ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

COVID-19 E INQUINAMENTO ATMOSFERICO: RACCOLTA E ANALISI DEI DATI CON METODI E TECNOLOGIE DI DATA SCIENCE

Elaborato in
Programmazione Di Applicazioni Data Intensive

Relatore
Prof. Gianluca Moro

Presentata da
Christian D'Errico

Seconda Sessione di Laurea
Anno Accademico 2019 – 2020

PAROLE CHIAVE

Covid-19

Data Intensive

Biostatistic Analysis

Negative-Binomial Mixed Regression

Python

A chi da lontano e a chi da vicino mi ha aiutato lungo il cammino.

Introduzione

L'emergenza sanitaria dovuta alla pandemia di Covid-19 è un evento che ha sconvolto completamente le nostre vite, con conseguenze terribili, sia da un punto di vista sanitario che economico.

L'improvvisa comparsa della malattia nel mondo e la velocità disarmante della sua diffusione hanno spinto i ricercatori ad effettuare analisi e ricercare informazioni cruciali per rallentare la devastazione di questo nemico silenzioso.

Tra gli aspetti importanti, sui quali gli scienziati di tutto il globo sono stati chiamati a far luce, c'è l'individuazione delle cause e dei fattori concomitanti alla comparsa del nuovo virus, con uno sguardo particolare alle componenti ambientali coinvolte, che potrebbero avere avuto e continuato poi a mantenere un ruolo importante sulla rapidità e la gravità con la quale la malattia ha colpito gli individui.

Diversi studi hanno indagato le possibili relazioni tra l'accelerazione dei contagi e l'aumento dell'inquinamento atmosferico. A sostenere tale tesi sono stati diversi ricercatori, che recentemente hanno dimostrato che frammenti di Rna del Sars-Cov-2 sono nel particolato atmosferico, cioè nel Pm, e questo fungerebbe da veicolo (carrier) e amplificatore (boost). L'idea che l'inquinamento da Pm10 sia un facilitatore delle infezioni non è nuova e la letteratura ha sollevato il problema anche in passato, suggerendo norme importanti per limitare l'inquinamento. Il presupposto con il Coronavirus è lo stesso: il particolato funge da "autostrada" per il trasporto del virus. Anche nell'etere. Forse tanto quanto una stretta di mano.

Altrettanto noto ai più è il fatto che l'esposizione a lungo termine al PM10 influisca negativamente sulle vie respiratorie e sul sistema cardiovascolare, inducendo un aumento generale dei problemi di salute (soprattutto nei soggetti più deboli, come i bambini e gli anziani) e una maggiore incidenza di malattie cardiovascolari, patologie respiratorie e tumori. Da ciò segue immediatamente l'accostamento fra inquinamento atmosferico e la nuova malattia.

Nuova, anche se da inizio pandemia ad oggi, la nostra conoscenza di Covid-19 sta progressivamente evolvendo: i dati provenienti dalla Cina hanno suggerito che la maggior parte dei decessi si è verificata in adulti di età maggiore o uguale a 60 anni e tra persone con patologie pregresse. Situazione analoga ha riguardato i pazienti COVID-19 negli Stati Uniti, dove i più alti tassi di mortalità si riscontrano in persone di 65 o più anni di età. Secondo rapporti ISTAT (Istituto nazionale di statistica), la quota di deceduti, in cui COVID-19 è la causa direttamente responsabile della morte, varia in base all'età, raggiungendo il valore massimo del 92% nella classe 60-69 anni e il minimo (82%) nelle persone di età inferiore ai 50 anni. Il virus, sempre secondo ISTAT, è risultato fatale sia in assenza di concause - circa il 28,2%, percentuale simile nei due sessi e nelle diverse classi di età, con un valore del 18% per persone fra 0-49 anni - che in presenza (Il 71,8%).

Tra le complicanze più frequentemente riportate nei pazienti risultati positivi al test per il Sars-Cov, ci sono le polmoniti, seguite dall'insufficienza respiratoria e altri sintomi e segni respiratori. Meno frequenti lo shock, la sindrome da distress respiratorio acuto (ARDS) ed edema polmonare, le complicanze cardiache, la sepsi e le infezioni non specificate.

Sebbene l'epidemiologia di COVID-19 stia progressivamente cambiando, si è notata una certa congruenza fra le cause di morte per questa nuova patologia e le malattie influenzate da un'esposizione a lungo termine al particolato atmosferico (PM10 e PM2.5).

L'Agenzia internazionale per la ricerca sul cancro (IARC) e l'Organizzazione mondiale della sanità (OMS) hanno da tempo classificato l'inquinamento dell'aria (di cui il particolato atmosferico è un indicatore) nel Gruppo 1, vale a dire tra le sostanze cancerogene per l'uomo: per la sua capacità di penetrare nei polmoni e nel sangue, causando quindi attacchi cardiaci e malattie, e il suo ruolo nella diffusione e nella persistenza dei virus in sospensione nell'atmosfera, l'inquinamento risulta essere una delle principali cause di morte prematura nel mondo.

Sulla base di queste idee e congetture è stato concepito il lavoro di tesi, che viene presentato suddiviso nei seguenti capitoli:

- **Capitolo 1** - Data science: la scienza dei dati;
- **Capitolo 2** - Riproposizione dell'analisi "Exposure to air pollution and COVID-19 mortality in the US";
- **Capitolo 3** - Covid-19 e inquinamento atmosferico in Italia;

Indice

Elenco delle figure

Capitolo 1

Data science: la scienza dei dati

Viviamo in un mondo sommerso dai dati. I siti web tengono traccia di ogni clic di ogni utente. Gli smartphone registrano le posizioni, gli spostamenti e persino la velocità con cui li effettuiamo, ogni secondo di ogni giorno. I "Quantified selfers" indossano pedometri che registrano sempre la frequenza cardiaca, le abitudini di movimento, la dieta e il ritmo del sonno. Le auto intelligenti raccolgono le abitudini di guida, le case intelligenti raccolgono le abitudini di vita e gli esperti di marketing raccolgono le abitudini di acquisto. Internet stesso rappresenta un enorme grafico della conoscenza che contiene (tra le altre cose), un'enorme enciclopedia con riferimenti incrociati; database di film, musica, risultati sportivi, flipper, meme e cocktail; e troppe statistiche governative (alcune quasi vere!) da troppi governi per capirci.

Lo scopo di questo capitolo è presentare la Data Science, la "scienza dei dati" e le sue tecnologie e sulle loro applicazioni con particolare attenzione a quelle utilizzate nel presente lavoro di tesi.

1.0.1 I Big Data

L'insieme di tutte queste informazioni oggi viene ricondotto al concetto di "Big Data".
I Big Data sono dati che contengono una maggiore varietà, che arrivano in volumi crescenti e con velocità sempre più elevate.
Si tratta infatti di set più grandi e complessi, soprattutto provenienti da fonti inedite.
Queste informazioni sono così voluminose che il software di elaborazione dati tradizionale non è in grado di gestirli, ma possono essere utilizzati per risolvere problemi che prima non erano affrontabili.

Tre sono le caratteristiche principali dei Big Data:

- Volume: quando si lavora con i big data lo si fa con elevati volumi di dati rarefatti e non strutturati.
  Possono essere valori sconosciuti, come feed di dati di Twitter, flussi di clic su una pagina Web o un'app mobile o apparecchiature abilitate per sensori.
  Per alcune organizzazioni, potrebbero essere decine di terabyte di dati. Per altri, potrebbero essere centinaia di petabyte.
- Velocità: normalmente, la velocità massima dei flussi di dati si raggiunge direttamente in memoria rispetto alla scrittura su disco. Alcuni prodotti smart "internet-enabled" funzionano in tempo reale o quasi in tempo reale e richiedono valutazioni e azioni in tempo reale.
- Varietà: i tipi di dati tradizionali erano strutturati e si adattavano perfettamente a un database relazionale.
  Con l'aumento dei big data, i dati arrivano in nuovi tipi, non strutturati. I tipi di dati non strutturati e semistrutturati, come testo, audio e video, richiedono un'ulteriore elaborazione preliminare per ricavare significato e supportare i metadati.

Oggi i dati rappresentano, senza ombra di dubbio, il nuovo paradigma economico mondiale da cui partire per qualsiasi tipologia di impresa, che sia commerciale, culturale o scientifica.
Diverse sono le tecnologie e le tecniche di analisi per scoprire patterns nascosti e connessioni tra i dati.
Lo scopo è effettuare un'analisi predittiva, ovvero conoscere anticipatamente cosa accadrà: ciò diventa possibile poiché se abbiamo un modello e abbiamo dati storici a sufficienza possiamo determinare cosa succederà in un futuro prossimo (una tendenza) con basi o fondamenti statistici. Sulla base di queste previsioni è possibile poi intervenire sul futuro mediante un'analisi prescrittiva, ovvero si vanno a cercare le condizioni affinché un certo evento accada.

I big data rappresentano quindi il nuovo strumento che rende "misurabile" la società: spingono verso una nuova scienza dei dati, in grado di misurare e, in prospettiva, prevedere crisi economiche, epidemie, diffusione di opinioni, distribuzione delle risorse economiche, bisogni di mobilità.

1.0.2 Introduzione alla Data Science

La Data Science è proprio il settore interdisciplinare che risponde a queste esigenze: i data scientist combinano le competenze in vari ambiti, tra cui

statistica, informatica ed economia aziendale, per analizzare i dati raccolti dal Web, dagli smartphone, dai clienti, dai sensori e da altre fonti.

La data science mostra i trend e produce insight che qualunque organizzazione o azienda o amministrazione può utilizzare per prendere decisioni più mirate e creare prodotti e servizi più innovativi. I dati costituiscono la base dell'innovazione, ma il loro valore deriva dalle informazioni che i data scientist possono ottenere e in base alle quali agire.

Facebook, ad esempio, ti chiede di elencare la tua città natale e la tua posizione attuale, apparentemente per permettere più facilmente ai tuoi amici di trovarti e connettersi con te. Ma analizza anche questi luoghi per identificare i tracciati delle migrazioni globali e scoprire dove vivono i tifosi delle diverse squadre di calcio.

Nel 2012, la campagna di Obama ha impiegato dozzine di data scientist che hanno estratto i dati e cercato di identificare gli elettori che avevano bisogno di maggiore attenzione, scegliendo appelli e programmi di raccolta fondi specifici e concentrando gli sforzi per ottenere il voto dove ritenevano fosse utile. E nel 2016 la campagna Trump ha testato un'incredibile varietà di annunci online e ha analizzato i dati per scoprire cosa avesse funzionato e cosa no.

1.0.3 Chi è il Data Scientist?

Il Data Scientist è colui che analizza un enorme insieme di dati e fonti differenti per fare scoperte.
Le informazioni analizzate possono essere strutturate o non strutturate e possono occupare quantità pari anche a Terabyte di hard disk, e migliaia di fogli in Excel.
Normalmente, lo scopo dell'analisi è quello di soddisfare esigenze specifiche o raggiungere obiettivi aziendali.

Il lavoro del Data Scientist può suddividersi in due macro stadi:

- Capire e Preparare i dati da analizzare: la prima domanda cui deve rispondere un Data Scientist è da dove ottenere i dati e come renderli fruibili. In questa fase lo scienziato dei dati deve cercare di capire i dati, individuare quelli che gli servono e che gli possono interessare. Utilizzano degli strumenti, i cosiddetti feature selector, che adottano euristiche per eliminare le informazioni che hanno poca probabilità di influire sui risultati finali. Alcuni sondaggi hanno stimato che preparare i dati e assemblarli tra loro occupa il 70-80% del tempo del loro lavoro.

- Il processo di scoperta: i dati che sono scelti per essere analizzati nella precedente fase vengono dati in pasto ad algoritmi sofisticati al fine di cercare modelli predittivi o nuova conoscenza. Successivamente i risultati trovati sono sintetizzati in uno o più report e comunicati a chi di interesse.

1.0.4 Analisi della regressione: predizione di variabili continue

L'analisi della regressione è una tecnica usata per analizzare una serie di dati che consistono in una variabile dipendente e una o più variabili indipendenti. Lo scopo è stimare un'eventuale relazione funzionale esistente tra la variabile dipendente e le variabili indipendenti.

Tipi di regressione

La funzione con la quale si cerca di rappresentare le relazioni fra le variabili in gioco può presentarsi in molteplici varianti.
Per prima cosa, occorre specificare che si parla di regressione monovariata quando l'input del modello è costituito da un'unica varibile indipendente:

$$Y_i = \beta_0 + \beta_1 X_1 + u_1$$

Mentre, quando ci riferiamo ad un modello di regressione multivariata, stiamo indicando un modello che prende in entrata più di una variabile:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = 1, \ldots, N.$$

Detto ciò, è importante sottolineare come esistano diverse "famiglie" di regressione a seconda della funzione messa in campo per la modellazione dei dati:

- Regressione lineare: i dati sono rappresentati da una funzione lineare.
- Regressione non lineare: la funzione è lineare nei parametri, ma il grado delle variabili indipendenti è superiore al primo.
- Modelli lineari generalizzati: a differenza delle prime due categorie che ammettono una distribuzione normale per la variabile output, un modello lineare generalizzato ammette la possibilità che la variabile dipendente possa essere distribuita come una qualsiasi variabile casuale

della famiglia esponenziale e dunque, oltre alla variabile continua normale anche le variabili casuali binomiale, poissoniana, gamma, normale inversa e altre.

Indipendentemente dalle scelte adottate sulla tipologia di modello / funzione però, la variabile dipendente nell'equazione di regressione risulta sempre essere una funzione delle variabili indipendenti più un termine d'errore.
Quest'ultimo è una variabile casuale e rappresenta una variazione non controllabile e imprevedibile nella variabile dipendente.

Lo scopo del buon Data Scientist è proprio quello di minimizzare l'errore delle previsioni del modello: questo rappresenta la soluzione a una sorta di problema di ottimizzazione.

L'approccio più famoso è costituito da una tecnica chiamata "discesa del gradiente" che si presta abbastanza bene a risolvere da zero la questione.

Discesa del gradiente

Una volta fissato un set di dati su cui è calcolato, l'errore è una funzione continua sui parametri del modello di predizione.
Denotando con theta la combinazione di valori dei parametri e con h-theta il modello che li utilizza, si può scrivere:

$$E(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

La funzione dell'errore su n parametri graficamente è costituita da una curva in n+1 dimensioni (nel caso di una regressione monovariata, la curva avrà 3 dimensioni, l'intercetta cioè il valore della variabile Y quando X=0 nonchè la stima del parametro reale b0, e la singola x, più l'errore, ma in generale ne può avere un numero arbitrario).
Testando una combinazione di parametri, l'errore calcolato su di essa fornisce un campione di tale funzione, ovvero un punto sulla curva.

L'obiettivo della regressione è quindi trovare i parametri per cui il valore di tale funzione sia minimo.
Quello che si cerca di fare è pertanto percorrere il gradiente della funzione di errore alla ricerca del punto di minimo: osservare il gradiente ci permette infatti di capire se una funzione sale o scende in un determinato punto.

Data una funzione f a più variabili, le derivate parziali, grazie alle quali siamo in grado di trovare il vettore gradiente, sono le derivate di f su ognuna delle

variabili che la compongono, calcolate trattando le variabili diverse da quella osservata nel calcolo come costanti:

$$f(a,b) = 3a + b^2 \quad \frac{\delta f}{\delta a} = 3 \quad \frac{\delta f}{\delta b} = 2b$$

Questi i passi della "discesa del gradiente":

- Si parte da un punto x0
- Al punto xk si valuta la funzione f ed il suo gradiente.
- Ad xk si sottrae un vettore proporzionale al gradiente per ottenere un punto xk+1 con f(xk+1) < f(xk)
- Si pone k ← k+1 e si esegue l'iterazione successiva dal punto 2, ripetendo fino alla convergenza ad un minimo

Al passo i dell'algoritmo di discesa del gradiente, calcolato il gradiente, il punto successivo xk+1 è calcolato così:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \cdot \nabla f(\mathbf{x}_k)$$

Eta costituisce la lunghezza del passo di discesa (step size), il rapporto tra gradiente e spostamento ad ogni passo.
Eta è un parametro dell'algoritmo di discesa da impostare:

- se troppo basso l'algoritmo impiega molte iterazioni per convergere
- se troppo alto l'algoritmo potrebbe rallentare nel trovare il minimo per via di salti troppo ampi da un punto all'altro della curva
- in alcuni metodi Eta viene fatto variare da un'iterazione all'altra

1.0.5 Valutazione del modello

Per definire il grado di accuratezza o l'efficacia di un qualsiasi modello di machine learning è necessario eseguire una o più valutazioni sugli errori che si ottengono nelle previsioni.
In genere, dopo la fase di training, ma prima della convalida, viene effettuata una stima dell'errore per il modello.
In questo processo, viene eseguita una stima numerica della differenza nelle risposte previste e originali, chiamata anche errore di addestramento (training error).
Tuttavia, questo ci dà solo un'idea di quanto bene il nostro modello faccia sui

dati utilizzati per addestrarlo, in quanto è possibile che il modello sia inadeguato o in eccesso ai dati.
Quindi, il problema con questa tecnica di valutazione è che non fornisce un'indicazione di come il modello di apprendimento si comporterà con un set di dati indipendente / invisibile, ossia su dati che non ha già visto.
Per ovviare a questo si utilizza la cross validation, o convalida incrociata.

Esistono diverse tipologie di cross validation:

- Holdout: il tipo più semplice di convalida dei dati, con un'unica suddivisione in due insiemi: set di training (per l'allenamento del modello) e set di test (per la validazione del modello).

- LOOCV (Leave One Out Cross Validation): in cui il numero delle suddivisioni è pari al numero delle osservazioni che abbiamo nel dataset. Il set di dati viene diviso in due parti. Una parte è costituita da un'unica osservazione, i dati di test, e l'altra parte da tutte le altre osservazioni del set di dati di allenamento (set di training). Se si dispone di un set di dati con n osservazioni, i dati di allenamento contengono n-1 osservazioni e i dati di test contengono un'unica osservazione. Questo processo è ripetuto per ciascun data points n volte e genera n volte la metrica di accuratezza del modello.

- K-fold CV: la convalida più diffusa, che permette di definire k suddivisioni del dataset. Si frammenta casualmente il set di dati in k gruppi (fold) di dimensioni approssimativamente uguali. Il primo fold viene conservato per i test, mentre il modello viene addestrato sui k-1 gruppi rimanenti. Il processo viene ripetuto K volte e ogni volta per la convalida vengono utilizzati fold diversi o un diverso gruppo di punti dati.

- Stratified Cross Validation: in cui in ogni fold o suddivisione la distribuzione dei campioni tra le classi viene mantenuta costante. I dati sono ordinati in modo tale che ogni fold abbia una buona rappresentazione dell'intero set di dati. Obbliga ogni gruppo ad avere almeno m istanze di ogni classe.

- ShuffleSplit: un metodo ibrido tra il metodo holdout e la convalida k-fold. Vengono create divisioni casuali dei dati nel set di test e di allenamento e quindi viene ripetuto il processo di suddivisione e valutazione dell'algoritmo più volte, proprio come nel metodo di convalida incrociata.

1.0.6 Metriche di errore

Le metriche più usate per analizzare l'errore (error analysis regression) di un modello di regressione si basano sui residui ossia sulle differenze tra le previsioni del modello e le risposte corrette (target):

- Mean Squared Error (MSE): la media delle differenze al quadrato tra previsioni (p) e valori reali (y). L'errore è nullo quando l'indicatore MSE=0. E' tanto maggiore quanto più l'indicatore MSE è maggiore di zero.

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - p_i)^2$$

- Mean Absolute Error (MAE). E' la media delle differenze assolute tra previsioni e valori reali. Una metrica altrettanto utilizzata è il MAPE (Mean Absolute Percentage Error), il MAE in percentuale.

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |y_i - p_i|$$

R2 score. E' il coefficiente di determinazione, una proporzione tra variabilità e correttezza dei dati del modello. L'indicatore varia da 0 (pessimo) a 1 (ottimo).

Capitolo 2

Exposure to air pollution and COVID-19 mortality in the US

Uno studio nazionale sull'esposizione a lungo termine all'inquinamento atmosferico e sulla mortalità da COVID-19 negli Stati Uniti

Gli studiosi di Harvard, prestigiosissima università del Massachusetts, negli Stati Uniti, hanno deciso di approfondire la correlazione tra i principali focolai di coronavirus presenti negli Stati Uniti, dove la stima delle vittime complessive potrebbe raggiungere anche le 240mila persone, scoprendo che le zone a più alta mortalità sono le stesse in cui l'inquinamento atmosferico è maggiore.

Lo studio di Harvard è il primo a mettere nero su bianco la questione in modo scientifico: i ricercatori hanno infatti raccolto i dati sui decessi da Covid-19 e l'esposizione media a lungo termine di polveri ultrasottili di circa tremila contee degli Stati Uniti, prendendo in considerazione anche variabili importanti come la disponibilità di posti letto in terapia intensiva in rapporto al numero di abitanti, il numero di tamponi effettuati, le variabili socio-economiche e comportamentali come l'alimentazione, la dipendenza da fumo, il peso corporeo, fino alle condizioni meteorologiche del territorio.

Di seguito viene riportato lo studio nelle sue tre fasi di svolgimento:

Preprocessing: I dati vengono incrociati, associati e preparati alla fase di analisi successiva. Il prodotto finale è un dataframe contenente le informazioni

precedentemente descritte per ogni contea interessata: in particolare, i dati sulla pandemia si collocano temporalmente al 24 Agosto 2020, mentre per "storici" si intendono i dati relativi al 30 Marzo 2020; tutte le altre variabili risultano essere delle osservazioni medie annuali, relative ad anni non superiori al 2016, tranne che per le informazioni sulle misure restrittive e il computo dei giorni trscorsi dallo scoppio della pandemia.

Analsys: Le informazioni prodotte durante la fase di "Preprocessing" vengono proposte ai sedici modelli elaborati in questa sezione. Per ogni modello vengono mostrati i coefficienti trovati, gli intervalli di confidenza al 95% entro i quali questi si trovano e i p-value.

Figure: Sono state riprodotte un paio di mappe che mostrano una fotografia degli Stati Uniti colpiti dal Covid-19 e intossicati dal PM2.5. A seguire, viene tracciato un grafico che presenta i coefficienti restituiti dai modelli e i loro confronti.

NOTA: *L'analisi, originalmente condotta in linguaggio R, è stata per lo più tradotta in Python, tranne che per alcune sezioni di cui non è stata possibile la trascrizione nella nuova versione: queste parti sono state conservate ed inserite utilizzando il framework Rpy2 per garantire l'interoperabilità con il resto delle istruzioni Python.*

2.1 PREPROCESSING

2.1.1 Dati Covid-19

[1]:
```
import pandas as pd
import numpy as np
from rpy2.robjects import pandas2ri
pandas2ri.activate()
```

[2]:
```
%load_ext rpy2.ipython
```

- Viene scelta come data di studio il 24 agosto 2020 (ultimo aggiornamento della analisi): a quella data fanno riferimento i dati "recenti" sulla pandemia negl'USA.
- I dati "storici" sono datati 30 marzo 2020: forniscono una "fotografia" del periodo di inizio della diffusione del Covid-19 negli Stati Uniti

[3]:
```
date_of_study = "08-24-2020"
#Historical data
covid_hist = pd.read_csv('https://raw.githubusercontent.com/
 ↪CSSEGISandData/COVID-19/master/csse_covid_19_data/
 ↪csse_covid_19_daily_reports/03-30-2020.csv')
covid_us_hist = covid_hist.loc[(covid_hist['Country_Region']␣
 ↪== 'US') & (~covid_hist['FIPS'].isna())]
```

- I dati più attuali vengono combinati con quelli più lontani nel tempo per avere un quadro della variazione dei numeri su contagi e decessi in tutto il periodo dell'anno investito dall'emergenza
- La fonte è la John Hopkins University, un'università di ricerca privata a Baltimora, nel Maryland, balzata agl'onori della cronaca come principale "cronista" della evoluzione della pandemia a livello globale

[4]:
```
# Import outcome data from JHU CSSE
url = 'https://raw.githubusercontent.com/CSSEGISandData/
 ↪COVID-19/master/csse_covid_19_data/
 ↪csse_covid_19_daily_reports/' + date_of_study + '.csv'

covid = pd.read_csv(url)
covid_us = covid.loc[covid['Country_Region'] == 'US'].iloc[:
 ↪,range(0,12)]
covid_us = covid_us.append(covid_us_hist.
 ↪loc[(~covid_us_hist['FIPS'].isin(covid_us['FIPS']))
                                    &␣
 ↪(covid_us_hist['Confirmed'] == 0)
                                    & (covid_us_hist['Deaths']␣
 ↪== 0)
                                    & (~covid_us_hist['FIPS'].
 ↪isna())])
covid_us['FIPS'] = [str(val)[:5] for val in covid_us['FIPS']]
```

Questi i dati sul Covid negli Stati Uniti:

[5]:
```
covid_us.head(5)
```

[5]:
```
      FIPS   Admin2 Province_State Country_Region          ␣
 ↪Last_Update  \
631  1001.  Autauga        Alabama             US  2020-08-25␣
 ↪04:28:02
```

```
632  1003.  Baldwin        Alabama             US  2020-08-25␣
↪04:28:02
633  1005.  Barbour        Alabama             US  2020-08-25␣
↪04:28:02
634  1007.     Bibb        Alabama             US  2020-08-25␣
↪04:28:02
635  1009.   Blount        Alabama             US  2020-08-25␣
↪04:28:02

           Lat      Long_  Confirmed  Deaths  Recovered ␣
↪Active  \
631  32.539527 -86.644082       1286      22          0  1264.0
632  30.727750 -87.722071       4083      32          0  4051.0
633  31.868263 -85.387129        633       7          0   626.0
634  32.996421 -87.125115        510       6          0   504.0
635  33.982109 -86.567906        946       6          0   940.0

             Combined_Key
631  Autauga, Alabama, US
632  Baldwin, Alabama, US
633  Barbour, Alabama, US
634     Bibb, Alabama, US
635   Blount, Alabama, US
```

2.1.2 Dati Ambientali: concentrazioni annuali di PM2.5

- Vengono introdotte le informazioni sulle concentrazioni di PM2.5: si tratta di stime, poiché le stazioni di monitoraggio dell'inquinamento atmosferico sono scarsamente distribuite negli Stati Uniti, con un'ampia maggioranza delle contee che non contengono centraline.
- Tali valori sono prodotti da van Donekelaar et al (2019) (1). Sono creati fondendo misure di PM2,5 da tre diverse fonti: rilvetori di terra, modelli di trasporto chimico GEOS-Chem (CTM) e osservazioni satellitari.

