

Sample Code for Covid19 Data Visualization

Sampo Suzuki, CC 4.0 BY-NC-SA

2021-03-02

Introduction

本資料は Covid19 Japan が独自に収集している陽性者のデータ（個票データ）を集計・可視化するためのサンプルコード例です。データ収集処理のためのソースは ソースリポジトリ（GitHub）、収集・処理したデータは データリポジトリ（GitHub） にて公開¹ されています。

¹ ライセンスに関してはリンク先にて確認してください。

Import and tidy

データリポジトリ（GitHub） にて公開されているデータは JSON 形式ですので、読み込みには `jsonlite` パッケージが必要です²。

² `tidyverse` パッケージと共にインストールされます。インストールされていない場合は以下のコードでインストールしてください。

個票データの読み込み

陽性と判定された陽性判定者（以降、陽性者と記述）単位で記録されている個票データは データリポジトリ（GitHub） から直接読み込めます³ が、必ず `raw.githubusercontent.com` のパスを使ってください。 `github.com` パスでは読み込めませんので注意してください。 コードは `read.csv` や `readr::read_csv` を使う場合と同様で下記のようになります。

```
install.packages("jsonlite")`
```

³ 時間帯によりデータが揃っていない場合があります。その場合は、時間帯を変更するか、前日までのデータを利用してください。

```
df <- "https://raw.githubusercontent.com/reustle/covid19japan-data/master/docs/patient_data/latest.json" %>%  
  jsonlite::fromJSON()
```

Table 1: 読み込んだデータの一部のみを表示。

patientId	dateAnnounced	ageBracket	gender	detectedPrefecture	patientStatus
15	2020-01-15	30	M	Kanagawa	Recovered
TOK1	2020-01-24	40	M	Tokyo	Recovered
TOK2	2020-01-25	30	F	Tokyo	Recovered
18	2020-01-26	40	M	Aichi	NA
19	2020-01-28	40	M	Aichi	Hospitalized

読み込んだ個票データを `skimr` パッケージを利用して要約すると下記の

通り⁴です。

⁴ 数値と論理型を除き文字型として扱われますので適切な型に変換する必要があります。

```
df %>%
  skimr::skim()
```

Table 2: Data summary

Name	Piped data
Number of rows	441581
Number of columns	17

Column type frequency:	
character	15
logical	1
numeric	1

Group variables	None


Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
patientId	0	1.00	1	16	0	433660	0
dateAnnounced	0	1.00	10	10	0	399	0
gender	330861	0.25	1	1	0	2	0
detectedPrefecture	0	1.00	3	15	0	49	0
patientStatus	431405	0.02	8	23	0	8	0
mhlwPatientNumber	441132	0.00	1	11	0	434	0
prefecturePatientNumber	324136	0.27	5	20	0	117436	0
residence	338951	0.23	1	38	0	1429	0
relatedPatients	428994	0.03	2	259	0	7470	0
knownCluster	439047	0.01	3	88	0	235	0
detectedCityTown	412548	0.07	2	22	0	667	0
cityPrefectureNumber	412849	0.07	1	34	0	28723	2
deceasedDate	433566	0.02	10	10	0	349	0
deceasedReportedDate	440296	0.00	10	62	0	209	0
deathSourceURL	440439	0.00	14	123	0	659	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
confirmedPatient	0	1	0.98	TRU: 433659, FAL: 7922

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ageBracket	330974	0.25	38.01	20.69	0	20	30	50	100	

データ型の変換

個票のデータフォーマット（GitHub）を参考に適切な型⁵に変換するとともに不要な項目を外しておきます。また、記録には重複や未確定なデータが含まれていますので、これらを外しておく⁶ ことも必要です。

⁵ 日付は日付型、属性は因子型に変換しておく処理しやすくなります。

⁶ `confirmedPatient == TRUE` でフィルタリングします。

```
dft <- df %>%
  dplyr::select(
    patientId, date = dateAnnounced, gender, pref = detectedPrefecture,
    patientStatus, knownCluster, confirmedPatient, ageBracket
  ) %>%
  dplyr::filter(confirmedPatient == TRUE) %>%
  dplyr::mutate(
    date = lubridate::as_date(date),
    gender = forcats::as_factor(gender),
    pref = stringr::str_to_lower(pref),
    patientStatus = forcats::as_factor(patientStatus),
    cluster = dplyr::if_else(!is.na(knownCluster), TRUE, FALSE),
    ageBracket = forcats::as_factor(ageBracket)
  )
```

