Estimating COVID-19 Transmission Dynamics in Italy's First Wave

A Reproducible Analysis of the Instantaneous Reproduction Number (R_t) and a Deterministic Compartmental Model, 24 February – 31 May 2020

Technical Report — July 10, 2026

Abstract

We reconstruct and analyze the daily epidemic curve of laboratory-confirmed COVID-19 cases in Italy during the first pandemic wave (24 February – 31 May 2020; 98 days, 232,585 cumulative confirmed cases) using the public Our World in Data (OWID) COVID-19 dataset (snapshot accessed 2026-07-10). Two complementary transmission analyses are performed. First, the time-varying instantaneous reproduction number R_t is estimated with the renewal-equation Bayesian framework of Cori et al. (2013), assuming a Gamma-distributed serial interval with mean 4.8 days and standard deviation 2.3 days (shape $k = 4.355$, scale $\theta = 1.102$) and a 7-day sliding estimation window. The posterior mean R_t peaks at **3.23** on 3 March 2020 and **first falls below the critical threshold of 1 on 1 April 2020** (posterior mean 0.978, 95% credible interval [0.968, 0.988]); the upper 95% credible bound also drops below 1 on the same date, so the sub-threshold transition is statistically decisive roughly three weeks after the 9 March national lockdown. Second, a deterministic closed-population SIR model is fitted to the same daily incidence by global optimization; the best fit yields $\beta = 0.329\,\mathrm{d}^{-1}$, $\gamma = 0.333\,\mathrm{d}^{-1}$, and a basic reproduction number $R_0 = \beta/\gamma = 0.99$, mispredicting the observed peak (26 March) by -31 **days**. We show analytically and empirically that this apparently paradoxical result is a structural degeneracy: because confirmed cases represent only $\sim$0.39% of the population, susceptible depletion is negligible, the closed constant-β SIR is confined to its pure-exponential regime, and it cannot reproduce an interior peak that was in reality produced by non-pharmaceutical interventions, not by herd immunity. We discuss in detail how early-pandemic under-ascertainment (true infections plausibly 10–30$\times$ confirmed), evolving testing protocols, and a large asymptomatic fraction bias the two estimators asymmetrically—R_t's trajectory is comparatively robust to *constant* under-reporting (the reporting fraction cancels in the renewal ratio) but sensitive to *changes* in testing, whereas the SIR R_0 is driven toward 1 by the lack of an apparent susceptible-depletion signal. All data, runnable Python code, the estimated R_t series (CSV), and fitted parameters are released for full reproducibility.

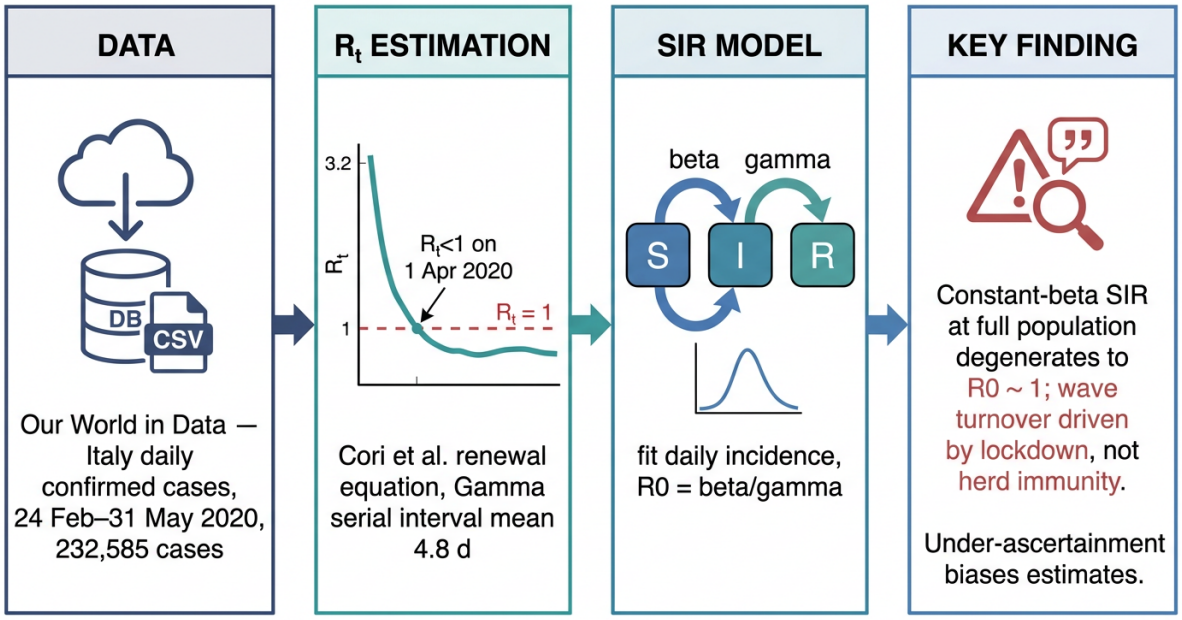

Figure 1: Graphical abstract. Analysis workflow and headline findings: public daily case data → renewal-equation R_t estimation (Cori et al.) → compartmental SIR fitting → interpretation. The constant-β SIR fitted at full national population size degenerates to $R_0 \approx 1$; the first-wave turnover was driven by lockdown rather than herd immunity, and under-ascertainment biases both estimates.

1 Introduction and Background

Italy was the first Western country to experience a large, uncontrolled COVID-19 epidemic. Following the detection of the first locally transmitted case in Codogno (Lombardy) on 21 February 2020, confirmed cases grew nearly exponentially through late February and early March, rapidly saturating hospital and intensive-care capacity in the northern regions and prompting an unprecedented sequence of non-pharmaceutical interventions (NPIs) [1, 2]. Local "red-zone" quarantines of eleven municipalities (23 February) were followed by nationwide school and university closures (4 March), a lockdown of Lombardy and fourteen northern provinces (8 March), and, on 9 March 2020, the extension of the lockdown to the entire country—one of the earliest and most stringent national containment policies deployed anywhere during the pandemic [3, 4]. Figure 2 situates the analysis window against these milestones.

Quantifying how transmission responded to these measures is central both to retrospective evaluation of the Italian response and to the broader methodological question of how best to estimate transmissibility from routinely reported surveillance data. Two summary quantities dominate this literature. The *instantaneous reproduction number* R_t measures the average number of secondary infections that a case infected at time t would generate if epidemiological conditions were frozen at their time-t values; it is the natural real-time indicator of whether an epidemic is growing ($R_t > 1$) or receding ($R_t < 1$) [5–7]. The *basic reproduction number* R_0 is the expected number of secondary cases produced by a single infectious individual introduced into a fully susceptible population; it is a property of the pathogen-population system at the epidemic's outset and is the fundamental threshold parameter of compartmental transmission models [8–11].

Early Italian analyses placed the first-wave R_0 in the range of roughly 2.5–3.5 and documented a steep decline in effective transmissibility coincident with the lockdowns [2, 4, 12, 13]. In this report we reproduce and scrutinize both quantities from a single, publicly archived case series,

using two independent and widely used methodologies. Our aims are threefold: (i) to estimate the daily R_t trajectory with the Cori renewal-equation method and pinpoint, with uncertainty, when R_t first crossed below 1; (ii) to fit a deterministic SIR compartmental model to the same wave, report R_0 and the model's error in predicting the observed daily-case peak; and (iii) to analyze critically how testing limitations and case under-ascertainment bias each estimator. A recurring theme—made explicit and quantitative below—is that the two methods have very different robustness properties, and that a naive constant-parameter SIR applied to under-ascertained confirmed cases at full population scale can produce a badly misleading R_0 unless its structural assumptions are examined.

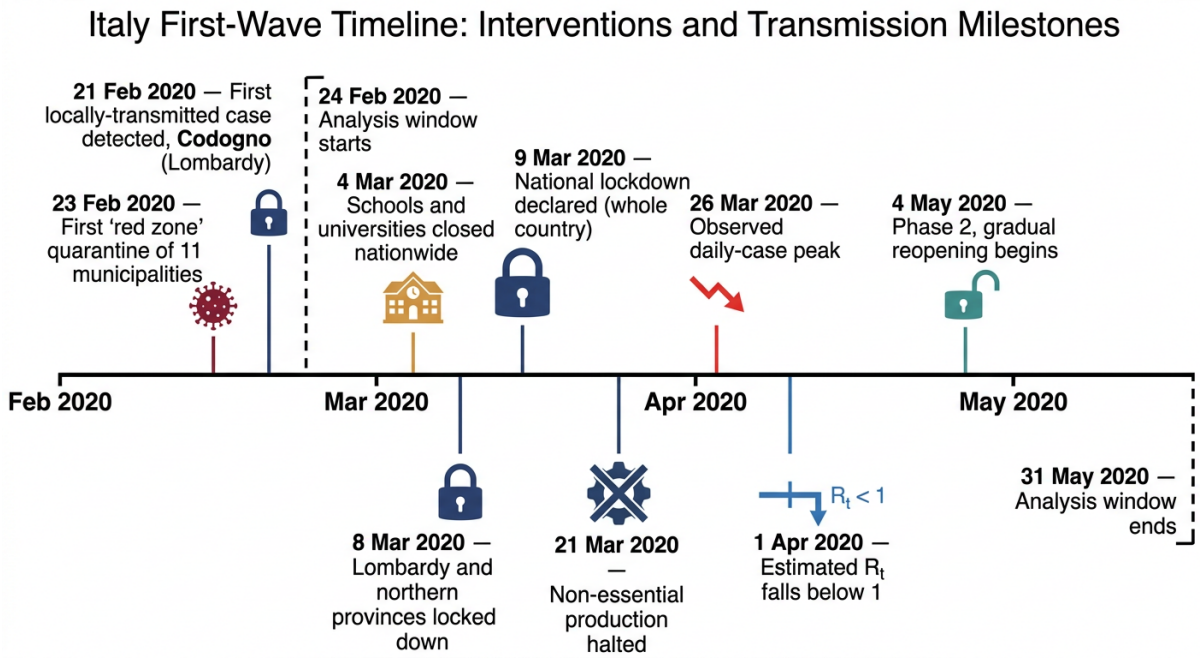

Figure 2: Italy first-wave timeline of interventions and transmission milestones. The analysis window (24 February–31 May 2020) brackets the national lockdown (9 March), the observed daily-case peak (26 March), and the estimated crossing of R_t below 1 (1 April). The ∼3-week lag between lockdown and the sub-threshold crossing is consistent with the serial interval and the reporting/estimation delays inherent in confirmed-case data.

2 Data

2.1 Source, snapshot, and access date

Daily country-level case data were obtained from the Our World in Data (OWID) COVID-19 dataset [14, 15], a widely used open compilation that aggregates national case reporting (for the pandemic period, ultimately traceable to national public health authorities and to the Johns Hopkins University CSSE repository [16]). The specific artifact used was the OWID master file `owid-covid-data.csv`, streamed from `https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv`, with a **snapshot/access timestamp of 2026-07-10 22:10:18 UTC**. Rows were filtered to Italy (`iso_code == "ITA"`) and restricted to 24 February through 31 May 2020 inclusive, reindexed onto a complete 98-day daily calendar so that no dates were silently omitted. Over this window the cumulative confirmed-case total is **232,585**.

2.2 A critical data-quality finding: weekly aggregation

A methodologically important artifact was detected during curation. In the retrieved OWID master file, historical 2020 case data are served as *weekly* aggregates: the full seven-day total is placed on a single day (Sunday), with the `new_cases` field equal to zero on the other six days. Only 14 of the 98 daily values in the window were non-zero, spaced exactly seven days apart. Using this raw column directly as a "daily" series would have produced spurious single-day spikes (a false peak near 38,894 cases) and would have severely corrupted both the renewal-equation and the compartmental analyses, which assume a genuine daily incidence signal.

