

第 1 章

序論

ソーシャルコーディング.

第 2 章

背景

ソーシャルコーディングに使われるGitHubを分析することで何か新しい発見があると思い分析を開始した。しかしプログラミング初心者には参考文献として本を読んでもまったく意味が分からなかった。そこで参考文献に乗っているデータの取得，変換，解析を実践し，実際に何が分からなかったかを見つけることにした。

第 3 章

目的

本研究では，参考文献に乗っているデータの取得，変換，解析を実践する．分からなかった点を見つけ出し解決する．

第 4 章

手法

4.1 利用するものの事前知識

4.1.1 R

Rとはプログラミング言語のひとつであり，今回の研究ではデータの変換，解析に用いる．



図 4.1 Rのロゴ

4.1.2 GitHub

GitHubとはソフトウェア開発のプラットフォームである。プラットフォームとは場所を指す英語。ここではソフトウェア開発を行う場所として用いる。GitHubはGitというバージョン管理システムから生まれたものである。バージョン管理システムとは図4.3、4.4のようにファイルの変更時間や変更場所、変更した人間を記録してくれるシステムである。

GitHubではそのバージョン管理システムやソーシャルコーディングを助ける様々な機能を提供している。ソーシャルコーディングとはプログラムなどをオンライン上で共有し開発していく事である。

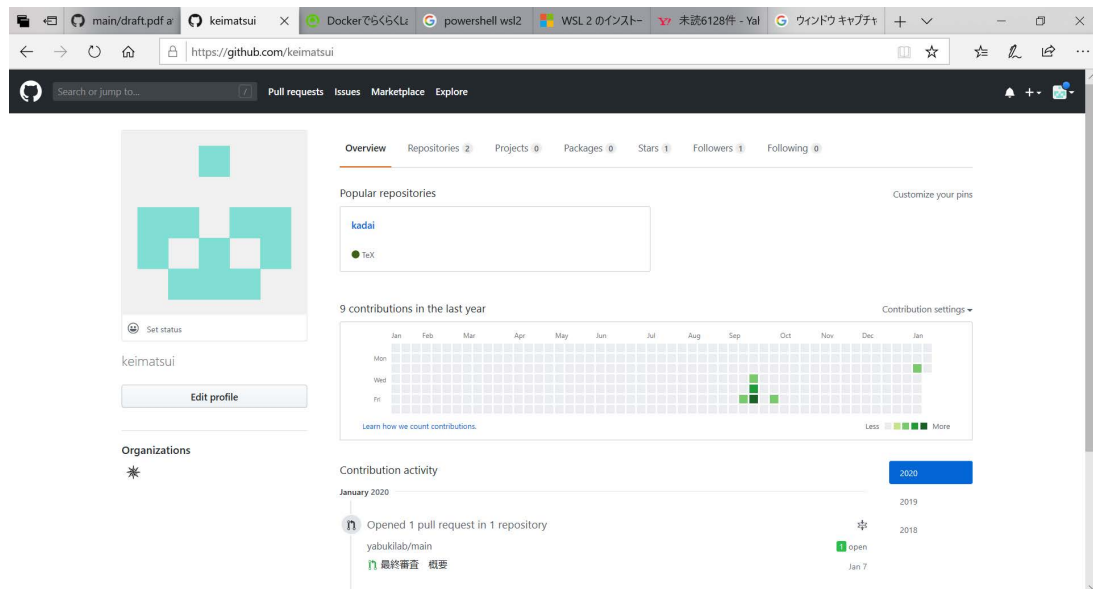


図 4.2 GitHub

4.1.3 GitHubの機能

GitHubには様々な機能がある。事前知識として必要なので順番に説明していく。

リポジトリ ファイル更新日時や変更点を記録している場。

クローン リモートリポジトリをコピーしPC内にローカルリポジトリを作成する。

ブランチ .

4.1.3 GitHubAPI

GitHubAPIはGitHubが提供するAPIである。APIとはアプリケーションプログラミングインタフェースの略で分かりやすく言うと特定ものに対するアクセスを簡略化したものである。GitHubAPIを使うことにより簡単にデータが取得出来る。

