

Figure S1. Missingness analysis of BMI with respect to Charlson Comorbidity Index. A box plot summarizing Charlson Comorbidity Index [1] of patients for whom BMI is missing (bottom box plot) or present (top box plot) is shown on the x-axis. A box plot summarizing BMI is shown on the y-axis. Graphic generated with the R VIM package [2]. There is no missing data for the Charlson Comorbidity Index. The plot suggests BMI data is missing completely at random.

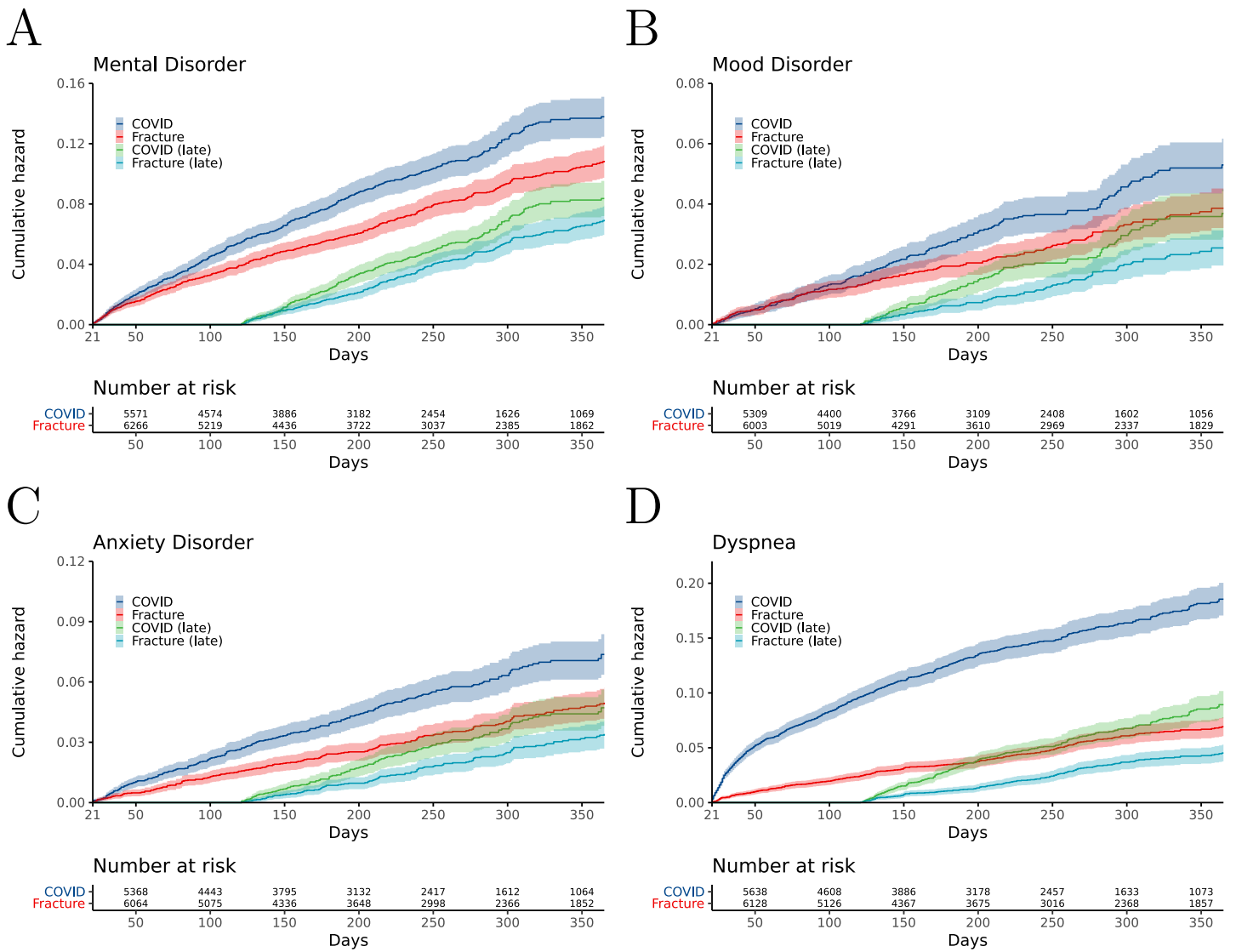


Figure S2. Kaplan-Meier curves for onset of first psychiatric diagnoses after COVID-19 diagnosis compared with large-bone fracture.

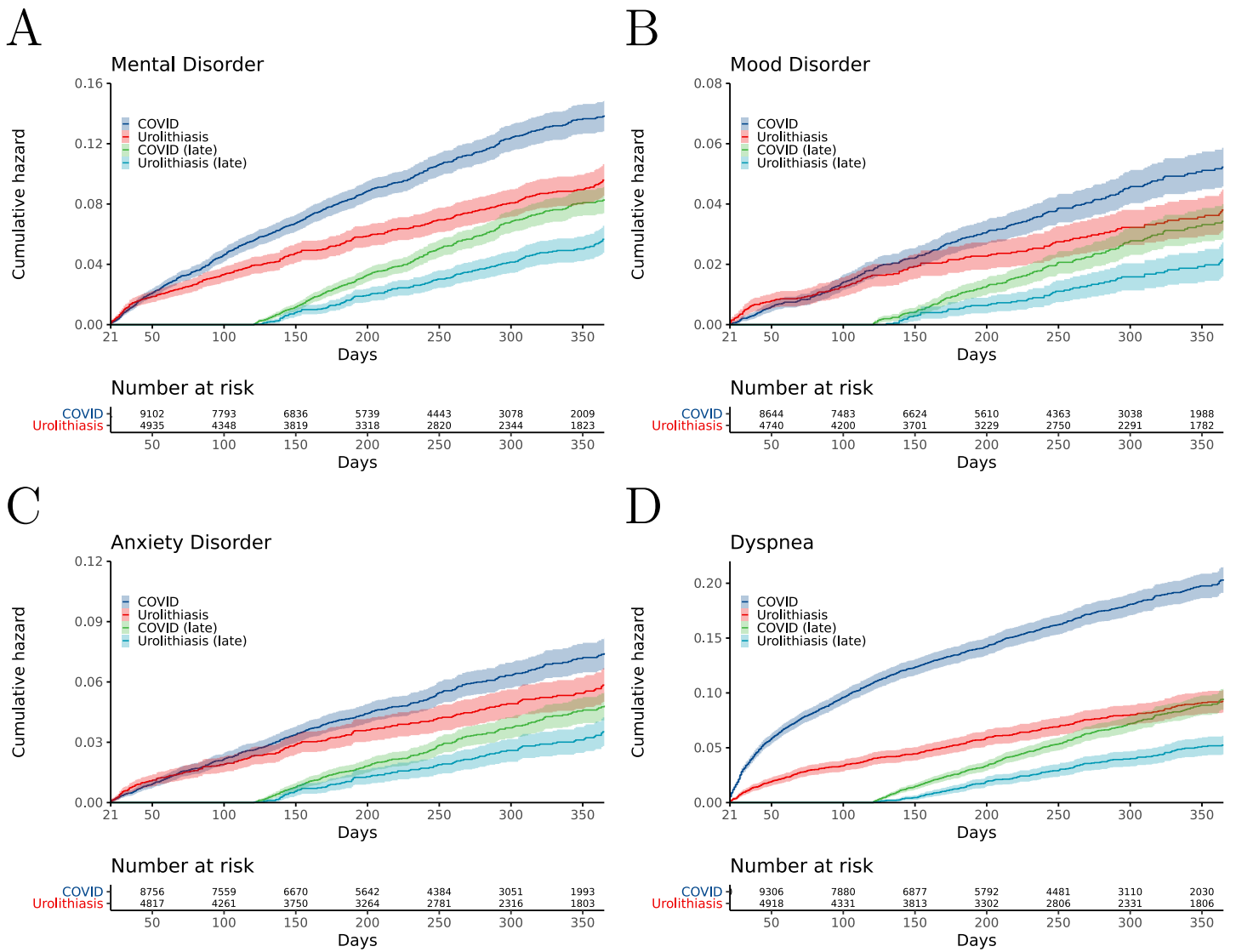


Figure S3. Kaplan-Meier curves for onset of first psychiatric diagnoses after COVID-19 diagnosis compared with urolithiasis.