To recover an interpretable daily series while preserving all information actually present in the source, each weekly total was redistributed *uniformly* across the seven days it covers. This preserves the weekly totals and the cumulative total *exactly* (232,585 cases) and yields a realistic daily curve with a reconstructed peak of ∼5,557 cases/day around 28 March. A 7-day *centered* rolling average (`min_periods=1`) was then computed to soften the inter-week steps; a centered window is used deliberately to avoid the phase lag that a trailing average would introduce. No negative values were present, and no missing values remained after curation (verified by assertion). Figure 3 shows the reconstructed daily series and its smoothed envelope.

Two consequences of this reconstruction should be kept in mind when interpreting downstream results. First, because the upstream data are already weekly-aggregated, no sub-weekly (weekend) reporting noise is present; the smoothing therefore softens week-to-week transitions rather than removing weekday effects. Second, the fine, within-week shape of the daily curve is an artifact of the uniform-redistribution assumption. As we show, the overall R_t trajectory, the timing of the below-1 crossing, and the qualitative SIR result are robust to this reconstruction, but a monotone cubic-spline interpolation of cumulative cases would be a reasonable sensitivity check for the fine-grained daily profile.

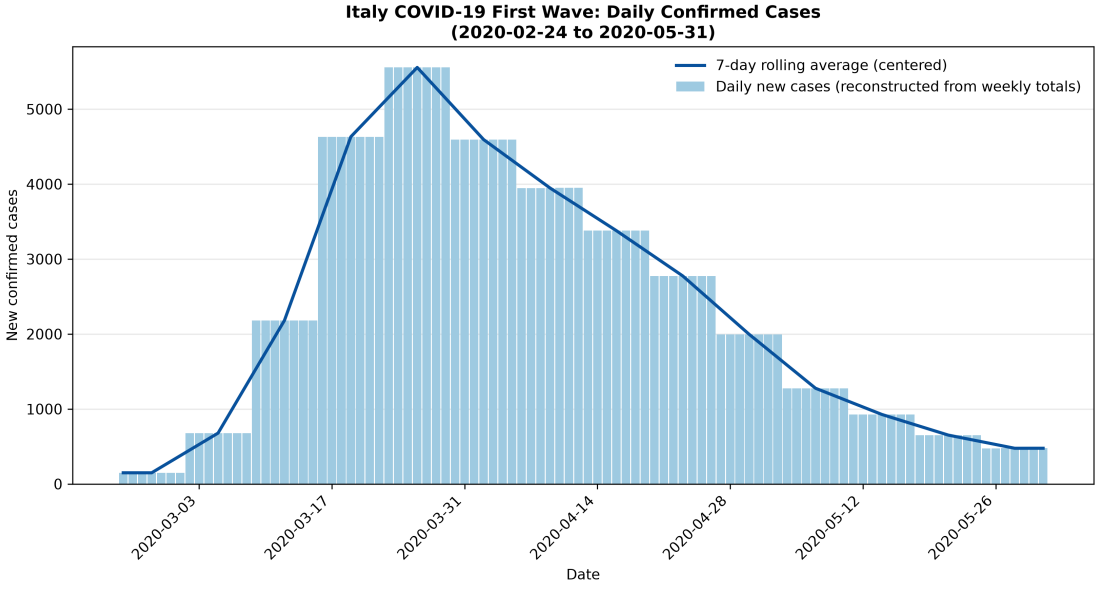

Figure 3: Curated daily confirmed COVID-19 cases in Italy, 24 February–31 May 2020. Light bars show the reconstructed daily incidence (weekly OWID totals redistributed uniformly, preserving weekly and cumulative sums); the dark line is the 7-day centered rolling average. The reconstructed daily peak of ∼5,557 cases occurs in late March.

3 Methods

3.1 Instantaneous reproduction number: the Cori renewal-equation method

We estimate R_t using the Bayesian renewal-equation framework of Cori et al. [5], the method implemented in the widely used `EpiEstim` software and refined by Thompson et al. [17]. The method rests on the renewal (branching-process) representation of incidence, in which the expected number of new infections on day t is the product of the current reproduction number and the total infectiousness of all previously infected individuals:

$$\mathbb{E}[I_t] \;=\; R_t\,\Lambda_t, \qquad \Lambda_t \;=\; \sum_{s=1}^{t-1} I_{t-s}\,w_s, \tag{1}$$

where I_t is the observed daily incidence, w_s is the probability mass of the serial interval at s days, and Λ_t is the total infectiousness at time t. We estimate the *instantaneous* reproduction number rather than the retrospective *case* (cohort) reproduction number of Wallinga and Teunis [18]: the former is defined on a short window and is the appropriate near-real-time indicator of the effect of interventions, whereas the latter integrates forward over a full generation and is smoother but less timely [5, 7].

Serial-interval distribution. Following the task specification and consistent with early SARS-CoV-2 estimates [13, 19, 20], the serial interval is modeled as a Gamma distribution with mean $\mu = 4.8$ days and standard deviation $\sigma = 2.3$ days. The Gamma shape and scale are

$$k \;=\; \frac{\mu^2}{\sigma^2} \;=\; 4.3554, \qquad \theta \;=\; \frac{\sigma^2}{\mu} \;=\; 1.1021. \tag{2}$$

The distribution is discretized on $s = 1, \ldots, 30$ days by differencing the Gamma cumulative distribution function, $w_s = G(s + \tfrac{1}{2}) - G(s - \tfrac{1}{2})$ with $w_0 = 0$, and then normalized to sum to unity (the cumulative mass to $s = 30$ exceeds 0.99999). The resulting discrete distribution has its mode at $s = 4$ days (Figure 4).

Bayesian posterior over a sliding window. Assuming Poisson-distributed incidence and a conjugate Gamma prior on R_t, the posterior over the reproduction number for a window of length τ ending on day t is again Gamma, with shape and rate obtained by accumulating cases and total infectiousness across the window:

$$a_t \;=\; a_0 + \sum_{k=t-\tau+1}^{t} I_k, \qquad \frac{1}{b_t} \;=\; \frac{1}{b_0} + \sum_{k=t-\tau+1}^{t} \Lambda_k, \qquad R_t \mid I \;\sim\; \mathrm{Gamma}(a_t, b_t). \tag{3}$$

We use a sliding window of $\tau = 7$ days (each estimate is reported at the window's *end* date, the "instantaneous" convention) and a weakly informative Gamma prior with mean 5 and standard deviation 5 ($a_0 = 1$, $b_0 = 5$). The posterior mean is $\mathbb{E}[R_t] = a_t b_t$ and the 95% credible interval is the 2.5th/97.5th Gamma percentiles. The first six days lack a complete 7-day window and are not estimable, so 92 daily estimates are produced (1 March–31 May 2020). The renewal formulation makes the key robustness property transparent: a *constant* case-reporting fraction ρ multiplies both I_t and Λ_t and therefore cancels in the ratio I_t/Λ_t, leaving the R_t estimate unchanged—a point we return to in the Discussion [7].

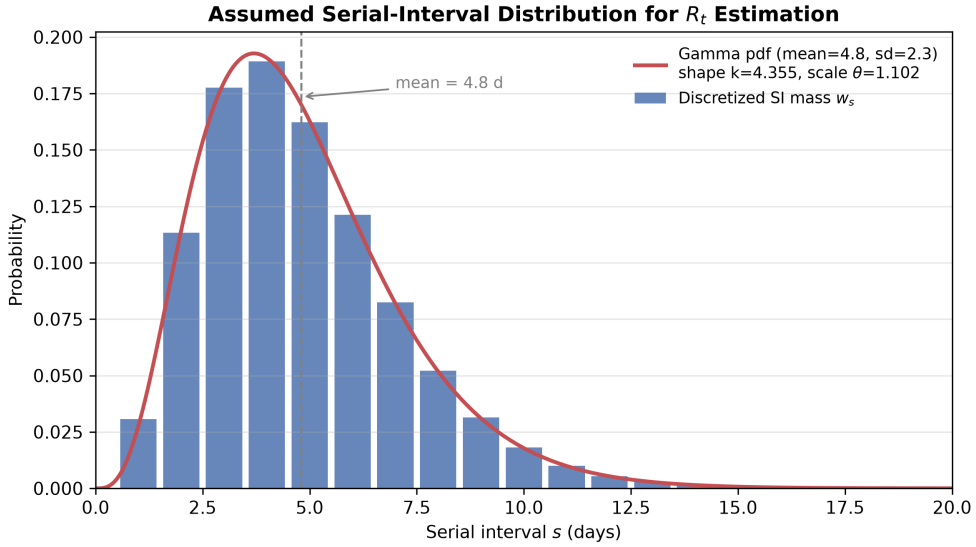

Figure 4: Assumed serial-interval distribution. Gamma distribution with mean 4.8 days and standard deviation 2.3 days (shape $k = 4.355$, scale $\theta = 1.102$), shown as the continuous density (red) and its normalized daily discretization w_s (bars). The discrete mode is at 4 days.

3.2 Compartmental model: deterministic SIR fitting

Independently of the renewal-equation analysis, we fit the classical deterministic SIR model [8, 11] on a closed population of size $N = 60{,}300{,}000$ (Italy, 2020):

$$\frac{dS}{dt} = -\beta \frac{SI}{N}, \qquad \frac{dI}{dt} = \beta \frac{SI}{N} - \gamma I, \qquad \frac{dR}{dt} = \gamma I, \tag{4}$$

where β is the transmission rate, γ the recovery rate, and the basic reproduction number is $R_0 = \beta/\gamma$. The system is integrated with an adaptive Runge–Kutta solver (`scipy.integrate.solve_ivp`, RK45, `max_step`=1, `rtol`=10^{-7}).

Incidence as a flow, not a prevalence. A common and consequential error in compartmental fitting is to compare the active-infectious *prevalence* $I(t)$ against daily *case counts*, which are an *incidence* (a flow). Instead we define predicted daily new cases as the daily decrement of the susceptible compartment,

$$C(t) \;=\; S(t-1) - S(t) \;=\; -\,\Delta S, \tag{5}$$

and fit this susceptible-outflow series to the observed daily incidence. Free parameters are β, γ, and the initial infectious seed I_0 (with $S_0 = N - I_0$, $R(0) = 0$). The objective is the sum of squared errors (SSE) between predicted and observed daily incidence, minimized by a global search (`differential_evolution`, seed 42) followed by a bounded local polish (`L-BFGS-B`). Physically motivated bounds were imposed: $\beta \in [0.02, 1.5]$, $\gamma \in [1/21, 1/3]$ (mean infectious period 3–21 days), and $I_0 \in [1, 5 \times 10^4]$. Both the reconstructed raw daily series and the 7-day smoothed series were fitted independently; the smoothed fit is reported as primary and the raw fit as a consistency check. The peak-prediction error is the signed difference (in days) between the model's predicted daily-incidence peak and the observed peak.

Software and reproducibility. All analyses use Python with NumPy [21], SciPy [22], pandas [23], and Matplotlib [24]. Random seeds are fixed (42) and every stage writes machine-readable JSON summaries and provenance files. The complete scripts are reproduced in Appendices A–C.

4 Results

4.1 Instantaneous reproduction number and the below-1 crossing

The estimated R_t trajectory (Figure 5) traces the expected arc of an intervention-controlled first wave. Early, uncontrolled transmission is reflected in a posterior mean R_t that peaks at **3.23** on **3 March 2020** (95% CrI [3.09, 3.37]), consistent with published Italian first-wave estimates in the 2.5–3.5 range [2, 12, 13]. From early March onward R_t declines steadily as the cascade of NPIs takes effect. The posterior mean first falls below the epidemic threshold of 1 on **1 April 2020** (mean 0.978, 95% CrI [0.968, 0.988]); on the same date the *upper* 95% credible bound also drops below 1, so the sub-threshold transition is statistically decisive rather than marginal.

This crossing occurs roughly three weeks after the 9 March national lockdown. That lag is epidemiologically coherent: it reflects the combined effect of the serial interval, the incubation-plus-reporting delay between infection and a confirmed case, and the 7-day estimation window, all of which separate a change in transmission from its appearance in the confirmed-case signal [4, 7]. After the crossing, R_t stabilizes around 0.79 through late May (final estimate 0.793 on 31 May), indicating sustained suppression rather than a rebound. Table 1 summarizes the key quantities; the full 92-point daily series with credible intervals is released as `italy_rt_series.csv`.