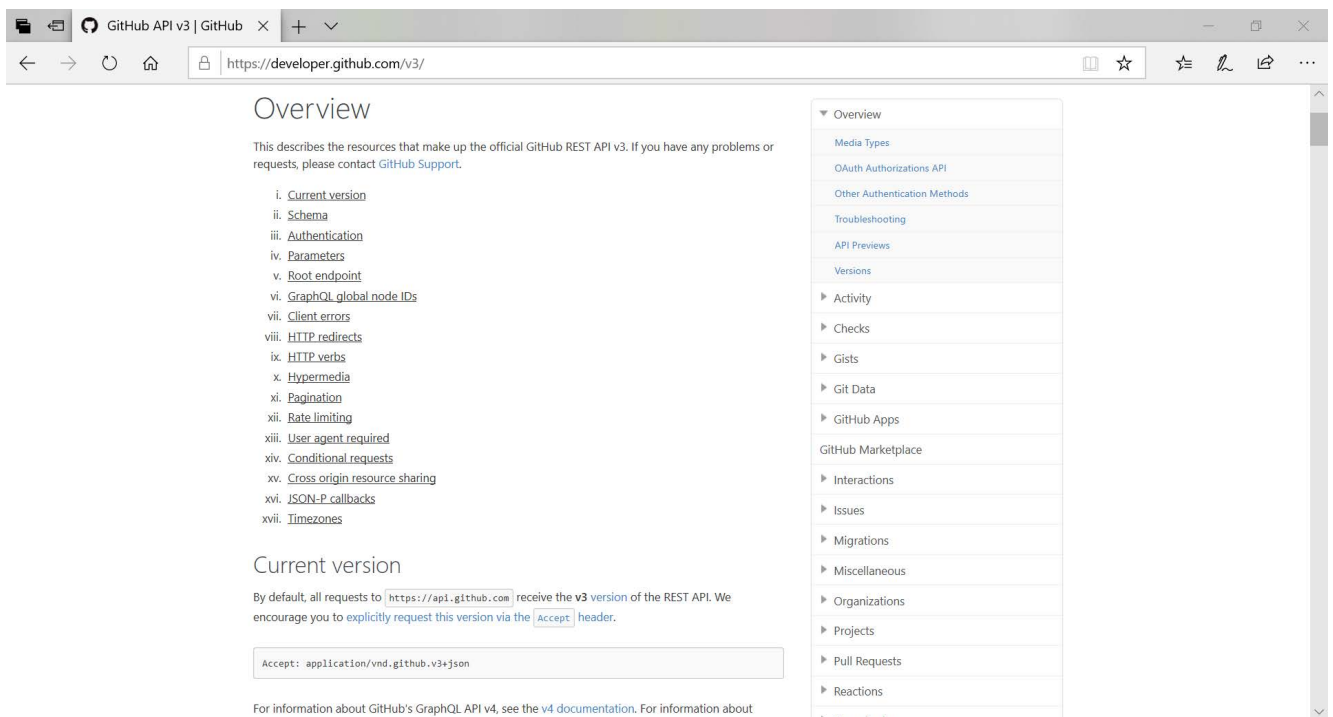


図 4.5 GitHubAPI公式サイト

4.2 主な手順

本節では研究の流れについて説明する．

1. Rをインストールする．
2. パッケージをインストールする．
3. アクセストークンを取得する．
4. データの取得，変換，分析を実践する．
5. 理解が出来なかったところを調べる．

4.3 Rの準備

4.3.1 Rのインストール

まず初めにRをインストールする．検索エンジンで「R インストール」と調べると，図4.6のサイトが上位に表示されると考えられる．見つからなかった場合は<http://www.okadajp.org/RWiki/>このリンク入力してもらえればたどり着ける．図4.6のページが表示されたら図下部にある最新版はこちらからを押す．



図 4.6 RjpWiki

図4.7画面に出たらbaseか install R for the first timeを押す。その後図4.8のページが表示される。一番上にあるDownload R 3.6.2 for Windows押すとRのインストールファイルのダウンロードが始まる。Download R 3.6.2 for Windowsとあるが3.6.2の部分はバージョンを表しているため変わっている場合もある。今回の研究ではR 3.6.2 for Windowsを用いている。

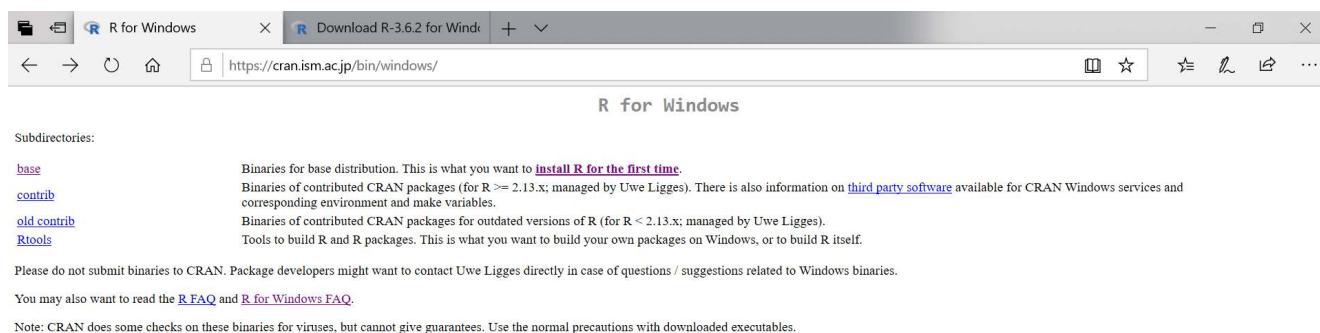


図 4.7 base

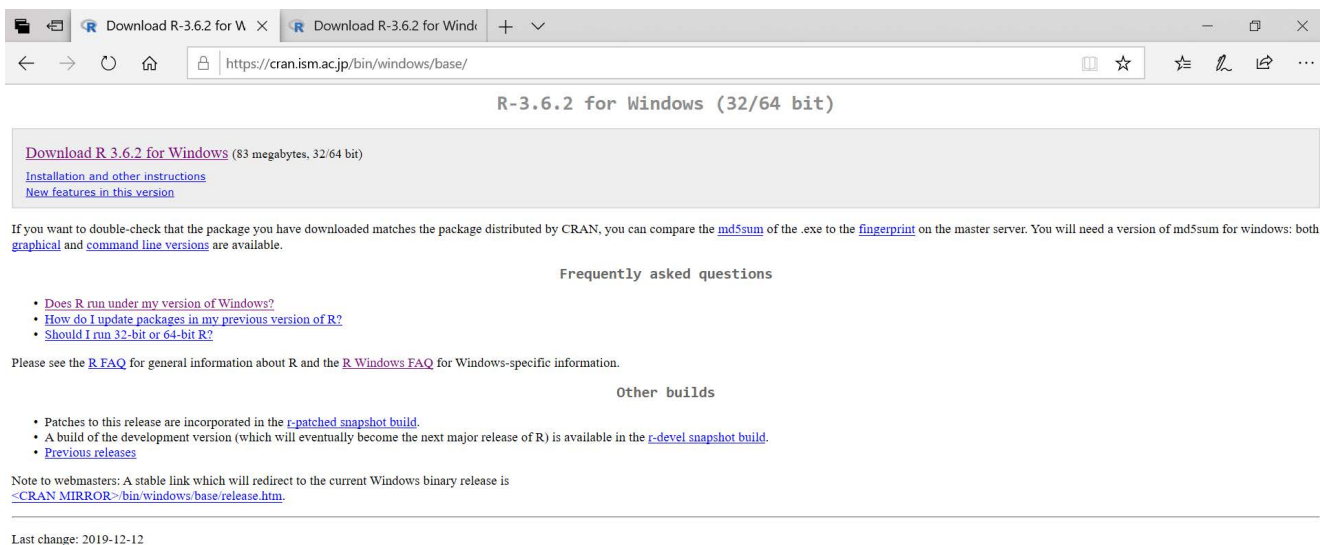


図 4.8 R バージョン

Rのインストールファイルをダウンロードし実行する。その後流れに沿って進めていくと図4.9の画面になる。まずCore FileとMessage translationsにはチェックを入れておく。32, 64に関しては各々必要な方を入れておく。分からなければどちらにも入れておけばよい。そのあとも説明に従って進めていけばインストールされる。