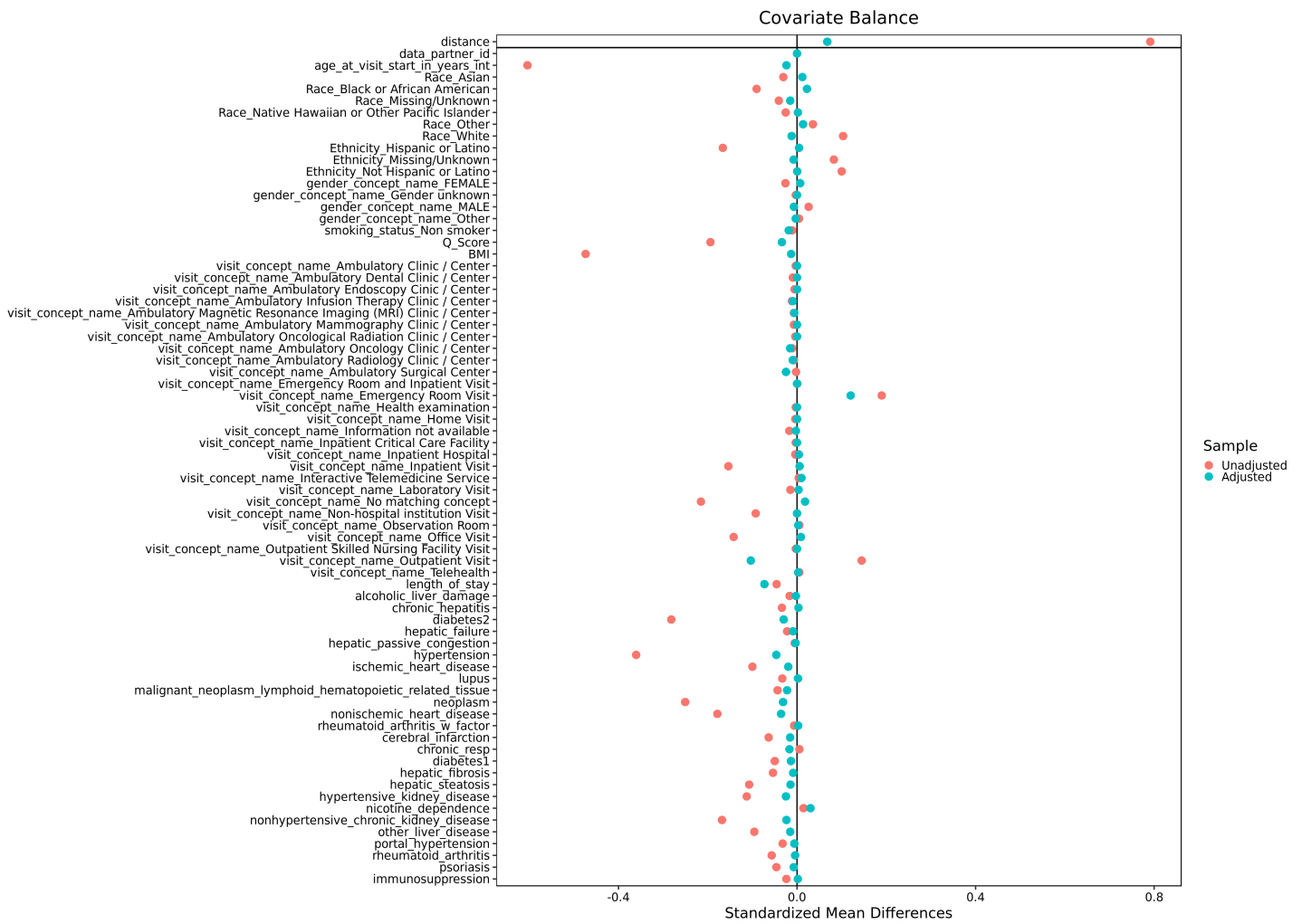


Figure S4. Love plot showing the results of propensity score matching for COVID-19 and other respiratory tract infections (RTI). Propensity matching of other groups showed in comparable results.