[6]:
```
# Import exposure PM2.5 data
county_pm = pd.read_csv("https://raw.githubusercontent.com/
↪wxwx1993/PM_COVID/master/Data/county_pm25.csv")
```

[7]:
```
county_pm.head(5)
```

[7]:
```
      fips  year       pm25
0  36103.0  2000  13.749745
1  36103.0  2001  13.681471
2  36103.0  2002  12.549986
3  36103.0  2003  12.436192
4  36103.0  2004  11.738100
```

2.1.3 Dati Ambientali: dati meteorologici, periodo 2000-2016

Sono stati ricavati i dati sulla temperatura massima giornaliera e umidità relativa su raster con celle di 4km x 4km tramite Gridmet di Google Earth Engine:

- Un raster è una matrice di celle (o pixel) organizzata in righe e colonne (o una griglia) in cui ogni cella contiene un valore che rappresenta le informazioni, come ad esempio la temperatura. Un raster può presentarsi come una fotografia aerea digitale, immagine da satellite, immagine digitale o persino una mappa scansionata.

- Delle temperature e umidità relativa è stata poi calcolata la media per l'estate (giugno-settembre) e l'inverno (dicembre-febbraio) nel periodo 2000-2016.

- La media ha coinvolto i raster di ciascuna contea.

[8]:
```
county_temp = pd.read_csv("https://raw.githubusercontent.com/
↪wxwx1993/PM_COVID/master/Data/temp_seasonal_county.csv")
```

[9]:
```
county_temp.head(5)
```

[9]:
```
   fips  year  summer_tmmx  summer_rmax  winter_tmmx ␣
↪winter_rmax
0  1001  2000   307.273692    95.056772   290.079938    85.
↪639748
1  1001  2001   304.385060    99.043182   286.548225    88.
↪898302
2  1001  2002   305.201921    97.726810   288.728068    86.
↪141829
3  1001  2003   303.722675    98.872965   286.703221    83.
↪966630
4  1001  2004   304.203438    97.278645   286.638994    87.
↪666546
```

2.1.4 Dati socio-economici americani

- Di seguito i dati su nove variabili di censimento a livello di contea:

- percentuale di residenti di età superiore ai 65 anni
- percentuale di residenti ispanici
- percentuale di residenti neri
- reddito medio familiare
- valore medio della casa
- percentuale di residenti in povertà
- percentuale di residenti con un diploma di scuola superiore
- densità di popolazione
- percentuale di residenti che vivono nella casa che possiedono.

Sono state raccolte anche le informazioni su due fattori di rischio per la salute, sempre a livello di contea, dal Behavioral Risk Factor Surveillance System (BRFSS), risalenti all'anno 2011 (l'ultimo anno per il quale sono state organizzate tali variabili a livello di contea).

- Le misure in questione sono:

- indice di massa corporea medio
- tasso di fumatori

[10]:
```
county_census = pd.read_csv("https://raw.githubusercontent.com/
 ↪wxwx1993/PM_COVID/master/Data/census_county_interpolated.
 ↪csv")

url = 'https://www.countyhealthrankings.org/sites/default/
 ↪files/media/document/analytic_data2020.csv'
county_brfss = pd.read_csv(url).drop(0)
county_brfss = county_brfss.loc[:,['5-digit FIPS Code','Adult␣
 ↪smoking raw value', 'Adult obesity raw value']]
county_brfss.rename(columns={'5-digit FIPS Code':'fips',
                             'Adult smoking raw value':
 ↪'smoke',
                             'Adult obesity raw value':
 ↪'obese'}, inplace=True)
```

```
county_brfss['fips'] = [str(val)[:5] for val in␣
 ↪county_brfss['fips']]
```

[11]:
```
#Dati su indicatori brfss - Media fumo e indice di massa␣
 ↪corporea
display(county_brfss.head(5))
#Dati socio economici sulle contee
display(county_census.head(5).drop(columns='Unnamed: 0'))
```

	fips	smoke	obese
1	00000	0.1708001743	0.29
2	01000	0.2092735311	0.355
3	01001	0.1808155718	0.333
4	01003	0.174890326	0.31
5	01005	0.2199998453	0.417

	fips	year	poverty	popdensity	medianhousevalue	pct_blk	\
0	36103.0	2000	0.058031	1875.609065	239247.243803	0.047576	
1	36103.0	2004	0.056872	1883.031226	269377.542545	0.047693	
2	36103.0	2008	0.047349	1942.548304	506122.762671	0.048704	
3	36103.0	2011	0.047716	1945.335549	510830.523810	0.048919	
4	36103.0	2014	0.049384	1956.409276	481809.390476	0.049218	

	medhouseholdincome	pct_owner_occ	hispanic	education	population	\
0	66265.088050	0.808135	0.072071	0.246606	1.450411e+06	
1	68979.083857	0.811235	0.075708	0.234804	1.457034e+06	
2	90873.962264	0.833869	0.105376	0.135350	1.511674e+06	
3	90295.504762	0.835889	0.104793	0.139874	1.494680e+06	
4	92309.723810	0.824198	0.110059	0.120635	1.508614e+06	

	pct_asian	pct_native	pct_white
0	0.018570	0.002206	0.893031
1	0.019281	0.002208	0.892249
2	0.025127	0.002184	0.886210
3	0.024687	0.002195	0.885766
4	0.027491	0.002044	0.880521

2.1.5 Dati sanitari: test Covid-19

- I dati raccolti fino a qui vengono poi integrati con le informazioni statistiche sul numero di test COVID-19 eseguiti fino al 24 agosto 2020.
- Le informazioni provengono dal progetto di monitoraggio COVID (https://covidtracking.com/)

[12]:
```
state_test = pd.read_csv('https://covidtracking.com/api/v1/
 ↪states/daily.csv')
state_test = state_test.loc[state_test['date'] ==␣
 ↪int(date_of_study.split('-')[2] + date_of_study.
 ↪split('-')[0] + date_of_study.split('-')[1])].
 ↪drop(columns=state_test.columns[20])
#Codici stati
statecode = pd.read_csv('https://raw.githubusercontent.com/
 ↪wxwx1993/PM_COVID/master/Data/statecode.csv')
```

[13]:
```
state_test.columns
```

[13]:
```
Index(['date', 'state', 'positive', 'negative', 'pending',
       'hospitalizedCurrently', 'hospitalizedCumulative',␣
 ↪'inIcuCurrently',
       'inIcuCumulative', 'onVentilatorCurrently',␣
 ↪'onVentilatorCumulative',
       'recovered', 'dataQualityGrade', 'lastUpdateEt',␣
 ↪'dateModified',
       'checkTimeEt', 'death', 'hospitalized', 'dateChecked',
       'totalTestsViral', 'negativeTestsViral',␣
 ↪'positiveCasesViral',
       'deathConfirmed', 'deathProbable',␣
 ↪'totalTestEncountersViral',
       'totalTestsPeopleViral', 'totalTestsAntibody',␣
 ↪'positiveTestsAntibody',
       'negativeTestsAntibody', 'totalTestsPeopleAntibody',
       'positiveTestsPeopleAntibody',␣
 ↪'negativeTestsPeopleAntibody',
       'totalTestsPeopleAntigen', 'positiveTestsPeopleAntigen',
       'totalTestsAntigen', 'positiveTestsAntigen', 'fips',␣
 ↪'positiveIncrease',
       'negativeIncrease', 'total', 'totalTestResultsSource',
```

```
       'totalTestResults', 'totalTestResultsIncrease',␣
↪'posNeg',
       'deathIncrease', 'hospitalizedIncrease', 'hash',␣
↪'commercialScore',
       'negativeRegularScore', 'negativeScore',␣
↪'positiveScore', 'score',
       'grade'],
      dtype='object')
```

2.1.6 Dati sanitari: ospedali

- Sono state aggiunte anche informazioni a livello di contea sulle strutture sanitarie
- Le informazioni sono datate 2019 e provengono da Homeland Infrastructure Foundation-Level Data (HIFLD)

[14]:
```
hospitals = pd.read_csv('https://opendata.arcgis.com/datasets/
↪6ac5e325468c4cb9b905f1728d6fbf0f_0.csv?
↪outSR=%7B%22latestWkid%22%3A3857%2C%22wkid%22%3A102100%7D')
hospitals['BEDS'].loc[hospitals['BEDS'] < 0] = 'NaN'
```

[15]:
```
hospitals.head(5)
```

[15]:

	X	Y	FID	ID	NAME	ADDRESS	CITY	STATE	ZIP	ZIP4
0	-1.331889e+07	4.346975e+06	1	5793230	CENTRAL VALLEY GENERAL HOSPITAL					
1	-1.322651e+07	4.049626e+06	2	53391362	LOS ROBLES HOSPITAL & MEDICAL CENTER - EAST CA...					
2	-1.315620e+07	4.031978e+06	3	11190023	EAST LOS ANGELES DOCTORS HOSPITAL					
3	-1.317190e+07	4.041752e+06	4	17090028	SOUTHERN CALIFORNIA HOSPITAL AT HOLLYWOOD					
4	-1.313208e+07	4.037270e+06	5	23691706	KINDRED HOSPITAL BALDWIN PARK					

```
0    1025 NORTH DOUTY STREET          HANFORD    CA  93230 ␣
↪NOT AVAILABLE
1             150 VIA MERIDA  WESTLAKE VILAGE    CA  91362 ␣
↪NOT AVAILABLE
2    4060 WHITTIER BOULEVARD      LOS ANGELES    CA  90023 ␣
↪NOT AVAILABLE
3     6245 DE LONGPRE AVENUE        HOLLYWOOD    CA  90028 ␣
↪NOT AVAILABLE
4  14148 FRANCISQUITO AVENUE     BALDWIN PARK    CA  91706 ␣
↪NOT AVAILABLE

   ...             VAL_DATE                            WEBSITE ␣
↪     STATE_ID  \
0  ...  2014/02/10 00:00:00      http://www.hanfordhealth.com ␣
↪NOT AVAILABLE
1  ...  2014/02/10 00:00:00  http://www.losrobleshospital.com ␣
↪NOT AVAILABLE
2  ...  2014/02/10 00:00:00             http://www.elalax.com ␣
↪NOT AVAILABLE
3  ...  2014/02/10 00:00:00         http://sch-hollywood.com/ ␣
↪NOT AVAILABLE
4  ...  2014/02/10 00:00:00      http://www.khbaldwinpark.com ␣
↪NOT AVAILABLE

                                      ALT_NAME ST_FIPS       ␣
↪OWNER TTL_STAFF  \
0                                NOT AVAILABLE       6 ␣
↪PROPRIETARY      -999
1                                NOT AVAILABLE       6 ␣
↪PROPRIETARY      -999
2                                NOT AVAILABLE       6 ␣
↪PROPRIETARY      -999
3  HOLLYWOOD COMMUNITY HOSPITAL OF HOLLYWOOD       6 ␣
↪PROPRIETARY      -999
4                                NOT AVAILABLE       6 ␣
↪PROPRIETARY      -999

   BEDS         TRAUMA  HELIPAD
0    49  NOT AVAILABLE        N
```

```
1    62  NOT AVAILABLE        N
2   127  NOT AVAILABLE        N
3   100  NOT AVAILABLE        N
4    95  NOT AVAILABLE        N

[5 rows x 34 columns]
```

2.1.7 Dati su mortalità nelle contee

- Fra i potenziali fattori che possano inficiare l'analisi svolta, viene considerata anche la mortalità per contea, divisa in fasce di età.
- In particolare, il periodo esaminato è quello che va fra il 2009 e il 2016
- Oltre al numero di persone censite nel periodo di interesse e il numero totale di decessi annoverati, sono presenti due ulteriori metriche:
  - Il tasso grezzo di mortalità, che corrisponde al numero di decessi segnalati ogni anno solare fratto un fattore selezionato (in questo caso, il fattore selezionato riporta il tasso di mortalità per 100.000 persone.).
  - Il tasso di mortalità aggiustato per età, che consiste di una media ponderata dei tassi di mortalità specifici per età, dove i pesi rappresentano una popolazione standard (in questo caso la popolazione degl'U.S.A al censimento del 2000). Tale metrica viene utilizzata per confrontare il rischio di mortalità relativo tra i gruppi nel tempo. Un tasso aggiustato per età rappresenta il tasso che esisterebbe se i tassi di mortalità specifici per età di un determinato anno fossero prevalenti in una popolazione la cui distribuzione per età fosse la stessa di quella della popolazione standard.

[16]:
```
county_base_mortality = pd.read_csv('https://raw.
 ↪githubusercontent.com/wxwx1993/PM_COVID/master/Data/
 ↪county_base_mortality.txt', sep='\t').reset_index().
 ↪drop(columns='index')
county_old_mortality = pd.read_table('https://raw.
 ↪githubusercontent.com/wxwx1993/PM_COVID/master/Data/
 ↪county_old_mortality.txt').reset_index().
 ↪drop(columns='index')
```

```
county_014_mortality = pd.read_csv("https://raw.
 ↪githubusercontent.com/wxwx1993/PM_COVID/master/Data/
 ↪county_014_mortality.txt", sep = "\t").reset_index().
 ↪drop(columns='index')
county_1544_mortality = pd.read_csv("https://raw.
 ↪githubusercontent.com/wxwx1993/PM_COVID/master/Data/
 ↪county_1544_mortality.txt", sep = "\t").reset_index().
 ↪drop(columns=['Unnamed: 0', 'index'])
county_4564_mortality = pd.read_csv("https://raw.
 ↪githubusercontent.com/wxwx1993/PM_COVID/master/Data/
 ↪county_4564_mortality.txt", sep = "\t").reset_index().
 ↪drop(columns='index')
```

[17]:
```
#Mortalità base (complessiva di tutte le classi di età)
display(county_base_mortality.head(5))
#Mortalità over 65
display(county_old_mortality.head(5))
#Mortalità 0-14
display(county_014_mortality.head(5))
#Mortalità 15-44
display(county_1544_mortality.head(5))
#Mortalità 45-64
display(county_4564_mortality.head(5))
```

```
                County  County Code  Deaths  Population  Crude Rate  \
0  Autauga County, AL          1001    7893      918492       859.3
1  Baldwin County, AL          1003   30292     3102984       976.2
2  Barbour County, AL          1005    5197      499262      1040.9
3     Bibb County, AL          1007    4089      397470      1028.8
4   Blount County, AL          1009    9912      997531       993.7

   Age Adjusted Rate % of Total Deaths
0               958.8              0.0%
1               812.6              0.1%
2               951.7              0.0%
3              1025.5              0.0%
4               929.4              0.0%


                County  County Code  Deaths  Population  Crude Rate
0  Autauga County, AL          1001    5338      109652      4868.1
```

```
1  Baldwin County, AL         1003   22452      526184      4266.9
2  Barbour County, AL         1005    3675       71542      5136.8
3     Bibb County, AL         1007    2685       51438      5219.9
4   Blount County, AL         1009    7076      146136      4842.1
```

```
                County  County Code  Deaths  Population  Crude Rate
0  Autauga County, AL          1001     137      202433        67.7
1  Baldwin County, AL          1003     368      590889        62.3
2  Barbour County, AL          1005      86       93954        91.5
3     Bibb County, AL          1007      79       76137       103.8
4   Blount County, AL          1009     132      203165        65.0
```

```
                County  County Code  Deaths  Population  Crude Rate
0  Autauga County, AL          1001     603      378529       159.3
1  Baldwin County, AL          1003    1761     1138248       154.7
2  Barbour County, AL          1005     349      205608       169.7
3     Bibb County, AL          1007     365      167942       217.3
4   Blount County, AL          1009     681      389634       174.8
```

```
                County  County Code  Deaths  Population  Crude Rate
0  Autauga County, AL          1001    1815      227878       796.5
1  Baldwin County, AL          1003    5710      847663       673.6
2  Barbour County, AL          1005    1087      128158       848.2
3     Bibb County, AL          1007     960      101953       941.6
4   Blount County, AL          1009    2023      258596       782.3
```

[18]:
```
county_old_mortality.rename(columns={'Population':
 ↪'older_Population'}, inplace=True)
county_014_mortality.rename(columns={'Population':
 ↪'014_Population'}, inplace=True)
county_1544_mortality.rename(columns={'Population':
 ↪'1544_Population'}, inplace=True)
county_4564_mortality.rename(columns={'Population':
 ↪'4564_Population'}, inplace=True)
```

[19]:
```
county_base_mortality = pd.merge(county_base_mortality,␣
 ↪county_old_mortality.iloc[:,[1,3]], on = "County Code",␣
 ↪how='left')
```

```
county_base_mortality = pd.merge(county_base_mortality,␣
 ↪county_014_mortality.iloc[:,[1,3]], on = "County Code",␣
 ↪how='left')
county_base_mortality = pd.merge(county_base_mortality,␣
 ↪county_1544_mortality.iloc[:,[1,3]], on = "County Code",␣
 ↪how='left')
county_base_mortality = pd.merge(county_base_mortality,␣
 ↪county_4564_mortality.iloc[:,[1,3]], on = "County Code",␣
 ↪how='left')
```

[20]:
```
#Mortalità finale
county_base_mortality.head(5)
```

[20]:
```
                County  County Code  Deaths  Population  Crude␣
 ↪Rate  \
0  Autauga County, AL         1001    7893      918492       ␣
 ↪859.3
1  Baldwin County, AL         1003   30292     3102984       ␣
 ↪976.2
2  Barbour County, AL         1005    5197      499262       ␣
 ↪1040.9
3     Bibb County, AL         1007    4089      397470       ␣
 ↪1028.8
4   Blount County, AL         1009    9912      997531       ␣
 ↪993.7

   Age Adjusted Rate % of Total Deaths  older_Population ␣
 ↪014_Population  \
0              958.8              0.0%          109652.0       ␣
 ↪ 202433.0
1              812.6              0.1%          526184.0       ␣
 ↪ 590889.0
2              951.7              0.0%           71542.0       ␣
 ↪  93954.0
3             1025.5              0.0%           51438.0       ␣
 ↪  76137.0
4              929.4              0.0%          146136.0       ␣
 ↪ 203165.0
```

```
   1544_Population  4564_Population
0         378529.0         227878.0
1        1138248.0         847663.0
2         205608.0         128158.0
3         167942.0         101953.0
4         389634.0         258596.0
```

[21]:
```
#mantenuto l'ordine originale delle colonne
county = county_base_mortality['County Code']
county_base_mortality.drop(labels=['County Code'],␣
 ↪axis=1,inplace = True)
county_base_mortality.insert(0, 'County Code', county)
```

Sono calcolate le percentuali di individui per classe di età sulla totalità della popolazione:

[22]:
```
county_base_mortality['older_pecent'] =␣
 ↪county_base_mortality['older_Population'] /␣
 ↪county_base_mortality['Population']
county_base_mortality['young_pecent'] =␣
 ↪county_base_mortality['014_Population'] /␣
 ↪county_base_mortality['Population']
county_base_mortality['prime_pecent'] =␣
 ↪county_base_mortality['1544_Population'] /␣
 ↪county_base_mortality['Population']
county_base_mortality['mid_pecent'] =␣
 ↪county_base_mortality['4564_Population'] /␣
 ↪county_base_mortality['Population']
county_base_mortality['older_pecent'].
 ↪loc[county_base_mortality['older_pecent'].isna()] = 0
county_base_mortality['prime_pecent'].
 ↪loc[county_base_mortality['prime_pecent'].isna()] = 0
county_base_mortality['mid_pecent'].
 ↪loc[county_base_mortality['mid_pecent'].isna()] = 0
county_base_mortality['young_pecent'].
 ↪loc[county_base_mortality['young_pecent'].isna()] = 0
```

2.1.8 Dati NCHS sulla popolazione

- La popolazione americana è stata successivamente rappresentata secondo lo schema adottato dal NCHS.
- I sistemi di dati del National Center for Health Statistics (NCHS) sono spesso utilizzati per studiare le associazioni tra il livello di residenza e la salute dell'urbanizzazione e per monitorare la salute dei residenti urbani e rurali. NCHS ha sviluppato uno schema di classificazione urbano-rurale a sei livelli per le contee statunitensi e le entità equivalenti a contea. La categoria più urbana è costituita da contee "centrali" di grandi aree metropolitane e la categoria più rurale è costituita da contee "non core", non metropolitane.
- Lo schema adottato per l'analisi risale al 2013

[23]:
```
NCHSURCodes2013 = pd.read_csv("https://raw.githubusercontent.
↪com/wxwx1993/PM_COVID/master/Data/NCHSURCodes2013.csv")
NCHSURCodes2013['FIPS'] = [str(var)[:5] for var in␣
↪NCHSURCodes2013['FIPS']]
```

[24]:
```
#Dati con suddivisione NCHSUR 2013
NCHSURCodes2013.head(5)
```

[24]:
```
   FIPS State Abr.     County name                 CBSA title␣
↪CBSA 2012 pop  \
0  1001         AL  Autauga County             Montgomery, AL ␣
↪      377149
1  1003         AL  Baldwin County  Daphne-Fairhope-Foley, AL ␣
↪      190790
2  1005         AL  Barbour County                         NaN ␣
↪           .
3  1007         AL     Bibb County      Birmingham-Hoover, AL ␣
↪     1136650
4  1009         AL   Blount County      Birmingham-Hoover, AL ␣
↪     1136650

  County 2012 pop  2013 code  2006 code 1990-based code ␣
↪Unnamed: 9
0           55514          3          3               3       ␣
↪ NaN
```

```
1          190790          4          5          3          ␣
 ↪ NaN
2           27201          6          5          5          ␣
 ↪ NaN
3           22597          2          2          6          ␣
 ↪ NaN
4           57826          2          2          3          ␣
 ↪ NaN
```

[25]:
```
%%R
library("dplyr");
library(stringr);
install.packages("RCurl")
library("RCurl");
library(httr);
```

[26]:
```
%R script <- getURL("https://raw.githubusercontent.com/
 ↪cmu-delphi/delphi-epidata/main/src/client/delphi_epidata.
 ↪R", ssl.verifypeer = FALSE)
%R eval(parse(text = script))
%R -o script
```

2.1.9 Dati sanitari: misure restrittive

- In questo momento dell'anno è stato necessario considerare anche le misure restrittive adottate da ogni stato per far fronte all'emergenza sanitaria
- Questi i dati, contea per contea

[27]:
```
state_policy = pd.read_csv("https://raw.githubusercontent.com/
 ↪wxwx1993/PM_COVID/master/Data/state_policy0410.csv")
state_policy.rename(columns={'Stay at home/ shelter in place':
 ↪'stay_at_home'}, inplace=True)
```

[28]:
```
#Misure restrittive Covid
state_policy.columns
```

[28]:
```
Index(['State', 'State of emergency', 'Date closed K-12␣
 ↪schools',
```

```
       'Closed day cares', 'Date banned visitors to nursing␣
↪homes',
       'stay_at_home', 'Closed non-essential businesses',
       'Religious Gatherings Exempt Without Clear Social␣
↪Distance Mandate*',
       'Alcohol/Liquor Stores Open', 'Keep Firearms Sellers␣
↪Open',
       'Closed restaurants except take out', 'Closed gyms',
       'Closed movie theaters', 'Froze evictions',
       'Order freezing utility shut offs', 'Froze mortgage␣
↪payments',
       'Waived one week waiting period for unemployment␣
↪insurance',
       'Waive work search requirement for unemployment␣
↪insurance',
       'Expand eligibility of unemployment insurance to␣
↪anyonewho is quarantined
and/or taking care of someone who is quarantined',
       'Expand eligibility of unemployment insurance to those␣
↪who have lost
childcare/school closures',
       'Extend the amount of time an individual can be on␣
↪unemployment
insurance',
       'Paid sick leave', 'Population density per square␣
↪miles',
       'Population 2018 ', 'Square Miles', 'Number Homeless␣
↪(2019)',
       'Percent Unemployed (2018)',
       'Percent living under the federal poverty line (2018)',
       'Percent at risk for serious illness due to COVID',
       'All-cause deaths 2016'],
      dtype='object')
```

[29]:

```
# merging data
state_test = state_test.merge(statecode.rename(columns={'Code':
↪'state'}), on ='state')
state_test = state_test.merge(state_policy.iloc[:,[0,5]], on =␣
↪"State")
```

Viene successivamente calcolato il numero di giorni trascorsi dall'inizio dell'applicazione delle misure restrittive anti-contagio, in particolare dall'inizio dei lockdown.

