ITPASS
Information Training Program
with a Spirit of Self-help

定期的に作業を実行するデーモン, cron
— COVID-19 の時系列データの図を描いてみる

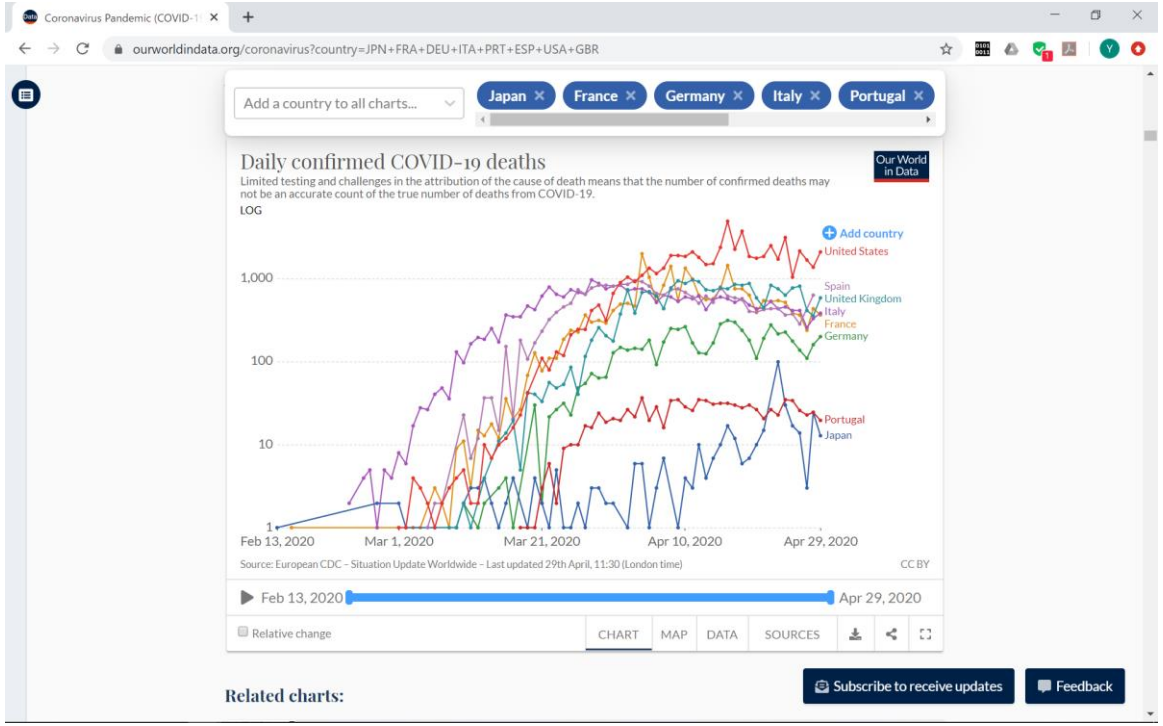
高橋芳幸
神戸大学大学院理学研究科惑星学専攻
2020 年 11 月 20 日

はじめに 1

- サーバの運用のためには定期的に行う作用がある.
  - サーバの状態の確認
    - 定期的にディスク使用量等を確認
    - …
  - サーバのデータの冗長性を確保
    - 定期的にファイルを他のサーバにコピー / 同期
    - …
- それらを実現するデーモン : cron
- これまで cron について解説したことがないので試みる.

はじめに 2-1

- COVID-19
  - コロナウイルス SARS-CoV-2 が引き起こす病気. 2019 年に確認され 2020 年 11 月現在感染拡大中.
- 感染拡大の動向が注目されており, 各国の研究機関, 政府機関, 個人が感染者数, 死者数などのデータを可視化するウェブサイトを立ち上げている.
  - 例
    - Our World in Data (https://ourworldindata.org/coronavirus)
    - 東洋経済 ONLINE (https://toyokeizai.net/sp/visual/tko/covid19/)
- 様々なデータが可視化されていて有益だが, 自分で図を描いてみたい.