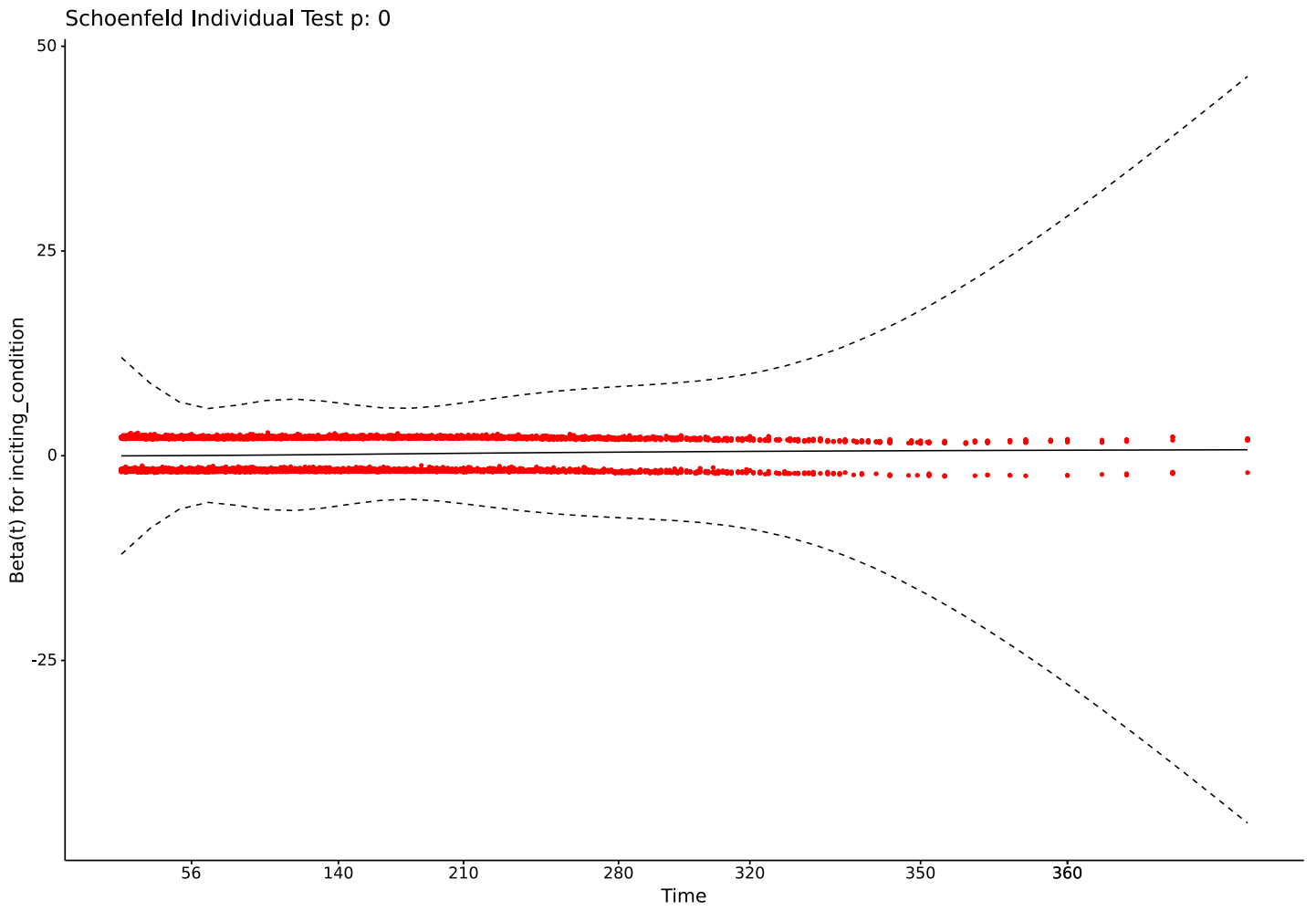


Figure S5. Schoenfeld residual plot for COVID-19 and other respiratory tract infections (RTI).

Outcome	COVID-19	Fracture	HR	HR <i>p</i> -val	PH <i>p</i> -val
		120 Days			
All psychiatric illness	5.3% (4.7-5.9)	3.8% (3.4-4.3)	1.4 (1.2-1.6)	0.00040	0.19
Mood disorder	1.6% (1.2-1.9)	1.3% (1.0-1.6)	1.2 (0.91-1.7)	0.17	0.09
Anxiety	2.6% (2.2-3.1)	1.5% (1.2-1.9)	1.7 (1.3-2.3)	<0.0001	0.70
Fatigue	6.8% (6.1-7.5)	3.9% (3.4-4.4)	1.8 (1.6-2.2)	<0.0001	0.0082
Dyspnea	9.2% (8.4-9.9)	2.4% (2-2.8)	4.0 (3.4-4.9)	<0.0001	0.026
		365 Days			
All psychiatric illness	12.9% (11.7-14.0)	10.3% (9.3-11.3)	1.3 (1.0-1.5)	0.014	0.27
Mood disorder	5.2% (4.3-6.0)	3.8% (3.2-4.4)	1.6 (1.2-2.1)	0.0034	0.37
Anxiety	7.1% (6.2-8.0)	4.8% (4.1-5.6)	1.5 (1.1-1.9)	0.0038	0.21
Fatigue	14.2% (13-15)	8.9% (8.0-9.8)	1.8 (1.4-2.2)	<0.0001	0.073
Dyspnea	16.9% (16-18)	6.7% (5.9-7.5)	2.1 (1.7-2.5)	<0.0001	0.16

Table S1. Fracture. HR: hazard ratio; PH: proportional hazards

Outcome	COVID-19	Urolithiasis	HR	HR <i>p</i> -val	PH <i>p</i> -val
120 Days					
All psychiatric illness	5.4% (4.9-5.9)	3.9% (3.3-4.4)	1.4 (1.2-1.6)	0.00017	0.0077
Mood disorder	1.8% (1.5-2.1)	1.6% (1.3-2)	1.0 (0.8-1.4)	0.73	0.029
Anxiety	2.6% (2.2-2.9)	2.3% (1.9-2.7)	1.1 (0.88-1.4)	0.40	0.077
Fatigue	6.6% (6.1-7.1)	3.7% (3.2-4.2)	1.8 (1.5-2.1)	<0.0001	0.26
Dyspnea	10.3% (9.7-11)	3.9% (3.4-4.5)	2.6 (2.3-3.1)	<0.0001	0.10
365 Days					
All psychiatric illness	13% (12.1-13.8)	9.1% (8.2-10.1)	1.5 (1.3-1.8)	<0.0001	0.15
Mood disorder	5.1% (4.5-5.7)	3.7% (3.1-4.4)	1.7 (1.2-2.2)	0.0010	0.29
Anxiety	7.2% (6.5-7.9)	5.7% (4.9-6.5)	1.4 (1.1-1.8)	0.0070	0.49
Fatigue	15% (13.8-15.7)	8.8% (7.9-9.8)	1.7 (1.4-2.1)	<0.0001	0.33
Dyspnea	18% (17.4-19.3)	8.8% (7.9-9.7)	1.7 (1.4-2.1)	<0.0001	0.39

Table S2. Urolithiasis. HR: hazard ratio; PH: proportional hazards

Predictor	HR (95% CI) w/o freq	HR (95% CI) w freq
COVID	1.3 (1.2-1.4)	1.2 (1.1-1.3)
BMI	1.0 (1.0-1.0)	1.0 (1.0-1.0)
Non-smoker	0.74 (0.68-0.81)	0.71 (0.65-0.77)
Carleson comorbidity	1.1 (1.1-1.1)	1.1 (1.0-1.1)
age	1.0 (1.0-1.0)	1.0 (1.0-1.0)
African American	1.5 (1.1-2.0)	1.5 (1.1-2.0)
White	1.7 (1.2-2.3)	1.6 (1.2-2.2)
Outpatient	0.62 (0.55-0.70)	0.64 (0.56-0.72)
Not hispanic	1.2 (1.1-1.4)	1.2 (1.1-1.4)
Male	0.73 (0.68-0.80)	0.71 (0.66-0.77)
visit frequency	-	3.9 (3.6-4.2)

Table S3. Predictors from regression with and without post-COVID visit frequency. The major predictors are shown together with their 95% confidence intervals. Cox regression was performed in the same way in both experiments except that visit frequency was included in only one experiment. Visit frequency was defined as the number of visits starting at day 21 following the initial event (diagnosis of COVID-19 or RTI). The number of visits was counted up to the final event in the survival curve (either diagnosis of a mental illness or the last day of observation), and was divided the the total number of days between day 21 and the final event.

S1 Supplementary Note: Code documentation

This project was conducted in the National Institute of Health (NIH) N3C Data Enclave on the Palantir Foundry platform (Palantir Technologies Inc., Denver, Colorado). This platform organizes code in nodes (transformations) that form a directed acyclic graph and cannot easily be presented in linearized fashion. The purpose of this documentation is to illustrate the approach taken by our analysis and to promote reproducibility and extensibility on the Palantir platform or (with appropriate changes) on other platforms.

The Palantir platform enables each node to use a different programming language. We chose SQL, Python, and R according to which technique was best adapted to the task at hand.

The N3C Data Enclave (hereafter “Enclave”) contains data from over 2.4 million COVID-19 positive patients from 65 health systems in the United States. The dataset utilized in this study was frozen on October 20, 2021. This dataset consists of 7,139,696 patients. Data were harmonized, integrated, and mapped with the Observational Medical Outcomes Partnership (OMOP) 5.3.1 vocabulary [3].

S1.1 Data Preparation

The first step of our analysis consisted in the definition of cohorts and the preparation of data needed for the statistical analysis.