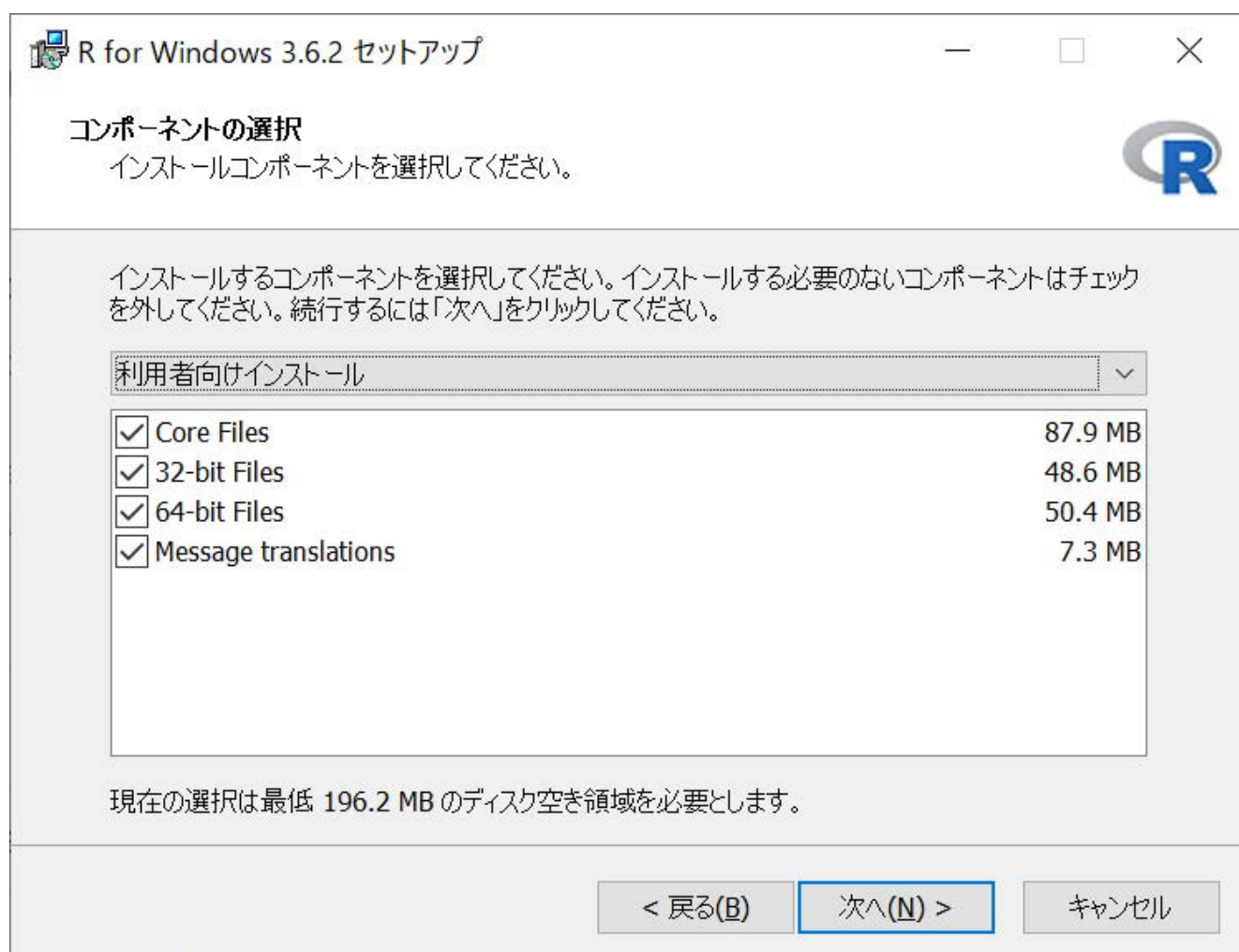
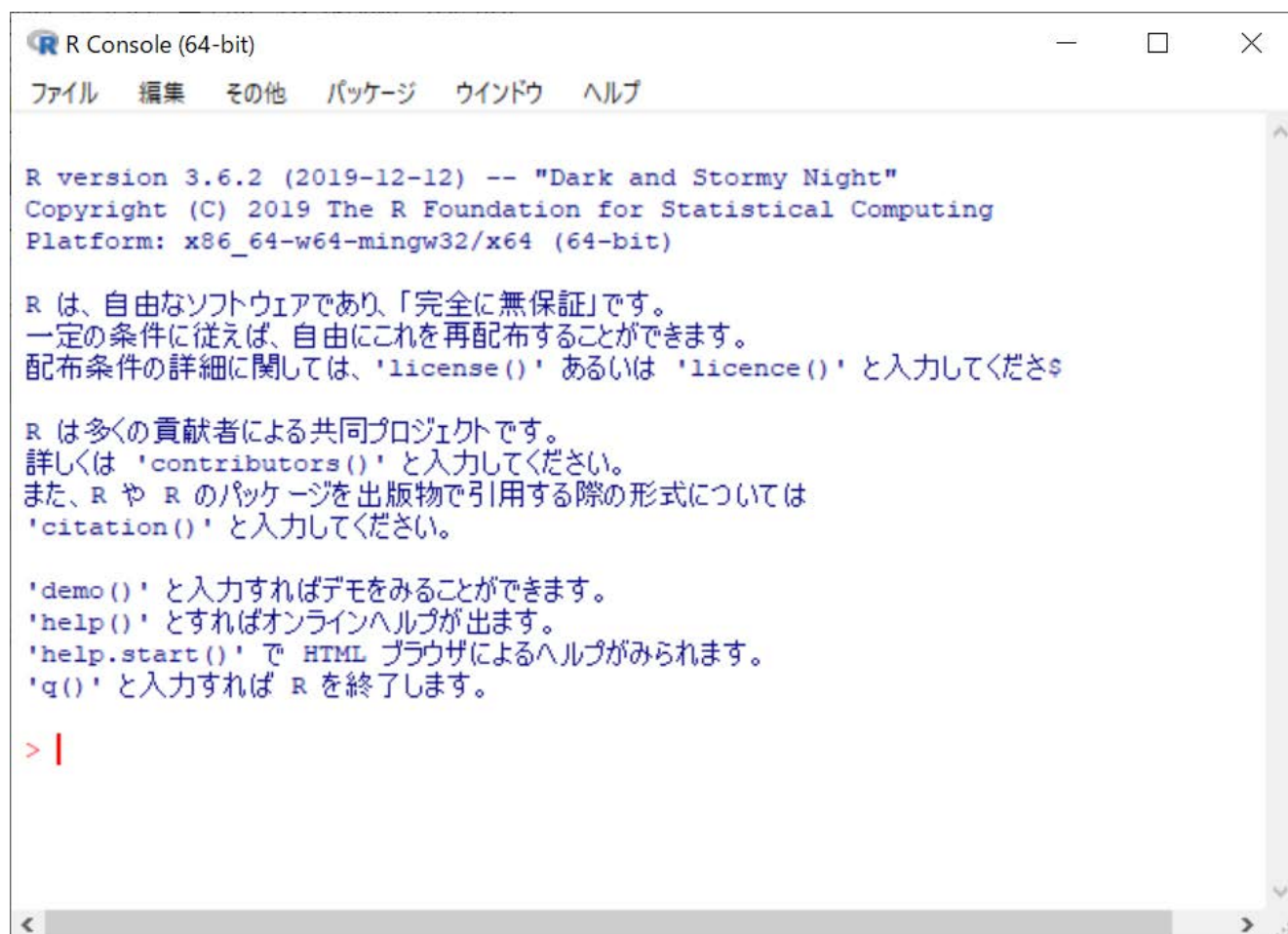