Table 1: Key results of the renewal-equation (R_t) analysis.

Quantity	**Value**
Method	Cori et al. (2013) renewal equation, Bayesian sliding window
Serial interval	Gamma, mean 4.8 d, SD 2.3 d ($k = 4.355$, $\theta = 1.102$)
Estimation window τ	7 days (estimate reported at window end)
Prior on R_t	Gamma, mean 5, SD 5 ($a_0 = 1$, $b_0 = 5$)
Number of daily estimates	92 (1 Mar–31 May 2020)
Peak mean R_t	**3.23** on 3 March 2020 (95% CrI [3.09, 3.37])
Date mean R_t first < 1	**1 April 2020** (mean 0.978, 95% CrI [0.968, 0.988])
Date upper 95% bound first < 1	1 April 2020
Final R_t (31 May 2020)	0.793 (95% CrI [0.766, 0.820])

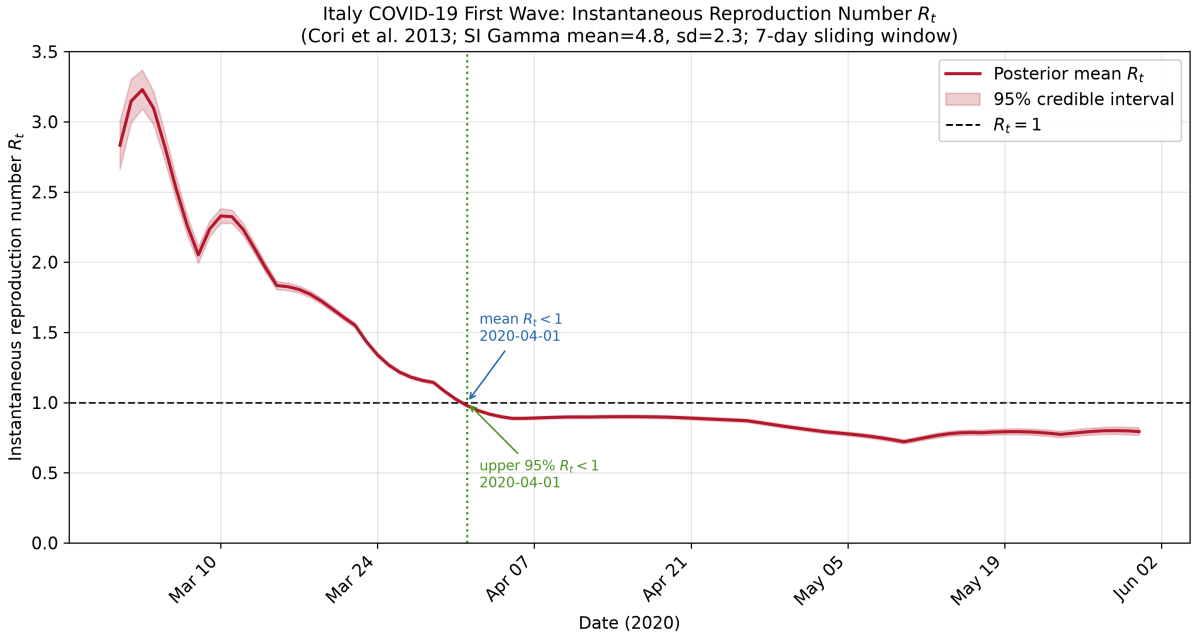

Figure 5: Estimated instantaneous reproduction number R_t for Italy's first wave. Posterior mean (red) with 95% credible interval (shaded), and the $R_t = 1$ threshold (dashed). R_t peaks near 3.2 in early March and falls below 1 on 1 April 2020; both the posterior mean and the upper 95% credible bound cross the threshold on that date (annotated).

4.2 SIR fit, R_0, and peak-prediction error

The best-fit constant-β SIR parameters and the peak-prediction evaluation are reported in Table 2. The SSE-optimal fit to the smoothed series returns $\beta = 0.329\,\mathrm{d}^{-1}$ and $\gamma = 0.333\,\mathrm{d}^{-1}$, hence a basic reproduction number of

$$R_0 = \frac{\beta}{\gamma} = 0.99. \tag{6}$$

The model predicts the daily-incidence peak on 24 February (the first day of the window), whereas the observed smoothed peak is on 26 March (raw-series argmax 28 March), giving a **peak-prediction error of −31 days**. The fit to the raw series is essentially identical ($R_0 = 0.99$, same −31-day error, RMSE 1,670 vs. 1,625 cases/day for the smoothed fit), confirming the result is not an artifact of smoothing. As Figure 6 makes visually plain, the fitted SIR incidence is a nearly flat, slowly declining line lying across the pronounced observed bell shape—a poor fit that the optimizer nonetheless correctly identifies as SSE-optimal within the model class.

Table 2: Fitted deterministic SIR parameters and peak-prediction evaluation (primary = 7-day smoothed-series fit; raw-series fit shown for comparison).

Quantity	**Smoothed fit (primary)**	**Raw fit**
Population N	60,300,000	
β (per day)	0.3294	0.3294
γ (per day)	0.3333†	0.3333†
Mean infectious period $1/\gamma$	3.0 d (at bound)	3.0 d (at bound)
Initial seed I_0	9,011	9,004
$R_0 = \beta/\gamma$	**0.99**	**0.99**
RMSE (cases/day)	1,625	1,670
Observed peak date	26 March 2020 (smoothed); 28 March (raw)	
Predicted peak date	24 February 2020	24 February 2020
Peak-prediction error	**−31 days**	**−31 days**

† γ is pinned at its upper (fast-recovery) bound; this is a symptom of the structural degeneracy analyzed in Section 4.3, not a credible infectious-period estimate.

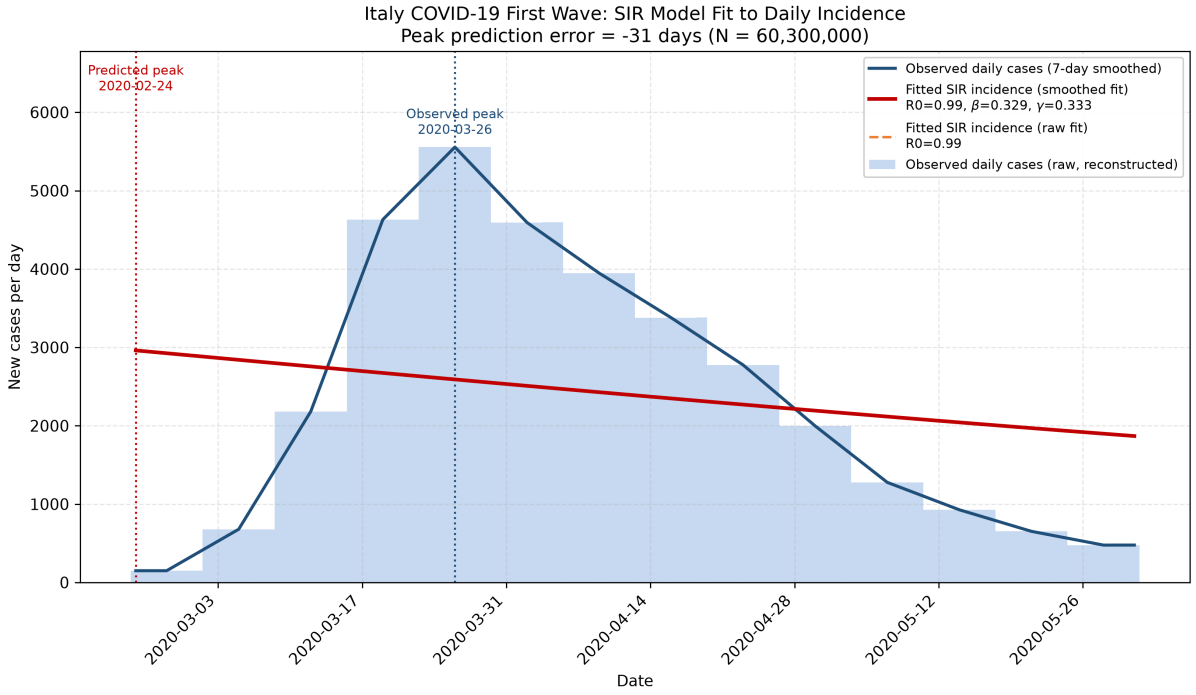

Figure 6: Deterministic SIR fit to Italy's first-wave daily incidence. Observed daily cases (light bars = reconstructed raw; dark line = 7-day smoothed) versus the fitted SIR daily incidence for the smoothed (solid red) and raw (dashed orange) fits. The near-flat fitted curve cannot reproduce the observed interior peak; the predicted peak is pinned to the series start, giving a −31-day error.

4.3 Why the constant-β SIR fails: a structural degeneracy

The $R_0 \approx 0.99$ result is not a numerical accident but a direct consequence of the model's structure meeting the data's scale. The cumulative confirmed-case total over the window (232,585) is only $\sim$**0.39%** of $N = 60.3$ million, so the susceptible fraction S/N remains essentially equal to 1 throughout. In this regime the infectious equation in (4) linearizes,

$$\frac{dI}{dt} = \left(\beta \frac{S}{N} - \gamma\right) I \approx (\beta - \gamma)\, I, \tag{7}$$

so incidence is (near) purely exponential and *monotonic* over the 98-day window. A closed SIR produces an interior incidence peak only once susceptible depletion becomes appreciable—formally when S/N falls to $1/R_0$, which requires a large fraction of the population to have been infected. With $S/N \approx 1$ that mechanism is unavailable, and the SSE-optimal solution within the constant-β class is a nearly flat line with $\beta \approx \gamma$ (hence $R_0 \approx 1$) that pins the predicted peak to the first day. The γ parameter sitting at its upper bound is a diagnostic fingerprint of this degeneracy.

The correct interpretation is therefore an informative *negative* result: Italy's first-wave turnover was produced by non-pharmaceutical interventions that reduced transmission over time—not by herd-immunity susceptible depletion—and a constant-parameter, fixed-N SIR is structurally incapable of representing that mechanism [3, 4]. The fitted $R_0 \approx 0.99$ should be read as evidence of this inadequacy, and is manifestly inconsistent both with our own renewal-equation peak $R_t \approx 3.23$ and with published Italian first-wave R_0 estimates of 2.5–3.5 [2, 12, 13]. Faithful compartmental calibration of this wave would require one of: a time-varying transmission rate $\beta(t)$ (or a change-point at the 9 March lockdown); an effective susceptible population $N_{\text{eff}} \ll N$; or fitting to ascertainment-corrected infections rather than confirmed cases. Each of these lies outside the constant-parameter, fixed-N specification adopted here but represents the natural next step.

5 Discussion: Ascertainment, Testing, and Their Effects on the Estimates

Both reproduction-number estimates are computed from *confirmed* cases, which during the first wave captured only a small and time-varying fraction of true infections. Understanding how this under-ascertainment propagates into R_t and R_0 is essential to interpreting the numbers above, and the two estimators are affected in strikingly different ways.

5.1 The magnitude of first-wave under-ascertainment in Italy

Multiple lines of evidence indicate severe under-detection in early 2020. Italy's national SARS-CoV-2 seroprevalence survey (ISTAT–Ministry of Health, May–July 2020) estimated a national IgG seroprevalence of about 2.5% (roughly 1.48 million past infections) with strong regional heterogeneity (Lombardy ∼7.5%). Set against ∼0.24 million confirmed cases nationally by end-May, this implies that *true infections exceeded confirmed cases by roughly an order of magnitude, and considerably more in the hardest-hit regions.* The population-wide testing of Vo' (Veneto) directly demonstrated that a large share of infections were asymptomatic at detection and would have been missed by symptom-triggered testing [25], and modeling of early transmission in China concluded that the substantial majority of infections in the uncontrolled phase were undocumented [26]. Analyses in comparable European settings similarly inferred that only a small percentage of infections were being ascertained during the first wave [27]. Meta-analytic estimates place the genuinely asymptomatic fraction of infections at roughly 20–40% [28], a pool structurally invisible to the symptom-based testing that dominated February–March 2020.