S1.2 Inputs

The inputs for our analysis included OMOP tables as well as some tables provided by the Palantir platform with processed data.

S1.2.1 OMOP Table

The following OMOP tables were used for our analysis (Table S4). We refer the reader to the original OMOP documentation for more information details [4].

Table	Summary
<code>condition_occurrence</code>	Dates when a condition is considered to have started and (if applicable) ended
<code>observation_period</code>	Defines the time period for which a patient’s demographics, conditions, procedures and drugs are recorded in the source system with the expectation of a reasonable sensitivity and specificity.

Table S4. OMOP tables used in this study.

In the code workbook, we refer to the `observation_period` table as `Observation_period_oct21` and `condition_occurrence` as `Condition_occurrence_oct21` because we used the October 21, 2021 data freeze.

S1.3 Tables offered by the Enclave

The following Enclave-specific tables were used for our analysis.

S1.3.1 `concept_set_members`

This table defines the relations between the codesets and concept sets (Table S5).

In the code workbooks, we refer to this table as `Concept_set_members_oct21`, because the October 21, 2021 data freeze was used.

codeset_id	concept_id	concept_set_name	concept_name
837398380	436540	Fracture	Open fracture axis
837398380	436824	Fracture	Open fracture of cervical vertebra without spinal cord injury
837398380	436826	Fracture	Closed fracture of shaft of radius
837398380	436832	Fracture	Closed fracture of sternum
837398380	437117	Fracture	Open fracture of intracapsular section of femur

Table S5. concept_set_members table. Some examples from this table, which is used to coordinate codesets created by N3C Enclave members (often using the OHDSI Atlas tool). The table has three additional columns (not shown here) that specify the version and were used to extract the latest versions for analysis.

S1.3.2 complete_patient_table_with_derived_scores

This table is prepared as described [5], and contains derived information that was used to provide information about some of the covariates used in our analysis (Table S6).

Column	data type	comment
person_id	string	unique patient identifier
data_partner_id	integer	identifier of N3C contributing center
visit_concept_id	integer	e.g., 9201 (Inpatient Visit)
visit_start_date	Date	The date of the initial medical encounter for COVID-19 or control condition
visit_concept_name	string	concept name corresponding to visit_concept_id
visit_occurrence_id	string	identifier for a visit
AKI_in_hospital	string	“Yes” or null
ECMO	string	“ECMO” or null
Invasive_Ventilation	string	“Invasive Ventilation” or null
positive_covid_test	string	“true” or null
negative_covid_test	string	“true” or null
Suspected_COVID	string	“true” or null
in_death_table	string	“true” or null. True if patient known to be deceased
age_at_visit_start_in_years_int	integer	age in years
length_of_stay	integer	length of stay in days
Race	string	e.g., “Missing/Unknown”
Ethnicity	string	e.g., “Missing/Unknown”
gender_concept_name	string	e.g., “FEMALE”
smoking_status	string	e.g., “Non smoker”
blood_type	string	e.g., “Unknown”
covid_status_name	string	e.g., “covid_confirmed_positive”
Severity_Type	string	e.g., “Mild”
InpatientOrED	boolean	true or false
Q_Score	integer	Charlson Comorbidity Index
BMI	double	e.g., 18.2
Height	double	in meters
Weight	double	in kilograms
Testcount	integer	Not used in current analysis

Table S6. Fields of the complete_patient_table_with_derived_scores table

S1.4 Data Preparation

The following subsections show the code used for preparing the data for analysis. Each subsection corresponds to a single node (transform) in the Enclave.

S1.4.1 Raw_Comorbidities_Table

Inputs:

- `Concept_set_members_oct21` (Section [S1.3.1](#))
- `Condition_occurrence_oct21` (Section [S1.2.1](#))

Here, we first retrieve the covariate `codeset_id`'s from the OMOP `concept_set_members` table (October 21, 2021 data freeze) (lines 1-4). Then we select all condition occurrences corresponding to these concept sets (lines 6-10). Finally, we select as covariates conditions that existed prior to the patient getting COVID-19 (or the control condition). The code reports the covariate as null if it occurred after covid onset to avoid eliminating any `person_id` from list. (lines 12-14).

```
WITH CSM AS (SELECT *
2 FROM Concept_set_members_oct21
WHERE concept_set_name IN ("alcoholic_liver_damage-bdc", "alzheimers_disease-bdc", "
cerebral_infarction-bdc", "chronic_hepatitis-bdc", "chronic_resp-bdc", "
dementia_associated_with_another_disease-bdc", "diabetes1-bdc", "diabetes2-bdc", "
hepatic_failure-bdc", "hepatic_fibrosis-bdc", "hepatic_passive_congestion-bdc", "
hepatic_steatosis-bdc", "hypertension-bdc", "hypertensive_kidney_disease-bdc", "
immunosuppression-bdc", "ischemic_heart_disease-bdc", "lupus-bdc", "
malignant_neoplasm_lymphoid_hematopoietic_related_tissue-bdc", "neoplasm-bdc", "
nicotine_dependence-bdc", "non-hypertensive_chronic_kidney_disease-bdc", "non-
ischemic_heart_disease-bdc", "other_liver_disease-bdc", "portal_hypertension-bdc", "
psoriasis-bdc", "rheumatoid_arthritis-bdc", "rheumatoid_arthritis_w_factor-bdc", "
unspecified_dementia-bdc", "vascular_dementia-bdc")
4 AND is_most_recent_version = true),
6 COV_OCC AS (SELECT person_id, concept_set_name, MIN(condition_start_date) AS start_date
FROM Condition_occurrence_oct21
8 INNER JOIN CSM
ON CSM.concept_id = Condition_occurrence_oct21.condition_concept_id
10 GROUP BY person_id, concept_set_name)
12 SELECT person_id, IF(start_date < visit_start_date, COV_OCC.concept_set_name, null) AS
covariate
FROM COV_OCC
14 RIGHT JOIN Severity_Table_for_COVID_and_Controls USING(person_id)
```

Listing 1: Data preparation: retrieve covariates.

S1.4.2 Severity_Table_for_COVID_and_Controls

Inputs:

- `Concept_set_members_oct21` (Section [S1.3.1](#))
- `Condition_occurrence_oct21` (Section [S1.2.1](#))
- `Complete_patient_table_with_derived_scores_missing_bmi_sites_removed_oct21` (Section [S1.3.2](#))

