を用いて分析を行った結果である．図 8.3 は図 7.16 を用いて分析を行った結果である．

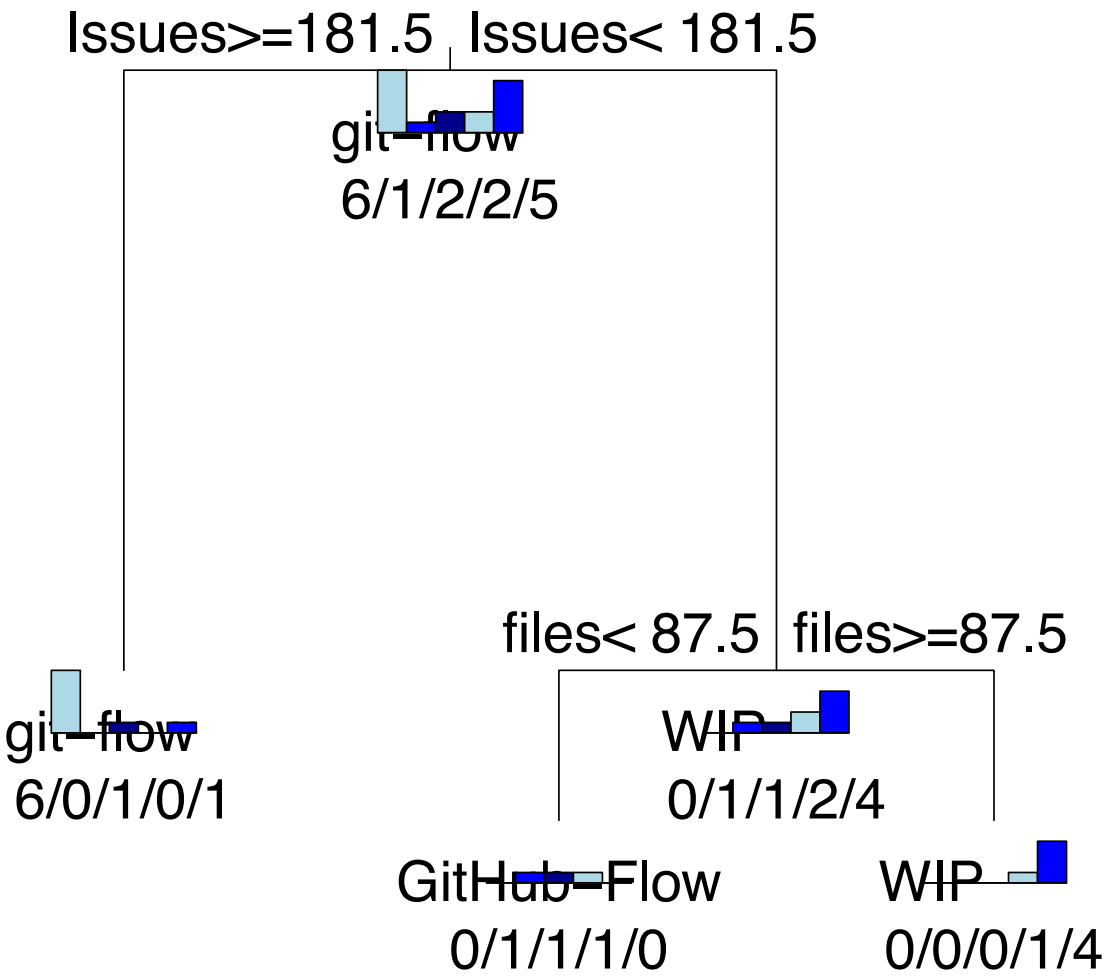

Issues>=181.5
Issues< 181.5
6/1/2/2/5
files< 87.5
files>=87.5
6/0/1/0/1
0/1/1/2/4
0/1/1/1/0
0/0/0/1/4