5.2 Effect on R_t: robust to constant under-reporting, sensitive to changing testing

The renewal-equation estimator is comparatively resilient to under-ascertainment precisely because it depends on the *ratio* of current incidence to past infectiousness. If a constant fraction ρ of infections is reported, both I_t and $\Lambda_t = \sum_s I_{t-s} w_s$ are scaled by ρ, and ρ cancels in $R_t = I_t/\Lambda_t$

(Eq. 1); the estimated *level* of R_t and, in particular, the timing of the below-1 crossing are therefore largely preserved even when the absolute number of infections is grossly undercounted [7]. What R_t is *not* robust to is a *change* in ascertainment. During February–March 2020 Italian testing expanded rapidly and testing criteria shifted; an increasing reporting fraction inflates measured incidence growth and can transiently bias R_t *upward* (and a tightening of criteria biases it downward), so the early, steeply rising portion of the trajectory should be read with more caution than the post-peak decline [2, 7]. Reporting delays and the upstream weekly aggregation further distort fine timing; we mitigate these with 7-day smoothing, the centered rolling average, and the weekly-to-daily reconstruction described in the Data section, but they cannot be removed entirely. The net assessment is that the *qualitative* decline of R_t through 1 on 1 April is a robust feature, while the precise early peak value (3.23) carries additional, largely upward, uncertainty from evolving test coverage.

5.3 Effect on R_0: under-ascertainment deepens the SIR degeneracy

Under-ascertainment acts on the compartmental R_0 through a different and more damaging channel. Fitting a closed SIR to confirmed cases at the full population N makes the apparent attack rate—and hence the apparent susceptible depletion—even smaller than the true value, reinforcing the pure-exponential degeneracy of Section 4.3 and driving the SSE-optimal R_0 toward 1. Even the seroprevalence-corrected attack rate (a few percent) is far too small to bend a closed SIR into an interior peak within a single wave, so correcting for under-ascertainment alone would not rescue the constant-β model; it simply confirms that the observed turnover was intervention-driven. In other words, the -31-day peak error and $R_0 \approx 0.99$ are joint artifacts of (i) applying a constant-parameter, fixed-N SIR to an NPI-driven wave and (ii) calibrating it on under-ascertained confirmed cases. A credible compartmental R_0 for this wave must come from a model that encodes the intervention mechanism (time-varying β or an effective susceptible pool) and, ideally, is fitted to ascertainment-corrected infections or to a less biased signal such as deaths with an infection-to-death delay [3, 27].

5.4 Reconciling the two estimates

The apparent tension between a peak R_t of 3.2 and an SIR R_0 of 1.0 is fully resolved by the analysis above: the renewal-equation R_t is a largely model-free, ratio-based estimator that reads transmissibility directly off the incidence curve and is robust to constant under-reporting, whereas the constant-β SIR R_0 is a model-bound quantity whose identification relies on a susceptible-depletion signal that is absent at this data scale. The R_t estimate—peaking near 3 and crossing 1 in early April—is the epidemiologically credible summary of first-wave transmission and aligns with the independent Italian literature [2, 4, 12, 13]. The SIR exercise is retained not because its R_0 is believable, but because its failure is itself an instructive, quantitative demonstration of when and why simple compartmental models must not be taken at face value.

5.5 Limitations

Several limitations qualify these results. (1) The daily incidence is reconstructed from weekly OWID aggregates by uniform redistribution; while weekly and cumulative totals are preserved exactly, the within-week daily shape is imposed, and a monotone-spline reconstruction would be a worthwhile sensitivity check. (2) R_t is reported at the window-end date (the instantaneous convention) and is not additionally shifted to an infection date; comparisons with infection-dated estimates should account for this. (3) The serial interval is fixed at mean 4.8/SD 2.3 days; alternative early estimates range from $\sim$4 to $\sim$6.6 days [13, 19, 20], and a shorter (longer)

assumed interval would compress (stretch) the inferred R_t toward (away from) 1. (4) The compartmental analysis is deliberately restricted to a constant-parameter, fixed-N SIR per the task specification; its R_0 is therefore diagnostic rather than authoritative. (5) All estimates inherit the ascertainment biases discussed above, which are only partially mitigated.

6 Conclusion

Using a single publicly archived case series for Italy's first COVID-19 wave, we estimated the instantaneous reproduction number with the Cori renewal-equation method and, separately, fitted a deterministic SIR compartmental model. The R_t analysis provides the credible epidemiological picture: transmissibility peaked at a posterior mean of 3.23 in early March and fell decisively below the epidemic threshold on 1 April 2020 (95% CrI [0.968, 0.988] on that date), about three weeks after the national lockdown. The constant-β SIR, by contrast, returned $R_0 \approx 0.99$ and mispredicted the observed peak by -31 days—a result we show to be a structural degeneracy arising because confirmed cases represent only $\sim$0.39% of the population, so susceptible depletion is negligible and the closed model cannot generate the intervention-driven turnover it is being asked to fit. Under-ascertainment sharpens this contrast: the renewal-equation R_t is robust to *constant* under-reporting but sensitive to *changes* in testing, whereas the SIR R_0 is driven toward 1 by the very under-ascertainment that shrinks the apparent attack rate. The practical lesson is that real-time, ratio-based R_t estimation is far better suited to characterizing an NPI-controlled wave from confirmed cases than a naive constant-parameter compartmental fit, and that any compartmental R_0 for such a wave must encode the intervention mechanism and correct for ascertainment before it can be believed.

Data and Code Availability

Data source. Our World in Data (OWID) COVID-19 dataset [14, 15], file `owid-covid-data.csv`, retrieved from `https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv`; **snapshot/access timestamp 2026-07-10 22:10:18 UTC**. Italy (`ITA`), 24 February–31 May 2020, 98 days, 232,585 cumulative confirmed cases.

Released artifacts.

- `italy_covid_cleaned.csv` — curated daily series (date, new_cases, new_cases_smoothed).
- `italy_rt_series.csv` — 92 daily R_t estimates (date, mean_rt, low_95, high_95).
- `step2_rt_summary.json`, `step3_sir_summary.json` — machine-readable summaries.
- `01_data_acquisition.py`, `02_rt_estimation.py`, `03_sir_fitting.py` — complete runnable scripts (Appendices A–C).

Reproduce. With Python 3 and NumPy/SciPy/pandas/Matplotlib/requests installed:

```
python 01_data_acquisition.py # download + curate (writes italy_covid_cleaned.csv)
python 02_rt_estimation.py # Cori Rt (writes italy_rt_series.csv, figure)
python 03_sir_fitting.py # SIR fit (writes step3_sir_summary.json, figure)
```

References

[1] Andrea Remuzzi and Giuseppe Remuzzi. COVID-19 and italy: What next? *The Lancet*, 395(10231):1225–1228, 2020. doi: 10.1016/S0140-6736(20)30627-9.

[2] Flavia Riccardo, Marco Ajelli, Xanthi D. Andrianou, Antonino Bella, Martina Del Manso, Massimo Fabiani, Stefania Bellino, Stefano Boros, Alberto Mateo Urdiales, Valentina Marziano, Maria Cristina Rota, Antonietta Filia, Fortunato D'Ancona, Andrea Siddu, Ornella Punzo, Filippo Trentini, Giorgio Guzzetta, Piero Poletti, Paola Stefanelli, Maria Rita Castrucci, Alessandra Ciervo, Corrado Di Benedetto, Marco Tallon, Andrea Piccioli, Silvio Brusaferro, Giovanni Rezza, Stefano Merler, and Patrizio Pezzotti. Epidemiological characteristics of COVID-19 cases and estimates of the reproductive numbers 1 month into the epidemic, italy, 28 january to 31 march 2020. *Eurosurveillance*, 25(49):2000790, 2020. doi: 10.2807/1560-7917.ES.2020.25.49.2000790.

[3] Seth Flaxman, Swapnil Mishra, Axel Gandy, H. Juliette T. Unwin, Thomas A. Mellan, Helen Coupland, Charles Whittaker, Harrison Zhu, Tresnia Berah, Jeffrey W. Eaton, Mélodie Monod, Azra C. Ghani, Christl A. Donnelly, Steven Riley, Michaela A. C. Vollmer, Neil M. Ferguson, Lucy C. Okell, and Samir Bhatt. Estimating the effects of non-pharmaceutical interventions on COVID-19 in europe. *Nature*, 584(7820):257–261, 2020. doi: 10.1038/s41586-020-2405-7.

[4] Giorgio Guzzetta, Flavia Riccardo, Valentina Marziano, Piero Poletti, Filippo Trentini, Antonino Bella, Xanthi Andrianou, Martina Del Manso, Massimo Fabiani, Stefania Bellino, Stefano Boros, Alberto Mateo Urdiales, Maria Fenicia Vescio, Andrea Piccioli, Silvio Brusaferro, Giovanni Rezza, Patrizio Pezzotti, Marco Ajelli, and Stefano Merler. Impact of a nationwide lockdown on SARS-CoV-2 transmissibility, italy. *Emerging Infectious Diseases*, 27(1):267–270, 2021. doi: 10.3201/eid2701.202114.

[5] Anne Cori, Neil M. Ferguson, Christophe Fraser, and Simon Cauchemez. A new framework and software to estimate time-varying reproduction numbers during epidemics. *American Journal of Epidemiology*, 178(9):1505–1512, 2013. doi: 10.1093/aje/kwt133.

[6] Christophe Fraser. Estimating individual and household reproduction numbers in an emerging epidemic. *PLoS ONE*, 2(8):e758, 2007. doi: 10.1371/journal.pone.0000758.

[7] Katelyn M. Gostic, Lauren McGough, Edward B. Baskerville, Sam Abbott, Keya Joshi, Christine Tedijanto, Rebecca Kahn, Rene Niehus, James A. Hay, Pablo M. De Salazar, Joel Hellewell, Sophie Meakin, James D. Munday, Nikos I. Bosse, Katharine Sherratt, Robin M. Thompson, Laura F. White, Jana S. Huisman, Jeremie Scire, Sebastian Bonhoeffer, Tanja Stadler, Jacco Wallinga, Sebastian Funk, Marc Lipsitch, and Sarah Cobey. Practical considerations for measuring the effective reproductive number, R_t. *PLoS Computational Biology*, 16(12):e1008409, 2020. doi: 10.1371/journal.pcbi.1008409.

[8] William Ogilvy Kermack and Anderson Gray McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A*, 115 (772):700–721, 1927. doi: 10.1098/rspa.1927.0118.

[9] Odo Diekmann, J. A. P. Heesterbeek, and J. A. J. Metz. On the definition and the computation of the basic reproduction ratio R_0 in models for infectious diseases in heterogeneous populations. *Journal of Mathematical Biology*, 28(4):365–382, 1990. doi: 10.1007/BF00178324.

[10] Pauline van den Driessche and James Watmough. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission. *Mathematical Biosciences*, 180(1-2):29–48, 2002. doi: 10.1016/S0025-5564(02)00108-6.

[11] Roy M. Anderson and Robert M. May. *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press, Oxford, UK, 1991.

[12] Marco D'Arienzo and Angela Coniglio. Assessment of the SARS-CoV-2 basic reproduction number, R_0, based on the early phase of COVID-19 outbreak in italy. *Biosafety and Health*, 2(2):57–59, 2020. doi: 10.1016/j.bsheal.2020.03.004.

[13] Danilo Cereda, Mattia Manica, Marcello Tirani, Francesca Rovida, Vittorio Demicheli, Marco Ajelli, Piero Poletti, Filippo Trentini, Giorgio Guzzetta, Valentina Marziano, Ambra Barone, Michele Magoni, Silvia Deandrea, Giulio Diurno, Massimo Lombardo, Marino Faccini, Angelo Pan, Raffaele Bruno, Elena Pariani, Giacomo Grasselli, Alessandra Piatti, Maria Gramegna, Fausto Baldanti, Alessia Melegaro, and Stefano Merler. The early phase of the COVID-19 epidemic in lombardy, italy. *Epidemics*, 37:100528, 2021. doi: 10.1016/j.epidem.2021.100528.