Our World in Data
(https://ourworldindata.org/coronavirus?country=JPN+FRA+DEU+ITA+PRT+ESP+USA+GBR)

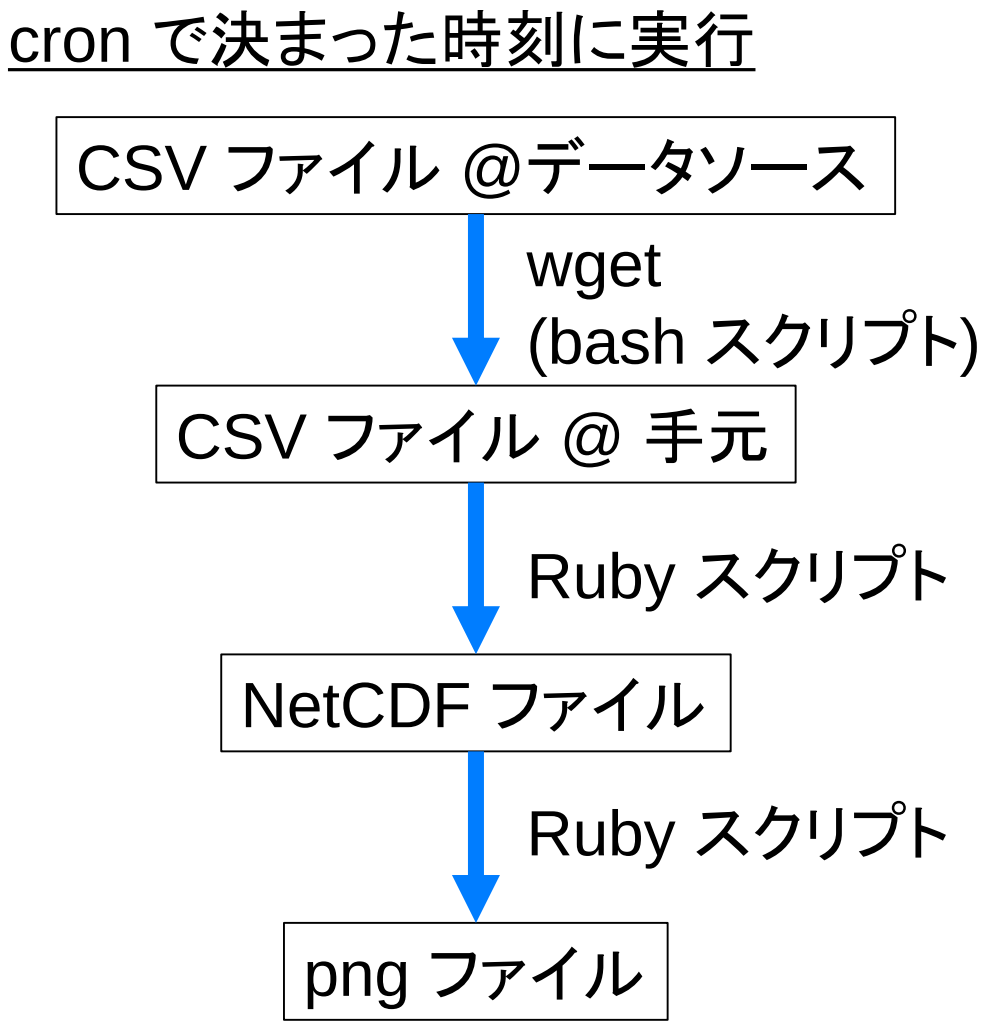
はじめに 2-2

- やってみたいこと
  - COVID-19 の死亡者数データを自分で可視化してみる.
    - 異なるソースのデータは (ほとんど) 同じなのか?
    - 好きな間隔で移動平均を取りたい.
    - 好きな軸で図を描きたい. 対数軸とか線形軸とか.
  - 毎日図を更新したい.
    - COVID-19 の関連データは日々更新される.
    - 自動的に毎日データをダウンロードしたい
    - 自動的に毎日図を描きたい.

はじめに 2-3

- cron を使って定時にデータをダウンロードして図を描く.
- データは wget でダウンロード.
- データファイルを Ruby で読んで成形して NetCDF ファイルに出力.
- NetCDF ファイルを GPhys/Ruby スクリプトで読んで図を描く.