変換結果を要約すると下記のように適切に変換されていることが分かります。

Table 6: Data summary

Name	Piped data
Number of rows	433659
Number of columns	9

Column type frequency:	
character	3
Date	1
factor	3
logical	2

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
patientId	0	1.00	1	16	0	433659	0
pref	0	1.00	3	15	0	49	0
knownCluster	431154	0.01	3	88	0	233	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
date	0	1	2020-01-15	2021-03-02	2020-12-26	399

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	323987	0.25	FALSE	2	M: 61370, F: 48302
patientStatus	431125	0.01	FALSE	8	Hos: 1261, Dec: 372, Hom: 315, Dis: 283
ageBracket	324084	0.25	FALSE	12	20: 29433, 30: 19042, 40: 16089, 50: 14139

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
confirmedPatient	0	1	1.00	TRU: 433659
cluster	0	1	0.01	FAL: 431154, TRU: 2505

都道府県データの読み込みと変換

個票データの都道府県（`detectedPrefecture`）を利用して都道府県に関するデータ⁷をマージすることで、地方などの地域を切り口とした集計が可能になります。 個票データを除く他のデータ⁸とのマージも考慮して項目名や型を適宜変換しておきます。

⁷ 著者が作成し Gist で公開しているもの。

⁸ Google 感染予測 など

```

prefs <- "https://gist.githubusercontent.com/k-metrics/9f3fc18e042850ff24ad9676ac34764b/raw/f4ea87f429e1ca28627feff94b67c8"
readr::read_csv() %>%
  dplyr::mutate(
    # Google の予測データと結合を考慮してコード体系を合わせておく
    japan_prefecture_code = paste0("JP-", `コード`)
  ) %>%
  dplyr::select(
    # Google の予測データと結合するために名称を変更する

```

```
japan_prefecture_code, prefecture_name = pref,
# 日本語の変数名は扱いにくいので英語名に変更する
pref = `都道府県`, region = `八地方区分`, pops = `推計人口`
) %>%
dplyr::mutate(
# 水準ごとに表示させるために因子化する（あらかじめデータをコード順に
# 並べておくことが因子化の際のポイントのひとつ）
japan_prefecture_code = forcats::fct_inorder(japan_prefecture_code),
pref = forcats::fct_inorder(pref),
region = forcats::fct_inorder(region),
pops = as.integer(pops)
)
```

48 番目に都道府県を除く空港検疫などからの報告数をまとめるための「空港検疫」を用意してあります。人口（pops）は千人単位になっています。

Table 11: 都道府県データ

japan_prefecture_code	prefecture_name	pref	region	pops
JP-01	hokkaido	北海道	北海道地方	5286
JP-02	aomori	青森県	東北地方	1263
JP-03	iwate	岩手県	東北地方	1241
JP-04	miyagi	宮城県	東北地方	2316
JP-05	akita	秋田県	東北地方	981
JP-06	yamagata	山形県	東北地方	1090
JP-07	fukushima	福島県	東北地方	1864
JP-08	ibaraki	茨城県	関東地方	2877
JP-09	tochigi	栃木県	関東地方	1946
JP-10	gunma	群馬県	関東地方	1952
JP-11	saitama	埼玉県	関東地方	7330
JP-12	chiba	千葉県	関東地方	6255
JP-13	tokyo	東京都	関東地方	13822
JP-14	kanagawa	神奈川県	関東地方	9177
JP-15	niigata	新潟県	中部地方	2246
JP-16	toyama	富山県	中部地方	1050
JP-17	ishikawa	石川県	中部地方	1143
JP-18	fukui	福井県	中部地方	774
JP-19	yamanashi	山梨県	中部地方	817
JP-20	nagano	長野県	中部地方	2063
JP-21	gifu	岐阜県	中部地方	1997
JP-22	shizuoka	静岡県	中部地方	3659
JP-23	aichi	愛知県	中部地方	7537
JP-24	mie	三重県	近畿地方	1791
JP-25	shiga	滋賀県	近畿地方	1412

japan_prefecture_code	prefecture_name	pref	region	pops
JP-26	kyoto	京都府	近畿地方	2591
JP-27	osaka	大阪府	近畿地方	8813
JP-28	hyogo	兵庫県	近畿地方	5484
JP-29	nara	奈良県	近畿地方	1339
JP-30	wakayama	和歌山県	近畿地方	935
JP-31	tottori	鳥取県	中国地方	560
JP-32	shimane	島根県	中国地方	680
JP-33	okayama	岡山県	中国地方	1898
JP-34	hiroshima	広島県	中国地方	2817
JP-35	yamaguchi	山口県	中国地方	1370
JP-36	tokushima	徳島県	四国地方	736
JP-37	kagawa	香川県	四国地方	962
JP-38	ehime	愛媛県	四国地方	1352
JP-39	kochi	高知県	四国地方	706
JP-40	fukuoka	福岡県	九州地方	5107
JP-41	saga	佐賀県	九州地方	819
JP-42	nagasaki	長崎県	九州地方	1341
JP-43	kumamoto	熊本県	九州地方	1757
JP-44	oita	大分県	九州地方	1144
JP-45	miyazaki	宮崎県	九州地方	1081
JP-46	kagoshima	鹿児島県	九州地方	1614
JP-47	okinawa	沖縄県	九州地方	1448
JP-48	NA	空港検疫	NA	NA