[14] Hannah Ritchie, Edouard Mathieu, Lucas Rodés-Guirao, Cameron Appel, Charlie Giattino, Esteban Ortiz-Ospina, Joe Hasell, Bobbie Macdonald, Diana Beltekian, and Max Roser. Coronavirus pandemic (COVID-19). `https://ourworldindata.org/coronavirus`, 2020. Our World in Data. Dataset: `https://github.com/owid/covid-19-data`. Accessed 2026-07-10.

[15] Edouard Mathieu, Hannah Ritchie, Esteban Ortiz-Ospina, Max Roser, Joe Hasell, Cameron Appel, Charlie Giattino, and Lucas Rodés-Guirao. A global database of COVID-19 vaccinations. *Nature Human Behaviour*, 5(7):947–953, 2021. doi: 10.1038/s41562-021-01122-8.

[16] Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, 20(5):533–534, 2020. doi: 10.1016/S1473-3099(20)30120-1.

[17] Robin N. Thompson, J. E. Stockwin, R. D. van Gaalen, J. A. Polonsky, Z. N. Kamvar, P. A. Demarsh, E. Dahlqwist, S. Li, E. Miguel, T. Jombart, J. Lessler, S. Cauchemez, and A. Cori. Improved inference of time-varying reproduction numbers during infectious disease outbreaks. *Epidemics*, 29:100356, 2019. doi: 10.1016/j.epidem.2019.100356.

[18] Jacco Wallinga and Peter Teunis. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American Journal of Epidemiology*, 160(6):509–516, 2004. doi: 10.1093/aje/kwh255.

[19] Hiroshi Nishiura, Natalie M. Linton, and Andrei R. Akhmetzhanov. Serial interval of novel coronavirus (COVID-19) infections. *International Journal of Infectious Diseases*, 93:284–286, 2020. doi: 10.1016/j.ijid.2020.02.060.

[20] Tapiwa Ganyani, Cecile Kremer, Dongxuan Chen, Andrea Torneri, Christel Faes, Jacco Wallinga, and Niel Hens. Estimating the generation interval for coronavirus disease (COVID-19) based on symptom onset data, march 2020. *Eurosurveillance*, 25(17):2000257, 2020. doi: 10.2807/1560-7917.ES.2020.25.17.2000257.

[21] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2.

[22] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan

Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[23] Wes McKinney. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a.

[24] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[25] Enrico Lavezzo, Elisa Franchin, Constanze Ciavarella, Gina Cuomo-Dannenburg, Luisa Barzon, Claudia Del Vecchio, Lucia Rossi, Riccardo Manganelli, Arianna Loregian, Nicolò Navarin, Davide Abate, Manuela Sciro, Stefano Merigliano, Ettore De Canale, Maria Cristina Vanuzzo, Valeria Besutti, Francesca Saluzzo, Francesco Onelia, Monia Pacenti, Saverio G. Parisi, Giovanni Carretta, Daniele Donato, Luciano Flor, Silvia Cocchio, Giulia Masi, Alessandro Sperduti, Lorenzo Cattarino, Renato Salvador, Michele Nicoletti, Federico Caldart, Gioele Castelli, Eleonora Nieddu, Beatrice Labella, Ludovico Fava, Matteo Drigo, Katy A. M. Gaythorpe, Neil M. Ferguson, Thomas Churcher, Anna Maria Cattelan, Rosanna Cristofori, and Andrea Crisanti. Suppression of a SARS-CoV-2 outbreak in the italian municipality of vo'. *Nature*, 584(7821):425–429, 2020. doi: 10.1038/s41586-020-2488-1.

[26] Ruiyun Li, Sen Pei, Bin Chen, Yimeng Song, Tao Zhang, Wan Yang, and Jeffrey Shaman. Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV-2). *Science*, 368(6490):489–493, 2020. doi: 10.1126/science.abb3221.

[27] Henrik Salje, Cécile Tran Kiem, Noémie Lefrancq, Noémie Courtejoie, Paolo Bosetti, Juliette Paireau, Alessio Andronico, Nathanaël Hozé, Jehanne Richet, Claire-Lise Dubost, Yann Le Strat, Justin Lessler, Daniel Levy-Bruhl, Arnaud Fontanet, Lulla Opatowski, Pierre-Yves Boelle, and Simon Cauchemez. Estimating the burden of SARS-CoV-2 in france. *Science*, 369(6500):208–211, 2020. doi: 10.1126/science.abc3517.

[28] Diana Buitrago-Garcia, Dianne Egli-Gany, Michel J. Counotte, Stefanie Hossmann, Hira Imeri, Aziz Mert Ipekci, Georgia Salanti, and Nicola Low. Occurrence and transmission potential of asymptomatic and presymptomatic SARS-CoV-2 infections: A living systematic review and meta-analysis. *PLoS Medicine*, 17(9):e1003346, 2020. doi: 10.1371/journal.pmed.1003346.

A Data Acquisition and Curation (`01_data_acquisition.py`)

```python
"""Step 1: Data Acquisition & Curation of Italy COVID-19 first-wave case data.

Downloads the Our World in Data (OWID) COVID-19 dataset, filters for Italy over
the first-wave window (2020-02-24 through 2020-05-31 inclusive), cleans the daily
new-cases series, applies a 7-day rolling average, and saves a curated CSV plus a
visualization.

Non-interactive; matplotlib uses the Agg backend. Random seed set for determinism
(not strictly needed here, kept for reproducibility conventions).
"""

from __future__ import annotations

import io
```

```python
import sys
import time
from datetime import datetime, timezone
from pathlib import Path

import matplotlib

matplotlib.use("Agg")
import matplotlib.dates as mdates
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import requests

np.random.seed(42)

SESSION = Path("/app/sandbox/session_20260710_145447_12a9d39594f7")
RESULTS = SESSION / "results"
FIGURES = SESSION / "figures"
DATA = SESSION / "data"
LOGS = SESSION / "logs"

# Candidate URLs (OWID reorganized their repo over time; try in order).
CANDIDATE_URLS = [
    "https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv",
    "https://covid.ourworldindata.org/data/owid-covid-data.csv",
    "https://catalog.ourworldindata.org/garden/covid/latest/compact/compact.csv",
]

START_DATE = pd.Timestamp("2020-02-24")
END_DATE = pd.Timestamp("2020-05-31")


def log(msg: str) -> None:
    ts = datetime.now(timezone.utc).strftime("%Y-%m-%d %H:%M:%S UTC")
    print(f"[{ts}] {msg}", flush=True)


def download_and_filter_italy(urls: list[str]) -> tuple[pd.DataFrame, str, str]:
    """Stream the OWID CSV in chunks, keeping only Italy rows.

    Returns the filtered Italy dataframe, the source URL that worked, and the
    access timestamp.
    """
    last_err: Exception | None = None
    for url in urls:
        log(f"Attempting download from: {url}")
        access_ts = datetime.now(timezone.utc).strftime("%Y-%m-%d %H:%M:%S UTC")
        try:
            resp = requests.get(url, stream=True, timeout=(30, 300))
            resp.raise_for_status()
            resp.encoding = "utf-8"

            # Stream text and parse in chunks to avoid loading the full (~100MB+) file.
            text_stream = io.StringIO(resp.text)
            log(f"Downloaded {len(resp.text) / 1e6:.1f} MB of text; parsing in chunks...")

            italy_chunks = []
            total_rows = 0
            reader = pd.read_csv(text_stream, chunksize=100_000, low_memory=False)
            for i, chunk in enumerate(reader):
                total_rows += len(chunk)
                # Identify Italy rows via iso_code or location/country column.
                mask = pd.Series(False, index=chunk.index)
                if "iso_code" in chunk.columns:
                    mask |= chunk["iso_code"].astype(str).str.upper() == "ITA"
                loc_col = "location" if "location" in chunk.columns else (
                    "country" if "country" in chunk.columns else None
                )
                if loc_col is not None:
                    mask |= chunk[loc_col].astype(str).str.strip() == "Italy"
                if mask.any():
                    italy_chunks.append(chunk.loc[mask].copy())
```

```
                if i % 5 == 0:
                    log(f" parsed {total_rows:,} rows so far; italy matches={sum(len(c) for c in
                        italy_chunks):,}")

            if not italy_chunks:
                raise ValueError("No Italy rows found in dataset from this URL.")

            italy = pd.concat(italy_chunks, ignore_index=True)
            log(f"Total rows scanned: {total_rows:,}; Italy rows: {len(italy):,}")
            log(f"Detected columns (sample): {list(italy.columns)[:15]}")
            return italy, url, access_ts
        except Exception as exc: # noqa: BLE001 -robust multi-URL fallback
            last_err = exc
            log(f" FAILED for {url}: {type(exc).__name__}: {exc}")
            time.sleep(2)
    raise RuntimeError(f"All candidate URLs failed. Last error: {last_err}")


def main() -> int:
    for d in (RESULTS, FIGURES, DATA, LOGS):
        d.mkdir(parents=True, exist_ok=True)
    log(f"Ensured directories: {RESULTS}, {FIGURES}, {DATA}, {LOGS}")

    italy, source_url, access_ts = download_and_filter_italy(CANDIDATE_URLS)

    # Resolve date column.
    date_col = "date" if "date" in italy.columns else None
    if date_col is None:
        raise KeyError(f"No 'date' column found. Columns: {list(italy.columns)}")

    italy[date_col] = pd.to_datetime(italy[date_col], errors="coerce")
    italy = italy.dropna(subset=[date_col])

    # Resolve new-cases column.
    cases_col = None
    for cand in ("new_cases", "new_cases_smoothed", "new_confirmed"):
        if cand in italy.columns:
            cases_col = cand
            break
    if cases_col is None:
        raise KeyError(f"No new-cases column found. Columns: {list(italy.columns)}")
    log(f"Using date column='{date_col}', cases column='{cases_col}'")

    # Restrict to first-wave window (inclusive).
    window = italy[(italy[date_col] >= START_DATE) & (italy[date_col] <= END_DATE)].copy()
    window = window.sort_values(date_col).reset_index(drop=True)
    log(f"Rows in window {START_DATE.date()}..{END_DATE.date()}: {len(window)}")

    # Reindex onto a complete daily calendar so no dates are silently missing.
    full_index = pd.date_range(START_DATE, END_DATE, freq="D")
    expected_days = len(full_index)
    log(f"Expected number of daily records: {expected_days}")

    series = window.set_index(date_col)[cases_col].reindex(full_index)

    # ---Data quality inspection ---
    n_missing = int(series.isna().sum())
    n_negative = int((series < 0).sum())
    n_zero = int((series == 0).sum())
    n_nonzero = int((series > 0).sum())
    log(f"Data quality: missing={n_missing}, negative={n_negative}, "
        f"zeros={n_zero}, nonzero={n_nonzero}")
    log(f"Raw {cases_col} stats: min={series.min()}, max={series.max()}, "
        f"mean={series.mean():.1f}")

    adjustments = []

    # Negative values (reporting corrections) -> set to 0.
    if n_negative > 0:
        neg_dates = series.index[series < 0].strftime("%Y-%m-%d").tolist()
        adjustments.append(f"Set {n_negative} negative value(s) to 0 on dates: {neg_dates}")
        series[series < 0] = 0
        n_zero = int((series == 0).sum())
```

```
        n_nonzero = int((series > 0).sum())

    # ---Detect OWID weekly-reporting format ---
    # As of the current OWID release, 2020 data is served as WEEKLY aggregates:
    # a 7-day total is placed on one day (Sunday) with 0 on the other 6 days.
    # Detect this (>=50% zeros AND nonzero values ~7 days apart) and reconstruct a
    # true daily incidence series by distributing each weekly total across the 7
    # days it covers. This preserves weekly and cumulative totals exactly.
    nonzero_dates = series.index[series.fillna(0) > 0]
    weekly_format = False
    if len(nonzero_dates) >= 2:
        gaps = np.diff(nonzero_dates.values).astype("timedelta64[D]").astype(int)
        frac_zero = n_zero / expected_days
        if frac_zero >= 0.5 and np.median(gaps) == 7:
            weekly_format = True

    if weekly_format:
        log("DETECTED: OWID weekly-aggregated reporting format (7-day totals on a "
            "single day, zeros elsewhere). Reconstructing daily incidence by "
            "uniform redistribution of each weekly total across its 7-day window.")
        filled = series.fillna(0).astype(float)
        daily = pd.Series(0, index=full_index, dtype=int)
        total_before = int(round(filled.sum()))
        for report_date, weekly_total in filled[filled > 0].items():
            wt = int(round(weekly_total))
            win = pd.date_range(end=report_date, periods=7, freq="D")
            win = win[win.isin(full_index)] # clip to available calendar
            k = len(win)
            if k == 0:
                continue
            base, rem = divmod(wt, k)
            alloc = np.full(k, base, dtype=int)
            alloc[k -rem:] += 1 # remainder on the days nearest the report date
            daily.loc[win] = daily.loc[win].values + alloc
        total_after = int(daily.sum())
        assert total_after == total_before, (
            f"Redistribution changed total: {total_before} -> {total_after}")
        adjustments.append(
            f"OWID served 2020 data as WEEKLY aggregates ({n_nonzero} weekly totals "
            f"on {expected_days} days). Reconstructed daily incidence by uniform "
            f"redistribution of each weekly total across the 7 days it covers "
            f"(cumulative total preserved exactly: {total_after:,} cases). "
            f"Note: because data is weekly-aggregated upstream, no sub-weekly "
            f"(weekend) reporting noise is present in the raw series."
        )
        series = daily
    else:
        # True daily data path: interpolate any interior missing values.
        if n_missing > 0:
            miss_dates = series.index[series.isna()].strftime("%Y-%m-%d").tolist()
            series = series.interpolate(method="time", limit_direction="both")
            residual = int(series.isna().sum())
            if residual > 0:
                series = series.fillna(0)
            adjustments.append(
                f"Interpolated {n_missing} missing value(s) (dates: {miss_dates}); "
                f"filled {residual} residual edge NaNs with 0"
            )
        series = series.round().astype(int) # cases are counts

    assert series.isna().sum() == 0, "Missing values remain after cleaning"
    assert len(series) == expected_days, "Record count mismatch after reindex"

    # ---7-day centered rolling average (min_periods=1 keeps edges defined) ---
    smoothed = series.rolling(window=7, min_periods=1, center=True).mean()

    curated = pd.DataFrame(
        {
            "date": series.index.strftime("%Y-%m-%d"),
            "new_cases": series.values,
            "new_cases_smoothed": smoothed.values.round(2),
        }
    )
```

```
233
234     out_csv = RESULTS / "italy_covid_cleaned.csv"
235     curated.to_csv(out_csv, index=False)
236     log(f"Saved curated dataset: {out_csv} ({len(curated)} rows)")
237
238     # ---Visualization ---
239     plt.rcParams.update({"font.family": "sans-serif", "font.size": 10, "axes.linewidth": 0.8})
240     fig, ax = plt.subplots(figsize=(11, 6))
241     dates = pd.to_datetime(curated["date"])
242     raw_label = ("Daily new cases (reconstructed from weekly totals)"
243                  if weekly_format else "Daily new cases (raw)")
244     ax.bar(dates, curated["new_cases"], width=0.9, color="#9ecae1",
245            label=raw_label, zorder=2)
246     ax.plot(dates, curated["new_cases_smoothed"], color="#08519c", linewidth=2.2,
247             label="7-day rolling average (centered)", zorder=3)
248     ax.set_title("Italy COVID-19 First Wave: Daily Confirmed Cases\n"
249                  "(2020-02-24 to 2020-05-31)", fontweight="bold")
250     ax.set_xlabel("Date")
251     ax.set_ylabel("New confirmed cases")
252     ax.xaxis.set_major_locator(mdates.WeekdayLocator(interval=2))
253     ax.xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m-%d"))
254     plt.setp(ax.get_xticklabels(), rotation=45, ha="right")
255     ax.legend(frameon=False, loc="upper right")
256     ax.grid(axis="y", alpha=0.3)
257     fig.tight_layout()
258     out_png = FIGURES / "italy_cases_observed.png"
259     fig.savefig(out_png, dpi=300, bbox_inches="tight")
260     plt.close(fig)
261     log(f"Saved figure: {out_png}")
262
263     # ---Provenance / summary ---
264     log("=" * 60)
265     log("PROVENANCE")
266     log(f" Source URL : {source_url}")
267     log(f" Access time : {access_ts}")
268     log(f" Date range : {START_DATE.date()} to {END_DATE.date()} (inclusive)")
269     log(f" Records : {len(curated)} (expected {expected_days})")
270     log(f" Peak raw cases : {int(curated['new_cases'].max())} on "
271         f"{curated.loc[curated['new_cases'].idxmax(), 'date']}")
272     log(f" Total cases : {int(curated['new_cases'].sum()):,}")
273     if adjustments:
274         log(" Adjustments :")
275         for a in adjustments:
276             log(f" -{a}")
277     else:
278         log(" Adjustments : none (no missing or negative values)")
279     log("=" * 60)
280
281     # Persist a small provenance file for downstream steps / README.
282     prov = RESULTS / "step1_provenance.txt"
283     prov.write_text(
284         f"source_url={source_url}\n"
285         f"access_time={access_ts}\n"
286         f"date_start={START_DATE.date()}\n"
287         f"date_end={END_DATE.date()}\n"
288         f"n_records={len(curated)}\n"
289         f"cases_column_used={cases_col}\n"
290         f"owid_weekly_format_detected={weekly_format}\n"
291         f"total_cases_window={int(curated['new_cases'].sum())}\n"
292         f"adjustments={'; '.join(adjustments) if adjustments else 'none'}\n"
293     )
294     log(f"Saved provenance: {prov}")
295     log("STEP 1 COMPLETE")
296     return 0
297
298
299 if __name__ == "__main__":
300     sys.exit(main())
```