- 補足
  - 本来, NetCDF ファイルを仲介する必要はありません.



cron

- cron : 「クロン」「クーロン」
- 予定されたコマンドを実行するデーモン
  - 決まった時刻に指定された作業を行う.
  - 決まった時間間隔で指定された作業を行う.
  - メールで作業を報告する.
- 様々な(システム/個人)処理が実行されている.
  - 例
    - 毎日 HH 時 MM 分にサーバのディスク使用量を確認(して報告のメールを管理者に送る).
    - 毎日 HH 時 MM 分にバックアップのためにファイルを他のサーバと同期(して報告のメールを管理者に送る).

itpass サーバでの cron 実行例

- 毎日実施
  - ディスク使用状況確認
  - quota 状況確認
  - ネットワーク状況確認
  - ログイン失敗状況確認
  - ika（本機）から tako（予備機）へ /home をバックアップ
  - Let’s encrypt の証明書の更新
- 毎週実施
  - パッケージ確認
  - hiki 領域のバックアップ

cron の設定の表示

- crontab コマンドを使用

  $ crontab -l

  – 各ユーザの設定が表示される

出力例

```
$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 7 * * * cd /home/xxx/public_html/covid-19; bash ./all.sh
0 19 * * * cd /home/xxx/public_html/covid-19; bash ./all.sh
```

分 時 日　　月　曜日　実行するコマンド

```
# m  h dom mon dow   command
  0   7   *     *     *     cd /home/xxx/public_html/covid-19; bash ./all.sh
  0 19   *     *     *     cd /home/xxx/public_html/covid-19; bash ./all.sh
```

上の 2 行の時刻指定設定は　“0　7,19　*　*　*”　のように 1 行でも書けるらしい.

cron の設定 1

- crontab コマンドを使用

  $ crontab -e

  – エディタが立ち上がる
    - 立ち上がるエディタは環境変数 EDITOR で決まる.

エディタ画面例

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 7 * * * cd /home/yot/public_html/covid-19; bash ./all.sh
0 19 * * * cd /home/yot/public_html/covid-19; bash ./all.sh
```

分 時 日　　月　曜日　実行するコマンド

```
# m  h dom mon dow   command
  0   7   *     *     *     cd /home/xxx/public_html/covid-19; bash ./all.sh
  0 19   *     *     *     cd /home/xxx/public_html/covid-19; bash ./all.sh
```

上の 2 行の時刻指定設定は　“0　7,19　*　*　*”　のように 1 行でも書けるらしい.

cron の設定 2

定時に実行

分 時 日 月 曜日 実行するコマンド

```
# m    h dom mon dow   command
  15   7  *     *    *     cd /home/xxx/public_html/covid-19; bash ./all.sh
   0 19  5     1    *     cd /home/xxx/public_html/covid-19; bash ./all.sh
   0 1,5 *     *    *     cd /home/xxx/public_html/covid-19; bash ./all.sh
   0 1-5 *     *    *     cd /home/xxx/public_html/covid-19; bash ./all.sh
```

毎日 7 時 15 分に実行
毎月 1 月 5 日 19 時 0 分に実行
毎日 1 時 0 分, 5 時 0 分に実行
毎日 1 時 0 分, 2 時 0 分, …,
5 時 0 分に実行

決めた時間間隔で実行

分 時 日 月 曜日 実行するコマンド

```
# m  h dom mon dow   command
 */5 *  *     *    *     cd /home/xxx/public_html/covid-19; bash ./all.sh
```

5 分ごとに実行

cron の設定ファイルの所在

- システム関係処理の設定ファイル
  - /etc/crontab
  - /etc/cron.* ディレクトリの下のファイル
    - 実行頻度ごとに別ディレクトリに置くのが通例
      - /etc/cron.monthly　月に一度実行
      - /etc/cron.daily　日に一度実行
      - /etc/cron.hourly　時間に一度実行
    - 個々のサーバ固有の（システム関係処理の）設定
      - /etc/cron.local/monthly
      - /etc/cron.local/weekly
      - /etc/cron.local/daily
      - /etc/cron.local/hourly
- 個人の設定ファイル
  - /var/spool/cron/crontabs/* の下のファイル

wget (1/2)

- 非対話型のダウンロードソフトウェア.
- 例えば下のように使う.

```
$ wget http://www.xxx.yyy.ac.jp/~xxx/data.csv
```

- たくさんオプションがあるけれど, ここでは触れない.
  - 今回は何もオプションを付けていない.

wget (2/2)

- 今回ダウンロードしたデータの中には, しばしば DNS による名前解決に失敗するものがあった.
  - wget の “EXIT STATUS” を確認して, ダウンロードに失敗したら 1 分待って再実行.