We first get `concept_IDs` from predefined concept sets for our control groups. We then select data on covariates from the complete patient table (Section [S1.3.2](#)). Finally, we add a column called `inciting_condition` that contains either "COVID" or the name of a control condition.

```

2 WITH CCS AS (
3     SELECT concept_id, concept_set_name
4     FROM Concept_set_members_oct21
5     WHERE concept_set_name IN ("RTI", "Fracture", "Urolithiasis", "Skin Infection")
6     AND is_most_recent_version = true),
7
8 ST AS (SELECT cpt.person_id, cpt.data_partner_id, visit_start_date,
9     cpt.visit_occurrence_id, positive_covid_test, negative_covid_test,
10    Suspected_COVID, in_death_table, age_at_visit_start_in_years_int, length_of_stay,
11    Race, Ethnicity, gender_concept_name, smoking_status, covid_status_name, Q_Score,
12    BMI, Height, Weight, Testcount, condition_start_date, condition_concept_id, Severity_Type,
13    visit_concept_name
14 FROM Complete_patient_table_with_derived_scores_missing_bmi_sites_removed_oct21 AS cpt
15 LEFT JOIN Condition_occurrence_oct21 USING(visit_occurrence_id))
16
17 SELECT DISTINCT visit_occurrence_id, person_id, IF(positive_covid_test = true, "COVID",
18    CCS.concept_set_name) as inciting_condition, data_partner_id, visit_start_date,
19    age_at_visit_start_in_years_int, length_of_stay, Race, Ethnicity, gender_concept_name,
20    smoking_status, covid_status_name, Q_Score, BMI, Height, Weight, visit_concept_name
21 FROM ST
22 LEFT JOIN CCS
23 ON CCS.concept_id = ST.condition_concept_id
24 WHERE ((ISNULL(ST.Suspected_COVID) AND NOT ISNULL(CCS.concept_id))
25    OR ST.positive_covid_test = true) AND ISNULL(in_death_table)
26 AND Severity_Type != "Dead_w_COVID"

```

Listing 2: The SQL command performs a left join on visit_occurrence_id, which is used in the CPTDS table to denote the visit in which acute COVID-19 or a control condition first occurs.

S1.4.3 Psych_Conditions

Inputs:

- Concept_set_members_oct21 (Section S1.3.1)
- Condition_occurrence_oct21 (Section S1.2.1)

This transform gets patients (person_id), concept names for the psychiatric outcome variables and their start dates.

```

2 WITH PCD AS (
3     SELECT concept_id, concept_set_name
4     FROM Concept_set_members_oct21
5     WHERE concept_set_name IN ("Dyspnea", "Mental Disorder", "Dementia",
6     "Anxiety Disorder", "fatigue", "Mood Disorder", "Psychosis")
7     AND is_most_recent_version = true)
8
9 SELECT person_id, concept_set_name, condition_start_date
10 FROM Condition_occurrence_oct21
11 INNER JOIN PCD
12 ON Condition_occurrence_oct21.condition_concept_id = PCD.concept_id

```

Listing 3: Concept IDs for the control conditions.

S1.4.4 pivot_table

Inputs:

- Raw_Comorbidities_Table (Section S1.4.1)

Create pivot table where each column is a covariate.

```
def pivot_table(filter_by_date):
2     df = filter_by_date
    data = df.groupby("person_id").pivot("covariate").count().drop("null")
4     return data
```

Listing 4: pivot table to have each covariate as a column.

S1.4.5 covariate_list

Here we create a pivot table that makes a column for each of the covariates (which were mentioned in rows in the input table). Note that the suffix `-bdc` was used to keep track of codesets for this project (the Enclave stores codesets from all projects in the same place, and so generally suffices are used for project-specific codesets). We did not include some covariates that are related to psychiatric outcomes such as vascular dementia and Alzheimer's disease, so they are removed here.

```
def covariate_list(pivot_table):
2     df = pivot_table
    # Replace counts greater than 0 with 1
4     def make_binary(inrow):
        # nulls get converted to "nan" with str(x), necessary because of table conversion
        peculiarities
6         outrow = [True if str(x) != "nan" else False for x in inrow]
        return outrow
8     df.iloc[:,1:] = df.iloc[:,1:].apply(make_binary)
    df.columns = df.columns.str.replace('-bdc', '')
10    df.columns = df.columns.str.replace('-', '')
    df = df.drop(columns=["alzheimers_disease", "dementia_associated_with_another_disease",
12    , "unspecified_dementia", "vascular_dementia"])
    return df
```

Listing 5: pivot table to have each covariate as a column.

S1.4.6 eligible_pt_table

Input:

- `Observation_period_oct21` (Section [S1.2.1](#))
- `Severity_Table_for_COVID_and_Controls` (Section [S1.4.2](#))

We first extract the period of observation (needed for the time to event analysis) in lines 1-4. We then remove patients with less than one year of history prior to the first medical encounter with COVID-19 or control event (This is done because we want to perform an analysis on new-onset psychiatric disease following COVID-19. Although the previous history of one year does not guarantee that there was no previous encounter, we define the filter as such because of the limited amount of history available in the Enclave) (lines 6-14). Note that there were rare end dates higher than 50,000 days that we interpreted as data problems; affected data points were removed. The next stanza selects patients who developed a psychiatric condition 21 days or later after the initial encounter with COVID-19 or control condition (lines 15-23).

Finally, the resulting cohort of patients is joined to the covariate list in line 34.

```
WITH OP AS (
2     SELECT person_id, observation_period_end_date, observation_period_start_date
    FROM Observation_period_oct21
4 ),
```

```

6 PIYH AS (
7     SELECT *, DATEDIFF(observation_period_end_date , visit_start_date) AS obs_after_visit
8     FROM Severity_Table_for_COVID_and_Controls
9     INNER JOIN OP USING(person_id)
10    WHERE ((date_sub(Severity_Table_for_COVID_and_Controls.visit_start_date , 364) > OP.
11           observation_period_start_date))
12    AND
13    DATEDIFF(observation_period_end_date , visit_start_date) < 50000
14 ),
15
16 BLP AS (
17     SELECT person_id
18     FROM PIYH
19     INNER JOIN Psych_Conditions AS PC USING(person_id)
20    WHERE PC.concept_set_name IN ( 'Mental Disorder' , 'Anxiety Disorder' , 'Mood Disorder' , '
21           Dementia' )
22    AND
23    date_add(PIYH.visit_start_date , 21) > PC.condition_start_date
24    GROUP BY person_id
25 ),
26
27 RBLP AS (
28     SELECT *
29     FROM PIYH
30    ANTI JOIN BLP USING(person_id)
31    WHERE obs_after_visit >= 21
32 )
33
34 SELECT *
35 FROM RBLP
36 LEFT JOIN covariate_list USING(person_id)

```

Listing 6: eligible patient table.

S1.4.7 mental_disorder_info

Input:

- eligible_pt_table (Section S1.4.6)
- Psych_Conditions (Section S1.4.3)

This transform identifies patients who were diagnosed with a new-onset psychiatric condition 21 or more days after the start of the medical encounter for COVID-19 or the control condition.

```

WITH PDD AS (
2     SELECT eligible_pt_table.person_id ,
3           Psych_Conditions.concept_set_name AS mental_illness ,
4           DATEDIFF(condition_start_date , visit_start_date) as psych_condition_date_diff
5     FROM eligible_pt_table
6     INNER JOIN Psych_Conditions using(person_id)
7 )
8
9 SELECT person_id , mental_illness , MIN(psych_condition_date_diff) AS
10    MIN_psych_condition_date_diff
11 FROM PDD
12 WHERE psych_condition_date_diff >= 21
13 GROUP BY person_id , mental_illness

```

Listing 7: Gather patients with a diagnosis of psychiatric illness that appears at least 21 days after COVID-19 or the control condition.

S1.5 Create Table 1

Input:

- `Eligible_pt_table` (Section [S1.4.6](#))

The following code was used to gather the information used in Table 1 of the manuscript.