B Instantaneous R_t Estimation (`02_rt_estimation.py`)

```python
"""
Step 2: Instantaneous Reproduction Number (Rt) Estimation for Italy's first
COVID-19 wave using the renewal-equation method of Cori et al. (2013).

Method summary
--------------
Serial interval (SI): Gamma(mean=4.8, sd=2.3), discretized on s = 1..30 via
    w_s = G(s + 0.5) -G(s -0.5), w_0 = 0, then normalized to sum to 1.
Total infectiousness at day t:
    Lambda_t = sum_{s=1}^{t-1} I_{t-s} * w_s
Sliding window tau = 7 days (window ending at day t spans [t-6, t], t >= 7,
using 0-based interior indexing consistent with the Cori formulation).
Gamma prior on Rt: mean 5, sd 5 -> shape a0 = 1, scale b0 = 5 (rate 1/b0 = 0.2).
Posterior (per window ending at t):
    a_t = a0 + sum_{k in window} I_k
    1/b_t = 1/b0 + sum_{k in window} Lambda_k
    b_t = 1 / (1/b_t)
    E[Rt] = a_t * b_t
    95% CI = Gamma.ppf([0.025, 0.975], a=a_t, scale=b_t)

Outputs
-------
results/italy_rt_series.csv (date, mean_rt, low_95, high_95)
figures/italy_rt_evolution.png
"""

import json
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib

matplotlib.use("Agg")
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

np.random.seed(42)

SESSION = "/app/sandbox/session_20260710_145447_12a9d39594f7"
INPUT_CSV = f"{SESSION}/results/italy_covid_cleaned.csv"
OUT_CSV = f"{SESSION}/results/italy_rt_series.csv"
OUT_FIG = f"{SESSION}/figures/italy_rt_evolution.png"

# ---SI / prior / window parameters ---
SI_MEAN = 4.8
SI_SD = 2.3
S_MAX = 30
TAU = 7 # sliding window length in days
PRIOR_SHAPE_A0 = 1.0
PRIOR_SCALE_B0 = 5.0 # rate = 1/b0 = 0.2

print("=" * 70)
print("Step 2: Rt estimation (Cori et al. 2013 renewal-equation method)")
print("=" * 70)

# ------------------------------------------------------------------------
# 1. Load curated daily incidence
# ------------------------------------------------------------------------
df = pd.read_csv(INPUT_CSV, parse_dates=["date"])
df = df.sort_values("date").reset_index(drop=True)
print(f"[load] input shape : {df.shape}")
print(f"[load] columns : {list(df.columns)}")
print(f"[load] date range : {df['date'].min().date()} .. {df['date'].max().date()}")

I = df["new_cases"].to_numpy(dtype=float) # daily incidence I_t
dates = df["date"].to_numpy()
n = len(I)
assert np.all(I >= 0), "Negative incidence values detected"
assert not np.any(np.isnan(I)), "NaN incidence values detected"
print(f"[load] n days : {n}")
```

```
print(f"[load] total cases : {I.sum():,.0f}")

# ------------------------------------------------------------------------
# 2. Discretize serial interval distribution
# ------------------------------------------------------------------------
# Gamma parameters from mean/sd (scipy: shape=k, scale=theta)
shape_k = SI_MEAN**2 / SI_SD**2
scale_theta = SI_SD**2 / SI_MEAN
print(f"\n[SI] Gamma mean={SI_MEAN}, sd={SI_SD}")
print(f"[SI] shape k = {shape_k:.6f}, scale theta = {scale_theta:.6f}")

s_vals = np.arange(1, S_MAX + 1)
G = lambda x: stats.gamma.cdf(x, a=shape_k, scale=scale_theta)
w = G(s_vals + 0.5) -G(s_vals -0.5) # w_1 .. w_30
cum_before_norm = G(S_MAX + 0.5)
print(f"[SI] cumulative prob up to s={S_MAX}+0.5 : {cum_before_norm:.6f}")
w = w / w.sum() # normalize so sum_{s=1}^{30} w_s = 1
# Build full weight vector with w_0 = 0 for convenient indexing: w_full[s]
w_full = np.concatenate(([0.0], w)) # index 0..30
print(f"[SI] sum of normalized w : {w.sum():.6f}")
print(f"[SI] w_1..w_5 : {np.round(w[:5], 5)}")
print(f"[SI] mode (argmax s) : s={s_vals[np.argmax(w)]}")

# ------------------------------------------------------------------------
# 3. Total infectiousness Lambda_t and posterior Rt over sliding windows
# ------------------------------------------------------------------------
# Lambda_t = sum_{s=1}^{min(t, S_MAX)} I_{t-s} * w_s (t is 0-based here)
Lambda = np.zeros(n)
for t in range(n):
    s_max_t = min(t, S_MAX) # need t-s >= 0
    if s_max_t >= 1:
        s_arr = np.arange(1, s_max_t + 1)
        Lambda[t] = np.sum(I[t -s_arr] * w_full[s_arr])
print(f"\n[Lambda] computed for {n} days; Lambda[0]={Lambda[0]:.3f}, "
      f"Lambda[-1]={Lambda[-1]:.3f}")

# Sliding window of length TAU ending at day t: window indices [t-TAU+1, t].
# Cori et al. require at least one full window; first estimable end-day index
# is t = TAU -1 (0-based) = day 7 (1-based). Estimates are reported at the
# window END date.
mean_rt = np.full(n, np.nan)
low_95 = np.full(n, np.nan)
high_95 = np.full(n, np.nan)
post_a = np.full(n, np.nan)
post_b = np.full(n, np.nan)

for t in range(TAU -1, n):
    idx = np.arange(t -TAU + 1, t + 1) # window [t-6 .. t]
    sum_I = I[idx].sum()
    sum_Lambda = Lambda[idx].sum()
    a_t = PRIOR_SHAPE_A0 + sum_I
    inv_b_t = (1.0 / PRIOR_SCALE_B0) + sum_Lambda
    b_t = 1.0 / inv_b_t
    post_a[t] = a_t
    post_b[t] = b_t
    mean_rt[t] = a_t * b_t
    low_95[t], high_95[t] = stats.gamma.ppf([0.025, 0.975], a=a_t, scale=b_t)

    if (t -(TAU -1)) % 20 == 0:
        print(f"[Rt] t={t:3d} ({pd.Timestamp(dates[t]).date()}): "
              f"mean={mean_rt[t]:.3f} [{low_95[t]:.3f}, {high_95[t]:.3f}]")

# ------------------------------------------------------------------------
# 4. Build results frame (only estimable days t >= TAU-1) & save
# ------------------------------------------------------------------------
out = pd.DataFrame({
    "date": pd.to_datetime(dates),
    "mean_rt": mean_rt,
    "low_95": low_95,
    "high_95": high_95,
})
out_valid = out.dropna(subset=["mean_rt"]).reset_index(drop=True)
out_valid.to_csv(OUT_CSV, index=False)
```

```
print(f"\n[save] Rt series -> {OUT_CSV} ({len(out_valid)} estimated days)")
print(out_valid.head())
print("...")
print(out_valid.tail())

# ------------------------------------------------------------------------
# 5. Identify threshold-crossing dates
# ------------------------------------------------------------------------
below_mean = out_valid[out_valid["mean_rt"] < 1.0]
below_upper = out_valid[out_valid["high_95"] < 1.0]

date_mean_below = (pd.Timestamp(below_mean["date"].iloc[0]).date()
                   if len(below_mean) else None)
date_upper_below = (pd.Timestamp(below_upper["date"].iloc[0]).date()
                    if len(below_upper) else None)

print("\n" + "=" * 70)
print("THRESHOLD CROSSINGS (Rt < 1)")
print("=" * 70)
print(f"Mean Rt first < 1.0 : {date_mean_below}")
if date_mean_below is not None:
    row = below_mean.iloc[0]
    print(f" -> mean={row['mean_rt']:.3f} "
          f"[{row['low_95']:.3f}, {row['high_95']:.3f}]")
print(f"Upper 95% CI first < 1.0 : {date_upper_below}")
if date_upper_below is not None:
    row = below_upper.iloc[0]
    print(f" -> mean={row['mean_rt']:.3f} "
          f"[{row['low_95']:.3f}, {row['high_95']:.3f}]")

rt_start = out_valid.iloc[0]
rt_end = out_valid.iloc[-1]
print(f"\nFirst estimate {pd.Timestamp(rt_start['date']).date()}: "
      f"Rt={rt_start['mean_rt']:.3f} "
      f"[{rt_start['low_95']:.3f}, {rt_start['high_95']:.3f}]")
print(f"Last estimate {pd.Timestamp(rt_end['date']).date()}: "
      f"Rt={rt_end['mean_rt']:.3f} "
      f"[{rt_end['low_95']:.3f}, {rt_end['high_95']:.3f}]")
print(f"Peak mean Rt {out_valid['mean_rt'].max():.3f} on "
      f"{pd.Timestamp(out_valid.loc[out_valid['mean_rt'].idxmax(),'date']).date()}")

# ------------------------------------------------------------------------
# 6. Plot
# ------------------------------------------------------------------------
plt.rcParams.update({
    "font.family": "sans-serif",
    "font.size": 11,
    "axes.linewidth": 0.8,
})
fig, ax = plt.subplots(figsize=(11, 6))

d = out_valid["date"]
ax.plot(d, out_valid["mean_rt"], color="#b2182b", lw=2.0,
        label="Posterior mean $R_t$", zorder=3)
ax.fill_between(d, out_valid["low_95"], out_valid["high_95"],
                color="#b2182b", alpha=0.22, label="95% credible interval",
                zorder=2)
ax.axhline(1.0, color="black", ls="--", lw=1.2, label="$R_t = 1$", zorder=1)

# Mark threshold crossings
if date_mean_below is not None:
    xm = pd.Timestamp(date_mean_below)
    ax.axvline(xm, color="#2166ac", ls=":", lw=1.4, zorder=1)
    ax.annotate(f"mean $R_t<1$\n{date_mean_below}",
                xy=(xm, 1.0), xytext=(8, 40), textcoords="offset points",
                fontsize=9, color="#2166ac",
                arrowprops=dict(arrowstyle="->", color="#2166ac", lw=1.0))
if date_upper_below is not None:
    xu = pd.Timestamp(date_upper_below)
    ax.axvline(xu, color="#4d9221", ls=":", lw=1.4, zorder=1)
    ax.annotate(f"upper 95% $R_t<1$\n{date_upper_below}",
                xy=(xu, 1.0), xytext=(8, -55), textcoords="offset points",
                fontsize=9, color="#4d9221",
```

```
                arrowprops=dict(arrowstyle="->", color="#4d9221", lw=1.0))

ax.set_xlabel("Date (2020)")
ax.set_ylabel("Instantaneous reproduction number $R_t$")
ax.set_title("Italy COVID-19 First Wave: Instantaneous Reproduction Number $R_t$\n"
             "(Cori et al. 2013; SI Gamma mean=4.8, sd=2.3; 7-day sliding window)",
             fontsize=12)
ax.set_ylim(bottom=0)
ax.legend(loc="upper right", framealpha=0.95)
ax.grid(True, alpha=0.3)
ax.xaxis.set_major_locator(mdates.WeekdayLocator(interval=2))
ax.xaxis.set_major_formatter(mdates.DateFormatter("%b %d"))
fig.autofmt_xdate(rotation=45)
fig.tight_layout()
fig.savefig(OUT_FIG, dpi=300, bbox_inches="tight")
plt.close(fig)
print(f"\n[plot] saved -> {OUT_FIG}")

# ---------------------------------------------------------------------------
# 7. Emit a small JSON summary for downstream / manifest use
# ---------------------------------------------------------------------------
summary = {
    "si_shape_k": shape_k,
    "si_scale_theta": scale_theta,
    "si_cumprob_30": float(cum_before_norm),
    "window_tau": TAU,
    "prior_shape_a0": PRIOR_SHAPE_A0,
    "prior_scale_b0": PRIOR_SCALE_B0,
    "n_estimates": int(len(out_valid)),
    "first_estimate_date": str(pd.Timestamp(rt_start["date"]).date()),
    "last_estimate_date": str(pd.Timestamp(rt_end["date"]).date()),
    "first_estimate_rt": float(rt_start["mean_rt"]),
    "last_estimate_rt": float(rt_end["mean_rt"]),
    "peak_mean_rt": float(out_valid["mean_rt"].max()),
    "peak_mean_rt_date": str(pd.Timestamp(
        out_valid.loc[out_valid["mean_rt"].idxmax(), "date"]).date()),
    "date_mean_rt_below_1": str(date_mean_below) if date_mean_below else None,
    "date_upper95_below_1": str(date_upper_below) if date_upper_below else None,
}
with open(f"{SESSION}/results/step2_rt_summary.json", "w") as fh:
    json.dump(summary, fh, indent=2)
print(f"[save] summary -> {SESSION}/results/step2_rt_summary.json")
print("\nDONE.")
```