図 4.9 コンポーネントの選択

4.3.2 Rの設定

Rを起動させると図4.9のような画面が表示される。まず初めに上部タブ欄にあるファイルタブからディレクトリを変更するを選択する。ディレクトリを変更するとは解析したデータの保存や読み込みを行う場所を決める事なのでフォルダを作って選択するか、自分が分かりやすいフォルダを選択する。getwd()と入力することで今選んでいるディレクトリを知ることが出来る。



```
R Console (64-bit)
ファイル 編集 その他 パッケージ ウィンドウ ヘルプ

R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R は、自由なソフトウェアであり、「完全に無保証」です。
一定の条件に従えば、自由にこれを再配布することができます。
配布条件の詳細に関しては、'license()' あるいは 'licence()' と入力してくださ$

R は多くの貢献者による共同プロジェクトです。
詳しくは 'contributors()' と入力してください。
また、R や R のパッケージを出版物で引用する際の形式については
'citation()' と入力してください。

'demo()' と入力すればデモをみることができます。
'help()' とすればオンラインヘルプが出ます。
'help.start()' で HTML ブラウザによるヘルプがみられます。
'q()' と入力すれば R を終了します。

> |
```

図 4.9 R初期画面

パッケージのインストールを行うためにミラーサイトを設定しておく。ミラーサイトとはサイト上のコンテンツにアクセスが集中した場合遅延、停止してしまう可能性があるため同じ情報を持ったサイトである。図4.10にあるようにパッケージタグからCRANミラーサイトの設定を押しJapanを押す。