[30]:
```
%R -i date_of_study

%R -i state_test

%R state_test$date_since_social = as.numeric(as.Date(Sys.
↪Date()) - as.Date((strptime(state_test$stay_at_home, "%m/%d/
↪%Y"))))

%R -o state_test
```

[31]:
```
state_test['date_since_social'][state_test['date_since_social'].
↪isna()] = 0
```

2.1.10 Aggregazione dei dati

È stata calcolata la media delle concentrazioni di Pm2.5:

[32]:
```
county_pm_aggregated = county_pm.fillna(0).groupby(by='fips',␣
↪as_index=False).agg({'pm25':'mean'}).rename(columns={'pm25':
↪'mean_pm25'})
county_pm_aggregated = county_pm_aggregated.replace({0:
↪float('NaN')})
```

[33]:
```
#Media del PM a livello di contea
county_pm_aggregated.head(5)
```

[33]:

	fips	mean_pm25
0	NaN	8.840183
1	1001.0	11.712587
2	1003.0	10.077723
3	1005.0	10.981967
4	1007.0	11.998715

E la stessa operazione è stata applicata ai dati metereologici:

[34]:
```
temp_dictio = {'winter_tmmx':'mean_winter_temp',
               'summer_tmmx':'mean_summer_temp',
```

```
                'winter_rmax':'mean_winter_rm',
                'summer_rmax':'mean_summer_rm'}
county_temp_aggregated = county_temp.
 ↪groupby(by='fips',as_index=False).agg({'winter_tmmx':'mean',
                                                        ␣
 ↪'summer_tmmx':'mean',
                                                        ␣
 ↪'winter_rmax':'mean',
                                                        ␣
 ↪'summer_rmax':'mean'}).rename(columns=temp_dictio)
```

[35]:
```
#Temperatura aggregata per contea
county_temp_aggregated.head(5)
```

[35]:
```
   fips  mean_winter_temp  mean_summer_temp  mean_winter_rm ␣
 ↪mean_summer_rm
0  1001        288.085091        306.023451       85.651845 ␣
 ↪   96.055417
1  1003        290.208861        305.516633       89.730972 ␣
 ↪   97.971544
2  1005        289.242107        306.062249       88.633572 ␣
 ↪   97.371675
3  1007        287.362832        305.982177       86.485866 ␣
 ↪   96.293077
4  1009        285.565676        305.178865       85.449139 ␣
 ↪   94.630949
```

[36]:
```
county_pm_aggregated = county_pm_aggregated.
 ↪merge(county_temp_aggregated, on='fips', how='left')
```

Per ogni contea è stato ricavato il numero totale di posti letto disponibili, sommando i valori di tutti gli istituti di cura presenti su suolo USA

[37]:
```
from numpy import nansum
hospitals['BEDS'] = [float(val) for val in hospitals['BEDS']]
county_hospitals_aggregated = hospitals.
 ↪groupby(by='COUNTYFIPS', as_index=False).agg({'BEDS':␣
 ↪nansum}).rename(columns={'BEDS':'beds'})
county_hospitals_aggregated['COUNTYFIPS'] = [str(var)[:5] for␣
 ↪var in county_hospitals_aggregated['COUNTYFIPS']]
```

[38]:
```
#Posti letto totali
county_hospitals_aggregated.head(5)
```

[38]:

	COUNTYFIPS	beds
0	01001	85.0
1	01003	398.0
2	01005	74.0
3	01007	35.0
4	01009	40.0

È stato scelto il 2016 come anno di riferimento per i dati socio economici:

[39]:
```
county_census_aggregated2 =␣
↪county_census[county_census['year'] == 2016]
```

La densità di popolazione è stata divisa in quantili e ne è stata prodotta una variabile categorica

[41]:
```
county_census_aggregated2['q_popdensity'] = 1
quantile_popdensity = np.
↪quantile(county_census_aggregated2['popdensity'], [0.2,0.
↪4,0.6,0.8])
county_census_aggregated2['q_popdensity'].
↪loc[county_census_aggregated2['popdensity'] <=␣
↪quantile_popdensity[0]] = 1
county_census_aggregated2['q_popdensity'].
↪loc[(county_census_aggregated2['popdensity'] >␣
↪quantile_popdensity[0])
                                                &␣
↪(county_census_aggregated2['popdensity'] <=␣
↪quantile_popdensity[1])] = 2
county_census_aggregated2['q_popdensity'].
↪loc[(county_census_aggregated2['popdensity'] >␣
↪quantile_popdensity[1])
                                                &␣
↪(county_census_aggregated2['popdensity'] <=␣
↪quantile_popdensity[2])] = 3
county_census_aggregated2['q_popdensity'].
↪loc[(county_census_aggregated2['popdensity'] >␣
↪quantile_popdensity[2])
```

```
                                                    &␣
 ↪(county_census_aggregated2['popdensity'] <=␣
 ↪quantile_popdensity[3])] = 4
county_census_aggregated2['q_popdensity'].
 ↪loc[county_census_aggregated2['popdensity'] >␣
 ↪quantile_popdensity[3]] = 5
```

[42]:
```
county_census_aggregated2['fips'] = [str(var)[:5] for var in␣
 ↪county_census_aggregated2['fips']]
county_census_aggregated2 = county_census_aggregated2.
 ↪merge(county_brfss, on='fips',how='left')
```

[43]:
```
county_pm_aggregated['fips'] = [str(var)[:5] for var in␣
 ↪county_pm_aggregated['fips']]
aggregate_pm = county_pm_aggregated.merge(covid_us.
 ↪rename(columns={'FIPS':'fips'}))
```

[44]:
```
aggregate_pm.columns
```

[44]:
```
Index(['fips', 'mean_pm25', 'mean_winter_temp',␣
 ↪'mean_summer_temp',
       'mean_winter_rm', 'mean_summer_rm', 'Admin2',␣
 ↪'Province_State',
       'Country_Region', 'Last_Update', 'Lat', 'Long_',␣
 ↪'Confirmed', 'Deaths',
       'Recovered', 'Active', 'Combined_Key'],
      dtype='object')
```

[45]:
```
aggregate_pm_census = aggregate_pm.
 ↪merge(county_census_aggregated2)
```

[46]:
```
#Pm aggregato con census e pm
aggregate_pm_census.columns
```

[46]:
```
Index(['fips', 'mean_pm25', 'mean_winter_temp',␣
 ↪'mean_summer_temp',
       'mean_winter_rm', 'mean_summer_rm', 'Admin2',␣
 ↪'Province_State',
       'Country_Region', 'Last_Update', 'Lat', 'Long_',␣
 ↪'Confirmed', 'Deaths',
```

```
       'Recovered', 'Active', 'Combined_Key', 'Unnamed: 0',␣
↪'year', 'poverty',
       'popdensity', 'medianhousevalue', 'pct_blk',␣
↪'medhouseholdincome',
       'pct_owner_occ', 'hispanic', 'education', 'population',␣
↪'pct_asian',
       'pct_native', 'pct_white', 'q_popdensity', 'smoke',␣
↪'obese'],
      dtype='object')
```

[47]:
```
county_base_mortality['County Code'] = [str(var)[:5] for var␣
↪in county_base_mortality['County Code']]
aggregate_pm_census_cdc = aggregate_pm_census.
↪merge(county_base_mortality.iloc[:,[0, 3, 11, 12, 13, 14]].
↪rename(columns={'County Code':'fips'}), how='left')
aggregate_pm_census_cdc = aggregate_pm_census_cdc.
↪replace({'nan':float('NaN')})
```

[48]:
```
#Aggiunta mortalità alla tabella in formazione
aggregate_pm_census_cdc.columns
```

[48]:
```
Index(['fips', 'mean_pm25', 'mean_winter_temp',␣
↪'mean_summer_temp',
       'mean_winter_rm', 'mean_summer_rm', 'Admin2',␣
↪'Province_State',
       'Country_Region', 'Last_Update', 'Lat', 'Long_',␣
↪'Confirmed', 'Deaths',
       'Recovered', 'Active', 'Combined_Key', 'Unnamed: 0',␣
↪'year', 'poverty',
       'popdensity', 'medianhousevalue', 'pct_blk',␣
↪'medhouseholdincome',
       'pct_owner_occ', 'hispanic', 'education', 'population',␣
↪'pct_asian',
       'pct_native', 'pct_white', 'q_popdensity', 'smoke',␣
↪'obese',
       'Population', 'older_pecent', 'young_pecent',␣
↪'prime_pecent',
       'mid_pecent'],
      dtype='object')
```

[49]:
```
aggregate_pm_census_cdc = aggregate_pm_census_cdc.
 ↪loc[~aggregate_pm_census_cdc['fips'].isna()]
```

[50]:
```
#mantenuto l'ordine originale delle colonne
State = state_test['State']
state_test.drop(labels=['State'], axis=1,inplace = True)
state_test.insert(0, 'State', State)
date = state_test['date']
state_test.drop(labels=['date'], axis=1,inplace = True)
state_test.insert(2, 'date', date)
```

[51]:
```
id = [var for var in state_test.columns if var != 'fips']
aggregate_pm_census_cdc_test = aggregate_pm_census_cdc.
 ↪merge(state_test.loc[:,id].rename(columns={'State':
 ↪'Province_State'}), on='Province_State')
```

[52]:
```
#Aggiunti dati su test per Covid
aggregate_pm_census_cdc_test.columns
```

[52]:
```
Index(['fips', 'mean_pm25', 'mean_winter_temp',␣
 ↪'mean_summer_temp',
       'mean_winter_rm', 'mean_summer_rm', 'Admin2',␣
 ↪'Province_State',
       'Country_Region', 'Last_Update', 'Lat', 'Long_',␣
 ↪'Confirmed', 'Deaths',
       'Recovered', 'Active', 'Combined_Key', 'Unnamed: 0',␣
 ↪'year', 'poverty',
       'popdensity', 'medianhousevalue', 'pct_blk',␣
 ↪'medhouseholdincome',
       'pct_owner_occ', 'hispanic', 'education', 'population',␣
 ↪'pct_asian',
       'pct_native', 'pct_white', 'q_popdensity', 'smoke',␣
 ↪'obese',
       'Population', 'older_pecent', 'young_pecent',␣
 ↪'prime_pecent',
       'mid_pecent', 'state', 'date', 'positive', 'negative',␣
 ↪'pending',
       'hospitalizedCurrently', 'hospitalizedCumulative',␣
 ↪'inIcuCurrently',
```

```
       'inIcuCumulative', 'onVentilatorCurrently',␣
↪'onVentilatorCumulative',
       'recovered', 'dataQualityGrade', 'lastUpdateEt',␣
↪'dateModified',
       'checkTimeEt', 'death', 'hospitalized', 'dateChecked',
       'totalTestsViral', 'negativeTestsViral',␣
↪'positiveCasesViral',
       'deathConfirmed', 'deathProbable',␣
↪'totalTestEncountersViral',
       'totalTestsPeopleViral', 'totalTestsAntibody',␣
↪'positiveTestsAntibody',
       'negativeTestsAntibody', 'totalTestsPeopleAntibody',
       'positiveTestsPeopleAntibody',␣
↪'negativeTestsPeopleAntibody',
       'totalTestsPeopleAntigen', 'positiveTestsPeopleAntigen',
       'totalTestsAntigen', 'positiveTestsAntigen',␣
↪'positiveIncrease',
       'negativeIncrease', 'total', 'totalTestResultsSource',
       'totalTestResults', 'totalTestResultsIncrease',␣
↪'posNeg',
       'deathIncrease', 'hospitalizedIncrease', 'hash',␣
↪'commercialScore',
       'negativeRegularScore', 'negativeScore',␣
↪'positiveScore', 'score',
       'grade', 'Abbrev', 'stay_at_home', 'date_since_social'],
      dtype='object')
```

[53]:
```
aggregate_pm_census_cdc_test_beds =␣
↪aggregate_pm_census_cdc_test.
↪merge(county_hospitals_aggregated.
↪rename(columns={"COUNTYFIPS":'fips'}), on='fips', how =␣
↪'left')
```

[54]:
```
#Aggiunti dati su ospedali
aggregate_pm_census_cdc_test_beds.columns
```

[54]:
```
Index(['fips', 'mean_pm25', 'mean_winter_temp',␣
↪'mean_summer_temp',
       'mean_winter_rm', 'mean_summer_rm', 'Admin2',␣
↪'Province_State',
```

```
      'Country_Region', 'Last_Update', 'Lat', 'Long_',␣
↪'Confirmed', 'Deaths',
      'Recovered', 'Active', 'Combined_Key', 'Unnamed: 0',␣
↪'year', 'poverty',
      'popdensity', 'medianhousevalue', 'pct_blk',␣
↪'medhouseholdincome',
      'pct_owner_occ', 'hispanic', 'education', 'population',␣
↪'pct_asian',
      'pct_native', 'pct_white', 'q_popdensity', 'smoke',␣
↪'obese',
      'Population', 'older_pecent', 'young_pecent',␣
↪'prime_pecent',
      'mid_pecent', 'state', 'date', 'positive', 'negative',␣
↪'pending',
      'hospitalizedCurrently', 'hospitalizedCumulative',␣
↪'inIcuCurrently',
      'inIcuCumulative', 'onVentilatorCurrently',␣
↪'onVentilatorCumulative',
      'recovered', 'dataQualityGrade', 'lastUpdateEt',␣
↪'dateModified',
      'checkTimeEt', 'death', 'hospitalized', 'dateChecked',
      'totalTestsViral', 'negativeTestsViral',␣
↪'positiveCasesViral',
      'deathConfirmed', 'deathProbable',␣
↪'totalTestEncountersViral',
      'totalTestsPeopleViral', 'totalTestsAntibody',␣
↪'positiveTestsAntibody',
      'negativeTestsAntibody', 'totalTestsPeopleAntibody',
      'positiveTestsPeopleAntibody',␣
↪'negativeTestsPeopleAntibody',
      'totalTestsPeopleAntigen', 'positiveTestsPeopleAntigen',
      'totalTestsAntigen', 'positiveTestsAntigen',␣
↪'positiveIncrease',
      'negativeIncrease', 'total', 'totalTestResultsSource',
      'totalTestResults', 'totalTestResultsIncrease',␣
↪'posNeg',
      'deathIncrease', 'hospitalizedIncrease', 'hash',␣
↪'commercialScore',
```

```
       'negativeRegularScore', 'negativeScore',␣
↪'positiveScore', 'score',
       'grade', 'Abbrev', 'stay_at_home', 'date_since_social',␣
↪'beds'],
      dtype='object')
```

[55]:
```
aggregate_pm_census_cdc_test_beds['beds']
[aggregate_pm_census_cdc_test_beds['beds'].isna()] = 0
```

2.1.11 Dati sanitari: giorni dall'inizio della pandemia

- Le operazioni svolte in seguito sono destinate al calcolo del numero di giorni, per ogni contea, trascorso fra la registrazione del primo caso di contagio e il periodo oggetto di studio.
- *N.B*: la data di inizio del calcolo è il 22 marzo 2020, per cui, per le contee che prima di tale data avessero registrato casi di covid confermati, il computo dei giorni è stato comunque fatto partire da tale giornata, per cui il valore risulterà in difetto rispetto al reale conteggio.

[56]:
```
%R date_of_all = format(seq(as.Date("2020-03-22"), as.
↪Date(strptime(date_of_study,"%m-%d-%Y")), by =␣
↪"days"),"%m-%d-%Y")

%R -o date_of_all
```

[57]:
```
covid_us_daily_confirmed = []
for date in date_of_all:
  covid_daily = pd.read_csv('https://raw.githubusercontent.com/
↪CSSEGISandData/COVID-19/master/csse_covid_19_data/
↪csse_covid_19_daily_reports/' + date + ".csv")
  covid_daily = covid_daily.iloc[covid_daily['FIPS'].
↪drop_duplicates().index, range(0,12)]
  covid_us_daily_confirmed.append(covid_daily.
↪loc[(covid_daily['Country_Region'] == 'US') &␣
↪(~covid_daily['FIPS'].isna()) & (covid_daily['Confirmed'] >␣
↪0)])
```

[58]:
```
for i in range(0, len(covid_us_daily_confirmed)):
  covid_us_daily_confirmed[i]['Admin2'] = [str(var) for var in␣
↪covid_us_daily_confirmed[i]['Admin2']]
```

[59]:
```
%R -i covid_us_daily_confirmed

%R covid_us_new_confirmed = list()

%R covid_us_new_confirmed[1] = covid_us_daily_confirmed[1]

%R covid_us_new_confirmed[[1]]$date_since =␣
↪length(covid_us_daily_confirmed)

%R -o covid_us_new_confirmed
```

[60]:
```
covid_us_new_confirmed = [var for var in␣
↪covid_us_new_confirmed]
```

[61]:
```
fips = set()
for i in range(1, len(covid_us_daily_confirmed)):
  for k in range(0, i):
    for f in covid_us_daily_confirmed[k]['FIPS']:
      fips.add(f)
  covid_us_n_c = covid_us_daily_confirmed[i].
↪loc[~covid_us_daily_confirmed[i]['FIPS'].isin(fips)]
  if len(covid_us_n_c) > 0:
    covid_us_n_c['date_since'] = len(covid_us_daily_confirmed)␣
↪- i + 1
    covid_us_new_confirmed.append(covid_us_n_c)
  else:
    covid_us_new_confirmed.append(pd.DataFrame())
```

[62]:
```
%R -i covid_us_new_confirmed

%R covid_us_new_confirmed_df <- do.call("rbind",␣
↪covid_us_new_confirmed)[,c("FIPS","date_since")]

%R -o covid_us_new_confirmed_df
```

[63]:
```
covid_us_new_confirmed_df['FIPS'] = [str(var)[:5] for var in␣
↪covid_us_new_confirmed_df['FIPS']]
```

[64]:
```
```

```
aggregate_pm_census_cdc_test_beds =␣
↪aggregate_pm_census_cdc_test_beds.
↪merge(covid_us_new_confirmed_df.rename(columns={"FIPS":
↪'fips'}), how='left')
```

La variabile "date_since" mostra esattamente il risultato del conteggio dei giorni trascorsi dall'inizio della pandemia nella contea osservata

[65]:
```
aggregate_pm_census_cdc_test_beds['date_since']
[aggregate_pm_census_cdc_test_beds['date_since'].isna()] = 0
```

[66]:
```
aggregate_pm_census_cdc_test_beds =␣
↪aggregate_pm_census_cdc_test_beds.merge(NCHSURCodes2013.
↪iloc[:,[0,6]].rename(columns={'FIPS':'fips'}), on='fips',␣
↪how='left')
```

2.1.12 Preprocessing finale: unificazione quartieri di NY

- I dati dei quarteri di New York City (focolaio principale della pandemia statunitense), che riguardano la popolazione e il numero di posti letto, sono stati genericamente raggruppati sotto l'Admin "New York City."
- I quartieri accorpati sono il Bronx, Kings, Queens e Richmond

[67]:
```
aggregate_pm_census_cdc_test_beds.
↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] == "New␣
↪York City") &␣
↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
↪"New York")]['population'] = \
aggregate_pm_census_cdc_test_beds.
↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] == "New␣
↪York City") &␣
↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
↪"New York")]['population'].values[0] \
+ aggregate_pm_census_cdc_test_beds.
↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
↪"Bronx") &␣
↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
↪"New York")]['population'].values[0] \
```

```
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
 ↪"Kings") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0] \
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
 ↪"Queens") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0] \
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
 ↪"Richmond") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0]
```

[68]:

```
aggregate_pm_census_cdc_test_beds.
 ↪loc[aggregate_pm_census_cdc_test_beds['Admin2'] == "New␣
 ↪York City", 'population'] = \
aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] == "New␣
 ↪York City") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0] \
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
 ↪"Bronx") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0] \
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
 ↪"Kings") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0] \
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
 ↪"Queens") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0] \
```

```
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2'] ==␣
 ↪"Richmond") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State'] ==␣
 ↪"New York")]['population'].values[0]
```

[69]:
```
aggregate_pm_census_cdc_test_beds.
 ↪loc[aggregate_pm_census_cdc_test_beds['Admin2'] == "New␣
 ↪York City", 'beds'] = \
aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2']=="New York␣
 ↪City") &␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State']=="New␣
 ↪York")]['beds']
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2']=="Bronx")␣
 ↪&␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State']=="New␣
 ↪York")]['beds']
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2']=="Kings")␣
 ↪&␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State']=="New␣
 ↪York")]['beds']
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2']=="Queens")␣
 ↪&␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State']=="New␣
 ↪York")]['beds']
+ aggregate_pm_census_cdc_test_beds.
 ↪loc[(aggregate_pm_census_cdc_test_beds['Admin2']=="Richmond")␣
 ↪&␣
 ↪(aggregate_pm_census_cdc_test_beds['Province_State']=="New␣
 ↪York")]['beds']
```

[69]:
```
1836    1444.0
Name: beds, dtype: float64
```

[70]:
```
aggregate_pm_census_cdc_test_beds['smoke'] = [float(var) for␣
 ↪var in aggregate_pm_census_cdc_test_beds['smoke']]
```

```
aggregate_pm_census_cdc_test_beds['obese'] = [float(var) for␣
 ↪var in aggregate_pm_census_cdc_test_beds['obese']]
```

I quartieri diversi da "New York City" vengono poi eliminati dal dataset

[71]:
```
%R -i aggregate_pm_census_cdc_test_beds

%R vars = c("mean_pm25","poverty",
"medianhousevalue","medhouseholdincome",
"pct_owner_occ","education","pct_blk","hispanic",
"older_pecent","prime_pecent","mid_pecent","obese","smoke",␣
 ↪"mean_summer_temp","mean_summer_rm",
"mean_winter_temp","mean_winter_rm")

%R aggregate_pm_census_cdc_test_beds[
aggregate_pm_census_cdc_test_beds$Admin2=="New York␣
 ↪City",][,vars] = \
  sapply(vars,function(var){ \
    (subset(aggregate_pm_census_cdc_test_beds,Admin2=="New␣
 ↪York City"& Province_State=="New York")[
   ␣
 ↪,var]*subset(aggregate_pm_census_cdc_test_beds,Admin2=="New␣
 ↪York City"& Province_State=="New York")$population + \
        subset(aggregate_pm_census_cdc_test_beds,␣
 ↪Admin2=="Bronx"& Province_State=="New York")[,
        var]*subset(aggregate_pm_census_cdc_test_beds,
        Admin2=="Bronx"& Province_State=="New York")$population␣
 ↪+ \
        subset(aggregate_pm_census_cdc_test_beds
        , Admin2=="Kings"& Province_State=="New␣
 ↪York")[,var]*subset(aggregate_pm_census_cdc_test_beds,
        Admin2=="Kings"& Province_State=="New York")$population␣
 ↪+ \
        subset(aggregate_pm_census_cdc_test_beds,␣
 ↪Admin2=="Queens"& Province_State=="New␣
 ↪York")[,var]*subset(aggregate_pm_census_cdc_test_beds,
        Admin2=="Queens"& Province_State=="New␣
 ↪York")$population + \
```

```
      subset(aggregate_pm_census_cdc_test_beds,␣
↪Admin2=="Richmond"& Province_State=="New␣
↪York")[,var]*subset(aggregate_pm_census_cdc_test_beds,
      Admin2=="Richmond"& Province_State=="New␣
↪York")$population)/( \
        subset(aggregate_pm_census_cdc_test_beds,Admin2=="New␣
↪York City"& Province_State=="New␣
↪York")$population+subset(aggregate_pm_census_cdc_test_beds,
        Admin2=="Bronx"& Province_State=="New␣
↪York")$population+ \
          subset(aggregate_pm_census_cdc_test_beds,␣
↪Admin2=="Kings"& Province_State=="New York")$population+␣
↪subset(aggregate_pm_census_cdc_test_beds,
          Admin2=="Queens"& Province_State=="New␣
↪York")$population + \
          subset(aggregate_pm_census_cdc_test_beds,
          Admin2=="Richmond"& Province_State=="New␣
↪York")$population) \
 })

%R aggregate_pm_census_cdc_test_beds =␣
↪subset(aggregate_pm_census_cdc_test_beds, \
                                          !(Admin2=="Bronx"&␣
↪Province_State=="New York")& \
                                            !
↪(Admin2=="Kings"& Province_State=="New York")& \
                                            !
↪(Admin2=="Queens"& Province_State=="New York")& \
                                            !
↪(Admin2=="Richmond"& Province_State=="New York"))

%R -o aggregate_pm_census_cdc_test_beds
```

2.2 ANALYSIS

[73]:
```
#ANALSYS PART
```

[74]:
```
%%R
library("dplyr")
library("MASS")
install.packages('lme4')
library("lme4")
install.packages('glmmTMB')
library("glmmTMB")
install.packages("gamm4")
library("gamm4")
```

2.2.1 Analisi principale

Viene addestrato un modello misto binomiale negativo utilizzando, a livello di contea, le morti per COVID-19 come risultato e la media a lungo termine del PM 2.5 come esposizione.

Generalized Models

Un modello generalizzato misto binomiale negativo appartiene all'insieme dei "Generalized Mixed Models": un modello lineare generalizzato (GLM) è una generalizzazione flessibile della regressione lineare ordinaria che permette alle variabili di risposta di conformarsi a modelli di distribuzione dell'errore diversi da una distribuzione normale. Il GLM generalizza la regressione lineare consentendo al modello lineare di essere correlato alla variabile di risposta tramite una "link function" e consentendo alla grandezza della varianza di ciascuna misurazione di essere una funzione del suo valore predetto.

Ad esempio, nei casi in cui si prevede che la variabile di risposta sia sempre positiva e che vari in un ampio intervallo, i cambiamenti costanti dell'input portano a variazioni di output che variano geometricamente (cioè in modo esponenziale), piuttosto che costantemente.

Supponiamo che un modello di previsione lineare apprenda da alcuni dati (forse tratti principalmente da grandi spiagge) che una diminuzione della temperatura di 10 gradi porterebbe a 1.000 persone in meno a visitare una spiaggia. È improbabile che questo modello si generalizzi bene su spiagge di dimensioni diverse. Più specificamente, il problema è che se si utilizza il modello per prevedere la nuova presenza con un calo di temperatura di 10 gradi per una spiaggia che riceve regolarmente 50 bagnanti, si prevede un valore impossibile di -950. Logicamente, un modello più realistico prevederebbe invece un tasso costante di maggiore frequentazione della spiaggia (ad esempio, un aumento di 10 gradi porta a un raddoppio della

presenza in spiaggia e un calo di 10 gradi porta a un dimezzamento delle presenze). Questo è esattamente quello che fa un modello generalizzato.

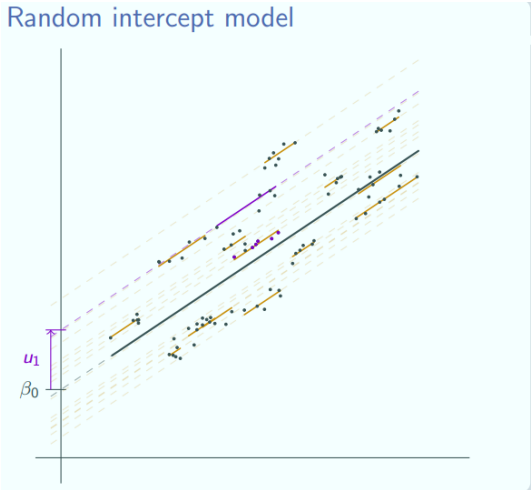
Generalized Mixed Models

Inoltre, quello utilizzato è un modello ad effetti misti, composto da una parte "fissa" e una parte "casuale" ("random").

$y_{ij} = b_0 + b_1 x_{ij}$ (parte "fissa") $+ u_j + e_{ij}$ (parte "random")

- La parte fissa stima i coefficienti b_0, b_1, b_2
- La parte casuale le varianze ${s_u}^2$, ${s_e}^2$

Il termine u_j (l'errore statistico legato ai gruppi, che nel nostro caso si riferisce agli Stati Americani) introduce un livello superiore rispetto alle singole contee: l'idea è che contee di uno stesso stato (raggruppate in cluster) presentano caratteristiche socio-economiche e ambientali uguali.



Per un modello a livello singolo, l'intercetta è costituita dal solo b_0, il parametro che proviene dalla parte "fissa".

Per un modello a più livelli, l'intercetta è ancora b_0 e a questa se ne aggiunge una per ogni linea di gruppo, data da $b_0 + u_j$

I modelli multilivello ci permettono di esplorare variabili a livello di "gruppo" e di "individuo" contemporaneamente.

Dall'esistenza di una gerarchia nei dati (nested data) può derivare una dipendenza fra osservazioni indivuali e spesso tale struttura caratterizza il contesto in cui avviene la rilevazione (dati raggruppati, quali ad esempio gli

studenti appartenenti a diverse classi) oppure l'operazione di rilevazione (dati longitudinali, quali ad esempio le misure effettuate da diversi soggetti sullo stesso paziente). In entrambi i casi è opportuno considerare il legame esistente tra le singole osservazioni all'interno di ciascun gruppo di unità statistiche.

Negative Binomial

La famiglia di riferimento per il modello è, come accennato sopra, "Negativo Binomiale": si tratta di una generalizzazione della regressione di Poisson, della quale ne rilassa l'ipotesi che vuole l'uguaglianza fra la media e la varianza della variabile modellata.

Nella nostra situazione, poiché le contee hanno registrato numeri di decessi COVID-19 molto differenti fra loro, la media dei nostri dati sui risultati è piccola, mentre la varianza è grande a causa della presenza di diversi epicentri di focolai. Ciò rende la distribuzione di Poisson inappropriata.