```
RC=1
irsync=1
while [[ $RC -ne 0 ]]
do
    wget ${csvurl}
    RC=$?
    if [ $irsync -gt 10 ]; then
        echo "ERROR: wget failed 10 times."
        break
    fi
    irsync=$(($irsync+1))
    sleep 60
done
```

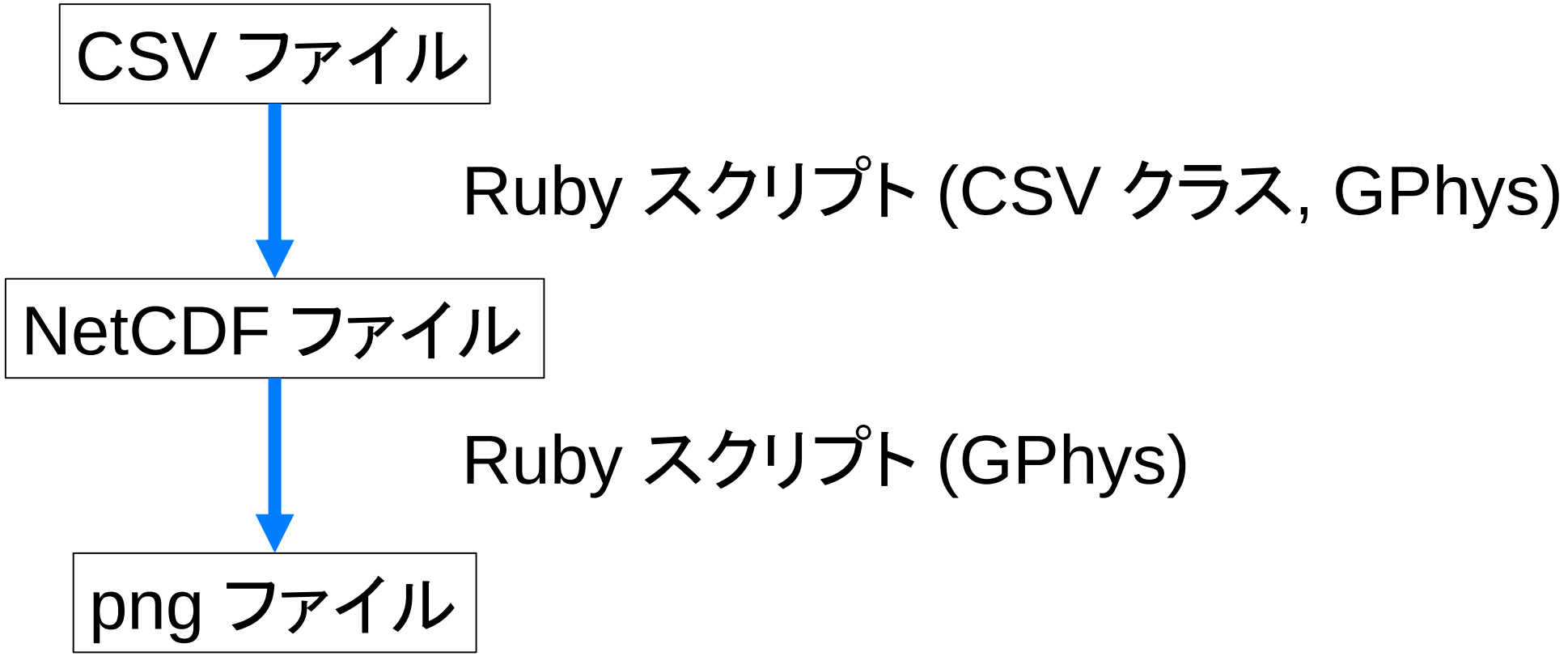
CSV ファイル

- CSV : Comma Separated Value
- よく使われるデータファイルの形式で, 文字通りコンマ (,) で値が区切られている.

European Center for Disease Prevention and Control (ECDC) で公開している COVID-19 データ
(https://opendata.ecdc.europa.eu/covid19/casedistribution/csv)