データの結合

前出の個票データと都道府県データを結合します。結合の際に因子型変数の水準が欠損値となっているものは適宜 `forcats` パッケージを用いて水準を割り当てておきます。

```
x <- dft %>%
  dplyr::left_join(prefs, by = c("pref" = "prefecture_name")) %>%
  dplyr::select(-pref) %>%
  dplyr::rename(pref = pref.y) %>%
  # 因子型の欠損値を水準化しておく
  dplyr::mutate(
    japan_prefecture_code = forcats::fct_explicit_na(japan_prefecture_code,
                                                       na_level = "JP-48"),
    pref = forcats::fct_explicit_na(pref, na_level = "空港検疫"),
    region = forcats::fct_explicit_na(region, na_level = "空港検疫"),
    gender = forcats::fct_explicit_na(gender, na_level = "非公表"),
```

```
ageBracket = forcats::fct_explicit_na(ageBracket, na_level = "非公表"),
patientStatus = forcats::fct_explicit_na(patientStatus,
                                          na_level = "Unknown")
)
```

これで集計対象となる個票データの完成です。

Table 12: 完成した個票データの一部

date	pref	region	ageBracket	gender
2021-01-17	神奈川県	関東地方	非公表	非公表
2021-01-18	埼玉県	関東地方	非公表	非公表
2021-01-24	埼玉県	関東地方	非公表	非公表
2020-12-28	福岡県	九州地方	非公表	非公表
2021-01-16	熊本県	九州地方	非公表	非公表
2020-07-29	東京都	関東地方	70	M
2021-02-04	福岡県	九州地方	非公表	非公表
2021-02-27	福島県	東北地方	非公表	非公表
2020-12-03	千葉県	関東地方	非公表	非公表
2021-01-29	愛知県	中部地方	非公表	非公表

Data Wrangling

完成した個票データを用いて様々な集計を行います。

日次集計（単純集計）

日付（date）を用いた単純な日次集計を行います。この集計結果は厚生労働省オープンデータの陽性者数データに相当⁹します。

集計には `dplyr::group_by` と `dplyr::summarize(n = n())`¹⁰ を用いることで簡単に計数することができます。ただし、個票データが存在しない日付は計数できませんので暗黙の欠落（欠損）となり明示的な欠落（欠損）に変換する¹¹ 必要があります。この処理は `tidyr::complete` を用いることで簡単にできます。

```
japan_daily <- x %>%
  dplyr::group_by(date) %>%
  dplyr::summarise(n = dplyr::n()) %>%
  dplyr::ungroup() %>%
  tidyr::complete(
    date = seq.Date(from = min(date), to = max(date), by = "day"),
```

⁹ 厚生労働省の場合、途中から個票からの集計を取りやめ各自治体が公表している集計値を積上げたものを公開しています。

¹⁰ `dplyr::count` を用いても同様の結果を得ることができます。

¹¹ Turns implicit missing values into explicit missing values.

```
fill = list(n = 0L)
)
```

dplyr::group_by と dplyr::summarize を行った後に続けて処理を行う場合には dplyr::ungroup で必ずアングループしてください。

Table 13: 最初の 10 日間の集計結果

date	n
2020-01-15	1
2020-01-16	0
2020-01-17	0
2020-01-18	0
2020-01-19	0
2020-01-20	0
2020-01-21	0
2020-01-22	0
2020-01-23	0
2020-01-24	1

前日差 (diff) ・累計 (cumsum) ・移動平均 (7 日 (ma7) ならびに 28 日 (ma28)) ¹² も求めます。

```
japan_daily <- japan_daily %>%
  dplyr::mutate(
    diff = lagdiff(n), # 前日差
    cum = cumsum(n),   # 累計
    ma7 = ma7(n),      # 移動平均 (7 日)
    ma28 = ma28(n)     # 移動平均 (28 日)
  )
```