Fattori confondenti

Nell'analisi principale, 19 sono i potenziali fattori di confusione:

- dimensione della popolazione, distribuzione per età, densità della popolazione, tempo dall'inizio dell'epidemia, tempo trascorso dall'inizio dell'applicazione delle misure restrittive, letti ospedalieri, condizioni meteorologiche e variabili socioeconomiche e comportamentali come l'obesità e il fumo.

Il confondimento è una situazione in cui un fattore (o una combinazione di fattori), diverso da quello in studio, è responsabile, almeno in parte, dell'associazione che abbiamo osservato. Quando è presente un fattore di confondimento, i dati grezzi mostrano un quadro sbagliato della correlazione tra causa ed effetto.

Offset

La popolazione americana viene trattata come un offset (offset(log(population))):

- Un offset è una variabile per cui il modello non calcola, durante l'addestramento, un coefficiente, che invece, per default, viene settato ad 1
- L'offset permette che le Y del modello non siano costituite semplicemente dal numero di decessi per Covid, ma dal tasso di

mortalità: con un paio di operazioni e sfruttando le proprietà dei logaritmi, risulta elementare trasformare l'output nel rapporto *decessi/popolazione* che costituisce il tasso di mortalità da Covid-19.

Modello

Questo il modello:

log(E[COVID-19 deaths]) = b_{c0} + b_{c1} PM2.5 + b_{c2} population density + b_{c4} percent living in poverty + b_{c5} median household income + b_{c6} percent black + b_{c7} percent hispanic + b_{c8} percent of the adult population with less than a high school education + b_{c9} median house value + $b_{c1}0$ percent of owner-occupied housing + b_{c11} average BMI + b_{c12} smoking rate + b_{c13} number of hospital beds + b_{c14} average summer temperature + b_{c15} average summer relative humidity + b_{c16} average winter temperature + b_{c17} average winter relative humidity + b_{c18} number of COVID-19 tests performed in state + b_{c19} date since the beginning of the pandemy + log(population size) + random intercept(State)

Risulta evidente come spostando il termine $log(population\_size)$ a sinistra si ottenga il tasso di mortalità

Si procede quindi con la prima analisi:

[75]:
```
%R mode.nb.random.off.main = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                    + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                    +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                    + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                    + scale(date_since_social)␣
↪+ scale(date_since) \
                                    + scale(beds/population) \
                                    + scale(obese) +␣
↪scale(smoke) \
                                    + scale(mean_summer_temp)␣
↪+ scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
```

```
                                        + (1|state) \
                                        + offset(log(population)),␣
↪data = aggregate_pm_census_cdc_test_beds)

%R -o mode.nb.random.off.main
```

[76]:
```
%R print(summary(mode.nb.random.off.main))
```

```
Generalized linear mixed model fit by maximum likelihood␣
↪(Laplace
 Approximation) [glmerMod]
 Family: Negative Binomial(1.5984)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
↪+
    scale(mid_pecent) + scale(date_since_social) +␣
↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) +
    (1 | state) + offset(log(population))
   Data: aggregate_pm_census_cdc_test_beds

     AIC      BIC   logLik deviance df.resid
 16889.7  17050.0  -8417.8  16835.7     2775

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2190 -0.6924 -0.3670  0.2779 17.1443

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5617   0.7495
Number of obs: 2802, groups:  state, 43
```

```
Fixed effects:
                                  Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                     -9.3861006  0.2095048 -44.801  <␣
 ↪2e-16 ***
mean_pm25                        0.1054663  0.0214422   4.919 8.
 ↪72e-07 ***
factor(q_popdensity)2           -0.0205156  0.0887846  -0.231 0.
 ↪817259
factor(q_popdensity)3            0.0006164  0.0926048   0.007 0.
 ↪994689
factor(q_popdensity)4           -0.0673699  0.0975005  -0.691 0.
 ↪489584
factor(q_popdensity)5            0.0817148  0.1133004   0.721 0.
 ↪470773
scale(poverty)                   0.0492793  0.0266351   1.850 0.
 ↪064290 .
scale(log(medianhousevalue))     0.1326918  0.0456861   2.904 0.
 ↪003679 **
scale(log(medhouseholdincome))   0.1217868  0.0466606   2.610 0.
 ↪009053 **
scale(pct_owner_occ)             0.0623875  0.0272968   2.286 0.
 ↪022282 *
scale(education)                 0.1498355  0.0326829   4.585 4.
 ↪55e-06 ***
scale(pct_blk)                   0.2154465  0.0304280   7.081 1.
 ↪44e-12 ***
scale(hispanic)                  0.1651193  0.0318755   5.180 2.
 ↪22e-07 ***
scale(older_pecent)              0.0458021  0.0416521   1.100 0.
 ↪271490
scale(prime_pecent)             -0.1861738  0.0505230  -3.685 0.
 ↪000229 ***
scale(mid_pecent)               -0.1399647  0.0375018  -3.732 0.
 ↪000190 ***
scale(date_since_social)         0.1573080  0.1337179   1.176 0.
 ↪239428
scale(date_since)                0.2646083  0.0416473   6.354 2.
 ↪10e-10 ***
```

```
scale(beds/population)           0.0177911  0.0250448   0.710 0.
↪477473
scale(obese)                     0.0282638  0.0241481   1.170 0.
↪241826
scale(smoke)                     0.2025125  0.0425663   4.758 1.
↪96e-06 ***
scale(mean_summer_temp)          0.1465511  0.0738753   1.984 0.
↪047282 *
scale(mean_winter_temp)          0.1323360  0.0929479   1.424 0.
↪154514
scale(mean_summer_rm)           -0.1588760  0.0746022  -2.130 0.
↪033201 *
scale(mean_winter_rm)           -0.0030305  0.0406289  -0.075 0.
↪940541
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[77]:
```
%R exp(summary(mode.nb.random.off.main)[10]$coefficients[2,1])
```

[77]:
```
array([1.11122866])
```

Estremo inferiore intervallo di confidenza 95%:

[78]:
```
%R exp(summary(mode.nb.random.off.main)[10]$coefficients[2,1]␣
↪- 1.96*summary(mode.nb.random.off.
↪main)[10]$coefficients[2,2])
```

[78]:
```
array([1.06549511])
```

[79]:
```
%R summary(mode.nb.random.off.main)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.off.main)[10]$coefficients[2,2]
```

[79]:
```
array([0.06343958])
```

Estremo superiore intervallo di confidenza 95%:

[80]:
```
%R exp(summary(mode.nb.random.off.main)[10]$coefficients[2,1]␣
↪+ 1.96*summary(mode.nb.random.off.
↪main)[10]$coefficients[2,2])
```

[80]:
```
array([1.15892521])
```

[81]:
```
%R summary(mode.nb.random.off.main)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.off.main)[10]$coefficients[2,2]
```

[81]:
```
array([0.14749303])
```

p-value per PM2.5:

[82]:
```
%R summary(mode.nb.random.off.main)[10]$coefficients[2,4]
```

[82]:
```
array([8.71514986e-07])
```

2.2.2 Seconda analisi: - beds

In questa seconda analisi vengono eliminati dalle confondenti i dati sul numero dei posti letto per contea rapportati alla popolazione residente

[83]:
```
%R mode.nb.random.off.beds = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                    + scale(poverty) +␣
↪scale(log(medianhousevalue)) \
                                    +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                    + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                    + scale(date_since_social)␣
↪+ scale(date_since) \
                                    + scale(obese) +␣
↪scale(smoke) \
                                    + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                    + (1|state) \
                                    + offset(log(population)),␣
↪data = aggregate_pm_census_cdc_test_beds)

%R -o mode.nb.random.off.beds
```

[84]:
```
%R print(summary(mode.nb.random.off.beds))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.5982)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(obese) + scale(smoke) + scale(mean_summer_temp) +
scale(mean_winter_temp) +
    scale(mean_summer_rm) + scale(mean_winter_rm) + (1 | state)␣
 ↪+
    offset(log(population))
   Data: aggregate_pm_census_cdc_test_beds

     AIC      BIC   logLik deviance df.resid
 16888.2  17042.6  -8418.1  16836.2     2776

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2189 -0.6930 -0.3674  0.2769 17.2692

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5618   0.7496
Number of obs: 2802, groups:  state, 43

Fixed effects:
                                 Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                    -9.3849960  0.2095085 -44.795  <␣
 ↪2e-16 ***
```

```
mean_pm25                          0.1053027  0.0214406   4.911 9.
 ↪04e-07 ***
factor(q_popdensity)2             -0.0229843  0.0887379  -0.259 0.
 ↪795625
factor(q_popdensity)3              0.0006724  0.0926184   0.007 0.
 ↪994208
factor(q_popdensity)4             -0.0648193  0.0974610  -0.665 0.
 ↪506000
factor(q_popdensity)5              0.0876068  0.1130290   0.775 0.
 ↪438291
scale(poverty)                     0.0492809  0.0266451   1.850 0.
 ↪064381 .
scale(log(medianhousevalue))       0.1305162  0.0455816   2.863 0.
 ↪004192 **
scale(log(medhouseholdincome))     0.1223407  0.0466607   2.622 0.
 ↪008744 **
scale(pct_owner_occ)               0.0602408  0.0271360   2.220 0.
 ↪026422 *
scale(education)                   0.1490269  0.0326665   4.562 5.
 ↪07e-06 ***
scale(pct_blk)                     0.2167799  0.0303736   7.137 9.
 ↪53e-13 ***
scale(hispanic)                    0.1643370  0.0318549   5.159 2.
 ↪48e-07 ***
scale(older_pecent)                0.0489160  0.0414410   1.180 0.
 ↪237850
scale(prime_pecent)               -0.1840716  0.0504507  -3.649 0.
 ↪000264 ***
scale(mid_pecent)                 -0.1402593  0.0374921  -3.741 0.
 ↪000183 ***
scale(date_since_social)           0.1563459  0.1337285   1.169 0.
 ↪242352
scale(date_since)                  0.2649378  0.0416757   6.357 2.
 ↪06e-10 ***
scale(obese)                       0.0279661  0.0241515   1.158 0.
 ↪246886
scale(smoke)                       0.2029917  0.0425623   4.769 1.
 ↪85e-06 ***
scale(mean_summer_temp)            0.1471086  0.0739104   1.990 0.
 ↪046551 *
```

```
scale(mean_winter_temp)          0.1319387  0.0929876   1.419 0.
↪155933
scale(mean_summer_rm)           -0.1601293  0.0746322  -2.146 0.
↪031907 *
scale(mean_winter_rm)           -0.0028565  0.0406291  -0.070 0.
↪943950
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[85]:
```
%R exp(summary(mode.nb.random.off.beds)[10]$coefficients[2,1])
```

[85]:
```
array([1.11104688])
```

Estremo inferiore intervallo di confidenza 95%:

[86]:
```
%R exp(summary(mode.nb.random.off.beds)[10]$coefficients[2,1]␣
↪- 1.96*summary(mode.nb.random.off.
↪beds)[10]$coefficients[2,2])
```

[86]:
```
array([1.06532409])
```

[87]:
```
%R summary(mode.nb.random.off.beds)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.off.beds)[10]$coefficients[2,2]
```

[87]:
```
array([0.06327906])
```

Estremo superiore intervallo di confidenza 95%:

[88]:
```
%R exp(summary(mode.nb.random.off.beds)[10]$coefficients[2,1]␣
↪+ 1.96*summary(mode.nb.random.off.
↪beds)[10]$coefficients[2,2])
```

[88]:
```
array([1.15873205])
```

[89]:
```
%R summary(mode.nb.random.off.beds)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.off.beds)[10]$coefficients[2,2]
```

[89]:
```
array([0.14732635])
```

p-value per PM2.5:

[90]:
```
%R summary(mode.nb.random.off.beds)[10]$coefficients[2,4]
```

[90]:
```
array([9.04462303e-07])
```

2.2.3 Terza analisi: - BRFSS, - Smoke

Esclusi i dati sui fattori di rischio per la salute dei cittadini americani (fumo e indice di massa corporea BRFSS)

[91]:
```
%R mode.nb.random.off.brfss = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                    + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                    +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                    + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                    + scale(date_since_social)␣
↪+ scale(date_since) \
                                    + scale(beds/population) \
                                    + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                    + (1|state) \
                                    + offset(log(population)),␣
↪data = aggregate_pm_census_cdc_test_beds)

%R -o mode.nb.random.off.brfss
```

[92]:
```
%R print(summary(mode.nb.random.off.brfss))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.5769)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
```

```
    scale(log(medianhousevalue)) +␣
↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
↪+
    scale(mid_pecent) + scale(date_since_social) +␣
↪scale(date_since) +
    scale(beds/population) + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) +
    scale(mean_summer_rm) + scale(mean_winter_rm) + (1 | state)␣
↪+
    offset(log(population))
  Data: aggregate_pm_census_cdc_test_beds

     AIC      BIC   logLik deviance df.resid
 16913.0  17061.4  -8431.5  16863.0     2777

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2136 -0.6928 -0.3650  0.2830 15.5919

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5274   0.7262
Number of obs: 2802, groups:  state, 43

Fixed effects:
                             Estimate Std. Error z value␣
↪Pr(>|z|)
(Intercept)                -9.341017   0.207429 -45.032  <␣
↪2e-16 ***
mean_pm25                   0.101915   0.021495   4.741 2.
↪12e-06 ***
factor(q_popdensity)2      -0.017728   0.088634  -0.200   0.
↪8415
factor(q_popdensity)3       0.006066   0.092342   0.066   0.
↪9476
factor(q_popdensity)4      -0.058037   0.097282  -0.597   0.
↪5508
```

```
factor(q_popdensity)5           0.067075   0.113330   0.592   0.
 ↪5539
scale(poverty)                  0.059233   0.026731   2.216   0.
 ↪0267 *
scale(log(medianhousevalue))    0.084657   0.044379   1.908   0.
 ↪0564 .
scale(log(medhouseholdincome))  0.034770   0.043590   0.798   0.
 ↪4251
scale(pct_owner_occ)            0.047520   0.027198   1.747   0.
 ↪0806 .
scale(education)                0.162457   0.032751   4.960 7.
 ↪04e-07 ***
scale(pct_blk)                  0.225174   0.030384   7.411 1.
 ↪25e-13 ***
scale(hispanic)                 0.127434   0.031212   4.083 4.
 ↪45e-05 ***
scale(older_pecent)            -0.018482   0.039624  -0.466   0.
 ↪6409
scale(prime_pecent)            -0.213869   0.048896  -4.374 1.
 ↪22e-05 ***
scale(mid_pecent)              -0.151667   0.037033  -4.095 4.
 ↪21e-05 ***
scale(date_since_social)        0.168689   0.129948   1.298   0.
 ↪1942
scale(date_since)               0.272666   0.041385   6.589 4.
 ↪44e-11 ***
scale(beds/population)          0.019035   0.025080   0.759   0.
 ↪4479
scale(mean_summer_temp)         0.155878   0.073896   2.109   0.
 ↪0349 *
scale(mean_winter_temp)         0.130907   0.092908   1.409   0.
 ↪1588
scale(mean_summer_rm)          -0.116682   0.073986  -1.577   0.
 ↪1148
scale(mean_winter_rm)          -0.013305   0.040281  -0.330   0.
 ↪7412
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[93]:
```
%R exp(summary(mode.nb.random.off.brfss)[10]$coefficients[2,1])
```

[93]:
```
array([1.10728959])
```

Estremo inferiore intervallo di confidenza 95%:

[94]:
```
%R exp(summary(mode.nb.random.off.brfss)[10]$coefficients[2,1]␣
↪- 1.96*summary(mode.nb.random.off.
↪beds)[10]$coefficients[2,2])
```

[94]:
```
array([1.06172142])
```

[95]:
```
%R summary(mode.nb.random.off.brfss)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.off.beds)[10]$coefficients[2,2]
```

[95]:
```
array([0.05989157])
```

Estremo superiore intervallo di confidenza 95%:

[96]:
```
%R exp(summary(mode.nb.random.off.brfss)[10]$coefficients[2,1]␣
↪+ 1.96*summary(mode.nb.random.off.
↪beds)[10]$coefficients[2,2])
```

[96]:
```
array([1.1548135])
```

[97]:
```
%R summary(mode.nb.random.off.brfss)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.off.beds)[10]$coefficients[2,2]
```

[97]:
```
array([0.14393886])
```

p-value per PM2.5:

[98]:
```
%R summary(mode.nb.random.off.brfss)[10]$coefficients[2,4]
```

[98]:
```
array([2.12328894e-06])
```

2.2.4 Quarta analisi: - weather

Vengono ora rimossi i dati sulle temperature e umidità relativa media

[99]:
```
```

```
%R mode.nb.random.off.weather = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                    + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                    +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                    + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                    + scale(date_since_social)␣
↪+ scale(date_since) \
                                    + scale(beds/population) \
                                    + scale(obese) +␣
↪scale(smoke) \
                                    + (1|state) \
                                    + offset(log(population)),␣
↪data = aggregate_pm_census_cdc_test_beds)

%R -o mode.nb.random.off.weather
```

[100]:
```
%R print(summary(mode.nb.random.off.weather))
```

```
Generalized linear mixed model fit by maximum likelihood␣
↪(Laplace
 Approximation) [glmerMod]
 Family: Negative Binomial(1.5759)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
↪+
    scale(mid_pecent) + scale(date_since_social) +␣
↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) + (1 |
    state) + offset(log(population))
   Data: aggregate_pm_census_cdc_test_beds
```

```
     AIC      BIC   logLik deviance df.resid
 16907.5  17044.1  -8430.8  16861.5     2779

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2174 -0.6931 -0.3715  0.2882 18.8241

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.4998   0.707
Number of obs: 2802, groups:  state, 43

Fixed effects:
                                Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                     -9.53554    0.18856 -50.570  <␣
 ↪2e-16 ***
mean_pm25                        0.12814    0.01944   6.591 4.
 ↪36e-11 ***
factor(q_popdensity)2           -0.06985    0.08762  -0.797 0.
 ↪425318
factor(q_popdensity)3           -0.06292    0.09088  -0.692 0.
 ↪488682
factor(q_popdensity)4           -0.15028    0.09488  -1.584 0.
 ↪113229
factor(q_popdensity)5           -0.01843    0.10923  -0.169 0.
 ↪866022
scale(poverty)                   0.04790    0.02676   1.790 0.
 ↪073395 .
scale(log(medianhousevalue))     0.10573    0.04371   2.419 0.
 ↪015580 *
scale(log(medhouseholdincome))   0.15278    0.04578   3.337 0.
 ↪000846 ***
scale(pct_owner_occ)             0.05597    0.02676   2.091 0.
 ↪036502 *
scale(education)                 0.15663    0.03268   4.793 1.
 ↪64e-06 ***
scale(pct_blk)                   0.24444    0.02998   8.153 3.
 ↪56e-16 ***
```

```
scale(hispanic)                0.19306    0.03117   6.193 5.
 ↪89e-10 ***
scale(older_pecent)            0.04352    0.04137   1.052 0.
 ↪292795
scale(prime_pecent)           -0.20227    0.05021  -4.028 5.
 ↪62e-05 ***
scale(mid_pecent)             -0.16667    0.03636  -4.584 4.
 ↪56e-06 ***
scale(date_since_social)       0.11647    0.12585   0.925 0.
 ↪354750
scale(date_since)              0.27117    0.04162   6.515 7.
 ↪25e-11 ***
scale(beds/population)         0.02070    0.02522   0.821 0.
 ↪411726
scale(obese)                   0.02901    0.02411   1.203 0.
 ↪228840
scale(smoke)                   0.19567    0.04243   4.612 3.
 ↪99e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[101]:
```
%R exp(summary(mode.nb.random.off.
 ↪weather)[10]$coefficients[2,1])
```

[101]:
```
array([1.1367168])
```

Estremo inferiore intervallo di confidenza 95%:

[102]:
```
%R exp(summary(mode.nb.random.off.
 ↪weather)[10]$coefficients[2,1] - 1.96*summary(mode.nb.
 ↪random.off.weather)[10]$coefficients[2,2])
```

[102]:
```
array([1.0942174])
```

[103]:
```
%R summary(mode.nb.random.off.weather)[10]$coefficients[2,1] -␣
 ↪1.96*summary(mode.nb.random.off.
 ↪weather)[10]$coefficients[2,2]
```

[103]:
```
array([0.0900394])
```

Estremo superiore intervallo di confidenza 95%:

[104]:
```
%R exp(summary(mode.nb.random.off.
↪weather)[10]$coefficients[2,1] + 1.96*summary(mode.nb.
↪random.off.weather)[10]$coefficients[2,2])
```

[104]:
```
array([1.18086688])
```

[105]:
```
%R summary(mode.nb.random.off.weather)[10]$coefficients[2,1] +␣
↪1.96*summary(mode.nb.random.off.
↪weather)[10]$coefficients[2,2]
```

[105]:
```
array([0.16624882])
```

p-value per PM2.5:

[106]:
```
%R summary(mode.nb.random.off.weather)[10]$coefficients[2,4]
```

[106]:
```
array([4.35768444e-11])
```

2.2.5 Quinta analisi: - date_since

I giorni passati dalla registrazione del primo caso di Covid in contea vengono ora rimossi

[107]:
```
%R mode.nb.random.off.outbreak = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                    + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                    +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                    + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                    + scale(date_since_social) \
                                    + scale(beds/population) \
                                    + scale(obese) +␣
↪scale(smoke) \
```

```
                                    + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                    + (1|state) \
                                    + offset(log(population)),␣
↪data = aggregate_pm_census_cdc_test_beds)

%R -o mode.nb.random.off.outbreak
```

[108]:
```
%R print(summary(mode.nb.random.off.outbreak))
```

```
Generalized linear mixed model fit by maximum likelihood␣
↪(Laplace
 Approximation) [glmerMod]
 Family: Negative Binomial(1.5669)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
↪+
    scale(mid_pecent) + scale(date_since_social) + scale(beds/
↪population) +
    scale(obese) + scale(smoke) + scale(mean_summer_temp) +
scale(mean_winter_temp) +
    scale(mean_summer_rm) + scale(mean_winter_rm) + (1 | state)␣
↪+
    offset(log(population))
   Data: aggregate_pm_census_cdc_test_beds

     AIC      BIC   logLik deviance df.resid
 16931.6  17085.9  -8439.8  16879.6     2776

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2086 -0.7003 -0.3810  0.2896 12.6175

Random effects:
 Groups Name        Variance Std.Dev.
```

```
 state  (Intercept) 0.5479   0.7402
Number of obs: 2802, groups:  state, 43

Fixed effects:
                             Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                 -9.49364    0.20746 -45.761  <␣
 ↪2e-16 ***
mean_pm25                    0.10822    0.02140   5.057 4.
 ↪27e-07 ***
factor(q_popdensity)2        0.09619    0.08643   1.113 0.
 ↪265699
factor(q_popdensity)3        0.15322    0.08944   1.713 0.
 ↪086678 .
factor(q_popdensity)4        0.10359    0.09384   1.104 0.
 ↪269647
factor(q_popdensity)5        0.25972    0.11009   2.359 0.
 ↪018313 *
scale(poverty)               0.04811    0.02659   1.809 0.
 ↪070398 .
scale(log(medianhousevalue)) 0.15060    0.04574   3.293 0.
 ↪000993 ***
scale(log(medhouseholdincome)) 0.12940    0.04661   2.776 0.
 ↪005498 **
scale(pct_owner_occ)         0.07088    0.02731   2.596 0.
 ↪009439 **
scale(education)             0.14798    0.03268   4.528 5.
 ↪94e-06 ***
scale(pct_blk)               0.22423    0.03063   7.320 2.
 ↪47e-13 ***
scale(hispanic)              0.17200    0.03185   5.401 6.
 ↪63e-08 ***
scale(older_pecent)          0.05275    0.04122   1.280 0.
 ↪200700
scale(prime_pecent)         -0.13490    0.04882  -2.763 0.
 ↪005728 **
scale(mid_pecent)           -0.12602    0.03698  -3.407 0.
 ↪000656 ***
scale(date_since_social)     0.16465    0.13221   1.245 0.
 ↪212984
```

```
scale(beds/population)          0.01849    0.02484   0.744 0.
↪456636
scale(obese)                    0.03899    0.02416   1.613 0.
↪106639
scale(smoke)                    0.20719    0.04245   4.881 1.
↪05e-06 ***
scale(mean_summer_temp)         0.17692    0.07333   2.413 0.
↪015839 *
scale(mean_winter_temp)         0.11059    0.09228   1.198 0.
↪230762
scale(mean_summer_rm)          -0.16481    0.07393  -2.229 0.
↪025800 *
scale(mean_winter_rm)           0.01057    0.04054   0.261 0.
↪794252
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[109]:
```
%R exp(summary(mode.nb.random.off.
↪outbreak)[10]$coefficients[2,1])
```

[109]:
```
array([1.11429264])
```

[110]:
```
%R summary(mode.nb.random.off.outbreak)[10]$coefficients[2,1]
```

[110]:
```
array([0.1082198])
```

Estremo inferiore intervallo di confidenza 95%:

[111]:
```
%R exp(summary(mode.nb.random.off.
↪outbreak)[10]$coefficients[2,1] - 1.96*summary(mode.nb.
↪random.off.outbreak)[10]$coefficients[2,2])
```

[111]:
```
array([1.06851851])
```

[112]:
```
%R summary(mode.nb.random.off.outbreak)[10]$coefficients[2,1]␣
↪- (1.96*summary(mode.nb.random.off.
↪outbreak)[10]$coefficients[2,2])
```

[112]:
```
array([0.06627312])
```

Estremo superiore intervallo di confidenza 95%:

[113]:
```
%R exp(summary(mode.nb.random.off.
↪outbreak)[10]$coefficients[2,1] + 1.96*summary(mode.nb.
↪random.off.outbreak)[10]$coefficients[2,2])
```

[113]:
```
array([1.16202767])
```

[114]:
```
%R summary(mode.nb.random.off.outbreak)[10]$coefficients[2,1]␣
↪+ 1.96*summary(mode.nb.random.off.
↪outbreak)[10]$coefficients[2,2]
```

[114]:
```
array([0.15016647])
```

p-value per PM2.5:

[115]:
```
%R summary(mode.nb.random.off.outbreak)[10]$coefficients[2,4]
```

[115]:
```
array([4.26624283e-07])
```

2.2.6 Sesta analisi: - NY state

In questa analisi vengono escluse le contee dello stato di New York, principale focolaio in suolo statutinitense:

[116]:
```
%R mode.nb.random.off.nyc = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                    + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                    +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                    + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                    + scale(date_since_social)␣
↪+ scale(date_since) \
                                    + scale(beds/population) \
                                    + scale(obese) +␣
↪scale(smoke) \
```