```
dateRep,day,month,year,cases,deaths,countriesAndTerritories,geoId,countryterritoryCode,popData2018,
28/04/2020,28,4,2020,172,0,Afghanistan,AF,AFG,37172386,Asia
27/04/2020,27,4,2020,68,10,Afghanistan,AF,AFG,37172386,Asia
26/04/2020,26,4,2020,112,4,Afghanistan,AF,AFG,37172386,Asia
…
28/04/2020,28,4,2020,191,25,Japan,JP,JPN,126529100,Asia
27/04/2020,27,4,2020,203,3,Japan,JP,JPN,126529100,Asia
26/04/2020,26,4,2020,290,14,Japan,JP,JPN,126529100,Asia
…
```

CSV ファイルから NetCDF ファイルを介して作図



Ruby CSV クラス

- Ruby で CSV ファイルやデータに対するインターフェースを提供.
- ファイル読み込み例:

```
require ‘csv’

# ファイルから一度に読む
# data_csv は二次元配列
data_csv = CSV.read("file.csv")
```

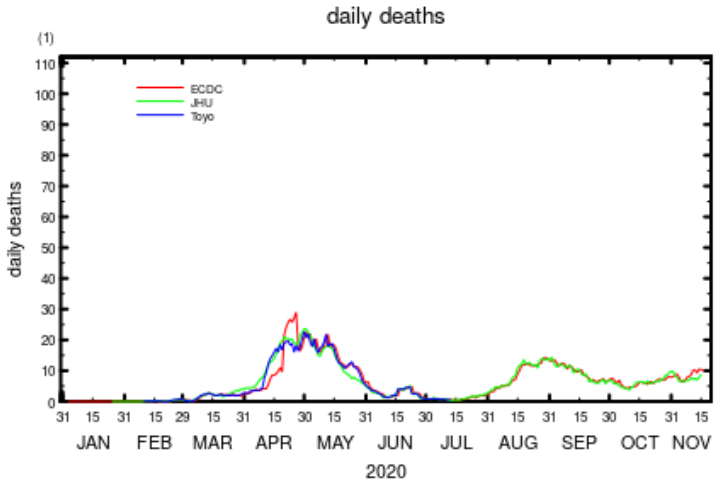
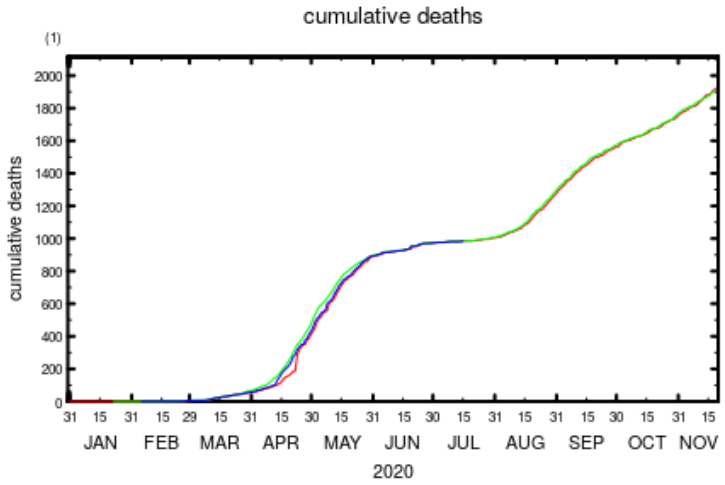
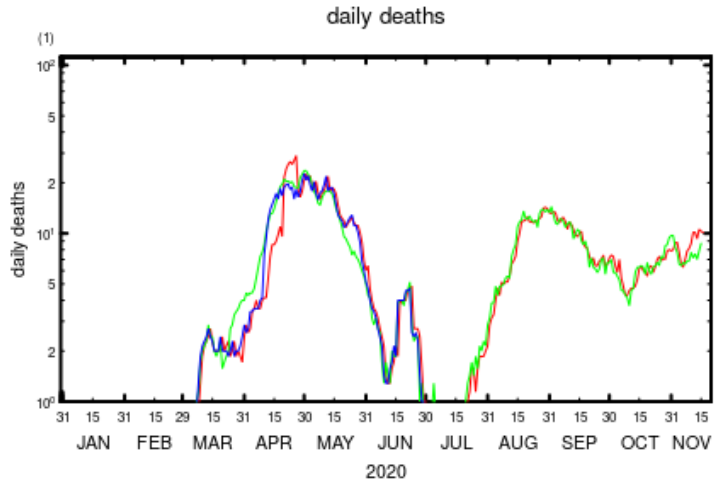
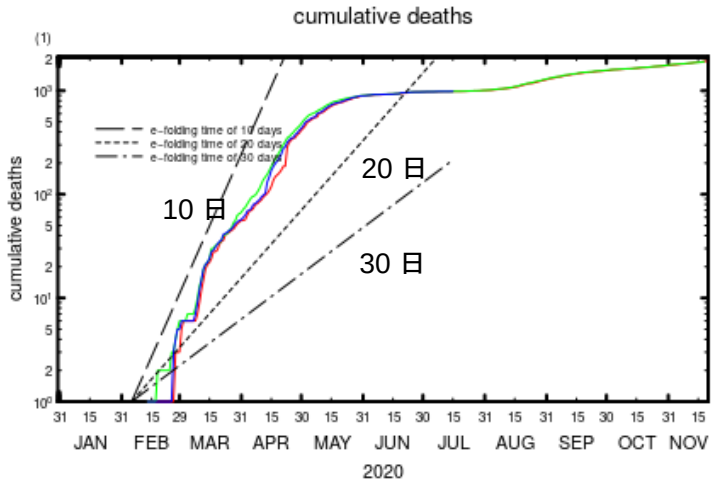
```
require ‘csv’