¹² lagdiff, ma7, ma28 は以下のような関数として定義してあります。

```
lagdiff <- function(n) { n -
  dplyr::lag(n, default = 0L) }

ma7 <- function(n) { zoo::rollmeanr(n,
  k = 7L, na.pad = TRUE) }

ma28 <- function(n) {
  zoo::rollmeanr(n, k = 28L, na.pad =
  TRUE) }
```

Table 14: 最初の 10 日間の集計結果と計算結果

date	n	diff	cum	ma7	ma28
2020-01-15	1	1	1	NA	NA
2020-01-16	0	-1	1	NA	NA
2020-01-17	0	0	1	NA	NA
2020-01-18	0	0	1	NA	NA
2020-01-19	0	0	1	NA	NA
2020-01-20	0	0	1	NA	NA
2020-01-21	0	0	1	0.1	NA
2020-01-22	0	0	1	0.0	NA
2020-01-23	0	0	1	0.0	NA
2020-01-24	1	1	2	0.1	NA

クロス集計

都道府県ごとの日次集計、年代別の日次集計など複数の変数の水準ごとの集計をクロス集計と呼びます。このクロス集計も単純集計と同様に `dplyr::group_by` と `dplyr::summrize(n =)`¹³ で計数することができます。

¹³ 単純集計と同様に `dplyr::count` を用いても同様の結果を得ることができます。

日次・地方区分別集計

地方区分 (region) を `dplyr::group_by` に追加指定するだけでクロス集計ができます。暗黙の欠落 (欠損) を明示的な欠落 (欠損) に変換する場合も `tidyr::complt` に地方区分 (region) を追加するだけ済みます。

```
region_daily <- x %>%
  dplyr::group_by(date, region) %>%
  dplyr::summarise(n = dplyr::n()) %>%
  dplyr::ungroup() %>%
  tidyr::complete(
    date = seq.Date(from = min(date), to = max(date), by = "day"), region,
    fill = list(n = 0L)
  ) %>%
  dplyr::group_by(region) %>%
  dplyr::mutate(
    diff = lagdiff(n),
    cum = cumsum(n),
    ma7 = ma7(n),
    ma28 = ma28(n)
  ) %>%
  dplyr::ungroup()
```

Table 15: 最初の 10 レコード

date	region	n	diff	cum	ma7	ma28
2020-01-15	北海道地方	0	0	0	NA	NA
2020-01-15	東北地方	0	0	0	NA	NA
2020-01-15	関東地方	1	1	1	NA	NA
2020-01-15	中部地方	0	0	0	NA	NA
2020-01-15	近畿地方	0	0	0	NA	NA
2020-01-15	中国地方	0	0	0	NA	NA
2020-01-15	四国地方	0	0	0	NA	NA
2020-01-15	九州地方	0	0	0	NA	NA
2020-01-15	空港検疫	0	0	0	NA	NA

date	region	n	diff	cum	ma7	ma28
2020-01-16	北海道地方	0	0	0	NA	NA

日次・都道府県別集計

同様に都道府県別の日次集計を行います。

Table 16: 最初の 10 レコード

pref	date	n	diff	cum	ma7	ma28
北海道	2020-01-15	0	0	0	NA	NA
北海道	2020-01-16	0	0	0	NA	NA
北海道	2020-01-17	0	0	0	NA	NA
北海道	2020-01-18	0	0	0	NA	NA
北海道	2020-01-19	0	0	0	NA	NA
北海道	2020-01-20	0	0	0	NA	NA
北海道	2020-01-21	0	0	0	0	NA
北海道	2020-01-22	0	0	0	0	NA
北海道	2020-01-23	0	0	0	0	NA
北海道	2020-01-24	0	0	0	0	NA