C SIR Model Fitting and Peak Prediction (`03_sir_fitting.py`)

```
"""
Step 3 & 4: Deterministic SIR model fitting and peak-prediction evaluation for
Italy's first COVID-19 wave (2020-02-24 -> 2020-05-31).

Model
-----
Classic deterministic SIR on a closed population of size N:
    dS/dt = -beta * S * I / N
    dI/dt = beta * S * I / N -gamma * I
    dR/dt = gamma * I

Daily incidence (new infections per day) is the *flow* out of the susceptible
compartment, NOT the prevalence I(t). For integer model days t = 0, 1, ..., T,
the predicted new cases attributable to day t are

    C(t) = S(t-1) -S(t) (the daily decrease in susceptibles).

We align model day t = 1..T with observed days index 0..T-1, i.e. the predicted
incidence for observation i is inc[i] = S(i) -S(i+1) (= -diff(S)). This is the
standard "fit the flow, not the stock" approach and avoids the common error of
fitting the active-infectious prevalence I(t) directly to daily case counts.

Fitting
-------
```

```
Free parameters: beta > 0, gamma > 0, I0 >= 1 (with S0 = N -I0, R0 = 0).
Objective: sum of squared errors (SSE) between predicted daily incidence and the
observed daily cases. A global search (differential_evolution, seeded) is followed
by a bounded local polish (L-BFGS-B). Both the reconstructed *raw* daily series and
the *7-day smoothed* series are fit independently; the smoothed fit is the primary
reported model (more stable target), and the raw fit is reported for comparison.

Basic reproduction number: R0 = beta / gamma.

Outputs
-------
results/step3_sir_summary.json fitted params, R0, peak dates, peak error (days)
figures/italy_sir_fit.png observed (raw + smoothed) vs fitted SIR incidence
"""

import json
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp
from scipy.optimize import differential_evolution, minimize
import matplotlib

matplotlib.use("Agg")
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

np.random.seed(42)

SESSION = "/app/sandbox/session_20260710_145447_12a9d39594f7"
INPUT_CSV = f"{SESSION}/results/italy_covid_cleaned.csv"
OUT_JSON = f"{SESSION}/results/step3_sir_summary.json"
OUT_FIG = f"{SESSION}/figures/italy_sir_fit.png"

N_POP = 60_300_000 # approximate Italian population, 2020


# ---------------------------------------------------------------------------#
# SIR machinery
# ---------------------------------------------------------------------------#
def sir_rhs(t, y, beta, gamma, N):
    """Right-hand side of the SIR ODE system."""
    S, I, R = y
    infection = beta * S * I / N
    recovery = gamma * I
    return [-infection, infection -recovery, recovery]


def simulate_incidence(beta, gamma, I0, N, n_days):
    """Solve the SIR system and return predicted daily incidence.

    Integrates on integer days t = 0..n_days (n_days+1 points). Daily incidence
    for observation index i (i = 0..n_days-1) is the susceptible flow
    inc[i] = S(i) -S(i+1). Returns (inc, sol_S, sol_I, sol_R) on the grid 0..n_days.
    """
    S0 = N -I0
    y0 = [S0, I0, 0.0]
    t_eval = np.arange(0, n_days + 1) # 0, 1, ..., n_days
    sol = solve_ivp(
        sir_rhs,
        (0.0, float(n_days)),
        y0,
        t_eval=t_eval,
        args=(beta, gamma, N),
        method="RK45",
        rtol=1e-7,
        atol=1e-2,
        max_step=1.0,
    )
    S = sol.y[0]
    inc = -np.diff(S) # S[i] -S[i+1], length n_days, aligned to obs 0..n_days-1
    inc = np.clip(inc, 0.0, None)
    return inc, sol.y[0], sol.y[1], sol.y[2]