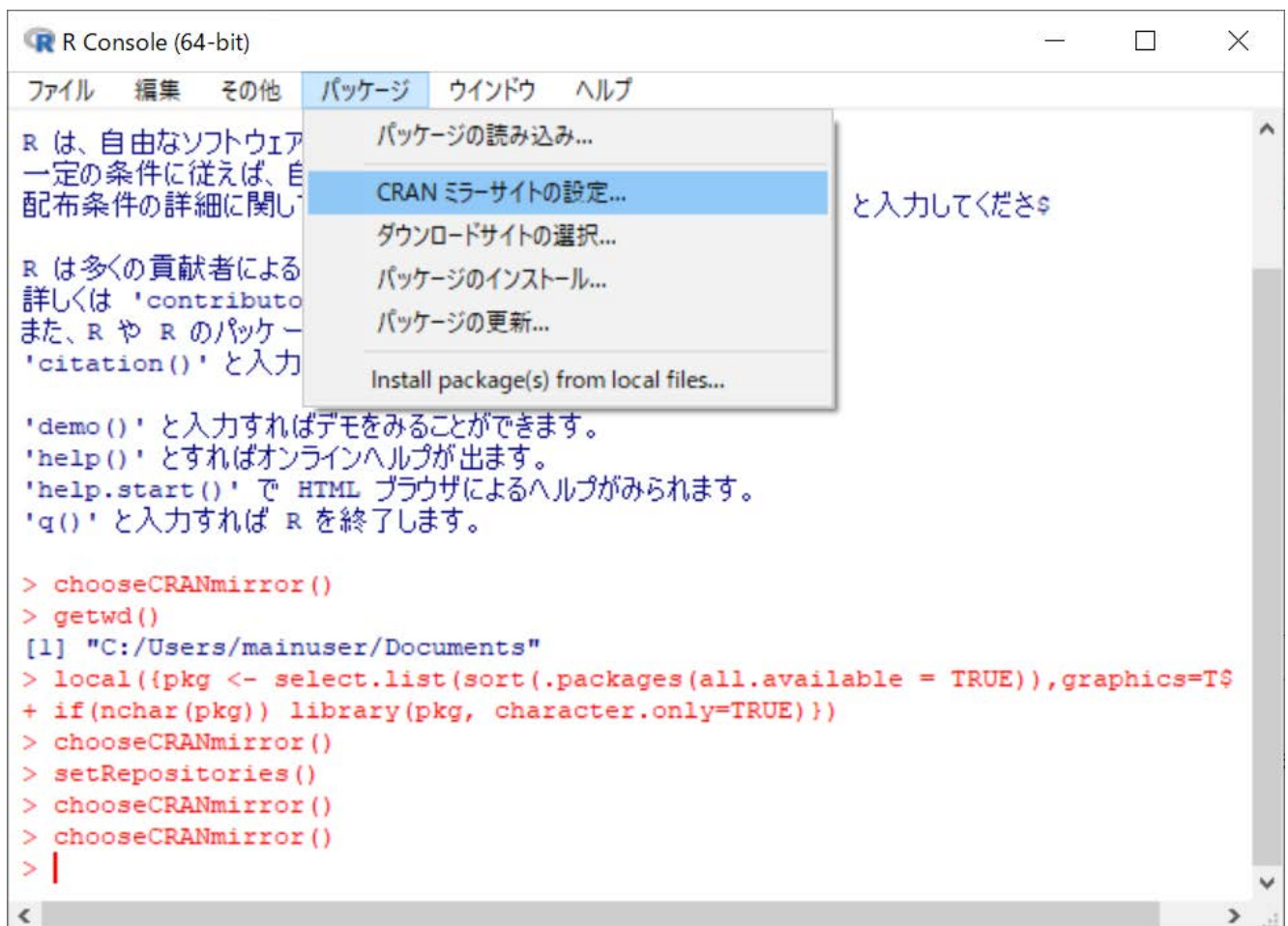
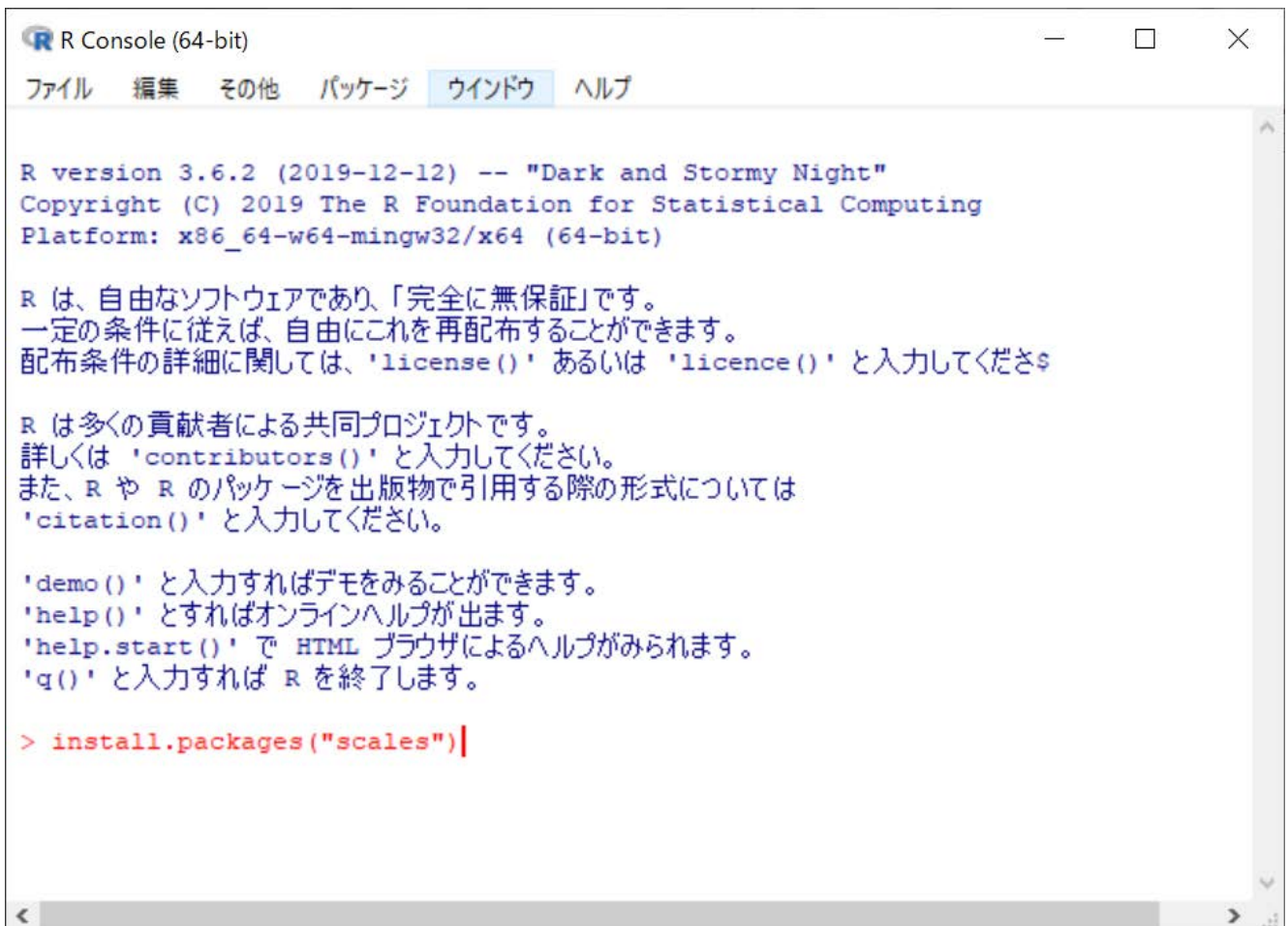