Visualize

集計結果を可視化してみます。

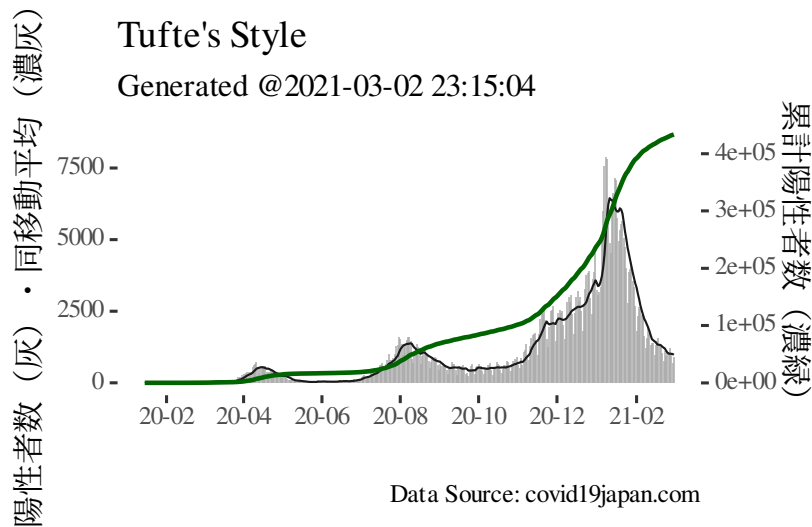
日次集計の可視化

日次集計データ (japan_daily) の単日陽性者数 (n)・累計陽性者数 (cum)・単日移動平均 (7 日) (ma7) を描きます。単日陽性者数は棒グラフ、累計陽性者数と単日移動平均は折線グラフで描きますが、累計の最大値は単日や移動平均と 2 桁異なりますので二軸のグラフとして描く必要があります。

```
title <- "Tufte's Style"
xlab <- ""
ylab <- ""
sec_scale <- 50 # 縦二軸用のスケール値
dbreaks <- "2 month"
dlabels <- "%y-%m"
axis1 <- "陽性者数 (灰)・同移動平均 (濃灰)"
```

```
axis2 <- "累計陽性者数（濃緑）"
```

```
japan_daily %>%
  ggplot2::ggplot(ggplot2::aes(x = date)) +
    ggplot2::geom_bar(ggplot2::aes(y = n), stat = "identity", width = 1.0,
                      fill = "dark gray", alpha = 0.75) +
    ggplot2::geom_line(ggplot2::aes(y = ma7), linetype = "solid",
                      colour = "gray10", size = 0.35) +
    # 第二軸を利用するグラフを描画する際はスケーリング調整する
    ggplot2::geom_line(ggplot2::aes(y = cum / sec_scale),
                      colour = "dark green", size = 0.75) +
    # 横軸表示の指定
    ggplot2::scale_x_date(date_breaks = "1 month", date_labels = "%y/%m") +
    # 二軸表示のための軸属性の指定
    ggplot2::scale_y_continuous(
      # 第一軸のラベル（スケールは自動調整）
      name = axis1,
      # 第二軸の指定（第一軸にスケーリング値をかけたスケール）
      sec.axis = ggplot2::sec_axis(~ . * sec_scale,
                                   name = axis2)) +
    ggthemes::theme_tufte() +
    ggplot2::scale_x_date(date_breaks = dbreaks, date_labels = dlabels) +
    ggplot2::labs(title = title, subtitle = subtitle, caption = caption,
                  x = xlab, y = ylab)
```



上グラフは `ggthemes::theme_tufte` を適用し Tufte スタイルで描画しています

Figure 1: 【全国】陽性者数の推移（単日／累計／移動平均（7日））

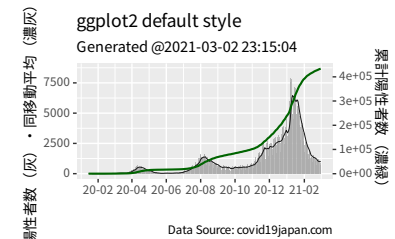


Figure 2: Draw with default theme.