```
                                    + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                    + (1|state) \
                                    +␣
↪offset(log(population)),data =␣
↪subset(aggregate_pm_census_cdc_test_beds,!(fips %in%␣
↪c("09001","42089","36111","09009","36059","36103","34013", \
                                                                  ␣
↪                                                  ␣
↪"34019","34027","34037","34039","42103","34023","34025","34029",␣
↪\
                                                                  ␣
↪                                                   "34035",␣
↪"34003", "34017", "34031","36005","36047","36061", \
                                                                  ␣
↪                                                  ␣
↪"36079","36081",  "36085",  "36087",  "36119",  "36027",  \
                                                                  ␣
↪                                                   "36071", ␣
↪"09005",  "34021"))))

%R -o mode.nb.random.off.nyc
```

[117]:
```
%R print(summary(mode.nb.random.off.nyc))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.5813)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
```

```
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    (1 | state) + offset(log(population))
   Data: subset(aggregate_pm_census_cdc_test_beds, !(fips %in%␣
 ↪c("09001",
    "42089", "36111", "09009", "36059", "36103", "34013",␣
 ↪"34019",
    "34027", "34037", "34039", "42103", "34023", "34025",␣
 ↪"34029",
    "34035", "34003", "34017", "34031", "36005", "36047",␣
 ↪"36061",
    "36079", "36081", "36085", "36087", "36119", "36027",␣
 ↪"36071",
    "09005", "34021")))

     AIC      BIC   logLik deviance df.resid
 16528.2  16688.3  -8237.1  16474.2     2751

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2076 -0.6923 -0.3700  0.2742 17.0670

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5126   0.716
Number of obs: 2778, groups:  state, 43

Fixed effects:
                                Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                    -9.384153   0.207495 -45.226  <␣
 ↪2e-16 ***
mean_pm25                       0.103918   0.021604   4.810 1.
 ↪51e-06 ***
factor(q_popdensity)2          -0.024713   0.088944  -0.278 0.
 ↪781128
factor(q_popdensity)3           0.007481   0.092857   0.081 0.
 ↪935786
```

```
factor(q_popdensity)4           -0.057230   0.097915  -0.584 0.
 ↪558895
factor(q_popdensity)5            0.096037   0.113982   0.843 0.
 ↪399472
scale(poverty)                   0.043625   0.026840   1.625 0.
 ↪104084
scale(log(medianhousevalue))     0.104355   0.045837   2.277 0.
 ↪022808 *
scale(log(medhouseholdincome))   0.113855   0.045945   2.478 0.
 ↪013209 *
scale(pct_owner_occ)             0.057836   0.027813   2.079 0.
 ↪037575 *
scale(education)                 0.143895   0.033072   4.351 1.
 ↪36e-05 ***
scale(pct_blk)                   0.213901   0.030806   6.944 3.
 ↪82e-12 ***
scale(hispanic)                  0.159140   0.032127   4.954 7.
 ↪29e-07 ***
scale(older_pecent)              0.043665   0.042394   1.030 0.
 ↪303010
scale(prime_pecent)             -0.181443   0.051115  -3.550 0.
 ↪000386 ***
scale(mid_pecent)               -0.137464   0.038145  -3.604 0.
 ↪000314 ***
scale(date_since_social)         0.154751   0.128602   1.203 0.
 ↪228846
scale(date_since)                0.267157   0.041837   6.386 1.
 ↪71e-10 ***
scale(beds/population)           0.017649   0.025238   0.699 0.
 ↪484359
scale(obese)                     0.030048   0.024107   1.246 0.
 ↪212608
scale(smoke)                     0.194712   0.042760   4.554 5.
 ↪27e-06 ***
scale(mean_summer_temp)          0.147947   0.074132   1.996 0.
 ↪045964 *
scale(mean_winter_temp)          0.130959   0.092772   1.412 0.
 ↪158064
scale(mean_summer_rm)           -0.165439   0.074325  -2.226 0.
 ↪026022 *
```

```
scale(mean_winter_rm)            0.020898   0.041152   0.508 0.
↪611576
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[118]:
```
%R exp(summary(mode.nb.random.off.nyc)[10]$coefficients[2,1])
```

[118]:
```
array([1.10950945])
```

Estremo inferiore intervallo di confidenza 95%:

[119]:
```
%R exp(summary(mode.nb.random.off.nyc)[10]$coefficients[2,1] -␣
↪1.96*summary(mode.nb.random.off.nyc)[10]$coefficients[2,2])
```

[119]:
```
array([1.06350915])
```

[120]:
```
%R summary(mode.nb.random.off.nyc)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.off.nyc)[10]$coefficients[2,2]
```

[120]:
```
array([0.06157396])
```

Estremo superiore intervallo di confidenza 95%:

[121]:
```
%R exp(summary(mode.nb.random.off.nyc)[10]$coefficients[2,1] +␣
↪1.96*summary(mode.nb.random.off.nyc)[10]$coefficients[2,2])
```

[121]:
```
array([1.15749942])
```

[122]:
```
%R summary(mode.nb.random.off.nyc)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.off.nyc)[10]$coefficients[2,2]
```

[122]:
```
array([0.14626201])
```

p-value per PM2.5:

[123]:
```
%R summary(mode.nb.random.off.nyc)[10]$coefficients[2,4]
```

[123]:
```
array([1.50850296e-06])
```

2.2.7 Settima analisi: - county < 10 cases

Questa volta ad essere escluse sono le contee che hanno registrato un numero di casi confermati di COVID-19 inferiore a 10:

[124]:
```
%R mode.nb.random.off.large = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                + scale(date_since_social) +␣
↪scale(date_since) \
                                + scale(beds/population) \
                                + scale(obese) + scale(smoke) \
                                + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                + (1|state) \
                                + offset(log(population)), data␣
↪= subset(aggregate_pm_census_cdc_test_beds,Confirmed >=10))␣
↪\

%R -o mode.nb.random.off.large
```

[125]:
```
%R print(summary(mode.nb.random.off.large))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.6228)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
```

```
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
↪+
    scale(mid_pecent) + scale(date_since_social) +␣
↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) +
    (1 | state) + offset(log(population))
   Data: subset(aggregate_pm_census_cdc_test_beds, Confirmed >=␣
↪10)

     AIC      BIC   logLik deviance df.resid
 16765.6  16924.4  -8355.8  16711.6     2621

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2310 -0.7107 -0.3789  0.3241 13.8832

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5481   0.7403
Number of obs: 2648, groups:  state, 43

Fixed effects:
                                Estimate Std. Error z value␣
↪Pr(>|z|)
(Intercept)                    -9.322595   0.211655 -44.046  <␣
↪2e-16 ***
mean_pm25                       0.107443   0.021467   5.005 5.
↪59e-07 ***
factor(q_popdensity)2          -0.008843   0.089630  -0.099  0.
↪92141
factor(q_popdensity)3           0.005689   0.093459   0.061  0.
↪95146
factor(q_popdensity)4          -0.058146   0.098232  -0.592  0.
↪55390
factor(q_popdensity)5           0.072761   0.113599   0.641  0.
↪52185
```

```
scale(poverty)                 0.034971   0.026420   1.324  0.
 ↪18561
scale(log(medianhousevalue))   0.144403   0.046991   3.073  0.
 ↪00212 **
scale(log(medhouseholdincome)) 0.101655   0.047815   2.126  0.
 ↪03350 *
scale(pct_owner_occ)           0.059586   0.027184   2.192  0.
 ↪02838 *
scale(education)               0.139477   0.032671   4.269 1.
 ↪96e-05 ***
scale(pct_blk)                 0.232439   0.030920   7.517 5.
 ↪59e-14 ***
scale(hispanic)                0.149666   0.032951   4.542 5.
 ↪57e-06 ***
scale(older_pecent)           -0.002535   0.042837  -0.059  0.
 ↪95281
scale(prime_pecent)           -0.233396   0.049837  -4.683 2.
 ↪82e-06 ***
scale(mid_pecent)             -0.187239   0.041246  -4.540 5.
 ↪64e-06 ***
scale(date_since_social)       0.123044   0.131078   0.939  0.
 ↪34788
scale(date_since)              0.126504   0.031633   3.999 6.
 ↪36e-05 ***
scale(beds/population)         0.014193   0.023558   0.602  0.
 ↪54686
scale(obese)                   0.020587   0.024006   0.858  0.
 ↪39113
scale(smoke)                   0.179540   0.042352   4.239 2.
 ↪24e-05 ***
scale(mean_summer_temp)        0.108067   0.073721   1.466  0.
 ↪14268
scale(mean_winter_temp)        0.164062   0.093238   1.760  0.
 ↪07848 .
scale(mean_summer_rm)         -0.148131   0.069721  -2.125  0.
 ↪03362 *
scale(mean_winter_rm)          0.001566   0.040505   0.039  0.
 ↪96916
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[126]:
```
%R exp(summary(mode.nb.random.off.large)[10]$coefficients[2,1])
```

[126]:
```
array([1.1134269])
```

Estremo inferiore intervallo di confidenza 95%:

[127]:
```
%R exp(summary(mode.nb.random.off.large)[10]$coefficients[2,1]␣
↪- 1.96*summary(mode.nb.random.off.
↪large)[10]$coefficients[2,2])
```

[127]:
```
array([1.0675506])
```

[128]:
```
%R summary(mode.nb.random.off.large)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.off.large)[10]$coefficients[2,2]
```

[128]:
```
array([0.06536686])
```

Estremo superiore intervallo di confidenza 95%:

[129]:
```
%R exp(summary(mode.nb.random.off.large)[10]$coefficients[2,1]␣
↪+ 1.96*summary(mode.nb.random.off.
↪large)[10]$coefficients[2,2])
```

[129]:
```
array([1.16127466])
```

[130]:
```
%R summary(mode.nb.random.off.large)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.off.large)[10]$coefficients[2,2]
```

[130]:
```
array([0.14951825])
```

p-value per PM2.5:

[131]:
```
%R summary(mode.nb.random.off.large)[10]$coefficients[2,4]
```

[131]:
```
array([5.58718498e-07])
```

2.2.8 Ottava analisi: NCHS Codes, rural people

L'analisi seguente considera solo la popolazione residente nelle zone rurali delle contee americane, sulla base dello schema realizzato dal National Center for Health Statistics (NCHS).

[132]:
```
aggregate_pm_census_cdc_test_beds.rename(columns={'2013 code':
↪'X2013_code'}, inplace=True)
```

[133]:
```
%R -i aggregate_pm_census_cdc_test_beds

%R mode.nb.random.off.rural = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                + scale(date_since_social) +␣
↪scale(date_since) \
                                + scale(beds/population) \
                                + scale(obese) + scale(smoke) \
                                + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                + (1|state) \
                                + offset(log(population)), data␣
↪= subset(aggregate_pm_census_cdc_test_beds,X2013_code>4))

%R -o mode.nb.random.off.rural
```

[134]:
```
%R print(summary(mode.nb.random.off.rural))
```

```
Generalized linear mixed model fit by maximum likelihood␣
↪(Laplace
 Approximation) [glmerMod]
Family: Negative Binomial(1.2995)  ( log )
```

```
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    (1 | state) + offset(log(population))
   Data: subset(aggregate_pm_census_cdc_test_beds, X2013_code >␣
 ↪4)

     AIC      BIC   logLik deviance df.resid
  8190.1   8338.1  -4068.1   8136.1     1744

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.0747 -0.6560 -0.3808  0.1750 20.7346

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.2128   0.4614
Number of obs: 1771, groups:  state, 39

Fixed effects:
                              Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                   -9.16204    0.24100 -38.017  <␣
 ↪2e-16 ***
mean_pm25                      0.05484    0.03045   1.801  0.
 ↪07173 .
factor(q_popdensity)2         -0.01944    0.10630  -0.183  0.
 ↪85492
factor(q_popdensity)3          0.07033    0.11712   0.601  0.
 ↪54815
```

```
factor(q_popdensity)4            0.03609    0.13018   0.277  0.
↪78161
factor(q_popdensity)5            0.13390    0.19770   0.677  0.
↪49823
scale(poverty)                   0.02690    0.03719   0.723  0.
↪46946
scale(log(medianhousevalue))     0.04455    0.05198   0.857  0.
↪39141
scale(log(medhouseholdincome))   0.05410    0.05464   0.990  0.
↪32218
scale(pct_owner_occ)             0.03011    0.03584   0.840  0.
↪40080
scale(education)                 0.10335    0.04614   2.240  0.
↪02510 *
scale(pct_blk)                   0.16917    0.04311   3.924 8.
↪70e-05 ***
scale(hispanic)                  0.11499    0.04377   2.627  0.
↪00861 **
scale(older_pecent)             -0.02067    0.06319  -0.327  0.
↪74357
scale(prime_pecent)             -0.20287    0.06945  -2.921  0.
↪00349 **
scale(mid_pecent)               -0.13758    0.05357  -2.568  0.
↪01022 *
scale(date_since_social)         0.13201    0.09797   1.347  0.
↪17784
scale(date_since)                0.30417    0.05469   5.562 2.
↪67e-08 ***
scale(beds/population)           0.04612    0.03779   1.221  0.
↪22227
scale(obese)                     0.08486    0.03442   2.465  0.
↪01368 *
scale(smoke)                     0.15696    0.06072   2.585  0.
↪00974 **
scale(mean_summer_temp)          0.27485    0.10585   2.597  0.
↪00941 **
scale(mean_winter_temp)          0.10580    0.11211   0.944  0.
↪34528
scale(mean_summer_rm)           -0.06478    0.06970  -0.929  0.
↪35271
```

```
scale(mean_winter_rm)            0.02571    0.04930   0.522  0.
↪60201
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[135]:
```
%R exp(summary(mode.nb.random.off.rural)[10]$coefficients[2,1])
```

[135]:
```
array([1.05636737])
```

Estremo inferiore intervallo di confidenza 95%:

[136]:
```
%R exp(summary(mode.nb.random.off.rural)[10]$coefficients[2,1]␣
↪- 1.96*summary(mode.nb.random.off.
↪rural)[10]$coefficients[2,2])
```

[136]:
```
array([0.99516487])
```

[137]:
```
%R summary(mode.nb.random.off.rural)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.off.rural)[10]$coefficients[2,2]
```

[137]:
```
array([-0.00484686])
```

Estremo superiore intervallo di confidenza 95%:

[138]:
```
%R exp(summary(mode.nb.random.off.rural)[10]$coefficients[2,1]␣
↪+ 1.96*summary(mode.nb.random.off.
↪rural)[10]$coefficients[2,2])
```

[138]:
```
array([1.12133381])
```

[139]:
```
%R summary(mode.nb.random.off.rural)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.off.rural)[10]$coefficients[2,2]
```

[139]:
```
array([0.11451888])
```

p-value per PM2.5:

[140]:
```
%R summary(mode.nb.random.off.rural)[10]$coefficients[2,4]
```

[140]:
```
array([0.07172998])
```

2.2.9 Nona analisi: NCHS Codes, urban people

Sulla base dello stesso modello di rappresentazione della popolazione americana, questa volta vengono considerate le persone residenti nelle zone urbane:

[141]:
```
%R mode.nb.random.off.urban = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                + scale(date_since_social) +␣
↪scale(date_since) \
                                + scale(beds/population) \
                                + scale(obese) + scale(smoke) \
                                + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                + (1|state) \
                                + offset(log(population)), data␣
↪= subset(aggregate_pm_census_cdc_test_beds,X2013_code<=4))

%R -o mode.nb.random.off.urban
```

[142]:
```
%R print(summary(mode.nb.random.off.urban))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(2.397)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
```

```
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    (1 | state) + offset(log(population))
   Data: subset(aggregate_pm_census_cdc_test_beds, X2013_code␣
 ↪<= 4)

     AIC      BIC   logLik deviance df.resid
  8624.1   8757.5  -4285.1   8570.1     1004

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.4751 -0.7290 -0.2172  0.4612  6.7801

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5486   0.7407
Number of obs: 1031, groups:  state, 43

Fixed effects:
                                Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                    -9.194149   0.330792 -27.794  <␣
 ↪2e-16 ***
mean_pm25                       0.105685   0.028176   3.751 0.
 ↪000176 ***
factor(q_popdensity)2           0.008073   0.234971   0.034 0.
 ↪972592
factor(q_popdensity)3          -0.011869   0.228134  -0.052 0.
 ↪958508
factor(q_popdensity)4          -0.058594   0.227657  -0.257 0.
 ↪796885
factor(q_popdensity)5           0.041066   0.235617   0.174 0.
 ↪861635
scale(poverty)                  0.076863   0.035342   2.175 0.
 ↪029642 *
```

```
scale(log(medianhousevalue))    0.163419   0.061461   2.659 0.
 ↪007840 **
scale(log(medhouseholdincome))  0.139554   0.067125   2.079 0.
 ↪037617 *
scale(pct_owner_occ)            0.090598   0.043215   2.096 0.
 ↪036041 *
scale(education)                0.102861   0.041323   2.489 0.
 ↪012803 *
scale(pct_blk)                  0.262245   0.038234   6.859 6.
 ↪94e-12 ***
scale(hispanic)                 0.249445   0.042090   5.926 3.
 ↪10e-09 ***
scale(older_pecent)             0.118814   0.052638   2.257 0.
 ↪023997 *
scale(prime_pecent)            -0.102136   0.071807  -1.422 0.
 ↪154918
scale(mid_pecent)              -0.090010   0.058497  -1.539 0.
 ↪123873
scale(date_since_social)        0.033091   0.118451   0.279 0.
 ↪779966
scale(date_since)               0.119260   0.055383   2.153 0.
 ↪031290 *
scale(beds/population)         -0.016663   0.028495  -0.585 0.
 ↪558698
scale(obese)                   -0.014378   0.033358  -0.431 0.
 ↪666446
scale(smoke)                    0.169067   0.052535   3.218 0.
 ↪001290 **
scale(mean_summer_temp)         0.067175   0.079922   0.841 0.
 ↪400624
scale(mean_winter_temp)         0.011439   0.110876   0.103 0.
 ↪917826
scale(mean_summer_rm)           0.007606   0.070560   0.108 0.
 ↪914163
scale(mean_winter_rm)          -0.019399   0.048677  -0.399 0.
 ↪690242
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[143]:
```
%R exp(summary(mode.nb.random.off.urban)[10]$coefficients[2,1])
```

[143]:
```
array([1.11147225])
```

Estremo inferiore intervallo di confidenza 95%:

[144]:
```
%R exp(summary(mode.nb.random.off.urban)[10]$coefficients[2,1]␣
↪- 1.96*summary(mode.nb.random.off.
↪urban)[10]$coefficients[2,2])
```

[144]:
```
array([1.05175457])
```

[145]:
```
%R summary(mode.nb.random.off.urban)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.off.urban)[10]$coefficients[2,2]
```

[145]:
```
array([0.05045978])
```

Estremo superiore intervallo di confidenza 95%:

[146]:
```
%R exp(summary(mode.nb.random.off.urban)[10]$coefficients[2,1]␣
↪+ 1.96*summary(mode.nb.random.off.
↪urban)[10]$coefficients[2,2])
```

[146]:
```
array([1.17458065])
```

[147]:
```
%R summary(mode.nb.random.off.urban)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.off.urban)[10]$coefficients[2,2]
```

[147]:
```
array([0.16091119])
```

p-value per PM2.5:

[148]:
```
%R summary(mode.nb.random.off.urban)[10]$coefficients[2,4]
```

[148]:
```
array([0.00017623])
```

2.2.10 Decima analisi: q_pm

Il livello di concentrazione di PM2.5 viene trasformato in una variabile categorica, dopo la suddivisione in 4 quantili (0.2, 0.4, 0.6, 0.8):

[149]:
```
aggregate_pm_census_cdc_test_beds['q_pm'] = 1

quantile_pm = np.
↪quantile(aggregate_pm_census_cdc_test_beds['mean_pm25'], [0.
↪2,0.4,0.6,0.8])

aggregate_pm_census_cdc_test_beds['q_pm'].
↪loc[aggregate_pm_census_cdc_test_beds['mean_pm25'] <=␣
↪quantile_pm[0]] = 1
aggregate_pm_census_cdc_test_beds['q_pm'].
↪loc[(aggregate_pm_census_cdc_test_beds['mean_pm25'] >␣
↪quantile_pm[0])
                                                    &␣
↪(aggregate_pm_census_cdc_test_beds['mean_pm25'] <=␣
↪quantile_pm[1])] = 2
aggregate_pm_census_cdc_test_beds['q_pm'].
↪loc[(aggregate_pm_census_cdc_test_beds['mean_pm25'] >␣
↪quantile_pm[1])
                                                    &␣
↪(aggregate_pm_census_cdc_test_beds['mean_pm25'] <=␣
↪quantile_pm[2])] = 3
aggregate_pm_census_cdc_test_beds['q_pm'].
↪loc[(aggregate_pm_census_cdc_test_beds['mean_pm25'] >␣
↪quantile_pm[2])
                                                    &␣
↪(aggregate_pm_census_cdc_test_beds['mean_pm25'] <=␣
↪quantile_pm[3])] = 4
aggregate_pm_census_cdc_test_beds['q_pm'].
↪loc[aggregate_pm_census_cdc_test_beds['mean_pm25'] >␣
↪quantile_pm[3]] = 5
```

[150]:
```
%R -i aggregate_pm_census_cdc_test_beds

%R mode.nb.random.off.catepm = glmer.nb(Deaths ~ factor(q_pm)␣
↪+ factor(q_popdensity) \
                                  + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                  +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
```

```
                                    + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                    + scale(date_since_social) +␣
↪scale(date_since) \
                                    + scale(beds/population) \
                                    + scale(obese) + scale(smoke) \
                                    + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                    + (1|state) \
                                    + offset(log(population)),␣
↪data = (aggregate_pm_census_cdc_test_beds))

%R -o mode.nb.random.off.catepm
```

[151]:
```
%R print(summary(mode.nb.random.off.catepm))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.6033)  ( log )
Formula: Deaths ~ factor(q_pm) + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    (1 | state) + offset(log(population))
   Data: (aggregate_pm_census_cdc_test_beds)

      AIC      BIC   logLik deviance df.resid
```

```
 16894.9  17073.1  -8417.5  16834.9     2772

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2216 -0.6960 -0.3670  0.2841 16.1886

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5776   0.76
Number of obs: 2802, groups:  state, 43

Fixed effects:
                                Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                    -8.848728   0.155702 -56.831  <␣
 ↪2e-16 ***
factor(q_pm)2                   0.131991   0.094123   1.402 0.
 ↪160818
factor(q_pm)3                   0.384744   0.109024   3.529 0.
 ↪000417 ***
factor(q_pm)4                   0.531599   0.123795   4.294 1.
 ↪75e-05 ***
factor(q_pm)5                   0.554606   0.135297   4.099 4.
 ↪15e-05 ***
factor(q_popdensity)2           0.003099   0.089182   0.035 0.
 ↪972281
factor(q_popdensity)3           0.020481   0.093381   0.219 0.
 ↪826398
factor(q_popdensity)4          -0.051280   0.098230  -0.522 0.
 ↪601646
factor(q_popdensity)5           0.119716   0.112152   1.067 0.
 ↪285772
scale(poverty)                  0.049949   0.026674   1.873 0.
 ↪061129 .
scale(log(medianhousevalue))    0.127196   0.045757   2.780 0.
 ↪005439 **
scale(log(medhouseholdincome))  0.136136   0.046678   2.916 0.
 ↪003540 **
scale(pct_owner_occ)            0.050831   0.027274   1.864 0.
 ↪062361 .
```

```
scale(education)                0.146091   0.032779   4.457 8.
 ↪32e-06 ***
scale(pct_blk)                  0.226128   0.030235   7.479 7.
 ↪49e-14 ***
scale(hispanic)                 0.175153   0.032042   5.466 4.
 ↪59e-08 ***
scale(older_pecent)             0.059628   0.042009   1.419 0.
 ↪155773
scale(prime_pecent)            -0.178236   0.050714  -3.515 0.
 ↪000440 ***
scale(mid_pecent)              -0.143096   0.037441  -3.822 0.
 ↪000132 ***
scale(date_since_social)        0.173251   0.135408   1.279 0.
 ↪200730
scale(date_since)               0.266854   0.041656   6.406 1.
 ↪49e-10 ***
scale(beds/population)          0.019781   0.025026   0.790 0.
 ↪429296
scale(obese)                    0.029013   0.024150   1.201 0.
 ↪229611
scale(smoke)                    0.192489   0.042570   4.522 6.
 ↪13e-06 ***
scale(mean_summer_temp)         0.162509   0.073528   2.210 0.
 ↪027094 *
scale(mean_winter_temp)         0.110958   0.093285   1.189 0.
 ↪234262
scale(mean_summer_rm)          -0.151229   0.075169  -2.012 0.
 ↪044234 *
scale(mean_winter_rm)          -0.005655   0.040957  -0.138 0.
 ↪890188
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

I dati per il primo quantile:

- MRR (Mortality rate ratios) prodotto:

[152]:
```
%R exp(summary(mode.nb.random.off.
 ↪catepm)[10]$coefficients[2,1])
```

[152]:
```
array([1.14109818])
```

- Estremo inferiore intervallo di confidenza 95%:

[153]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[2,1] - 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[2,2])
```

[153]:
```
array([0.9488638])
```

[154]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[2,1] -␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[2,2]
```

[154]:
```
array([-0.05249001])
```

- Estremo superiore intervallo di confidenza 95%:

[155]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[2,1] + 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[2,2])
```

[155]:
```
array([1.37227814])
```

[156]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[2,1] +␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[2,2]
```

[156]:
```
array([0.31647223])
```

- p-value per PM2.5:

[157]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[2,4]
```

[157]:
```
array([0.16081807])
```

I dati per il secondo quantile:

- MRR (Mortality rate ratios) prodotto:

[158]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[3,1])
```

[158]:
```
array([1.46923874])
```

- Estremo inferiore intervallo di confidenza 95%:

[159]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[3,1] - 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[3,2])
```

[159]:
```
array([1.18655863])
```

[160]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[3,1] -␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[3,2]
```

[160]:
```
array([0.17105721])
```

- Estremo superiore intervallo di confidenza 95%:

[161]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[3,1] + 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[3,2])
```

[161]:
```
array([1.81926323])
```

[162]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[3,1] +␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[3,2]
```

[162]:
```
array([0.5984316])
```

- p-value per PM2.5:

[163]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[3,4]
```

[163]:
```
array([0.00041716])
```

I dati per il terzo quantile:

- MRR (Mortality rate ratios) prodotto:

[164]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[4,1])
```

[164]:
```
array([1.70165185])
```

- Estremo inferiore intervallo di confidenza 95%:

[165]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[4,1] - 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[4,2])
```

[165]:
```
array([1.33504082])
```

[166]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[4,1] -␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[4,2]
```

[166]:
```
array([0.28896187])
```

- Estremo superiore intervallo di confidenza 95%:

[167]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[4,1] + 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[4,2])
```

[167]:
```
array([2.16893668])
```

[168]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[4,1] +␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[4,2]
```