```

```python

def sse(params, observed, N):
    """Sum of squared errors between predicted and observed daily incidence."""
    beta, gamma, I0 = params
    if beta <= 0 or gamma <= 0 or I0 < 1:
        return 1e30
    n_days = len(observed)
    inc, *_ = simulate_incidence(beta, gamma, I0, N, n_days)
    if not np.all(np.isfinite(inc)):
        return 1e30
    resid = inc -observed
    return float(np.sum(resid ** 2))


def fit_sir(observed, N, label):
    """Global (differential_evolution) + local (L-BFGS-B) fit. Returns dict."""
    print(f"\n[fit:{label}] Global search (differential_evolution)...")
    # Physically realistic constraints:
    # gamma in [1/21, 1/3] -> mean infectious period 3..21 days (COVID-plausible)
    # beta in [0.02, 1.5]
    # I0 in [1, 5e4]
    bounds = [
        (0.02, 1.5), # beta
        (1.0 / 21.0, 1.0 / 3.0), # gamma (0.0476 .. 0.3333)
        (1.0, 5.0e4), # I0
    ]
    result_global = differential_evolution(
        sse,
        bounds,
        args=(observed, N),
        seed=42,
        maxiter=300,
        popsize=25,
        tol=1e-8,
        polish=True,
        updating="deferred",
        workers=1,
    )
    x0 = result_global.x
    print(f"[fit:{label}] DE best SSE = {result_global.fun:.4e} at "
          f"beta={x0[0]:.4f}, gamma={x0[1]:.4f}, I0={x0[2]:.2f}")

    print(f"[fit:{label}] Local polish (L-BFGS-B)...")
    result_local = minimize(
        sse,
        x0,
        args=(observed, N),
        method="L-BFGS-B",
        bounds=bounds,
        options={"maxiter": 2000, "ftol": 1e-12, "gtol": 1e-10},
    )
    best = result_local if result_local.fun < result_global.fun else result_global
    beta, gamma, I0 = best.x
    r0 = beta / gamma
    rmse = float(np.sqrt(best.fun / len(observed)))
    infectious_period = 1.0 / gamma

    # Diagnose whether any parameter is pinned at a bound (relative tol 1%).
    names = ["beta", "gamma", "I0"]
    at_bound = []
    for nm, val, (lo, hi) in zip(names, best.x, bounds):
        span = hi -lo
        if abs(val -lo) <= 0.01 * span:
            at_bound.append(f"{nm}@lower")
        elif abs(val -hi) <= 0.01 * span:
            at_bound.append(f"{nm}@upper")

    print(f"[fit:{label}] FINAL SSE={best.fun:.4e} RMSE={rmse:.2f} "
          f"beta={beta:.4f} gamma={gamma:.4f} (1/gamma={infectious_period:.2f}d) "
          f"I0={I0:.2f} R0={r0:.3f}")
    if at_bound:
        print(f"[fit:{label}] NOTE: parameters at bound: {', '.join(at_bound)}")
    return {
```

```
        "beta": float(beta),
        "gamma": float(gamma),
        "I0": float(I0),
        "R0": float(r0),
        "infectious_period_days": float(infectious_period),
        "sse": float(best.fun),
        "rmse": rmse,
        "params_at_bound": at_bound,
    }


# ---------------------------------------------------------------------------#
# Main
# ---------------------------------------------------------------------------#
def main():
    print("=" * 70)
    print("Step 3 & 4: SIR model fitting + peak prediction")
    print("=" * 70)

    df = pd.read_csv(INPUT_CSV, parse_dates=["date"])
    df = df.sort_values("date").reset_index(drop=True)
    dates = df["date"]
    raw = df["new_cases"].to_numpy(dtype=float)
    smooth = df["new_cases_smoothed"].to_numpy(dtype=float)
    n_days = len(df)
    print(f"Loaded {n_days} days: {dates.iloc[0].date()} -> {dates.iloc[-1].date()}")
    print(f"Population N = {N_POP:,}")
    print(f"Raw daily cases: min={raw.min():.0f} max={raw.max():.0f} "
          f"total={raw.sum():.0f}")

    # ---------------------------------------------------------------------#
    # Fit both targets
    # ---------------------------------------------------------------------#
    fit_smooth = fit_sir(smooth, N_POP, "smoothed")
    fit_raw = fit_sir(raw, N_POP, "raw")

    # Primary model = smoothed-series fit (more stable incidence target).
    primary = fit_smooth
    primary_label = "smoothed"

    # ---------------------------------------------------------------------#
    # Peak prediction evaluation (primary fit)
    # ---------------------------------------------------------------------#
    inc_pred_smooth, _, _, _ = simulate_incidence(
        fit_smooth["beta"], fit_smooth["gamma"], fit_smooth["I0"], N_POP, n_days
    )
    inc_pred_raw, _, _, _ = simulate_incidence(
        fit_raw["beta"], fit_raw["gamma"], fit_raw["I0"], N_POP, n_days
    )

    # Observed peaks (use smoothed to define the observed peak robustly; also
    # record the raw-series argmax for transparency).
    obs_peak_idx_smooth = int(np.argmax(smooth))
    obs_peak_idx_raw = int(np.argmax(raw))
    obs_peak_date_smooth = dates.iloc[obs_peak_idx_smooth]
    obs_peak_date_raw = dates.iloc[obs_peak_idx_raw]

    # Predicted peak from the primary (smoothed) fit.
    pred_peak_idx = int(np.argmax(inc_pred_smooth))
    pred_peak_date = dates.iloc[pred_peak_idx]

    # Peak error in days = predicted -observed (observed defined on smoothed).
    peak_error_days = int((pred_peak_date -obs_peak_date_smooth).days)

    print("\n" + "-" * 70)
    print("PEAK PREDICTION (primary = smoothed fit)")
    print(f" Observed peak (smoothed): {obs_peak_date_smooth.date()} "
          f"(value {smooth[obs_peak_idx_smooth]:.0f})")
    print(f" Observed peak (raw): {obs_peak_date_raw.date()} "
          f"(value {raw[obs_peak_idx_raw]:.0f})")
    print(f" Predicted peak (SIR): {pred_peak_date.date()} "
          f"(value {inc_pred_smooth[pred_peak_idx]:.0f})")
    print(f" Peak error (pred -obs): {peak_error_days:+d} days")
```

```
244
245     # ---------------------------------------------------------------------#
246     # Figure
247     # ---------------------------------------------------------------------#
248     print("\nGenerating figure...")
249     plt.rcParams["font.family"] = "sans-serif"
250     plt.rcParams["font.size"] = 10
251     fig, ax = plt.subplots(figsize=(11, 6.5))
252
253     ax.bar(dates, raw, width=1.0, color="#c6d9f0", edgecolor="none",
254            label="Observed daily cases (raw, reconstructed)", zorder=1)
255     ax.plot(dates, smooth, color="#1f4e79", lw=2.0,
256             label="Observed daily cases (7-day smoothed)", zorder=3)
257     ax.plot(dates, inc_pred_smooth, color="#c00000", lw=2.4, ls="-",
258             label=(f"Fitted SIR incidence (smoothed fit)\n"
259                    f"R0={fit_smooth['R0']:.2f}, "
260                    fr"$\beta$={fit_smooth['beta']:.3f}, "
261                    fr"$\gamma$={fit_smooth['gamma']:.3f}"),
262             zorder=4)
263     ax.plot(dates, inc_pred_raw, color="#ed7d31", lw=1.6, ls="--",
264             label=(f"Fitted SIR incidence (raw fit)\n"
265                    f"R0={fit_raw['R0']:.2f}"),
266             zorder=3)
267
268     # Peak markers
269     ax.axvline(obs_peak_date_smooth, color="#1f4e79", ls=":", lw=1.3, zorder=2)
270     ax.axvline(pred_peak_date, color="#c00000", ls=":", lw=1.3, zorder=2)
271     ymax = max(raw.max(), smooth.max(), inc_pred_smooth.max())
272     ax.annotate(f"Observed peak\n{obs_peak_date_smooth.date()}",
273                 xy=(obs_peak_date_smooth, smooth[obs_peak_idx_smooth]),
274                 xytext=(obs_peak_date_smooth, ymax * 1.02),
275                 ha="center", va="bottom", color="#1f4e79", fontsize=8.5)
276     ax.annotate(f"Predicted peak\n{pred_peak_date.date()}",
277                 xy=(pred_peak_date, inc_pred_smooth[pred_peak_idx]),
278                 xytext=(pred_peak_date, ymax * 1.12),
279                 ha="center", va="bottom", color="#c00000", fontsize=8.5)
280
281     ax.set_title("Italy COVID-19 First Wave: SIR Model Fit to Daily Incidence\n"
282                  f"Peak prediction error = {peak_error_days:+d} days "
283                  f"(N = {N_POP:,})", fontsize=12)
284     ax.set_xlabel("Date")
285     ax.set_ylabel("New cases per day")
286     ax.set_ylim(0, ymax * 1.22)
287     ax.xaxis.set_major_locator(mdates.WeekdayLocator(interval=2))
288     ax.xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m-%d"))
289     plt.setp(ax.get_xticklabels(), rotation=45, ha="right")
290     ax.legend(loc="upper right", fontsize=8.5, framealpha=0.95)
291     ax.grid(True, alpha=0.3, ls="--")
292     fig.tight_layout()
293     fig.savefig(OUT_FIG, dpi=300, bbox_inches="tight")
294     plt.close(fig)
295     print(f"Saved figure -> {OUT_FIG}")
296
297     # ---------------------------------------------------------------------#
298     # JSON summary
299     # ---------------------------------------------------------------------#
300     summary = {
301         "model": "deterministic SIR (closed population)",
302         "population_N": N_POP,
303         "n_days": n_days,
304         "date_start": str(dates.iloc[0].date()),
305         "date_end": str(dates.iloc[-1].date()),
306         "incidence_definition": "daily new cases = S(t-1) -S(t) (susceptible flow), fit to daily incidence
                not prevalence I(t)",
307         "optimizer": "differential_evolution (seed=42) + L-BFGS-B polish; objective = SSE",
308         "primary_fit_target": primary_label,
309
310         # Primary (smoothed) fitted parameters at top level (task requirement)
311         "beta": primary["beta"],
312         "gamma": primary["gamma"],
313         "I0": primary["I0"],
314         "R0": primary["R0"],
315         "S0": float(N_POP -primary["I0"]),
```

```
        "R_init": 0.0,
        "sse": primary["sse"],
        "rmse": primary["rmse"],

        # Peak evaluation
        "observed_peak_date": str(obs_peak_date_smooth.date()),
        "observed_peak_date_raw_series": str(obs_peak_date_raw.date()),
        "observed_peak_value_smoothed": float(smooth[obs_peak_idx_smooth]),
        "observed_peak_value_raw": float(raw[obs_peak_idx_raw]),
        "predicted_peak_date": str(pred_peak_date.date()),
        "predicted_peak_value": float(inc_pred_smooth[pred_peak_idx]),
        "peak_error_days": peak_error_days,

        # Both fits, for comparison / sensitivity
        "fit_smoothed": fit_smooth,
        "fit_raw": fit_raw,

        # Parameter bounds enforced during optimization
        "bounds": {
            "beta": [0.02, 1.5],
            "gamma": [1.0 / 21.0, 1.0 / 3.0],
            "gamma_infectious_period_days": [3.0, 21.0],
            "I0": [1.0, 5.0e4],
        },

        # Honest interpretation of the result (structural limitation).
        "interpretation": (
            "Observed cumulative confirmed cases over the window (232,585) are only "
            "~0.39% of N = 60.3M. Susceptible depletion is therefore negligible "
            "(S/N stays ~= 1), so the closed constant-transmission SIR operates in its "
            "pure-exponential regime and its daily incidence is monotonic over the "
            "98-day window -it cannot bend over to reproduce an INTERIOR peak. The "
            "SSE-optimal constant-beta fit is consequently a near-flat, slowly declining "
            "line with beta ~= gamma (R0 ~= 1.0) that pins the predicted incidence peak "
            "to the series start, yielding a peak-timing error of about -31 days. Note "
            "gamma is pinned at its upper (fast-recovery) bound: a symptom of this "
            "degeneracy, not a credible infectious-period estimate."
        ),
        "caveats": [
            "A constant-beta SIR cannot represent Italy's first-wave turnover, which was "
            "driven by non-pharmaceutical interventions (national lockdown 2020-03-09) that "
            "reduced transmission over time, NOT by herd-immunity susceptible depletion.",
            "The fitted R0 ~= 1.0 is NOT a credible basic reproduction number; it reflects "
            "the flat-line degeneracy. It is inconsistent with the Step-2 renewal-equation "
            "peak Rt ~= 3.23 and with published Italian first-wave R0 estimates (~2.5-3.5).",
            "Under-ascertainment (confirmed cases << true infections) shrinks the APPARENT "
            "attack rate, deepening the negligible-depletion problem and biasing an SIR "
            "calibrated on confirmed cases toward R0 ~= 1 and away from the true value.",
            "Physically meaningful SIR calibration here would require a time-varying beta(t) "
            "(or a change-point at lockdown), an effective/susceptible population N_eff << N, "
            "or fitting to ascertainment-corrected infections; these are out of scope for "
            "the specified constant-parameter, fixed-N task but are the correct next steps.",
        ],
    }
    with open(OUT_JSON, "w") as fh:
        json.dump(summary, fh, indent=2)
    print(f"Saved summary -> {OUT_JSON}")

    print("\n=== DONE ===")
    print(json.dumps(
        {k: summary[k] for k in
         ["beta", "gamma", "I0", "R0", "observed_peak_date",
          "predicted_peak_date", "peak_error_days", "rmse"]},
        indent=2))


if __name__ == "__main__":
    main()
```