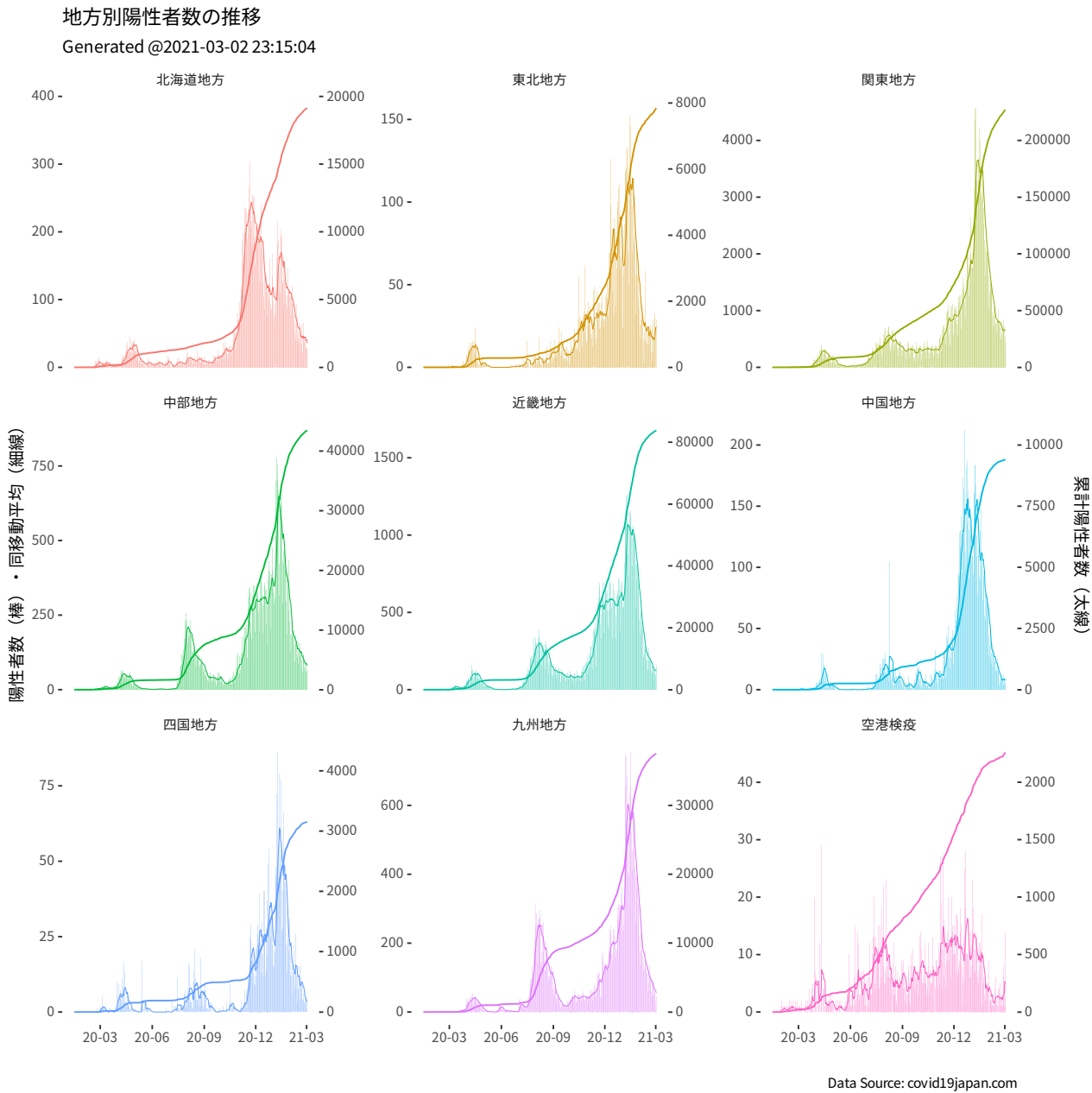
す。右側のデフォルトテーマ (`ggplot2::theme_gray`) で描画したグラフと比べるとインクレシオが高いことが分かります。

クロス集計の可視化（地方別）

`ggthemes::theme_tufte` は凡例を非表示にできませんので `ggplot2::theme` を利用して Tufte スタイルに近い設定にします。

```
title <- " 地方別陽性者数の推移"
xlab <- ""
ylab <- ""
sec_scale <- 50      # 縦二軸用のスケーリング値
ncol <- 3
dbreaks <- "3 month"
dlabels <- "%y-%m"
axis1 <- " 陽性者数（棒）・同移動平均（細線）"
axis2 <- " 累計陽性者数（太線）"

region_daily %>%
  dplyr::mutate(key = region) %>%
  ggplot2::ggplot(ggplot2::aes(x = date)) +
    ggplot2::geom_bar(ggplot2::aes(y = n, fill = key), stat = "identity",
                      alpha = 0.25, width = 1.0) +
    ggplot2::geom_line(ggplot2::aes(y = ma7, colour = key),
                      linetype = "solid", size = 0.25) +
    ggplot2::geom_line(ggplot2::aes(y = cum / sec_scale, colour = key)) +
    ggplot2::scale_x_date(date_breaks = dbreaks, date_labels = dlabels) +
    ggplot2::theme_gray() +
    ggplot2::theme(panel.grid = ggplot2::element_blank(),
                  panel.background = ggplot2::element_blank(),
                  strip.background = ggplot2::element_blank(),
                  legend.position = "none") +
    ggplot2::facet_wrap(~ key, ncol = ncol, scales = "free_y") +
    ggplot2::scale_y_continuous(
      name = axis1,
      sec.axis = ggplot2::sec_axis(~ . * sec_scale, name = axis2)) +
    ggplot2::labs(title = title, subtitle = subtitle, caption = caption,
                  x = xlab, y = ylab)
```