図 4.10 ミラーサイト選択

4.3.3 パッケージ

前述にもあったパッケージとは動作に対して必要なプログラムやファイルをまとめたものである。パッケージをインストールすることで機能が増えると考えればよい。図 4.11のように`install.packages("scales")`と入力するとscalesというパッケージがインストールされる。



```
R Console (64-bit)
ファイル 編集 その他 パッケージ ウィンドウ ヘルプ

R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R は、自由なソフトウェアであり、「完全に無保証」です。
一定の条件に従えば、自由にこれを再配布することができます。
配布条件の詳細に関しては、'license()' あるいは 'licence()' と入力してくださ$

R は多くの貢献者による共同プロジェクトです。
詳しくは 'contributors()' と入力してください。
また、R や R のパッケージを出版物で引用する際の形式については
'citation()' と入力してください。

'demo()' と入力すればデモをみることができます。
'help()' とすればオンラインヘルプが出ます。
'help.start()' で HTML ブラウザによるヘルプがみられます。
'q()' と入力すれば R を終了します。

> install.packages("scales")|
```

図 4.11 パッケージのインストール

今回の研究では以下のパッケージを使用するためインストールする.

scales

extrafontdb

Rttf2pt1

httr

jsonlite

dplyr

ggplot2

reshape2

hrbrthemes

sqldf

lubridate

corrplot

devtools

date.table

下のコードが入力例である．一つだけdevtools::install_githubというものがあるがこれはGitHubからインストールしているためである．devtoolsを使っているため先にdevtoolsをインストールしていると良い．

```
install.packages("scales")
install.packages('extrafontdb')
install.packages('Rttf2pt1')
install.packages("devtools")
devtools::install_github("hrbrmstr/hrbrthemes")
install.packages("httr")
install.packages("jsonlite")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("resharp2")
install.packages("hrbrthemes")
install.packages("sqldf")
install.packages("lubridate")
install.packages("corrplot")
install.packages("date.table").
```

4.4 アクセストークン

GitHubAPIは特に認証せずとも使用できる。しかし認証しなかった場合は一時間に60回しかリクエストを行えない。認証をすることにより5000回にまで増えるため、認証を行う。

まず初めに自分のGitHubアカウントのセッティング画面に入る。右上にある自分のアイコンを押せば行ける。図4.12の画面まで来たら左側にあるカテゴリから一番下にあるDeveloper settingを押す。