[168]:
```
array([0.77423704])
```

- p-value per PM2.5:

[169]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[4,4]
```

[169]:
```
array([1.75322409e-05])
```

I dati per il quarto quantile:

- MRR (Mortality rate ratios) prodotto:

[170]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[5,1])
```

[170]:
```
array([1.74125408])
```

- Estremo inferiore intervallo di confidenza 95%:

[171]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[5,1] - 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[5,2])
```

[171]:
```
array([1.33565701])
```

[172]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[5,1] -␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[5,2]
```

[172]:
```
array([0.28942331])
```

- Estremo superiore intervallo di confidenza 95%:

[173]:
```
%R exp(summary(mode.nb.random.off.
↪catepm)[10]$coefficients[5,1] + 1.96*summary(mode.nb.random.
↪off.catepm)[10]$coefficients[5,2])
```

[173]:
```
array([2.27001824])
```

[174]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[5,1] +␣
↪1.96*summary(mode.nb.random.off.
↪catepm)[10]$coefficients[5,2]
```

[174]:
```
array([0.81978787])
```

- p-value per PM2.5:

[175]:
```
%R summary(mode.nb.random.off.catepm)[10]$coefficients[5,4]
```

[175]:
```
array([4.14636324e-05])
```

2.2.11 Undicesima analisi: + tests

Alle potenziali confondenti vengono aggiunti i dati sul numero di test per COVID-19 effettuati:

[176]:
```
%R mode.nb.random.off.tested = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                  + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                  +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                  + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                  + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                  + scale(date_since_social) +␣
↪scale(date_since) \
                                  + scale(beds/population) \
                                  + scale(obese) + scale(smoke) \
                                  + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                  + (1|state) \
                                  + scale(totalTestResults) \
                                  + offset(log(population)),␣
↪data = (aggregate_pm_census_cdc_test_beds))

%R -o mode.nb.random.off.tested
```

[177]:
```
%R print(summary(mode.nb.random.off.tested))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.5976)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
```

```
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    (1 | state) + scale(totalTestResults) +␣
 ↪offset(log(population))
   Data: (aggregate_pm_census_cdc_test_beds)

     AIC      BIC   logLik deviance df.resid
 16887.7  17054.0  -8415.9  16831.7     2774

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2200 -0.6936 -0.3691  0.2850 17.2220

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.4958   0.7042
Number of obs: 2802, groups:  state, 43

Fixed effects:
                                 Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                    -9.2898019  0.2097595 -44.288  <␣
 ↪2e-16 ***
mean_pm25                       0.1036583  0.0213897   4.846 1.
 ↪26e-06 ***
factor(q_popdensity)2          -0.0205772  0.0887520  -0.232 0.
 ↪816654
factor(q_popdensity)3           0.0000714  0.0925628   0.001 0.
 ↪999385
factor(q_popdensity)4          -0.0677783  0.0974601  -0.695 0.
 ↪486775
factor(q_popdensity)5           0.0819853  0.1132645   0.724 0.
 ↪469164
scale(poverty)                  0.0501667  0.0266435   1.883 0.
 ↪059716 .
scale(log(medianhousevalue))    0.1373755  0.0457310   3.004 0.
 ↪002665 **
scale(log(medhouseholdincome))  0.1200027  0.0466215   2.574 0.
 ↪010054 *
```

```
scale(pct_owner_occ)            0.0628388  0.0272882   2.303 0.
 ↪021291 *
scale(education)                0.1503254  0.0326738   4.601 4.
 ↪21e-06 ***
scale(pct_blk)                  0.2170125  0.0304501   7.127 1.
 ↪03e-12 ***
scale(hispanic)                 0.1637085  0.0318102   5.146 2.
 ↪66e-07 ***
scale(older_pecent)             0.0449036  0.0416434   1.078 0.
 ↪280905
scale(prime_pecent)            -0.1868161  0.0505263  -3.697 0.
 ↪000218 ***
scale(mid_pecent)              -0.1405626  0.0374990  -3.748 0.
 ↪000178 ***
scale(date_since_social)        0.1169356  0.1275040   0.917 0.
 ↪359083
scale(date_since)               0.2641516  0.0416527   6.342 2.
 ↪27e-10 ***
scale(beds/population)          0.0182413  0.0250648   0.728 0.
 ↪466758
scale(obese)                    0.0291162  0.0241565   1.205 0.
 ↪228082
scale(smoke)                    0.2024460  0.0424997   4.763 1.
 ↪90e-06 ***
scale(mean_summer_temp)         0.1482043  0.0735912   2.014 0.
 ↪044022 *
scale(mean_winter_temp)         0.1174271  0.0924568   1.270 0.
 ↪204057
scale(mean_summer_rm)          -0.1647576  0.0732390  -2.250 0.
 ↪024475 *
scale(mean_winter_rm)          -0.0012752  0.0403778  -0.032 0.
 ↪974805
scale(totalTestResults)         0.3077156  0.1520541   2.024 0.
 ↪042999 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[178]:
```
%R exp(summary(mode.nb.random.off.
↪tested)[10]$coefficients[2,1])
```

[178]:
```
array([1.10922139])
```

Estremo inferiore intervallo di confidenza 95%:

[179]:
```
%R exp(summary(mode.nb.random.off.
↪tested)[10]$coefficients[2,1] - 1.96*summary(mode.nb.random.
↪off.tested)[10]$coefficients[2,2])
```

[179]:
```
array([1.0636799])
```

[180]:
```
%R summary(mode.nb.random.off.tested)[10]$coefficients[2,1] -␣
↪1.96*summary(mode.nb.random.off.
↪tested)[10]$coefficients[2,2]
```

[180]:
```
array([0.0617345])
```

Estremo superiore intervallo di confidenza 95%:

[181]:
```
%R exp(summary(mode.nb.random.off.
↪tested)[10]$coefficients[2,1] + 1.96*summary(mode.nb.random.
↪off.tested)[10]$coefficients[2,2])
```

[181]:
```
array([1.15671273])
```

[182]:
```
%R summary(mode.nb.random.off.tested)[10]$coefficients[2,1] +␣
↪1.96*summary(mode.nb.random.off.
↪tested)[10]$coefficients[2,2]
```

[182]:
```
array([0.14558213])
```

p-value per PM2.5:

[183]:
```
%R summary(mode.nb.random.off.tested)[10]$coefficients[2,4]
```

[183]:
```
array([1.25862357e-06])
```

2.2.12 Dodicesima analisi: + log(population)

Oltre alle già citate confondenti, si aggiunge il logaritmo della densità di popolazione:

[190]:
```
%R mode.nb.random.off.main.logpopdensity = glmer.nb(Deaths ~␣
↪mean_pm25 + scale(log(popdensity)) \
                                                + scale(poverty) ␣
↪+ scale(log(medianhousevalue)) \
                                                +␣
↪scale(log(medhouseholdincome)) + scale(pct_owner_occ) \
                                                + scale(education)␣
↪+ scale(pct_blk) + scale(hispanic) \
                                                +␣
↪scale(older_pecent) + scale(prime_pecent) +␣
↪scale(mid_pecent) \
                                                +␣
↪scale(date_since_social) + scale(date_since) \
                                                + scale(beds/
↪population) \
                                                + scale(obese) +␣
↪scale(smoke) \
                                                +␣
↪scale(mean_summer_temp) + scale(mean_winter_temp) +␣
↪scale(mean_summer_rm) + scale(mean_winter_rm) \
                                                + (1|state) \
                                                +␣
↪offset(log(population)), data =␣
↪aggregate_pm_census_cdc_test_beds) \

%R -o mode.nb.random.off.main.logpopdensity
```

[191]:
```
%R print(summary(mode.nb.random.off.main.logpopdensity))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.5941)  ( log )
Formula: Deaths ~ mean_pm25 + scale(log(popdensity)) +␣
 ↪scale(poverty) +
```

```
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    (1 | state) + offset(log(population))
   Data: aggregate_pm_census_cdc_test_beds

     AIC      BIC   logLik deviance df.resid
 16886.7  17029.2  -8419.3  16838.7     2778

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2191 -0.6879 -0.3655  0.2760 17.1111

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5535   0.744
Number of obs: 2802, groups:  state, 43

Fixed effects:
                                Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                    -9.294803   0.222789 -41.720  <␣
 ↪2e-16 ***
mean_pm25                       0.094058   0.022578   4.166 3.
 ↪10e-05 ***
scale(log(popdensity))          0.076360   0.041843   1.825 0.
 ↪068015 .
scale(poverty)                  0.053955   0.026533   2.033 0.
 ↪042004 *
scale(log(medianhousevalue))    0.118921   0.045846   2.594 0.
 ↪009489 **
```

```
scale(log(medhouseholdincome))  0.121747   0.046233   2.633 0.
 ↪008455 **
scale(pct_owner_occ)            0.066360   0.027603   2.404 0.
 ↪016214 *
scale(education)                0.146942   0.032689   4.495 6.
 ↪95e-06 ***
scale(pct_blk)                  0.221555   0.030073   7.367 1.
 ↪74e-13 ***
scale(hispanic)                 0.169173   0.031674   5.341 9.
 ↪24e-08 ***
scale(older_pecent)             0.048099   0.041633   1.155 0.
 ↪247959
scale(prime_pecent)            -0.196106   0.050593  -3.876 0.
 ↪000106 ***
scale(mid_pecent)              -0.139536   0.037468  -3.724 0.
 ↪000196 ***
scale(date_since_social)        0.155350   0.132861   1.169 0.
 ↪242296
scale(date_since)               0.242730   0.041272   5.881 4.
 ↪07e-09 ***
scale(beds/population)          0.016587   0.024967   0.664 0.
 ↪506465
scale(obese)                    0.023672   0.024022   0.985 0.
 ↪324420
scale(smoke)                    0.201948   0.042608   4.740 2.
 ↪14e-06 ***
scale(mean_summer_temp)         0.171737   0.075259   2.282 0.
 ↪022491 *
scale(mean_winter_temp)         0.118149   0.093520   1.263 0.
 ↪206458
scale(mean_summer_rm)          -0.160029   0.074690  -2.143 0.
 ↪032147 *
scale(mean_winter_rm)          -0.004449   0.040665  -0.109 0.
 ↪912885
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[192]:
```
%R exp(summary(mode.nb.random.off.main.
↪logpopdensity)[10]$coefficients[2,1])
```

[192]:
```
array([1.09862357])
```

Estremo inferiore intervallo di confidenza 95%:

[193]:
```
%R exp(summary(mode.nb.random.off.main.
↪logpopdensity)[10]$coefficients[2,1] - 1.96*summary(mode.nb.
↪random.off.main.logpopdensity)[10]$coefficients[2,2])
```

[193]:
```
array([1.05106604])
```

[194]:
```
%R summary(mode.nb.random.off.main.
↪logpopdensity)[10]$coefficients[2,1] - 1.96*summary(mode.nb.
↪random.off.main.logpopdensity)[10]$coefficients[2,2]
```

[194]:
```
array([0.04980492])
```

Estremo superiore intervallo di confidenza 95%:

[195]:
```
%R exp(summary(mode.nb.random.off.main.
↪logpopdensity)[10]$coefficients[2,1] + 1.96*summary(mode.nb.
↪random.off.main.logpopdensity)[10]$coefficients[2,2])
```

[195]:
```
array([1.14833294])
```

[196]:
```
%R summary(mode.nb.random.off.main.
↪logpopdensity)[10]$coefficients[2,1] + 1.96*summary(mode.nb.
↪random.off.main.logpopdensity)[10]$coefficients[2,2]
```

[196]:
```
array([0.13831127])
```

p-value per PM2.5:

[197]:
```
%R summary(mode.nb.random.off.main.
↪logpopdensity)[10]$coefficients[2,4]
```

[197]:
```
array([3.10139777e-05])
```

2.2.13 Tredicesima analisi: no offset (log(population))

L'offset viene rimosso e il logaritmo della popolazione si aggiunge a tutte le precedenti confondenti:

[198]:
```
%R mode.nb.random.log = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                          + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                          + scale(log(medhouseholdincome)) +␣
↪scale(pct_owner_occ) \
                          + scale(education) + scale(pct_blk) +␣
↪scale(hispanic) \
                          + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                          + scale(date_since_social) +␣
↪scale(date_since) \
                          + scale(beds/population) \
                          + scale(obese) + scale(smoke) \
                          + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                          + (1|state) \
                          + scale(log(population)), data =␣
↪(aggregate_pm_census_cdc_test_beds))

%R -o mode.nb.random.log
```

[199]:
```
%R print(summary(mode.nb.random.log))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.6019)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
```

```
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    (1 | state) + scale(log(population))
   Data: (aggregate_pm_census_cdc_test_beds)

     AIC      BIC   logLik deviance df.resid
 16886.7  17053.0  -8415.3  16830.7     2774

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.2201 -0.6901 -0.3655  0.2781 16.7046

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.5457   0.7387
Number of obs: 2802, groups:  state, 43

Fixed effects:
                                Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                     1.027530   0.217846   4.717 2.
 ↪40e-06 ***
mean_pm25                       0.095669   0.021899   4.369 1.
 ↪25e-05 ***
factor(q_popdensity)2          -0.062446   0.090890  -0.687 0.
 ↪492049
factor(q_popdensity)3          -0.062101   0.096916  -0.641 0.
 ↪521669
factor(q_popdensity)4          -0.152393   0.104774  -1.454 0.
 ↪145809
factor(q_popdensity)5          -0.059985   0.129720  -0.462 0.
 ↪643781
scale(poverty)                  0.046780   0.026602   1.759 0.
 ↪078655 .
scale(log(medianhousevalue))    0.119124   0.046071   2.586 0.
 ↪009718 **
```

```
scale(log(medhouseholdincome))  0.115984   0.046661   2.486 0.
 ↪012931 *
scale(pct_owner_occ)            0.065607   0.027289   2.404 0.
 ↪016211 *
scale(education)                0.148056   0.032660   4.533 5.
 ↪81e-06 ***
scale(pct_blk)                  0.214475   0.030415   7.052 1.
 ↪77e-12 ***
scale(hispanic)                 0.167168   0.031947   5.233 1.
 ↪67e-07 ***
scale(older_pecent)             0.052532   0.041866   1.255 0.
 ↪209564
scale(prime_pecent)            -0.189896   0.050727  -3.744 0.
 ↪000181 ***
scale(mid_pecent)              -0.134098   0.037701  -3.557 0.
 ↪000375 ***
scale(date_since_social)        0.159057   0.131979   1.205 0.
 ↪228139
scale(date_since)               0.239978   0.042896   5.594 2.
 ↪21e-08 ***
scale(beds/population)          0.014482   0.024971   0.580 0.
 ↪561933
scale(obese)                    0.028459   0.024137   1.179 0.
 ↪238372
scale(smoke)                    0.207757   0.042670   4.869 1.
 ↪12e-06 ***
scale(mean_summer_temp)         0.173640   0.074834   2.320 0.
 ↪020322 *
scale(mean_winter_temp)         0.105795   0.093332   1.134 0.
 ↪256992
scale(mean_summer_rm)          -0.138016   0.074773  -1.846 0.
 ↪064922 .
scale(mean_winter_rm)          -0.004354   0.040620  -0.107 0.
 ↪914644
scale(log(population))          1.578120   0.044543  35.429  <␣
 ↪2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[200]:
```
%R exp(summary(mode.nb.random.log)[10]$coefficients[2,1])
```

[200]:
```
array([1.10039424])
```

Estremo inferiore intervallo di confidenza 95%:

[201]:
```
%R exp(summary(mode.nb.random.log)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.log)[10]$coefficients[2,2])
```

[201]:
```
array([1.05416247])
```

[202]:
```
%R summary(mode.nb.random.log)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.log)[10]$coefficients[2,2]
```

[202]:
```
array([0.05274659])
```

Estremo superiore intervallo di confidenza 95%:

[203]:
```
%R exp(summary(mode.nb.random.log)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.log)[10]$coefficients[2,2])
```

[203]:
```
array([1.14865357])
```

[204]:
```
%R summary(mode.nb.random.log)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.log)[10]$coefficients[2,2]
```

[204]:
```
array([0.13859045])
```

p-value per PM2.5:

[205]:
```
%R summary(mode.nb.random.log)[10]$coefficients[2,4]
```

[205]:
```
array([1.25024935e-05])
```

2.2.14 Quattordicesima analisi: no offset (population)

L'offset (ovvero la variabile per cui il modello non calcola, durante l'addestramento, un coefficiente, che per default viene settato ad 1), viene rimosso e la popolazione si aggiunge a tutte le precedenti confondenti:

[206]:
```

```

```
%R mode.nb.random.nonlog = glmer.nb(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                                + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                                + scale(log(medhouseholdincome))␣
↪+ scale(pct_owner_occ) \
                                + scale(education) +␣
↪scale(pct_blk) + scale(hispanic) \
                                + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                                + scale(date_since_social) +␣
↪scale(date_since) + scale(beds/population) \
                                + scale(obese) + scale(smoke) \
                                + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                                + (1|state) \
                                + scale((population)), data =␣
↪(aggregate_pm_census_cdc_test_beds))

%R -o mode.nb.random.nonlog
```

[207]:
```
%R print(summary(mode.nb.random.nonlog))
```

```
Generalized linear mixed model fit by maximum likelihood␣
 ↪(Laplace
  Approximation) [glmerMod]
 Family: Negative Binomial(1.1174)  ( log )
Formula: Deaths ~ mean_pm25 + factor(q_popdensity) +␣
 ↪scale(poverty) +
    scale(log(medianhousevalue)) +␣
 ↪scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
```

```
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) +
    (1 | state) + scale((population))
   Data: (aggregate_pm_census_cdc_test_beds)

     AIC      BIC   logLik deviance df.resid
 17574.4  17740.7  -8759.2  17518.4     2774

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.0571 -0.6891 -0.3489  0.2630 24.0791

Random effects:
 Groups Name        Variance Std.Dev.
 state  (Intercept) 0.594    0.7707
Number of obs: 2802, groups:  state, 43

Fixed effects:
                              Estimate Std. Error z value␣
↪Pr(>|z|)
(Intercept)                   -0.18183    0.23932  -0.760 0.
↪447392
mean_pm25                      0.13454    0.02537   5.303 1.
↪14e-07 ***
factor(q_popdensity)2          0.60533    0.09800   6.177 6.
↪53e-10 ***
factor(q_popdensity)3          0.99446    0.10266   9.687  <␣
↪2e-16 ***
factor(q_popdensity)4          1.33502    0.10967  12.174  <␣
↪2e-16 ***
factor(q_popdensity)5          1.99094    0.13200  15.083  <␣
↪2e-16 ***
scale(poverty)                 0.02809    0.03109   0.903 0.
↪366343
scale(log(medianhousevalue))   0.25744    0.05433   4.738 2.
↪16e-06 ***
scale(log(medhouseholdincome)) 0.10078    0.05449   1.850 0.
↪064379 .
scale(pct_owner_occ)           0.12258    0.03273   3.746 0.
↪000180 ***
```

```
scale(education)                0.13387    0.03854   3.474 0.
 ↪000513 ***
scale(pct_blk)                  0.15357    0.03623   4.239 2.
 ↪25e-05 ***
scale(hispanic)                 0.14874    0.03632   4.096 4.
 ↪21e-05 ***
scale(older_pecent)            -0.03619    0.04574  -0.791 0.
 ↪428858
scale(prime_pecent)            -0.06563    0.05636  -1.164 0.
 ↪244254
scale(mid_pecent)              -0.21145    0.04281  -4.939 7.
 ↪86e-07 ***
scale(date_since_social)        0.20532    0.13858   1.482 0.
 ↪138449
scale(date_since)               0.62413    0.04785  13.043  <␣
 ↪2e-16 ***
scale(beds/population)          0.07982    0.03058   2.610 0.
 ↪009059 **
scale(obese)                    0.08491    0.02816   3.015 0.
 ↪002569 **
scale(smoke)                    0.19654    0.04886   4.023 5.
 ↪75e-05 ***
scale(mean_summer_temp)         0.02849    0.08602   0.331 0.
 ↪740449
scale(mean_winter_temp)         0.23828    0.10557   2.257 0.
 ↪024006 *
scale(mean_summer_rm)          -0.23307    0.08362  -2.787 0.
 ↪005313 **
scale(mean_winter_rm)           0.03412    0.04684   0.729 0.
 ↪466300
scale((population))             0.89164    0.06391  13.951  <␣
 ↪2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MRR (Mortality rate ratios) prodotto:

[208]:
```
%R exp(summary(mode.nb.random.nonlog)[10]$coefficients[2,1])
```

[208]:
```
array([1.14400888])
```

Estremo inferiore intervallo di confidenza 95%:

[209]:
```
%R exp(summary(mode.nb.random.nonlog)[10]$coefficients[2,1] -␣
↪1.96*summary(mode.nb.random.nonlog)[10]$coefficients[2,2])
```

[209]:
```
array([1.08851248])
```

[210]:
```
%R summary(mode.nb.random.nonlog)[10]$coefficients[2,1] - 1.
↪96*summary(mode.nb.random.nonlog)[10]$coefficients[2,2]
```

[210]:
```
array([0.08481207])
```

Estremo superiore intervallo di confidenza 95%:

[211]:
```
%R exp(summary(mode.nb.random.nonlog)[10]$coefficients[2,1] +␣
↪1.96*summary(mode.nb.random.nonlog)[10]$coefficients[2,2])
```

[211]:
```
array([1.20233469])
```

[212]:
```
%R summary(mode.nb.random.nonlog)[10]$coefficients[2,1] + 1.
↪96*summary(mode.nb.random.nonlog)[10]$coefficients[2,2]
```

[212]:
```
array([0.18426524])
```

p-value per PM2.5:

[213]:
```
%R summary(mode.nb.random.nonlog)[10]$coefficients[2,4]
```

[213]:
```
array([1.13969156e-07])
```

2.2.15 Quindicesima analisi: zero inflated

Per l'analisi che segue viene costruito un modello statistico basato su una distribuzione di probabilità a inflazione zero, ovvero una distribuzione che consente frequenti osservazioni a valore zero.

[214]:
```
%R glmmTMB.off.main = glmmTMB(Deaths ~ mean_pm25 +␣
↪factor(q_popdensity) \
                            + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                            + scale(log(medhouseholdincome)) +␣
↪scale(pct_owner_occ) \
```

```
                            + scale(education) + scale(pct_blk)␣
↪+ scale(hispanic) \
                            + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                            + scale(date_since_social) +␣
↪scale(date_since) \
                            + scale(beds/population) \
                            + scale(obese) + scale(smoke) \
                            + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                            + offset(log(population)) + (1 |␣
↪state), data = aggregate_pm_census_cdc_test_beds, \
                            family = nbinom2, ziformula  = ~ 1)

%R -o glmmTMB.off.main
```

MRR (Mortality rate ratios) prodotto:

[215]:
```
%R exp(summary(glmmTMB.off.main)[6]$coefficients$cond[2,1])
```

[215]:
```
array([1.111365])
```

Estremo inferiore intervallo di confidenza 95%:

[216]:
```
%R exp(summary(glmmTMB.off.main)[6]$coefficients$cond[2,1] - 1.
↪96*summary(glmmTMB.off.main)[6]$coefficients$cond[2,2])
```

[216]:
```
array([1.0655919])
```

[217]:
```
%R summary(glmmTMB.off.main)[6]$coefficients$cond[2,1] - 1.
↪96*summary(glmmTMB.off.main)[6]$coefficients$cond[2,2]
```

[217]:
```
array([0.06353042])
```

Estremo superiore intervallo di confidenza 95%:

[218]:
```
%R exp(summary(glmmTMB.off.main)[6]$coefficients$cond[2,1] + 1.
↪96*summary(glmmTMB.off.main)[6]$coefficients$cond[2,2])
```

[218]:
```
array([1.15910431])
```

[219]:
```
%R summary(glmmTMB.off.main)[6]$coefficients$cond[2,1] + 1.
↪96*summary(glmmTMB.off.main)[6]$coefficients$cond[2,2]
```

[219]:
```
array([0.14764756])
```

p-value per PM2.5:

[220]:
```
%R summary(glmmTMB.off.main)[6]$coefficients$cond[2,4]
```

[220]:
```
array([8.62684764e-07])
```

2.2.16 Sedicesima analisi: fixed NB

Lo "stato", per il quale era stata predisposta un'intercetta casuale, viene convertito in un effetto misto, in particolare in una variabile categorica:

[221]:
```
%R glm.nb.off = glm.nb(Deaths ~  mean_pm25 +␣
↪factor(q_popdensity) \
                    + scale(poverty)  +␣
↪scale(log(medianhousevalue)) \
                    + scale(log(medhouseholdincome)) +␣
↪scale(pct_owner_occ) \
                    + scale(education) + scale(pct_blk) +␣
↪scale(hispanic) \
                    + scale(older_pecent) +␣
↪scale(prime_pecent) + scale(mid_pecent) \
                    + scale(date_since_social) +␣
↪scale(date_since) \
                    + scale(beds/population) \
                    + scale(obese) + scale(smoke) \
                    + scale(mean_summer_temp) +␣
↪scale(mean_winter_temp) + scale(mean_summer_rm) +␣
↪scale(mean_winter_rm) \
                    + factor(state) \
                    + offset(log(population)), data =␣
↪aggregate_pm_census_cdc_test_beds) \

%R print(summary(glm.nb.off))
```

```
Call:
```

```
glm.nb(formula = Deaths ~ mean_pm25 + factor(q_popdensity) +
    scale(poverty) + scale(log(medianhousevalue)) +
scale(log(medhouseholdincome)) +
    scale(pct_owner_occ) + scale(education) + scale(pct_blk) +
    scale(hispanic) + scale(older_pecent) + scale(prime_pecent)␣
 ↪+
    scale(mid_pecent) + scale(date_since_social) +␣
 ↪scale(date_since) +
    scale(beds/population) + scale(obese) + scale(smoke) +
scale(mean_summer_temp) +
    scale(mean_winter_temp) + scale(mean_summer_rm) +␣
 ↪scale(mean_winter_rm) +
    factor(state) + offset(log(population)), data =
aggregate_pm_census_cdc_test_beds,
    init.theta = 1.646882449, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9441  -0.9541  -0.4339   0.2584   5.1130

Coefficients: (1 not defined because of singularities)
                                Estimate Std. Error z value␣
 ↪Pr(>|z|)
(Intercept)                   -1.027e+01  6.646e-01 -15.453  <␣
 ↪2e-16 ***
mean_pm25                      1.008e-01  2.187e-02   4.610 4.
 ↪02e-06 ***
factor(q_popdensity)2         -1.851e-02  8.572e-02  -0.216 0.
 ↪829063
factor(q_popdensity)3          8.819e-04  8.973e-02   0.010 0.
 ↪992159
factor(q_popdensity)4         -6.943e-02  9.465e-02  -0.733 0.
 ↪463276
factor(q_popdensity)5          7.695e-02  1.091e-01   0.705 0.
 ↪480573
scale(poverty)                 4.671e-02  2.582e-02   1.809 0.
 ↪070432 .
scale(log(medianhousevalue))   1.421e-01  4.449e-02   3.193 0.
 ↪001407 **
```