The following transform is written as an R function. In the Enclave, each R or Python transform is written as a function that is executed by the system. The print statement on line 14 causes the table to be printed in another dialog for downstream use.

```
1 library(tableone)
2
3
4 table_1_cohort_description <- function(Eligible_pt_table) {
5
6     cto_nonfactor_vars <- c("data_partner_id", "age_at_visit_start_in_years_int", "Race", "
7     Ethnicity", "gender_concept_name", "smoking_status", "blood_type", "InpatientOrED", "Q_
8     Score", "BMI", "has_condition", "inciting_condition", "visit_concept_name", "obs_after_
9     visit", "length_of_stay", "alcoholic_liver_damage", "chronic_hepatitis", "diabetes2",
10    "hepatic_failure", "hepatic_passive_congestion", "hypertension", "ischemic_heart_
11    disease", "lupus", "malignant_neoplasm_lymphoid_hematopoietic_related_tissue", "
12    neoplasm", "nonischemic_heart_disease", "rheumatoid_arthritis_w_factor", "vascular_
13    dementia", "alzheimers_disease", "cerebral_infarction", "chronic_resp", "dementia_
14    associated_with_another_disease", "diabetes1", "hepatic_fibrosis", "hepatic_steatosis"
15    , "hypertensive_kidney_disease", "nicotine_dependence", "nonhypertensive_chronic_
16    kidney_disease", "other_liver_disease", "portal_hypertension", "rheumatoid_arthritis",
17    "unspecified_dementia", "psoriasis", "immunosuppression")
18
19     cto_factor_vars <- c("Race", "Ethnicity", "gender_concept_name", "smoking_status", "
20     blood_type", "InpatientOrED", "visit_concept_name", "inciting_condition")
21
22     print(paste(date(), "cohort description"))
23     tab1 <- CreateTableOne(
24         vars = c(cto_nonfactor_vars, cto_factor_vars),
25         data = Eligible_pt_table,
26         factorVars = cto_factor_vars,
27     )
28     print(tab1, showAllLevels = TRUE)
29     return(NULL)
30 }
```

Listing 8: Creating table 1 with the tableone package.

S1.6 Statistical analysis

The statistical analysis is performed using different transforms for different control groups. Here we show a typical example.

S1.7 Create case propensity-matched control cohorts

In the following code, we create case-control datasets for analysis. In the example we show here, we use “RTI” (non-COVID-19 respiratory tract infection); an analogous analysis was performed for the other cohorts described in the main manuscript.

S1.7.1 RTI_COVID_Matched

Input:

- `Eligible_pt_table` (Section [S1.4.6](#))

For each of the case control groups, a function called `run_matchit` is called that uses the R package `matchit` to perform propensity score matching [6]. For instance, for the RTI group, the following transform is run.

```
RTI_COVID_Matched <- function(Eligible_pt_table) {
  return(run_matchit(Eligible_pt_table, "RTI", matching_ratio=1))
}
```

Listing 9: COVID-19 vs. RTI matched cohort.

A similar analysis was performed for COVID-19 vs. Fracture.

```
Fracture_COVID_Matched <- function(Eligible_pt_table) {
  return(run_matchit(Eligible_pt_table, "Fracture", matching_ratio=1))
}
```

Listing 10: COVID-19 vs. Fracture matched cohort.

A similar analysis was performed for COVID-19 vs. Urolithiasis.

```
Urolithiasis_COVID_Matched <- function(Eligible_pt_table) {
  return(run_matchit(Eligible_pt_table, "Urolithiasis", matching_ratio=2))
}
```

Listing 11: COVID-19 vs. Urolithiasis matched cohort.

We do not show further code related to Fracture or Urolithiasis because the code is completely analogous to the code for RTI.

S1.7.2 `run_matchit`

We run propensity matching by considering the cohort assignment (COVID-19 vs. RTI) as the “treatment” (i.e., `has_condition`) in preparation for the investigation of whether there is a significant difference between the two groups with respect to time to event of a psychiatric diagnosis.

```
run_matchit <- function(data, inciting_condition_name, matching_ratio=1) {
  set.seed(42)
  drop_inc_rows <- TRUE

  data <- data[(data$inciting_condition == "COVID" | data$inciting_condition == inciting_
    _condition_name),]
  factor_cols <- c("Race", "Ethnicity", "gender_concept_name", "smoking_status", "visit_
    concept_name")

  cols_of_interest <- c("data_partner_id", "age_at_visit_start_in_years_int", "Race",
    "Ethnicity", "gender_concept_name", "smoking_status", "Q_Score", "BMI", "has_condition",
    "person_id", "inciting_condition", "visit_concept_name", "obs_after_visit",
    "length_of_stay", "alcoholic_liver_damage", "chronic_hepatitis", "diabetes2",
    "hepatic_failure", "hepatic_passive_congestion", "hypertension",
    "ischemic_heart_disease", "lupus",
    "malignant_neoplasm_lymphoid_hematopoietic_related_tissue", "neoplasm",
    "nonischemic_heart_disease", "rheumatoid_arthritis_w_factor", "cerebral_infarction",
    "chronic_resp", "diabetes1", "hepatic_fibrosis", "hepatic_steatosis",
    "hypertensive_kidney_disease", "nicotine_dependence",
    "nonhypertensive_chronic_kidney_disease", "other_liver_disease",
    "portal_hypertension", "rheumatoid_arthritis", "psoriasis", "immunosuppression")

  matchit_formula <- as.formula("has_condition ~ data_partner_id +
    age_at_visit_start_in_years_int + Race + Ethnicity + gender_concept_name +
    smoking_status + Q_Score + BMI + visit_concept_name + length_of_stay +
    alcoholic_liver_damage+chronic_hepatitis+diabetes2 + hepatic_failure +
```

```

26     hepatic_passive_congestion + hypertension + ischemic_heart_disease + lupus +
28     malignant_neoplasm_lymphoid_hematopoietic_related_tissue + neoplasm +
30     nonischemic_heart_disease + rheumatoid_arthritis_w_factor + cerebral_infarction +
32     chronic_resp + diabetes1 + hepatic_fibrosis + hepatic_steatosis +
34     hypertensive_kidney_disease + nicotine_dependence +
36     nonhypertensive_chronic_kidney_disease + other_liver_disease +
38     portal_hypertension + rheumatoid_arthritis + psoriasis + immunosuppression")
40
42     data$has_condition <- as.integer(data$inciting_condition==inciting_condition_name)
44
46     data <- dplyr::select(data, cols_of_interest)
48
50     for(x in factor_cols){data[[x]] <- as.factor(c(data[[x]])})
52
54     # assess missingness, drop incomplete rows
56     pMiss <- function(x){sum(is.na(x))/length(x)*100}
58     print("Missingness:"); print(apply(data,2,pMiss))
60     print("Missingness pattern:"); print(md.pattern(data))
62     if (drop_inc_rows){data <- data %>% drop_na()}
64
66     matched <- MatchIt::matchit(formula = matchit.formula, ratio=matching_ratio,
68     method = "nearest", data = data, exact=c("data_partner_id"),
70     caliper=c(age_at_visit_start_in_years_int=5), std.caliper=c(FALSE))
72     matched.data <- match.data(matched)
74
76     print(summary(matched))
78
80     m.out <- matched
82     print(love.plot(m.out, binary = "std"))
84
86     return(matched.data)
88 }

```

Listing 12: Run matchit.