図4.13左側にあるOAuth Apps押しNew OAuth Appsを押す。

図4.14ではアプリケーションの情報を書く。

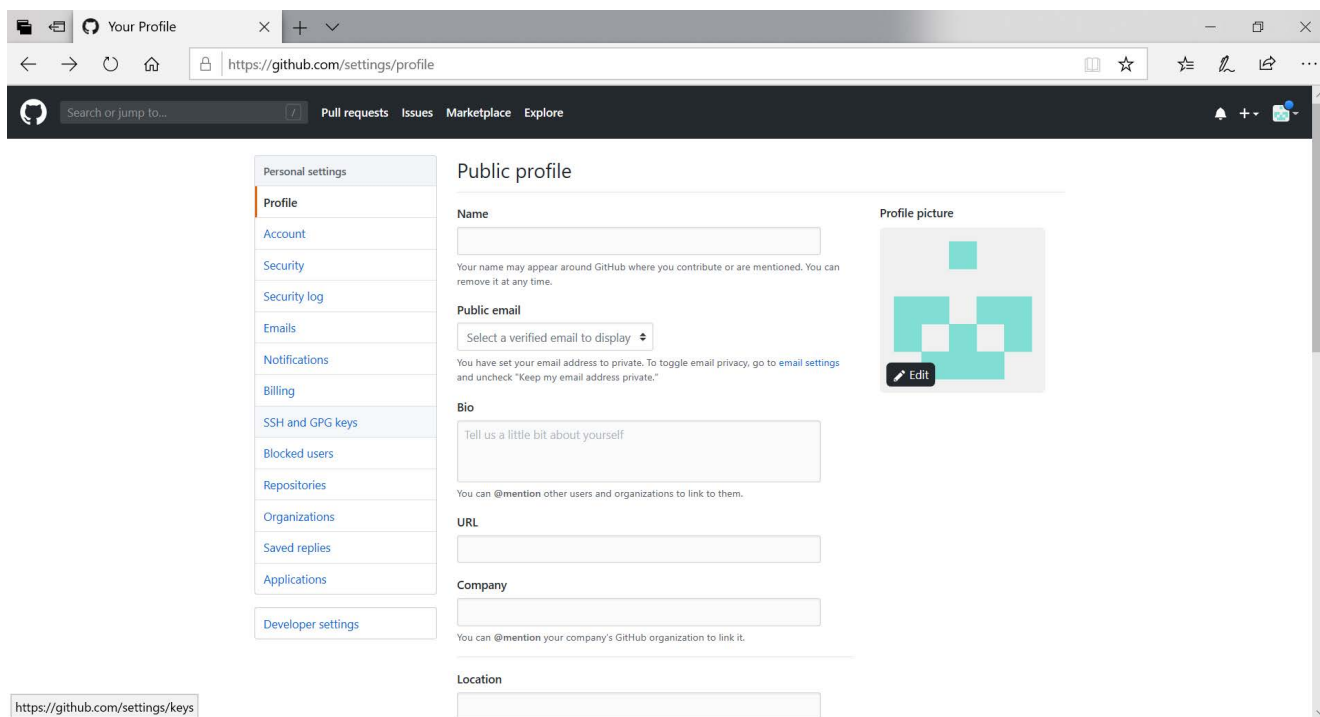


図 4.12 アクセストークン取得

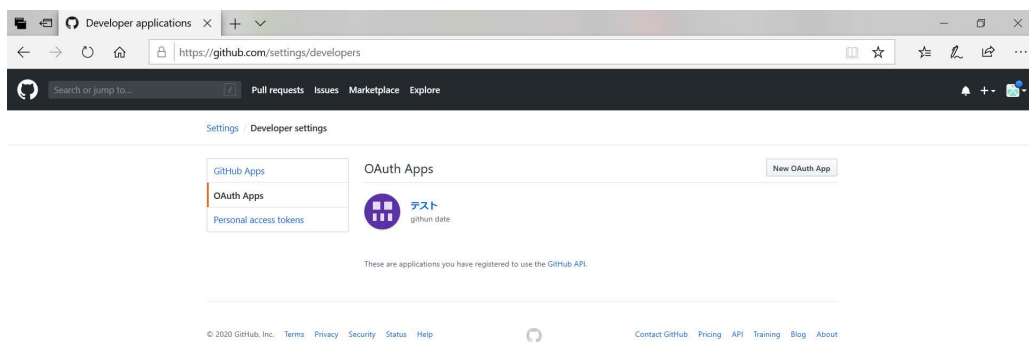


図 4.13 アプリケーション一覧

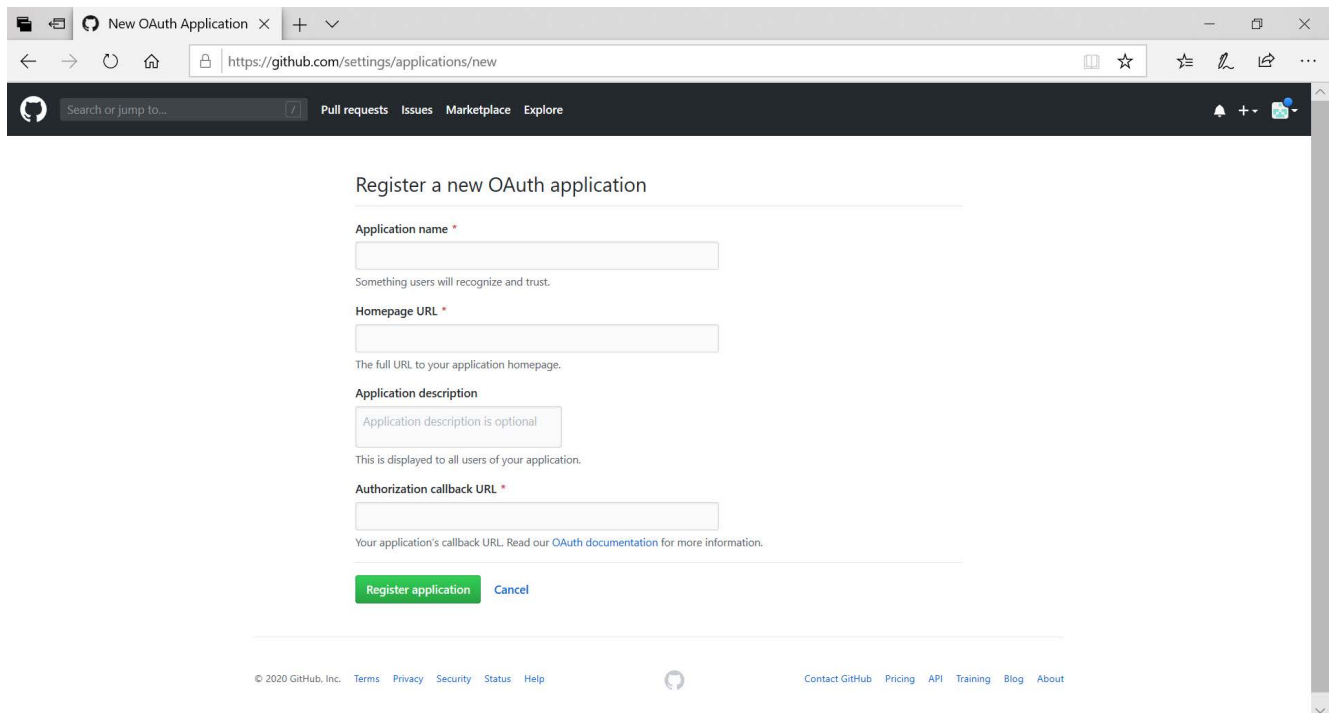


図 4.14 アプリケーション作成

これでアクセストークンが取得できた。トークンは必ずどこかに保存しておくこと。
トークンが取得出来たら次はどうやって使うか。下のコードのように打てば認証される。!!!の部分は自分のトークンを入れる。

```
auth.id <- '3f442ac5a258894328e0'  
auth.pwd <- 'a5831a4c762278c9935f15a55794858ddefe9862'  
api_id_param <- paste0('client_id=', auth.id)  
api_pwd_param <- paste0('client_secret=', auth.pwd)  
arg_sep = '&'
```

4.5 データの取得, 変換

4.5.1 データの取得

まず試しに自分のアカウントの情報を取得する。下のコードで取得できる。
<は格納することを意味しこの場合responseに取得した情報が格納された。もちろんresponseは変えても大丈夫である。

```
base_url <- 'https://api.github.com/users/keimatsui?'
my_profile_url <- paste0(base_url, api_id_param, arg_sep, api_pwd_param)
response <- GET(my_profile_url)
```

その後responseと調べることで格納された情報が見れる。

```
> response
Response [https://api.github.com/users/keimatsui?
client_id=3f442ac5a258894328e0&client_secret=a5831a4c762278c9935f15a55
794858ddefe9862]
  Date: 2020-01-14 21:51
  Status: 200
  Content-Type: application/json; charset=utf-8
  Size: 1.27 kB
{
  "login": "keimatsui",
  "id": 38268031,
  "node_id": "MDQ6VXNlcjM4MjY4MDMx",
  "avatar_url": "https://avatars1.githubusercontent.com/u/38268031?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/keimatsui",
  "html_url": "https://github.com/keimatsui",
  "followers_url": "https://api.github.com/users/keimatsui/followers",
  "following_url": "https://api.github.com/users/keimatsui/following{/othe...
  ...
}
```