```
scale(log(medhouseholdincome))  1.121e-01  4.570e-02   2.453 0.
 ↪014174 *
scale(pct_owner_occ)            6.288e-02  2.703e-02   2.326 0.
 ↪020020 *
scale(education)                1.506e-01  3.135e-02   4.805 1.
 ↪55e-06 ***
scale(pct_blk)                  2.097e-01  3.024e-02   6.934 4.
 ↪09e-12 ***
scale(hispanic)                 1.528e-01  3.176e-02   4.811 1.
 ↪50e-06 ***
scale(older_pecent)             3.942e-02  4.099e-02   0.962 0.
 ↪336116
scale(prime_pecent)            -1.832e-01  4.720e-02  -3.882 0.
 ↪000104 ***
scale(mid_pecent)              -1.328e-01  3.472e-02  -3.825 0.
 ↪000131 ***
scale(date_since_social)        2.218e+00  6.002e-01   3.696 0.
 ↪000219 ***
scale(date_since)               2.679e-01  4.163e-02   6.435 1.
 ↪23e-10 ***
scale(beds/population)          1.735e-02  2.280e-02   0.761 0.
 ↪446526
scale(obese)                    2.883e-02  2.368e-02   1.218 0.
 ↪223289
scale(smoke)                    2.083e-01  4.283e-02   4.864 1.
 ↪15e-06 ***
scale(mean_summer_temp)         9.752e-02  7.623e-02   1.279 0.
 ↪200810
scale(mean_winter_temp)         2.209e-01  1.008e-01   2.192 0.
 ↪028382 *
scale(mean_summer_rm)          -2.723e-01  8.541e-02  -3.188 0.
 ↪001434 **
scale(mean_winter_rm)           1.647e-02  4.339e-02   0.380 0.
 ↪704229
factor(state)DE                 9.850e-03  9.455e-01   0.010 0.
 ↪991688
factor(state)FL                -2.466e-01  8.056e-01  -0.306 0.
 ↪759532
factor(state)GA                 1.362e-01  7.897e-01   0.172 0.
 ↪863121
```

```
factor(state)IA                5.265e+00  1.029e+00   5.118 3.
 ↪08e-07 ***
factor(state)ID               -7.586e-01  8.841e-01  -0.858 0.
 ↪390848
factor(state)IL               -4.732e-01  8.584e-01  -0.551 0.
 ↪581441
factor(state)IN               -1.889e-01  8.426e-01  -0.224 0.
 ↪822654
factor(state)KS               -9.199e-01  8.191e-01  -1.123 0.
 ↪261408
factor(state)KY                4.024e+00  1.030e+00   3.907 9.
 ↪36e-05 ***
factor(state)LA               -3.300e-02  8.518e-01  -0.039 0.
 ↪969099
factor(state)MA                1.252e+00  8.586e-01   1.458 0.
 ↪144853
factor(state)MD               -9.237e-02  8.153e-01  -0.113 0.
 ↪909797
factor(state)ME               -3.791e-01  8.311e-01  -0.456 0.
 ↪648302
factor(state)MI               -1.386e-02  8.475e-01  -0.016 0.
 ↪986955
factor(state)MN               -1.071e-01  8.349e-01  -0.128 0.
 ↪897906
factor(state)MO               -1.048e+00  7.764e-01  -1.350 0.
 ↪177089
factor(state)MS                2.383e-01  8.088e-01   0.295 0.
 ↪768266
factor(state)MT               -6.431e-01  8.539e-01  -0.753 0.
 ↪451356
factor(state)NC               -5.612e-01  8.083e-01  -0.694 0.
 ↪487503
factor(state)ND                4.836e+00  1.044e+00   4.632 3.
 ↪62e-06 ***
factor(state)NE                4.928e+00  1.028e+00   4.795 1.
 ↪63e-06 ***
factor(state)NH               -1.157e-01  8.651e-01  -0.134 0.
 ↪893647
factor(state)NJ                7.865e-01  8.619e-01   0.912 0.
 ↪361515
```

```
factor(state)NM                -8.118e-01  8.760e-01  -0.927 0.
 ↪354060
factor(state)NV                -1.489e+00  8.964e-01  -1.661 0.
 ↪096754 .
factor(state)NY                 2.418e-01  8.545e-01   0.283 0.
 ↪777168
factor(state)OH                -4.277e-01  8.421e-01  -0.508 0.
 ↪611527
factor(state)OK                 4.042e+00  1.027e+00   3.936 8.
 ↪29e-05 ***
factor(state)OR                -8.848e-01  8.829e-01  -1.002 0.
 ↪316263
factor(state)PA                -1.457e-02  8.001e-01  -0.018 0.
 ↪985468
factor(state)RI                -4.071e+01  2.729e+07   0.000 0.
 ↪999999
factor(state)SC                 1.075e-01  7.780e-01   0.138 0.
 ↪890106
factor(state)SD                 4.629e+00  1.034e+00   4.477 7.
 ↪57e-06 ***
factor(state)TN                -7.436e-01  7.938e-01  -0.937 0.
 ↪348910
factor(state)TX                 4.575e+00  1.028e+00   4.450 8.
 ↪60e-06 ***
factor(state)UT                 3.259e+00  1.051e+00   3.100 0.
 ↪001935 **
factor(state)VA                -5.324e-01  8.024e-01  -0.664 0.
 ↪506991
factor(state)VT                -8.126e-01  8.832e-01  -0.920 0.
 ↪357530
factor(state)WA                -5.530e-01  8.831e-01  -0.626 0.
 ↪531155
factor(state)WI                -6.675e-01  8.428e-01  -0.792 0.
 ↪428330
factor(state)WV                -1.428e+00  8.488e-01  -1.682 0.
 ↪092488 .
factor(state)WY                        NA         NA      NA
 ↪    NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for Negative Binomial(1.6469) family␣
↪taken to be 1)

    Null deviance: 5435.9  on 2801  degrees of freedom
Residual deviance: 3033.0  on 2736  degrees of freedom
  (287 observations deleted due to missingness)
AIC: 16763

Number of Fisher Scoring iterations: 1


              Theta:  1.6469
          Std. Err.:  0.0626
Warning while fitting theta: alternation limit reached

 2 x log-likelihood:  -16628.8760
```

MRR (Mortality rate ratios) prodotto:

[222]:
```
%R exp(summary(glm.nb.off)$coefficients[2,1])
```

[222]:
```
array([1.10607473])
```

Estremo inferiore intervallo di confidenza 95%:

[223]:
```
%R exp(summary(glm.nb.off)$coefficients[2,1] - 1.
↪96*summary(glm.nb.off)$coefficients[2,2])
```

[223]:
```
array([1.05967072])
```

[224]:
```
%R summary(glm.nb.off)$coefficients[2,1] - 1.96*summary(glm.nb.
↪off)$coefficients[2,2]
```

[224]:
```
array([0.05795822])
```

Estremo superiore intervallo di confidenza 95%:

[225]:
```
%R exp(summary(glm.nb.off)$coefficients[2,1] + 1.
↪96*summary(glm.nb.off)$coefficients[2,2])
```

[225]:
```
array([1.15451082])
```

[226]:
```
%R summary(glm.nb.off)$coefficients[2,1] + 1.96*summary(glm.nb.
↪off)$coefficients[2,2]
```

[226]:
```
array([0.14367672])
```

p-value per PM2.5:

[227]:
```
%R summary(glm.nb.off)$coefficients[2,4]
```

[227]:
```
array([4.01716378e-06])
```

2.3 FIGURE

In questa sezione vengono mostrati i risultati ottenuti dalle analisi precedenti:

[228]:
```
tab = pd.read_csv('https://bitbucket.org/christianderrico/
↪file_condivisi/raw/ba560ce38a449801a9ac7a195c10e2b832d12ca6/
↪Tabella_sottoposta_ad_analisi.csv')
```

[229]:
```
tab.drop(columns='Unnamed: 0', inplace=True)
```

[230]:
```
tab['fips'] = ['0' + str(int(f)) if f < 10_000 else␣
↪str(int(f)) for f in tab['fips']]
```

[231]:
```
df = pd.read_csv("https://raw.githubusercontent.com/plotly/
↪datasets/master/fips-unemp-16.csv",
                 dtype={"fips": str})
```

[232]:
```
table = tab.merge(df)
```

[233]:
```
table = table.drop(columns='unemp')
```

[234]:
```
table['mortality'] = table['Deaths'] / table['population'] *␣
↪np.power(10, 6)
table['logmortality'] = np.log10(table['Deaths'] /␣
↪table['population'] * np.power(10, 6))
```

[235]:
```
table['logmortality'][table['logmortality'] < 0] = -1
```

[236]:
```
table['logmortality'][table['logmortality'].isna()] = -1
```

[238]:
```
import plotly.express as px

def plot_map(counties, table, column):
  fig = px.choropleth_mapbox(table, geojson=counties,␣
 ↪locations='fips', color=column,
                            color_continuous_scale=["#1e90ff",␣
 ↪"#ffffba", "#8b0000"],
                            range_color=(0, 12),
                            mapbox_style="carto-positron",
                            zoom=3, center = {"lat": 37.0902,␣
 ↪"lon": -95.7129},
                            opacity=0.5,
                           )
  fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
  fig.show()
```

2.3.1 Mappa 1: Morti da Coronavirus su una popolazione di un milione di abitanti

La prima mappa ci offre una fotografia degli Stati Uniti colpiti dal Coronavirus: i decessi di ogni contea sono rapportati ad una popolazione standard di un milione di abitanti.

[239]:
```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/
 ↪datasets/master/geojson-counties-fips.json') as response:
    counties = json.load(response)

plot_map(counties, table, 'logmortality')
```

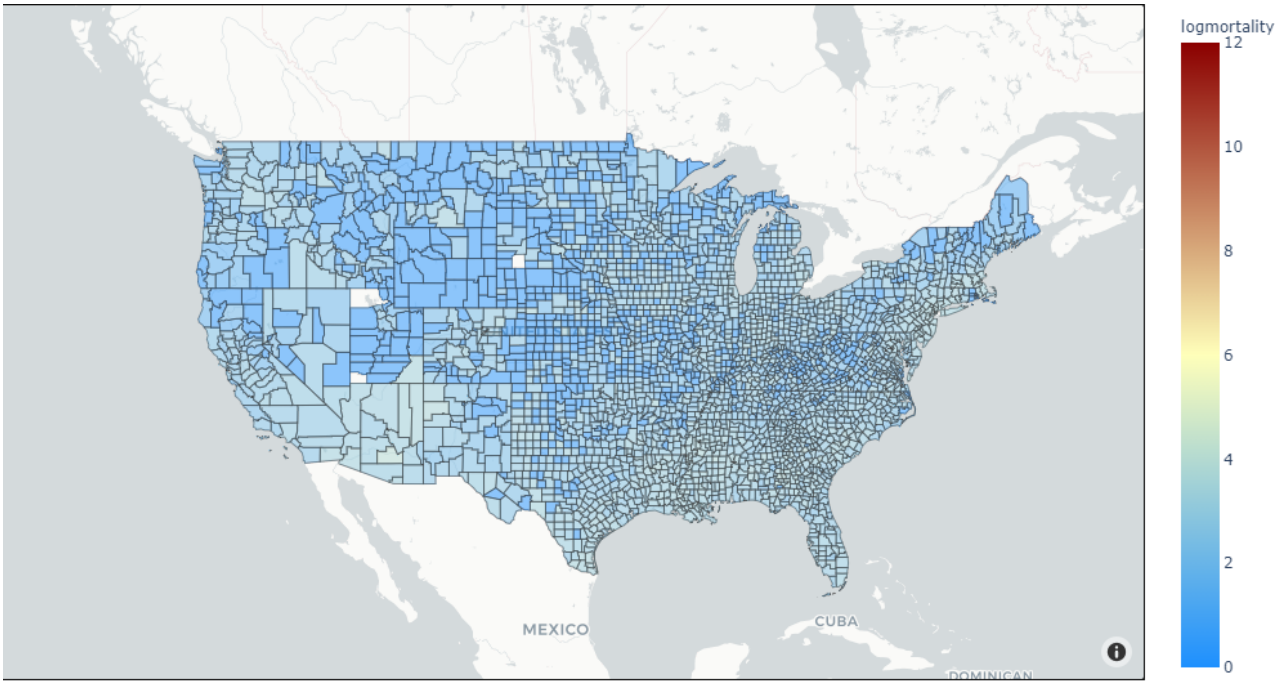
[240]:
```
county_pm = pd.read_csv('https://raw.githubusercontent.com/
 ↪wxwx1993/PM_COVID/master/Data/county_pm25.csv')
```

[241]:
```
county_pm_aggregated = county_pm.groupby(by='fips',␣
 ↪as_index=False).agg({'pm25':'mean'}).rename(columns={'pm25':
 ↪'mean_pm25'})
```

[242]:
```

```

```
county_pm_aggregated['fips'] = ['0' + str(int(f)) if f <␣
↪10_000 else str(int(f)) for f in␣
↪county_pm_aggregated['fips']]
```

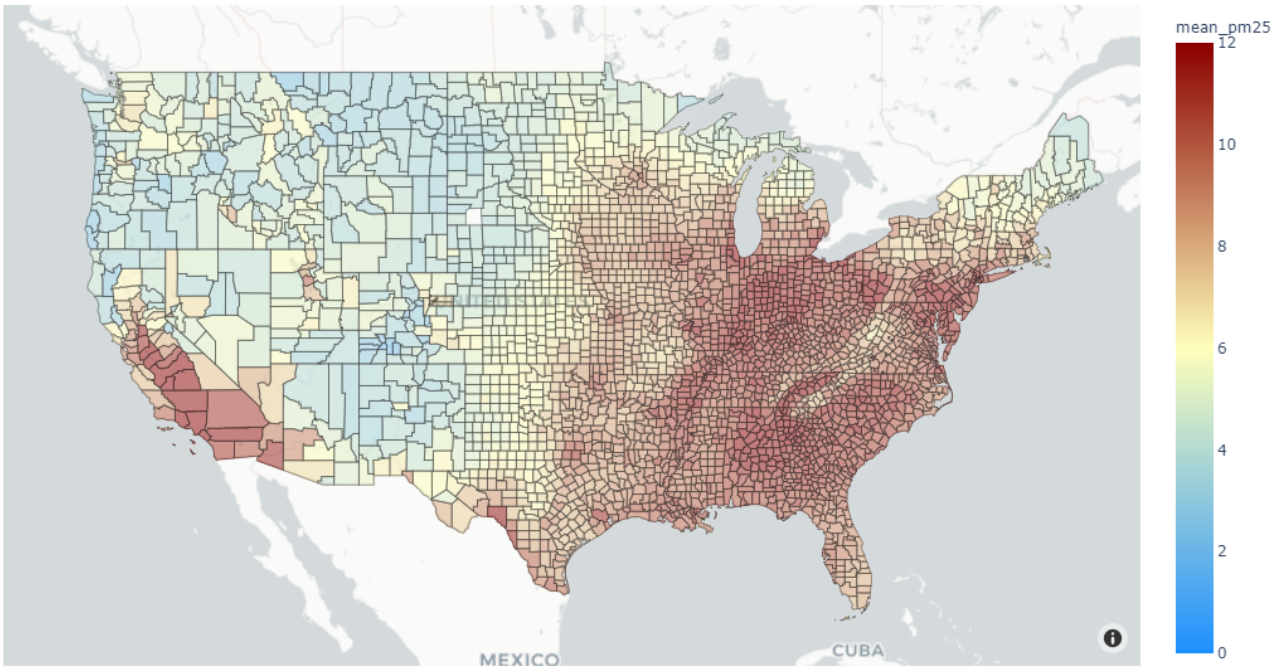
[243]:
```
statesPM = county_pm_aggregated.merge(df)
```



2.3.2 Mappa 2: Concentrazioni medie di PM 2.5 per il periodo 2000 - 2016 negl'US (in mg/m^3)

La seconda mappa mostra le contee USA per concentrazioni medie di PM2.5: risulta pertanto evidente come quelle che hanno registrato più decessi da Covid-19 siano le anche le più inquinate.

[244]:
```
plot_map(counties, statesPM, 'mean_pm25')
```

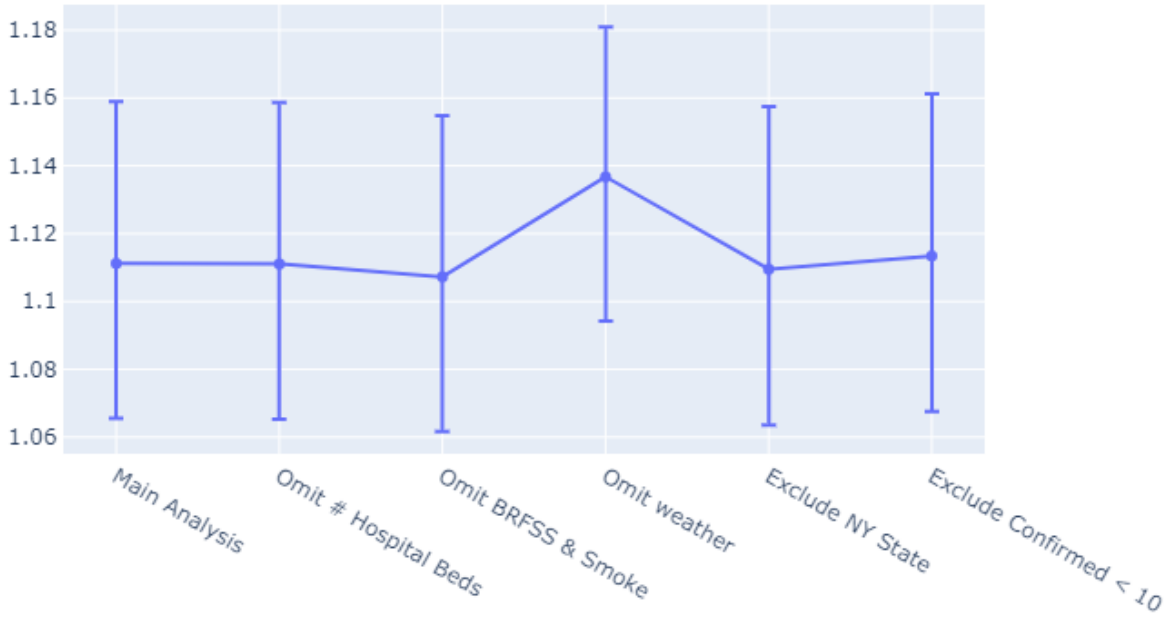


2.3.3 Risultati analisi effettuate

[248]:
```
df = pd.DataFrame({'Method' : ['Main Analysis',
                               'Omit # Hospital Beds',
                               'Omit BRFSS & Smoke',
                               'Omit weather',
                               'Exclude NY State',
                               'Exclude Confirmed < 10'],
                   'RR' : [1.11122866, 1.11104688, 1.10728959,␣
 ↪1.1367168, 1.10950945, 1.1134269],
                   'Lower_CI': [1.06549511, 1.06532409, 1.
 ↪06172142, 1.0942174, 1.06350915, 1.0675506],
                   'Upper_CI': [1.15892521, 1.15873205, 1.
 ↪1548135, 1.18086688, 1.15749942, 1.16127466],
                   'Methods' : [i for i in range(1, 7)]
})
```

[44]:
```
import plotly.express as px
import plotly.graph_objects as go
fig = go.Figure(data=go.Scatter(
            x=df['Method'],
            y=df["RR"],
```

```
            error_y=dict(
                type='data',
                symmetric=False,
                array=df['Upper_CI'] - df['RR'],
                arrayminus=df['RR'] - df['Lower_CI'])
        ))

fig.show()
```



Risultati: è stato scoperto che un aumento di solo 1 mg/m^3 di PM 2.5 è associato ad un aumento dell'11% del tasso di mortalità da COVID-19 (con intervallo di confidenza al 95% [CI]: 6%, 15%). I risultati sono statisticamente significativi e robusti, grazie alle analisi secondarie e di sensibilità.

Conclusioni: un piccolo aumento dell'esposizione a lungo termine al PM 2.5 porta a un grande aumento del tasso di mortalità COVID-19. I risultati sottolineano l'importanza di continuare a far rispettare le normative esistenti sull'inquinamento atmosferico per proteggere la salute umana sia durante che dopo la crisi del COVID-19.

Capitolo 3

Covid-19 e inquinamento atmosferico in Italia

In questo capitolo, si descrivono le ricerche effettuate sulla penisola italiana: l'obiettivo è quello di confermare quanto già ampiamente documentato dagli studi americani e di trovare ulteriori legami fra pandemia e inquinamento.

[1]:
```
import warnings
warnings.filterwarnings("ignore")
```

3.0.1 Lettura dati su Covid-19 della protezione civile

- I dati raccolti per l'analisi provengono dal repository ufficiale della Protezione civile italiana, che dal 24 Febbraio 2020 sta raccogliendo i dati epidemiologici sull'andamento della pandemia da Covid-19.
- **Fonte**: Protezione civile

[2]:
```
import pandas as pd
import datetime
```

[3]:
```
data_inizio = "2020-02-24"
today = datetime.date.today()
yesterday = today - datetime.timedelta(days=1)
```

[4]:
```
dates = [str(v)[:-9].split('-')[0] + str(v)[:-9].split('-')[1]␣
 ↪+ str(v)[:-9].split('-')[2] for v in pd.
 ↪date_range(start=str(data_inizio),end=str(yesterday)).
 ↪to_list()]
```

3.0.2 Dati sulle regioni

La tabella *Covid_regioni* raccoglie i dati regionali:

- **data**: data a cui fanno riferimento le rilevazioni
- **stato**: stato italiano per cui sono prodotti questi dati
- **codice_regione**: codice assegnato alla regione
- **denominazine_regione**: nome della regione
- **codice_provincia**: codice assegnato alla provincia
- **denominazione_provincia**: nome della provincia
- **sigla_provincia**: sigla della provincia
- **lat**: latitudine della provincia osservata
- **long**: longitudine della provincia osservata
- **ricoverati_con_sintomi**: individui sintomatici ricoverati
- **terapia_intensiva**: individui ricoverati in terapia intensiva
- **totale_ospedalizzati**: esito della somma fra "ricoverati_con_sintomi" e "terapia_intensiva"
- **isolamento_domiciliare**: individui positivi sottoposti a cure domiciliari
- **totale_positivi**: esito della somma fra "totale_ospedalizzati" e "isolamento_domiciliare"
- **variazione_totale_positivi**: variazione nel numero dei positivi rispetto al giorno precedente
- **nuovi_positivi**: nuovi casi di positività riscontrati
- **dimessi_guariti**: pazienti dimessi dagli ospedali (ma non necessariamente guariti) sia i casi di accertata guarigione, tramite diagnosi in ospedale o tramite esito dei tamponi con i test di laboratorio.
- **deceduti**: deceduti risultati positivi al coronavirus; il dato viene poi certificato in un secondo momento dall'Istituto Superiore di Sanità.
- **casi_da_sospetto_diagnostico** = casi positivi al tampone emersi da attività clinica.

- **casi_da_screening** = casi di positività emersi da indagini e test, pianificati a livello nazionale o regionale.
- **tamponi** = numero di tamponi effettuati
- **casi testati** = singole persone sottoposte a tampone.
- **totale_casi**: totale dei casi di positività al COVID-19 registrati
- **note**: annotazioni in caso di anomalie nelle rilevazioni

[5]:
```
Covid_regioni = pd.DataFrame()
for d in dates:
  Covid_regioni = Covid_regioni.append(pd.read_csv('https://
 ↪raw.githubusercontent.com/pcm-dpc/COVID-19/master/
 ↪dati-regioni/dpc-covid19-ita-regioni-' + d + '.csv',␣
 ↪parse_dates=['data']))
Covid_regioni = Covid_regioni.reset_index().
 ↪drop(columns='index')
```

3.0.3 Dati sulle province

La tabella *Covid_province* raccoglie i dati provinciali:

- **data**: data a cui fanno riferimento le rilevazioni
- **stato**: stato italiano per cui sono prodotti questi dati
- **codice_regione**: codice assegnato alla regione
- **denominazione_regione**: nome della regione
- **codice_provincia**: codice assegnato alla provincia
- **denominazione_provincia**: nome della provincia
- **sigla_provincia**: sigla della provincia
- **lat**: latitudine della provincia osservata
- **long**: longitudine della provincia osservata
- **totale_casi**: totale dei casi di positività al COVID-19
- **note**: annotazioni in caso di anomalie nelle rilevazioni

[6]:
```

```

```
Covid_province = pd.read_csv("https://raw.githubusercontent.
↪com/pcm-dpc/COVID-19/master/dati-province/
↪dpc-covid19-ita-province.csv", keep_default_na=False,␣
↪na_values=[''], parse_dates=['data'])
```

- La funzione *plot_storical_averages()* mostra graficamente la serie temporale che le viene proposta

[7]:
```
# Using plotly.express
def plot_storical_averages(x, y, df, title, multi=False):
  if (multi == True):
    fig = px.line(df, x=x, y=y, color="Reg",
                  line_group="Reg", hover_name="Reg", title=title)
  else:
    fig = px.line(df, x=x, y=y, title=title, log_y=True)
  fig.show()
```

[10]:
```
Covid_regioni.drop(columns=['lat', 'long', 'note'],␣
↪inplace=True)
Covid_province.drop(columns=['lat', 'long', 'note'],␣
↪inplace=True)