S1.7.3 RTI_matched_w_psych_info

Input:

- RTI_COVID_Matched (Section S1.7.1)
- Mental_disorder_info (Section S1.4.7)

This query merges information from previous transforms and adds a category for the type of visit (inpatient, outpatient, other).

```

2 SELECT *, IF(visit_concept_name IN ("Inpatient Visit", "Inpatient Hospital",
4 "Emergency Room and Inpatient Visit", "Inpatient Critical Care Facility"), "Inpatient",
6 IF((visit_concept_name IN ("Outpatient Visit", "Emergency Room Visit", "Office Visit",
8 "Non-hospital institution Visit", "Observation Room", "Laboratory Visit",
10 "Ambulatory Radiology Clinic / Center", "Ambulatory Surgical Center",
12 "Ambulatory Infusion Therapy Clinic / Center",
14 "Ambulatory Community Health Clinic / Center",
    "Ambulatory Clinic / Center", "Pharmacy visit", "Ambulatory Rehabilitation Visit",
    "Ambulatory Dental Clinic / Center", "Ambulatory Oncology Clinic / Center",
    "Ambulatory Magnetic Resonance Imaging (MRI) Clinic / Center",
    "Ambulatory Mammography Clinic / Center", "Ambulatory Endoscopy Clinic / Center",
    "Mass Immunization Center", "Ambulatory Oncological Radiation Clinic / Center",
    "Outpatient Skilled Nursing Facility Visit", "Health examination")),"Outpatient",
    "Other")) as visit_type

```

```

FROM RTI_COVID_Matched
LEFT JOIN Mental_disorder_info USING(person_id)

```

Listing 13: Run matchit.

S1.8 coxRegression

Input:

- RTI_matched_w_psych_info (Section S1.7.3)

Here, we perform cox regression to compare time to event (i.e., of a psychiatric diagnosis) of COVID-19 patients vs. a control group. We use short helper functions to avoid code duplication. The following was used for the comparison between COVID-19 and RTI, and analogous drivers were used for the other comparisons.

```

cox_RTI_all_psych <- function(RTI_matched_w_psych_info) {
  return(cox_regression(RTI_matched_w_psych_info, "Mental Disorder", 0.16, "RTI"))
}

```

Listing 14: Helper function for Cox regression script.

The code first defines the outcome variable by dividing patients into those who receive a diagnosis of `psych_disorder` as compared to those who did not have any diagnosis (for instance, if we are testing “Mood disorder”, patients with “Anxiety” are excluded from the control group). Then the number of days following the initial medical encounter (COVID-19 or control) is calculated. Cox regression over the entire time course is performed first.

We use the `survSplit` function, which takes a survival data set and a set of specified cut times and splits each record into multiple subrecords at each cut time. This allows us to perform Cox regression separately on this first and the second half of the year following the initial medical encounter. Note that we disregard results after the second cut because of a substantially smaller amount of data. Finally, the code displays a Kaplan Meier plot (this is the code that was used to generate plots shown in the manuscript and supplement).

```

cox_regression <- function(data, psych_disorder, y_limit = 0.08,
  control_condition = "control", cut_point = 120) {
  data$mental_illness[is.na(data$mental_illness)] <- 'none'
  data = data[(data$mental_illness == "none") | (data$mental_illness == psych_disorder)
  ,]
  data$result <- !is.na(data$MIN_psych_condition_date_diff)
  data$time <- pmin(data$MIN_psych_condition_date_diff, data$obs_after_visit, na.rm=TRUE)
)

# Limit in days for survival analysis
end_point <- 365

# Create list of dataframes for second time group
data_t2 <- data[data$time > cut_point,]
data_t2$result[data_t2$time > end_point] <- 0
data_t2$time[data_t2$time >= end_point] <- end_point

# Get KM fits for two time groups
km_fit2 <- survfit(Surv(time, result) ~ inciting_condition, data=data_t2)
km_fit1 <- survfit(Surv(time, result) ~ inciting_condition, data=data)
km_fit <- list(T1 = km_fit1, T2 = km_fit2)

# Make temporary plot for risk table for COVID and RTI groups at t1 only

```

```

24 temp_plot = ggsvplot(km_fit1, data=data, title=psych_disorder, xlim=c(21,365),
risk.table = TRUE, axes.offset=FALSE, conf.int=TRUE, fun='cumhaz', xlab="Days",
26 break.x.by=50, font.family = "Arial", fontsize = 5.5,
ggtheme = theme_classic2(base_size=26, base_family = "Arial"), palette = c("lancet")
,
legend.labs =c("COVID", control_condition), legend.title="")
28 my_plot = ggsvplot_combine(km_fit, data=data, title=psych_disorder, xlim=c(21,365),
ylim=c(0,y_limit), risk.table = TRUE, axes.offset=FALSE, conf.int=TRUE, censor.shape
=""),
30 fun='cumhaz', xlab="Days", break.x.by=50,
legend.labs= c('COVID', control_condition, 'COVID (late)', paste0(control_condition,
"(late)")),
32 ggtheme = theme_classic2(base_size=26, base_family = "Arial"), font.family = "Arial"
,
palette = c("lancet"), legend.title="")
34 my_plot$plot = my_plot$plot + scale_x_continuous(breaks = c
(21,50,100,150,200,250,300,350),
expand=c(0,0), limits=c(0,365)) + theme(legend.position=c(0.15,0.8))
36 my_plot$table <- temp_plot$table

38 # Factorize data
factor_cols = c("Race", "Ethnicity", "gender_concept_name", "smoking_status", "
39 inciting_condition", "visit_type")
40 for(x in factor_cols){data[[x]] <- as.factor(data[[x]])}
cox_fit <- coxph(Surv(time, result) ~ BMI + smoking_status + Q_Score + age_at_
41 visit_start_in_years_int + Race + visit_type + Ethnicity + gender_concept_name +
inciting_condition, data=data)

42
43 print(paste(date(), "doing zph"))
44 temp <- cox.zph(cox_fit) # apply the cox.zph function to the desired model
45 print(temp) # display the results
46 print(summary(cox_fit))

48 # image: svg
49 print(my_plot)
50
51 return(data)
52 }

```

Listing 15: Cox Regression.

S1.9 make_table_2_RTI

Input:

- `cox_regression` (Section S1.8)
- Analogous inputs for the other comparisons.

The following code combines all of our input datasets and extracts information that is used to generate table 2.

```

2 make_table_2_RTI <- function(cox_RTI_all_psych, cox_RTI_Mood, cox_RTI_Anxiety,
cox_RTI_Fatigue, cox_RTI_Dyspnea) {
3   sequelae_data <- list(cox_RTI_all_psych, cox_RTI_Mood, cox_RTI_Anxiety,
4     cox_RTI_Fatigue, cox_RTI_Dyspnea)
5   return(make_table_2(sequelae_data, "RTI", cut_point = 120))
6 }

```

Listing 16: Make Table 2 help function.