クロス集計の可視化（都道府県別）

```

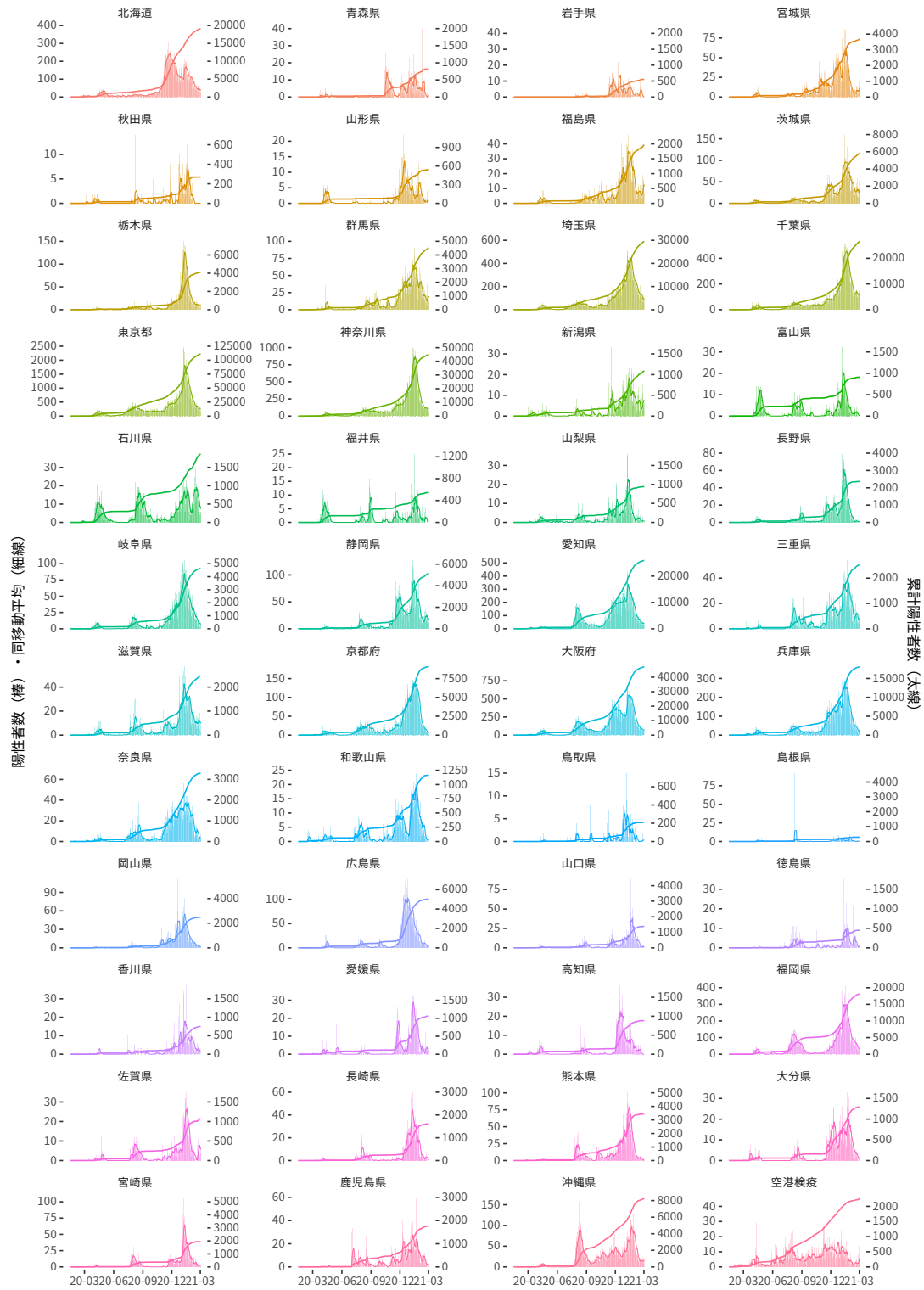
title <- " 都道府県別陽性者数の推移"
xlab <- ""
ylab <- ""
sec_scale <- 50      # 縦二軸用のスケーリング値
ncol <- 4
dbreaks <- "3 month"
dlabels <- "%y-%m"
axis1 <- " 陽性者数（棒）・同移動平均（細線）"
axis2 <- " 累計陽性者数（太線）"

pref_daily %>%
  dplyr::mutate(key = pref) %>%
  ggplot2::ggplot(ggplot2::aes(x = date)) +
    ggplot2::geom_bar(ggplot2::aes(y = n, fill = key), stat = "identity",
                      alpha = 0.25, width = 1.0) +
    ggplot2::geom_line(ggplot2::aes(y = ma7, colour = key),
                      linetype = "solid", size = 0.25) +
    ggplot2::geom_line(ggplot2::aes(y = cum / sec_scale, colour = key)) +
    ggplot2::scale_x_date(date_breaks = dbreaks, date_labels = dlabels) +
    ggplot2::theme_gray() +
    ggplot2::theme(panel.grid = ggplot2::element_blank(),
                  panel.background = ggplot2::element_blank(),
                  strip.background = ggplot2::element_blank(),
                  legend.position = "none") +
    ggplot2::facet_wrap(~ key, ncol = ncol, scales = "free_y") +
    ggplot2::scale_y_continuous(
      name = axis1,
      sec.axis = ggplot2::sec_axis(~ . * sec_scale, name = axis2)) +
    ggplot2::labs(title = title, subtitle = subtitle, caption = caption,
                  x = xlab, y = ylab)