4.5.2 データの変換

先ほどの形式では見づらいためで他を変換する。下コードを打つことにより変換できる。二つあるがどちらかどちらでも同じ結果が得られる。

上のコードはJSONとしてのデータフレームに落とし込んでいると考えられる。

下はそのまま要素を定義していると考えられる。

```
me <- fromJSON(my_profile_url)
me <- as.data.frame(t(as.matrix(me)))
View(me[,c('login', 'public_repos', 'public_gists', 'followers',
           'created_at', 'updated_at')])
```

```
me <- content(response)
me <- as.data.frame(t(as.matrix(me)))
View(me[,c('login', 'public_repos', 'public_gists', 'followers',
           'created_at', 'updated_at')])
```

4.6 リポジトリ活動の分析

4.6.1 コミット頻度の分析

コミット頻度の文分析のためまずはデータを取得する。取得したデータはエポック基準のため変換を行う。エポックとは1970/01/01 00:00:00.000のこと。エポック基準は分かりづらいので日時形式に変換を行う。

コミットアクティビティを取得

```
base_url <- 'https://api.github.com/repos/torvalds/linux/stats/commit_activity?'
repo_url <- paste0(base_url, api_id_param, arg_sep, api_pwd_param)
response <- fromJSON(repo_url)
```

エポックから日時形式に変換

```
response$week_conv <- as.POSIXct(response$week, origin="1970-01-01")
```

POSIXctとは日時型のデータ

週次のコミット頻度を可視化

```
ggplot(response, aes(week_conv, total)) +  
  geom_line(aes(color=total), size=1.5) +  
  labs(x="Time", y="Total commits",  
        title="Weekly GitHub commit frequency",  
        subtitle="Commit history for the linux repository") +  
  theme_ipsum_rc()
```

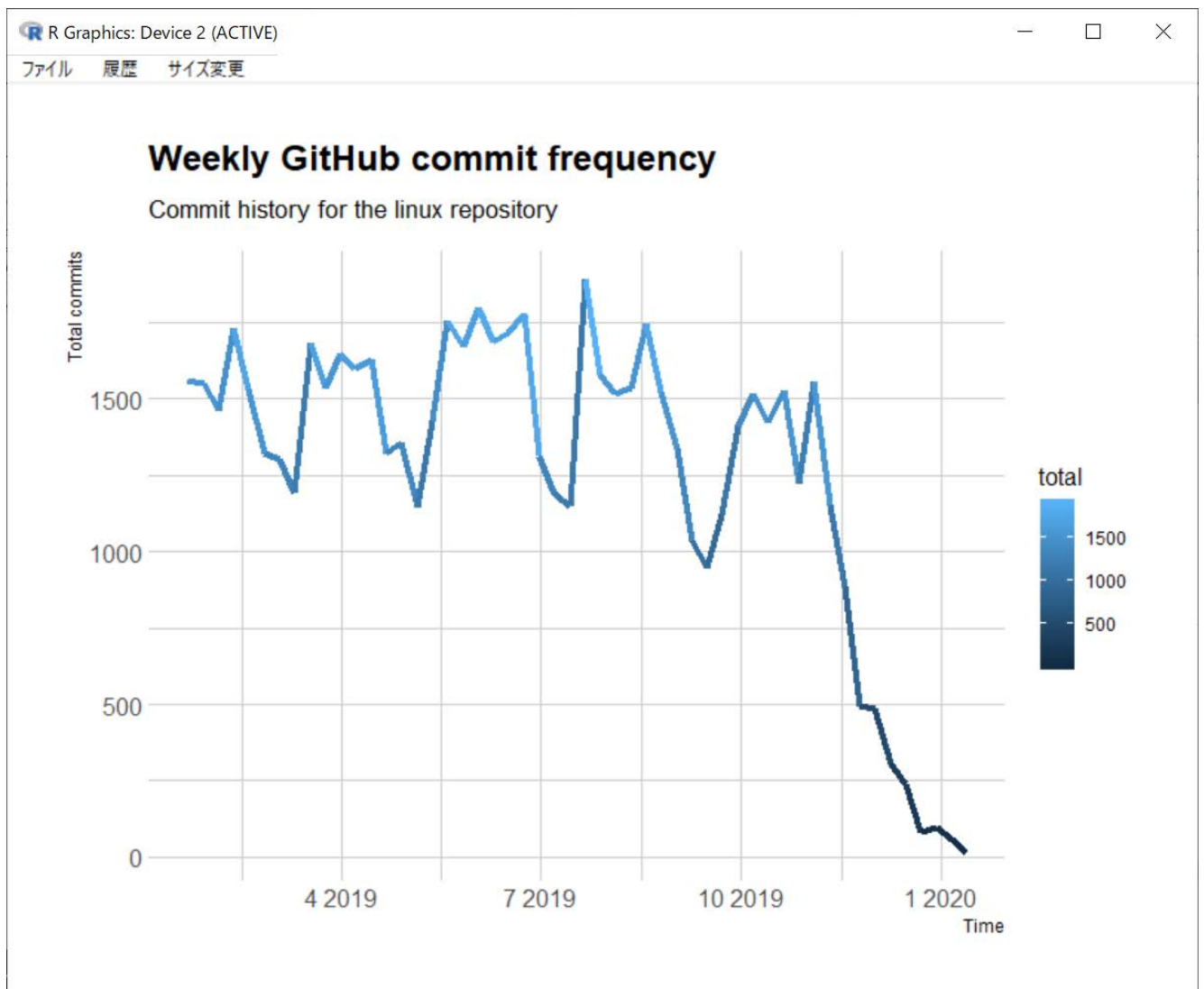


図 4.15 コミット頻度