図 8.2　決定木 1

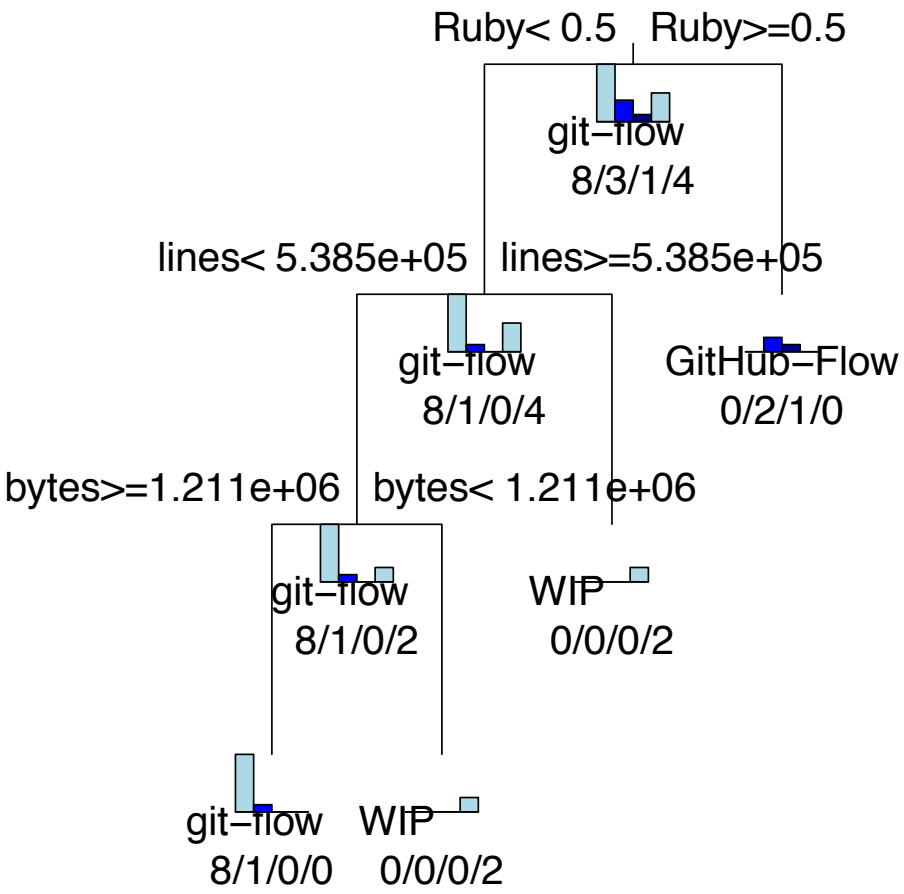


図 8.3　決定木 2

データをランダムに 2 つに分けた場合，異なった結果がでた．図 8.2 は，Issue 数，ファイル数，図 8.3 は，Ruby，行数，バイト数で別れた．

最も重要なファクターが全ての決定木で異なることがわかる．図 8.1 では，Star 数，図 8.2 では，Issue 数，図 8.3 では，Ruby である．また，別の決定木では，これらファクターは出てきていない．

参考文献

[1] 小野寺航己. バージョン管理システムを活用するソフトウェア開発の開発フロー. 卒業論文, 千葉工業大学, 2015.

[2] 池田尚史, 藤倉和明, 井上史彰. チーム開発実践入門〜共同作業を円滑に行うツール・メソッド. 技術評論社, 2014.

[3] Ken Nishimura. 970 万人の開発者を擁する github が「ギットハブ・ジャパン」を始動、法人向けでマクニカが販売. http://jp.techcrunch.com/2015/06/04/github-launches-the-first-branch-in-japan/（2015.10.05 閲覧）.

[4] 大塚弘記. GitHub 実践入門 Pull Request による開発の変革. 技術評論社, 2014.

[5] harada4atsushi. Github flow で pull request ベースな開発フローの進め方. http://qiita.com/harada4atsushi/items/527d5f98320d993b3072（2015.10.03 閲覧）.

[6] hayaishi. Line ios アプリ開発についてのご紹介. http://developers.linecorp.com/blog/?p=2921（2015.10.03 閲覧）.

[7] 竹内俊彦. はじめての S-PLUS/R 言語プログラミング– 例題で学ぶ S-PLUS/R 言語の基本. オーム社, 2005.