```

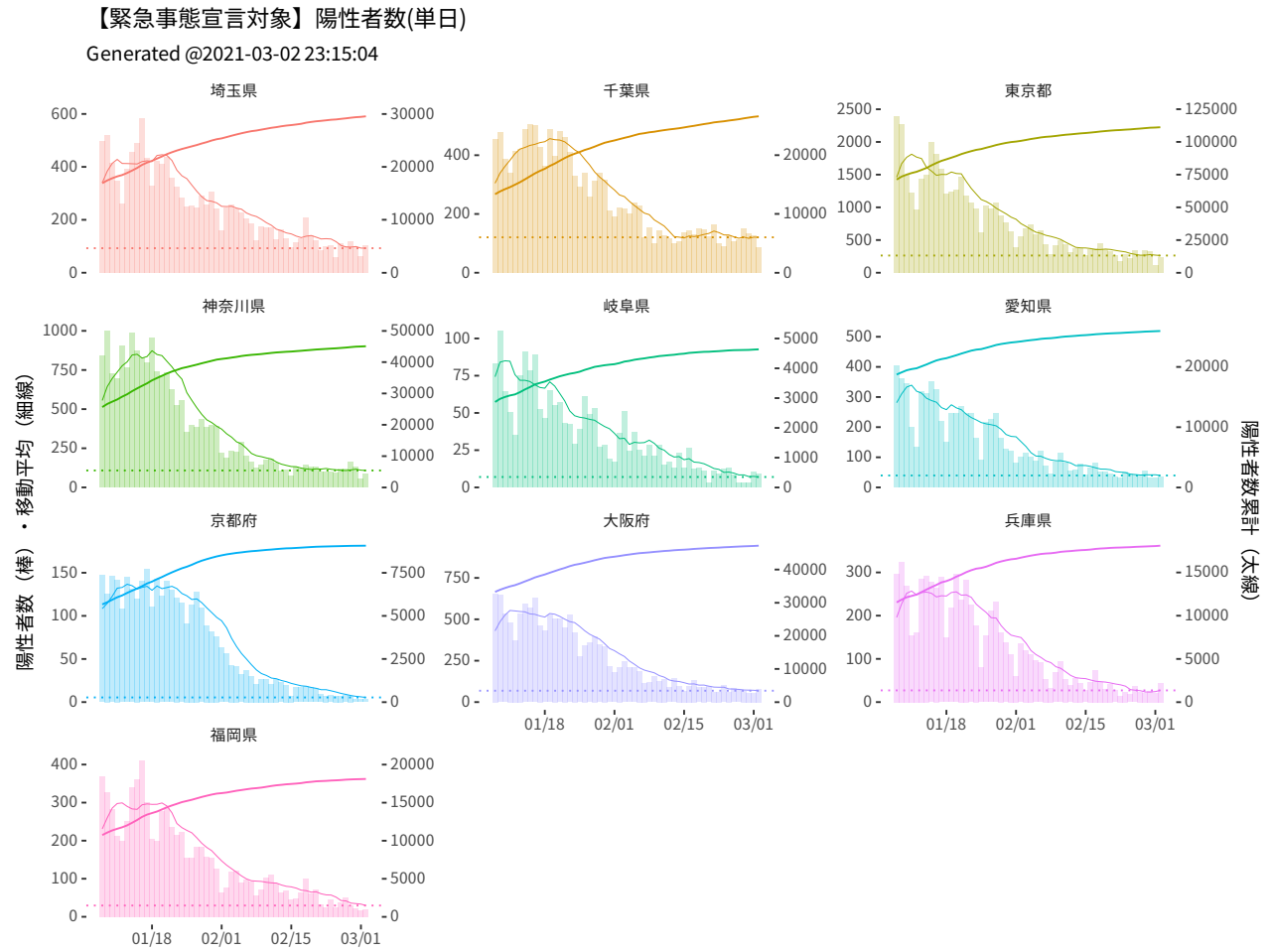
都道府県別陽性者数の推移

Generated @2021-03-02 23:15:04



Data Source: covid19japan.com

緊急事態宣言地域（前倒解除府県を含む）



Data Source: covid19japan.com