# ファイルから一行ずつ読んで処理
# row は一次元配列
CSV.foreach(“file.csv”) do |row|
   # use row here...
end
```

- ファイル書き込み例:
  - 使ってないから知らない. 調べればすぐわかるだろう.

今回使ったデータの所在

- European Centre for Disease Prevention and Conrtol (ECDC)
  - データソース
    - たくさん (500 くらいらしい)
      - websites of ministries of health (43% of the total number of sources), websites of public health institutes (9%), websites from other national authorities (ministries of social services and welfare, governments, prime minister cabinets, cabinets of ministries, websites on health statistics and official response teams) (6%), WHO websites and WHO situation reports (2%), and official dashboards and interactive maps from national and international institutions (10%), などなど.
  - https://opendata.ecdc.europa.eu/covid19/casedistribution/csv
- Centers for Systems Science and Engineering at Johns Hopkins University
  - データソース
    - WHO, CDC, ECDC, NHC, DXY, 1point3acres, Worldometers.info, BNO, the COVID Tracking Project (testing and hospitalizations), state and national government health departments, and local media reports.
  - https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv
- 東洋経済 ONLINE
  - データソース
    - 厚生労働省の報道発表資料. ただし, 東京都のみと発表データに基づく.
  - https://raw.githubusercontent.com/kaz-ogiwara/covid19/master/data/summary.csv

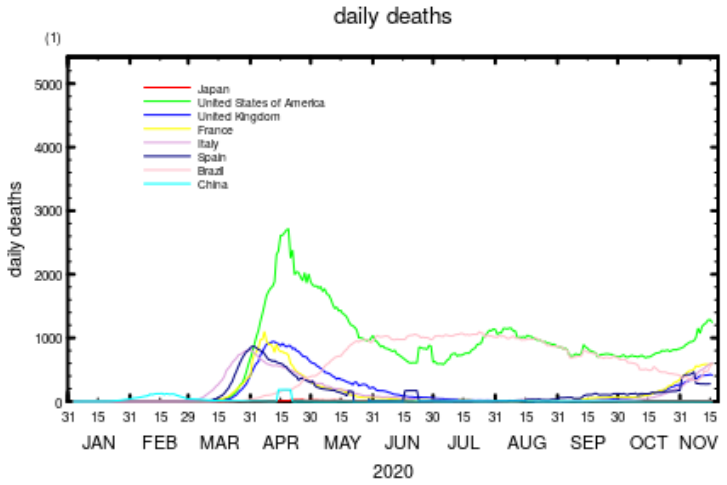
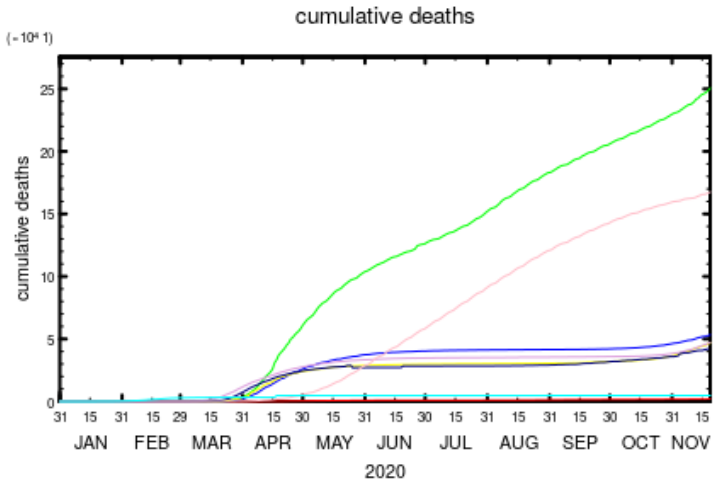
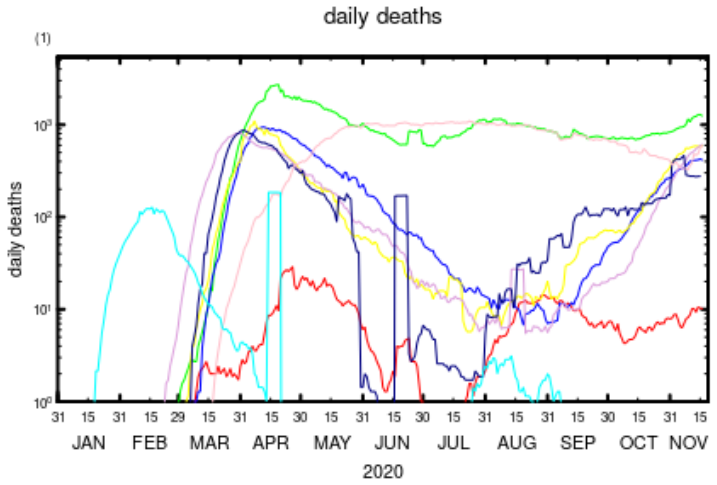
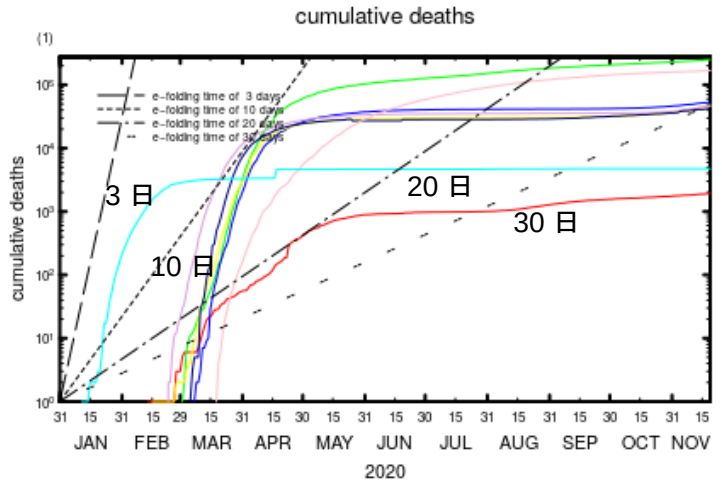
COVID-19 死亡者数の時間変化:
日本

	日毎(7 日間の移動平均)	累積
線形軸		
対数軸		

赤　：ECDC
緑　：JHU
青　：東洋経済

図中の日数は e-folding time

COVID-19 死亡者数の時間変化: 国別 (ECDC データ)

	日毎(7 日間の移動平均)	累積
線形軸		
対数軸		

赤　：日本
緑　：USA
青　：UK
黄　：フランス
紫　：イタリア
濃青　：スペイン
ピンク：中国

図中の日数は e-folding time

まとめ

- cron を使うことで, 定時に決まった作業を実施できる.
  - 決まった作業を定時に行うには cron は便利.
    - 一般ユーザでも (管理者でなくても) 実行できる.
    - 定期的なデータのダウンロードやバックアップに便利.
- 実際, cron, wget, bash スクリプト, Ruby (CSV クラス, GPhys) を使うことで日々更新されるデータの自動ダウンロードや自動可視化ができた.
  - ダウンロードの失敗 (今回は名前解決の失敗) はスクリプトで Exit status を判定などすれば対処可能.

- COVID-19 についてさらに何か?
  - 図をもっと「格好良く」する?
    - CGI でグラフを描く国を選択?
    - インタラクティブなグラフにする?
      - chart.js, plotly, …
  - モデルと比較?