The `make_table_2` function is as follows.

```
make_table_2 <- function(sequelae_data, comparator, cut_point) {
2   # Set global constants
   end_point <- 365
4   round_dec_places <- 1
   condition <- "COVID-19"
6   hazard_ratio_name <- "Hazard_Ratio"
   ph_pval <- "Proportional_hazard_p_value"
8   hr_pval <- "Hazard_Ratio_P_Value"
   sequelae_names <- c("All psychiatric illness (t1)", "Mood disorder (t1)", "Anxiety (t1)",
   "Fatigue (t1)", "Dyspnea (t1)", "All psychiatric illness (t2)", "Mood disorder (t2)",
   "Anxiety (t2)", "Fatigue (t2)", "Dyspnea (t2)")
10  # Get hazards at cut_point and end_point
   hazard <- lapply(sequelae_data, function(dat){
12     # Run Kaplan-Meier fit
     km_fit <- survfit(Surv(time, result) ~ inciting_condition, data=dat)
14     # Get Hazards
     s1 <- summary(km_fit, times = cut_point)
     s2 <- summary(km_fit, times = end_point)
16     # Format hazards with confidence interval
     percent1=paste0(round(100*(1-s1$surv), round_dec_places), " (", round(100*(1-s1$
18 upper), round_dec_places), "-" , round(100*(1-s1$lower), round_dec_places), ")")
     km_fit2 <- survfit(Surv(time, result) ~ inciting_condition, data=dat)
     percent2=paste0(round(100*(1-s2$surv), round_dec_places), " (", round(100*(1-s2$
20 upper), round_dec_places), "-" , round(100*(1-s2$lower), round_dec_places), ")")
     # Throw data in a sloppy dataframe
     return.df <- data.frame(
22         percent1[1],
         percent1[2],
         percent2[1],
         percent2[2]
24     )
     colnames(return.df) <- c(condition, comparator, "cond_t2", "comp_t2")
     return(return.df)
30  })
   # Make a nice dataframe
   hazard_df_raw <- do.call(rbind, hazard)
   hazard_df <- hazard_df_raw[,c(condition, comparator)]
32   hazard_temp <- hazard_df_raw[,c("cond_t2", "comp_t2")]
   names(hazard_temp) <- names(hazard_df)
34   hazard_df <- rbind(hazard_df, hazard_temp)
   # Create list of dataframes for first time group
36   sequelae_t1 <- lapply(sequelae_data, function(data) {
     data_t1 <- data
     data_t1$result[data_t1$time > cut_point] <- 0
     data_t1$time[data_t1$time > cut_point] <- cut_point
40     return(data_t1)
   })
42   # Create list of dataframes for second time group
   sequelae_t2 <- lapply(sequelae_data, function(data) {
     data_t2 <- data[data$time > cut_point,]
     data_t2$result[data_t2$time > end_point] <- 0
     data_t2$time[data_t2$time > end_point] <- end_point
44     return(data_t2)
   })
46   # Put time groups into single iterable dataframe
   sequelae_tgroups = c(sequelae_t1, sequelae_t2)
48   # Test proportional hazards assumption and do Wald test for each sequela and time
   group
50   models <- lapply(sequelae_tgroups, function(dat){
52     # Test proportional hazards assumption and do Wald test for each sequela and time
   group
54     # Test proportional hazards assumption and do Wald test for each sequela and time
   group
```

```

56   # run cox model and get wald test results
    dat$inciting_condition = factor(dat$inciting_condition, levels = c(comparator, "
COVID"))
    cox_fit <- coxph(Surv(time, result) ~ inciting_condition + BMI + smoking_status
+ Q_Score + age_at_visit_start_in_years_int + Race + visit_type + Ethnicity + gender_
concept_name, data=dat)
58   cox_conf <- summary(cox_fit)$conf.int
    HR <- format(cox_conf["inciting_conditionCOVID", "exp(coef)"], digits=2)
60   low_conf <- format(cox_conf["inciting_conditionCOVID", "lower .95"], digits=2)
    high_conf <- format(cox_conf["inciting_conditionCOVID", "upper .95"], digits=2)
62   hazard_ratio <- paste0(HR, " (", low_conf, "-", high_conf, ")")
    hazard_ratio_p_value <- summary(cox_fit)$coefficients["inciting_conditionCOVID", "
Pr(>|z|)"]
64   # Run proportional hazard assumption test
    cox_temp <- coxph(Surv(time, result) ~ inciting_condition, data=dat)
66   cox_zph <- cox.zph(cox_temp)
    prop_haz_p <- cox_zph$table["inciting_condition", "p"]
68   # Return dataframe blanks for columns that will be filled with hazards
    return_df <- data.frame(
70       "0",
       "0",
72       hazard_ratio,
       hazard_ratio_p_value,
74       prop_haz_p
    )
76   colnames(return_df) <- c(condition, comparator, hazard_ratio_name, hr_pval, ph_
pval)
    return(return_df)
78 }
# Create final output dataframe
80 df <- do.call(rbind, models)
    df[, condition] <- hazard_df[, condition]
82 df[, comparator] <- hazard_df[, comparator]
    row.names(df) <- sequelae_names
84 print(df)
    return(df)
86 }

```

Listing 17: Make Table 2.

References

- [1] Caroline E. Roffman, John Buchanan, and Garry T. Allison. Charlson comorbidities index. *Journal of physiotherapy*, 62:171, July 2016.
- [2] Alexander Kowarik and Matthias Templ. Imputation with the R package VIM. *Journal of Statistical Software*, 74(7):1–16, 2016.
- [3] Richard D. Boyce, Patrick B. Ryan, G. Niklas Norén, Martijn J. Schuemie, Christian Reich, Jon Duke, Nicholas P. Tatonetti, Gianluca Trifirò, Rave Harpaz, J. Marc Overhage, Abraham G. Hartzema, Mark Khayter, Erica A. Voss, Christophe G. Lambert, Vojtech Huser, and Michel Dumontier. Bridging islands of information to establish an integrated knowledge base of drugs and health outcomes of interest. *Drug safety*, 37:557–567, August 2014.
- [4] George Hripcsak, Jon D. Duke, Nigam H. Shah, Christian G. Reich, Vojtech Huser, Martijn J. Schuemie, Marc A. Suchard, Rae Woong Park, Ian Chi Kei Wong, Peter R. Rijnbeek, Johan van der Lei, Nicole Pratt, G. Niklas Norén, Yu-Chuan Li, Paul E. Stang, David Madigan, and Patrick B. Ryan. Observational health data sciences and informatics (ohdsi): Opportunities for observational researchers. *Studies in health technology and informatics*, 216:574–578, 2015.
- [5] Tellen D Bennett, Richard A Moffitt, Janos G Hajagos, Benjamin Amor, Adit Anand, Mark M Bissell, Katie Rebecca Bradwell, Carolyn Bremer, James Brian Byrd, Alina Denham, Peter E DeWitt, Davera Gabriel, Brian T Garibaldi, Andrew T Girvin, Justin Guinney, Elaine L Hill, Stephanie S Hong, Hunter Jimenez, Ramakanth Kavuluru, Kristin Kostka, Harold P Lehmann, Eli Levitt, Sandeep K Mallipattu, Amin Manna, Julie A McMurry, Michele Morris, John Muschelli, Andrew J Neumann, Matvey B Palchuk, Emily R Pfaff, Zhenglong Qian, Nabeel Qureshi, Seth Russell, Heidi Spratt, Anita Walden, Andrew E Williams, Jacob T Wooldridge, Yun Jae Yoo, Xiaohan Tanner Zhang, Richard L Zhu, Christopher P Austin, Joel H Saltz, Ken R Gersing, Melissa A Haendel, and Christopher G Chute. The national covid cohort collaborative: Clinical characterization and early severity prediction. *medRxiv : the preprint server for health sciences*, January 2021.
- [6] D Ho, K Imai, G King, and EA Stuart. Matchit: Nonparametric preprocessing for parametric causal inference. *Journal of Statistical Software*, 42(8):1–28, 2011.