Covid_province.rename(columns={'totale_casi':
↪'totale_casi_per_provincia'}, inplace=True)

columns = Covid_regioni.columns[0:4].to_list()
columns.extend([val + ('_per_regione') for val in␣
↪Covid_regioni.columns[4:]])
Covid_regioni.columns = columns
```

[11]:
```
Covid_province['data'] = [pd.to_datetime(str(v)[:10]) for v in␣
↪Covid_province['data']]
```

N.B.: la regione Trentino Alto-Adige è stata, per semplicità di trattazione, adeguata alle altre regioni, pur essendo una regione a statuto speciale con due province autonome.

[12]:
```
trentino = Covid_regioni.
↪loc[(Covid_regioni['denominazione_regione'] == 'P.A.␣
↪Bolzano') | (Covid_regioni['denominazione_regione'] == 'P.A.
↪ Trento')]
```

[13]:
```
trenti = {"data":"max",
          "stato":"max",
          "codice_regione":"max",
          "denominazione_regione":"max",
          "ricoverati_con_sintomi_per_regione":"sum",
          "terapia_intensiva_per_regione":"sum",
          "totale_ospedalizzati_per_regione":"sum",
          "isolamento_domiciliare_per_regione":"sum",
          "totale_positivi_per_regione":"sum",
          "variazione_totale_positivi_per_regione":"sum",
          "nuovi_positivi_per_regione":"sum",
          "dimessi_guariti_per_regione":"sum",
          "deceduti_per_regione":"sum",
          "totale_casi_per_regione":"sum",
          "tamponi_per_regione":"sum",
          }
```

[14]:
```
trentino = trentino.groupby(by='data',as_index=False).
 ↪agg(trenti).replace('P.A. Trento', 'Trentino Alto Adige')
cod_regione = trentino['codice_regione'].unique()[0]
```

[15]:
```
Covid_regioni = Covid_regioni.
 ↪loc[~Covid_regioni['denominazione_regione'].isin(['P.A.␣
 ↪Trento', 'P.A. Bolzano'])].append(trentino)
```

[16]:
```
Covid_province['denominazione_regione'] =␣
 ↪Covid_province['denominazione_regione'].
 ↪replace(to_replace=['P.A. Bolzano', 'P.A. Trento'],␣
 ↪value='Trentino Alto Adige')
Covid_province['codice_regione'] =␣
 ↪Covid_province['codice_regione'].replace(to_replace=[21,␣
 ↪22], value=cod_regione)
```

3.0.4 Dati sull' inquinamento

- La tabella *Ita_ inquinamento* contiene i dati sulle rilevazioni provinciali di PM2.5 e PM10 per l'anno 2020.
- **Fonte**: Arpae regionali

[17]:
```
Ita_inquinamento = pd.read_csv("https://bitbucket.org/
↪christianderrico/file_condivisi/raw/
↪6f0bd3d9d8eebb11e146dd7bae68a1386709f2fb/dati_inquinamento/
↪Italia_finale.csv", keep_default_na=False, na_values='').
↪drop(columns='Unnamed: 0')
```

[18]:
```
Ita_inquinamento['data'] = pd.
↪to_datetime(Ita_inquinamento['data'])
```

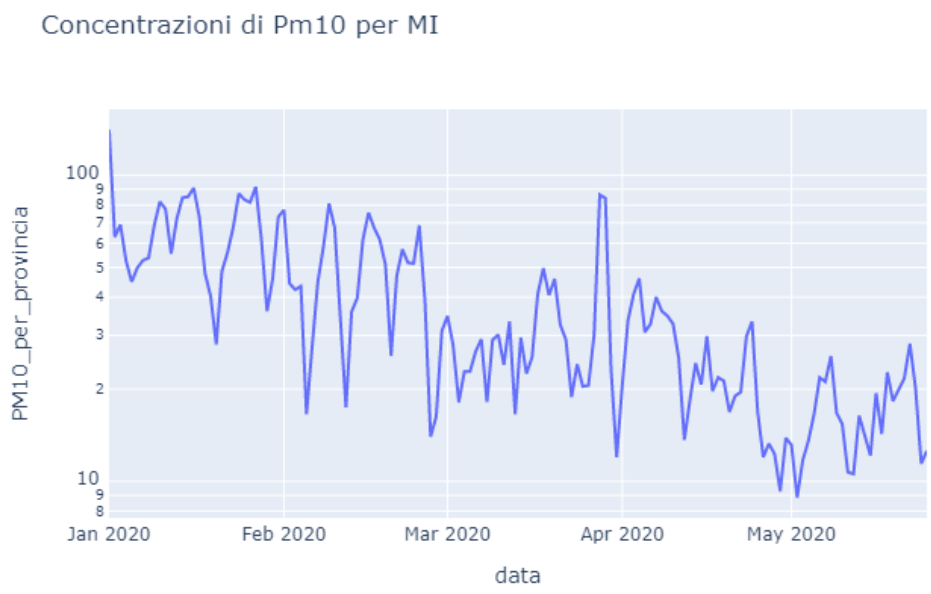
[19]:
```
Ita_inquinamento = Ita_inquinamento.rename(columns={'PM10':
↪'PM10_per_provincia', 'PM2.5':'PM2.5_per_provincia'})
```

[20]:
```
Ita_inquinamento['sigla_provincia'].
↪loc[Ita_inquinamento['sigla_provincia'].isna()] = 'NA'
```

Inserendo la sigla di una provincia è possibile visualizzarne i livelli giornalieri di PM 10 rilevati dalle centraline ARPAE presenti sul territorio:

[22]:
```
import plotly.express as px
sig = input("Inserisci provincia: ")
title = "Concentrazioni di Pm10 per " + sig
test = Ita_inquinamento[Ita_inquinamento['sigla_provincia'] ==␣
↪sig]
if len(test) > 0 & len(test[test['PM10_per_provincia'].
↪isna()]) < (len(test) / 2):
  plot_storical_averages(test['data'],␣
↪test['PM10_per_provincia'], test, title=title)
else:
  print("Rilevazioni non presenti o non sufficienti")
```

```
Inserisci provincia: MI
```



- La funzione *generate_dataframe()* crea il dataset per l'analisi, contenente dati su inquinamento e Covid
- La funzione *draw_correlation_heat_map()* crea la matrice con ivalori delle correlazioni fra le variabili
- La funzione *create_plot()* crea uno scatter plot con le caratteristiche passate in input
- La funzione *draw_scatter_plot()* disegna tutti gli scatter plot passati in input

[23]:
```
def generate_dataframe(start_data, end_data):
    Ita_inquinamento_1 =␣
↪Ita_inquinamento[(Ita_inquinamento['data'] >= start_data) &␣
↪(Ita_inquinamento['data'] <= end_data)]
    Covid_province_1 = Covid_province[(Covid_province['data']␣
↪>= start_data)
                                      &␣
↪(Covid_province['data'] <= end_data)
                                      &␣
↪(Covid_province['denominazione_provincia'] != 'In fase di␣
↪definizione/aggiornamento')]
```

```
    Covid_province_1['data'] = [pd.to_datetime(str(s).split(" ")[0]) for s in Covid_province_1['data']]
    Covid_prov_inq = Covid_province_1.merge(Ita_inquinamento_1, on=['data', 'sigla_provincia'], how='left')
    return Covid_prov_inq
```

[24]:
```
def draw_correlation_heat_map(df):
    correlation = df.corr()
    #display(correlation.style.background_gradient(cmap='coolwarm').set_precision(2))
    heat = go.Heatmap(z=correlation,
                      x=correlation.columns,
                      y=correlation.columns,
                      xgap=1, ygap=1,
                      colorbar_thickness=20,
                      colorbar_ticklen=3,
                     )
    title = 'Correlation Matrix'
    layout = go.Layout(title_text=title, title_x=0.5,
                       width=600, height=600,
                       xaxis_showgrid=False,
                       yaxis_showgrid=False,
                       yaxis_autorange='reversed')

    fig=go.Figure(data=[heat], layout=layout)
    fig.show()
```

[25]:
```
def create_plot(df, col_1, col_2, temp=None, title=None, mode= 'markers', add = True, line = dict(color="#2271b3")):
    if temp not in df.columns:
        hover = temp
    else:
        hover = [str(v) for v in df[temp]]

    plot = go.Scatter(x = df[col_1],
                      y = df[col_2],
                      mode=mode,
                      hoverlabel = dict(font=dict(color='white')),
                      showlegend = True,
```

```
                    line=line,
                    hovertemplate=hover)
    plot_info = {'plot' : plot,
                 'adding_info': add,
                 'x_axis' : col_1,
                'y_axis': col_2,
                'title': title}
    return plot_info
```

[26]:
```
import plotly.graph_objects as go

def draw_scatter_plot(plot_list):
    fig = go.Figure()
    for v in plot_list:
        fig.add_trace(v['plot'])
        if v['adding_info'] == True:
            fig.update_layout(
                title = v['title'],
                xaxis_title=v['x_axis'],
                yaxis_title=v['y_axis'],
                font=dict(
                    family="Times New Roman",
                    size=12,
                    color="black"
                ))
    fig.show()
```

- La funzione *create_model()* crea il modello che verrà utilizzato per l'analisi
- La funzione *split_datas()* divide i dati nei set utilizzati per la fasi di addestramento e validazione
- La funzione *mean_absolute_percentage_error()* calcola il MAPE fra i valori predetti e i valori reali del modello.

N.B.: per ovviare al problema della divisione per 0, (dato che alcuni valori per le y reali valgono 0 e la letteratura vuole che i valori reali si trovino al di sotto della linea di frazione), si è assunto che il denominatore del rapporto sia costituito dal valore predetto invece che dal valore reale.

[27]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from regressors import stats

def create_model(model_dict):
    prm = Pipeline(model_dict)
    return prm
```

[28]:
```
def split_datas(X, y):
    X_train, X_val, y_train, y_val = train_test_split(X, y,␣
 ↪test_size=0.33, random_state=42)
    return X_train, X_val, y_train, y_val
```

[29]:
```
import numpy as np
def mean_absolute_percentage_error(y_true, y_pred):
    # i valori = 0 si assumono 1
    #y_true = y_true.where(y_true > 0, 1)
    return np.mean(np.abs((y_true - y_pred) / y_pred)) * 100
```

Il modello utilizzato ha queste caratteristiche:

- si tratta di un modello **lineare** di regressione.
- i dati preliminarmente passano attraverso un filtro **Standard Scaler**, il cui fine è la standardizzare delle features, con la rimozione della media e ridimensionamento della varianza unitaria.

La standardizzazione di un set di dati è un requisito comune per molti stimatori di machine learning: i modelli potrebbero comportarsi male se le singole caratteristiche non assomigliano più o meno a dati standard normalmente distribuiti (es. Gaussiana con media 0 e varianza unitaria). Inoltre, avere dati su scale di grandezza diverse fra loro compromette il lavoro degl'algoritmi e comporta un livello di accuratezza inferiore

[30]:
```
scaler = StandardScaler()
lean_reg = LinearRegression()
```

- La funzione *define_ datas()* seleziona dal dataset in ingresso le variabili che interverranno nelle analisi

- La funzione *get_ statistics()* effettua l'addestramento del modello e la validazione, mostrando poi a video le statistiche legate alle metriche di valutazione del modello
- La funzione *plot_ paramethers()* mostra a video i parametri individuati dal modello
- La funzione *calculate_ line()* rappresenta la linea che cerca di approssimare i dati selezionati
- la funzione *plot_ p_ values()* restituisce i p-value per le variabili utilizzate nell'addestramento del modello

[31]:
```
def define_datas(modelling, outcome, income, single=False):
    y = modelling[outcome]
    if single == False:
        X = modelling.loc[:, (income)]
    else:
        X = modelling[[income]]
    return X, y
```

[32]:
```
def get_statistics(model, X_train, y_train, X_val, y_val):
    model.fit(X_train, y_train)
    display("Model accuracy (R^2): " + str(round(model.
 ↪score(X_val, y_val),2)))
    display("Mean Squared Error: " +␣
 ↪str(round(mean_squared_error(y_val, model.predict(X_val)),␣
 ↪1)))
    display("Mean Absolute Percentage Error: " +␣
 ↪str(round(mean_absolute_percentage_error(y_val, model.
 ↪predict(X_val)), 1)))
    legend = "MSE: " + str(round(mean_squared_error(y_val,␣
 ↪model.predict(X_val)), 1)) \
             + " MAPE: " +␣
 ↪str(round(mean_absolute_percentage_error(y_val, model.
 ↪predict(X_val)), 1))
    return legend
```

[33]:
```
from regressors import stats

def plot_p_values(model, X_train, y_train):
    columns = ['Intercept']
```

```
    for v in X_train.columns:
        columns.append(v)
    display(pd.DataFrame(stats.coef_pval(model, X_train,␣
↪y_train), index=columns))
```

[34]:
```
def plot_paramethers(model, X):
    b = pd.DataFrame(model.coef_[0:len(X.columns)], X.columns)
    return b
```

[35]:
```
def calculate_line(model, X, df_or, income, outcome, temp,␣
↪line = True):
    b = plot_paramethers(model, X).loc[income, 0]
    y = df_or[income] * b
    if line == False:
        y = model.predict(X)
    df = pd.DataFrame({income: X[income], outcome: y})
    draw_scatter_plot([create_plot(df_or, income, outcome,␣
↪temp = temp, title = titolo_primo_grafico),
                       create_plot(df, income, outcome, temp =␣
↪legend, mode='lines', line = dict(color="#cb3234"))])
```

3.0.5 Analisi: diffusione del Covid-19 in relazione ai giorni di superamento dei limiti

Cercare una correlazione fra diffusione del nuovo coronavirus e l'inquinamento ci porta a considerare una possibile relazione tra i superamenti dei limiti di legge delle concentrazioni di PM10 registrati nel periodo 10 Febbraio (data che precede di 14 gg il 24 Febbraio 2020, da cui hanno inizio le rilevazioni della protezione civile, considerando un ritardo temporale intermedio relativo a due settimane, approssimativamente pari al tempo di incubazione del virus fino alla identificazione della infezione contratta) - 10 Marzo (giorno di inzio della quarantena) e il numero di casi infetti da COVID-19 aggiornati al 24 Marzo (giorno scelto con la stessa considerazione fatta per il 10 Febbraio).

Questi i limiti di legge giornalieri per gli inquinanti:

- 50 mg/m^3 per il PM10.
- 25 mg/m^3 per il PM2.5

N.B: Il limite giornaliero per il PM2.5 non è ufficialmente adottato come soglia limite di legge, ma viene raccomandato dall'OMS (Organizzazione modiale della sanità)

[36]:
```
limite_giornaliero_PM10 = 50
limite_giornaliero_PM2_5 = 25
```

Isoliamo i casi di positività nel periodo che arriva fino al 31 Marzo per avere una copertura dei giorni che realmente ci interessano nell'analisi:

[37]:
```
analisi_1 = generate_dataframe('2020-02-24', '2020-03-31')
```

[38]:
```
periodo_osservato = Ita_inquinamento[(Ita_inquinamento['data']␣
↪>= '2020-02-10') & (Ita_inquinamento['data'] <=␣
↪'2020-03-10')]
```

Per il periodo osservato sono 99 le province considerate: per le assenti non è stato possibile reperire le misurazioni, data la mancanza di centraline o di servizi di distribuzione dei dati adeguati.

[39]:
```
periodo_osservato =␣
↪periodo_osservato[(~periodo_osservato['PM10_per_provincia'].
↪isna()) | (~periodo_osservato['PM2.5_per_provincia'].
↪isna())]
province = [p for p in periodo_osservato['sigla_provincia'].
↪unique()]
```

Vengono calcolati i superamenti dei limiti per i due agenti inquinanti:

[40]:
```
superamenti =␣
↪periodo_osservato[(periodo_osservato['PM10_per_provincia']␣
↪> limite_giornaliero_PM10) | (periodo_osservato['PM2.
↪5_per_provincia'] >␣
↪limite_giornaliero_PM2_5)]['sigla_provincia'].
↪value_counts().reset_index().
↪rename(columns={'sigla_provincia':'numero_superamenti',␣
↪'index':'sigla_provincia'})
```

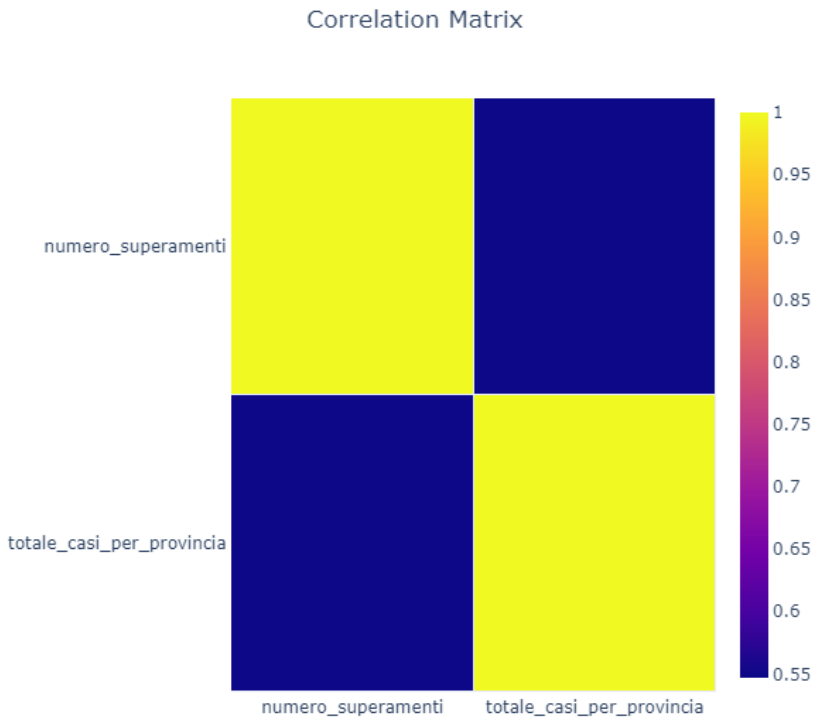
[41]:
```
superamenti_totale = pd.DataFrame(analisi_1['sigla_provincia'].
↪unique()).rename(columns={0:'sigla_provincia'}).
↪merge(superamenti, how='left')
```

```
superamenti_totale =␣
 ↪superamenti_totale[superamenti_totale['sigla_provincia'].
 ↪isin(province)]
superamenti_totale['numero_superamenti']
[superamenti_totale['numero_superamenti'].isna()] = 0
superamenti_totale['numero_superamenti'] = [int(v) for v in␣
 ↪superamenti_totale['numero_superamenti']]
```

[42]:
```
superamenti_totale = superamenti_totale.
 ↪merge(analisi_1[analisi_1['data'] == '2020-03-24'].loc[:,␣
 ↪['sigla_provincia', 'totale_casi_per_provincia']])
```

Di seguito mostrata le correlazione fra superamenti e totale dei casi di positività:

[43]:
```
draw_correlation_heat_map(superamenti_totale)
```



- La funzione *quantile_ divison()* divide in 5 quantili la serie di dati in input.

- In statistica il quantile di ordine o -quantili (con a un numero reale nell'intervallo [0,1]) è un valore qa che divide la popolazione in due parti, proporzionali ad a e (1-a) e caratterizzate da valori rispettivamente minori e maggiori di qa. Per poter calcolare un quantile di ordine è necessario che il carattere sia almeno ordinato, cioè sia possibile definire un ordinamento sulle modalità.

- La funzione *categorize_var()* opera una "categorizzazione" di una variabile in input sulla base della divisione in quantili operata da *quantile_division()*

[44]:
```
def quantile_division(values):
    return np.quantile(values, [0.2, 0.4, 0.6, 0.8])
```

[45]:
```
def categorize_var(df, old_column, new_column):
    quantiles = quantile_division(df[old_column])

    df[new_column] = 1
    df[new_column].loc[df[old_column] <= quantiles[0]] = 1
    df[new_column].loc[(df[old_column] > quantiles[0])
                                                    &␣
 ↪(df[old_column] <= quantiles[1])] = 2
    df[new_column].loc[(df[old_column] > quantiles[1])
                                                    &␣
 ↪(df[old_column] <= quantiles[2])] = 3
    df[new_column].loc[(df[old_column] > quantiles[2])
                                                    &␣
 ↪(df[old_column] <= quantiles[3])] = 4
    df[new_column].loc[df[old_column] > quantiles[3]] = 5

    return df
```

Sulla base del numero di casi per provincia di positività al Covid, le province vengono suddivise in 5 fasce, post divisione in quantili del valore. Queste le soglie decretate:

[46]:
```
cases_quantile = quantile_division(
superamenti_totale['totale_casi_per_provincia'])
cases_quantile
```

[46]:
```
array([  91.8,  192.2,  368.8, 1042.6])
```

Pertanto, le province vengono raggruppate in questa maniera:

- alla prima fascia appartengono le province con meno di 92 casi.
- alla seconda le province con un numero di casi compreso fra i 92 e i 192 casi.
- alla terza le province con un numero di casi fra i 192 e i 369 casi.
- alla quarta le province fra i 369 e 1043 casi.
- alla quinta le province con più di 1043 casi.

[47]:
```
superamenti_totale = categorize_var(superamenti_totale,␣
↪'totale_casi_per_provincia', 'q_cases')
```

A questo punto per ogni fascia viene calcolata la media dei giorni di superamento:

[48]:
```
superamenti_totale.groupby(by='q_cases', as_index=False).mean()
```

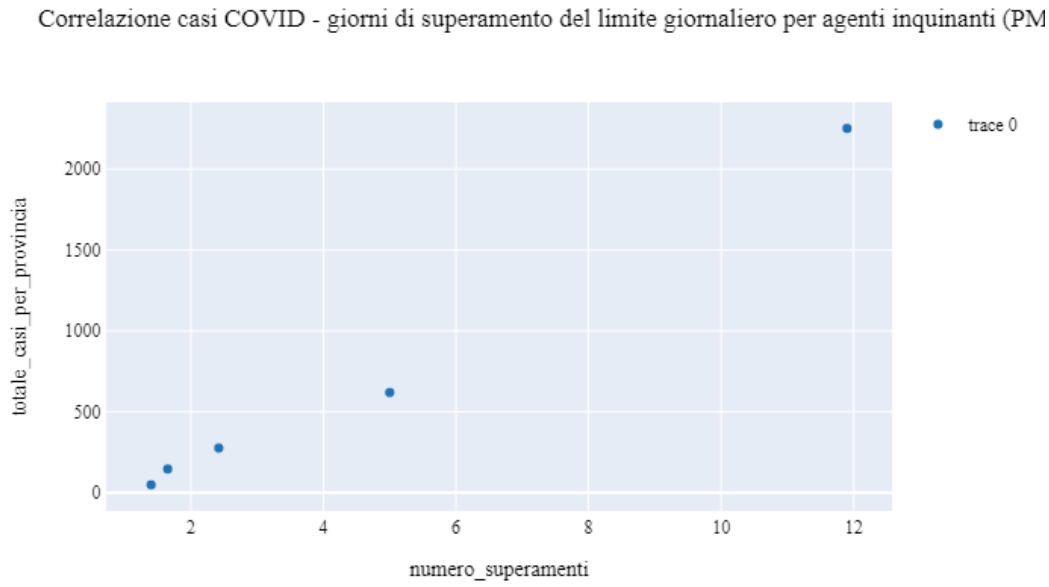
[48]:

	q_cases	numero_superamenti	totale_casi_per_provincia
0	1	1.400000	48.100000
1	2	1.650000	145.900000
2	3	2.421053	275.631579
3	4	5.000000	619.150000
4	5	11.900000	2251.600000

[49]:
```
test = superamenti_totale.groupby(by='q_cases',␣
↪as_index=False).mean()
```

Il grafico mostra la correlazione fra le 2 variabili:

[50]:
```
titolo_quarto_grafico = 'Correlazione casi COVID - giorni di␣
↪superamento del limite giornaliero per agenti inquinanti␣
↪(PM2.5 e PM10)'
draw_scatter_plot([create_plot(test, 'numero_superamenti'␣
↪,'totale_casi_per_provincia', 'q_cases',␣
↪titolo_quarto_grafico)])
```



[51]:
```
cases = test['q_cases']
test = test.drop(columns='q_cases')
test_2 = pd.DataFrame(scaler.fit_transform(test), columns=test.
 ↪columns)
```

Queste le statistiche osservate utilizzando un modello di regressione lineare:

[52]:
```
legend = get_statistics(lean_reg,␣
 ↪test_2[['numero_superamenti']],␣
 ↪test_2['totale_casi_per_provincia'],␣
 ↪test_2[['numero_superamenti']],␣
 ↪test_2['totale_casi_per_provincia'])
```

```
'Model accuracy (R^2): 0.99'


'Mean Squared Error: 0.0'


'Mean Absolute Percentage Error: 33.7'
```

Questi i parametri del modello:

[53]:
```
plot_paramethers(lean_reg, test_2[['numero_superamenti']])
```

[53]:

	0
numero_superamenti	0.995132

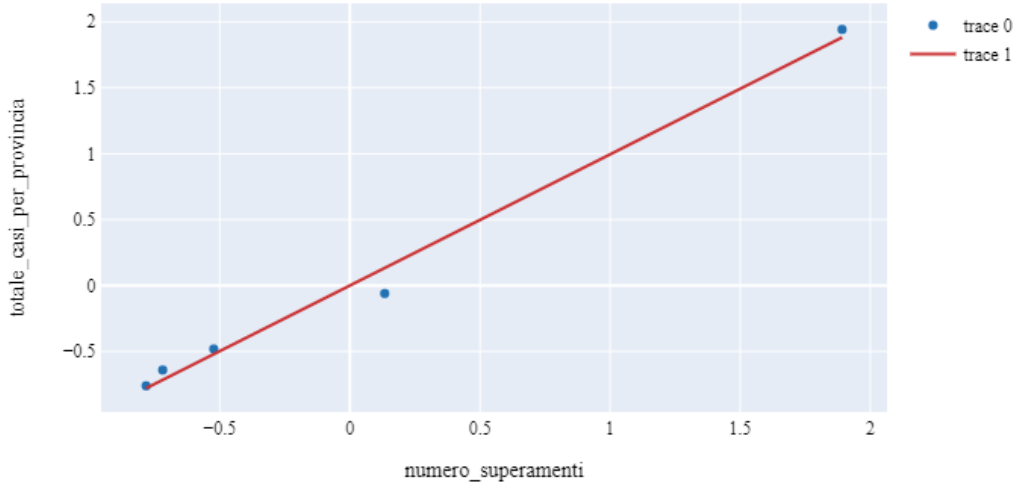
E questi i relativi p-value:

[54]:
```
plot_p_values(lean_reg, test_2[['numero_superamenti']],␣
↪test_2['totale_casi_per_provincia'])
```

	0
Intercept	1.000000
numero_superamenti	0.000023

Il grafico sottostante evidenzia una relazione lineare (R2 = 0,99), a seguito del raggruppamento delle Province in 5 classi sulla base del numero di casi infetti, in relazione ai superamenti del limite delle concentrazioni di PM10 e PM2.5 per ognuna delle 5 classi di Province:

[55]:
```
titolo_primo_grafico = 'Correlazione fasce di positivi -␣
↪numero superamenti'
test_2['q_cases'] = cases
calculate_line(lean_reg, test_2[['numero_superamenti']],␣
↪test_2, 'numero_superamenti', 'totale_casi_per_provincia',␣
↪'q_cases', line=False)
```



[56]:
```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/Neurality/
 ↪Covid19/master/map/GeoJSON/limits_IT_provinces_simple.
 ↪json') as response:
    counties = json.load(response)
```

- La funzione *draw_map()* realizza una mappa che mostra la situazione della penisola italiana in relazione ad una variabile passata in input

[57]:
```
def draw_map(drawing, color_column, label):
    fig = px.choropleth_mapbox(drawing, geojson=counties,␣
 ↪locations='prov_acr', color=color_column,
                           color_continuous_scale="Inferno",
                           featureidkey='properties.prov_acr',
                           mapbox_style="carto-positron",
                           zoom=4.6, center = {"lat": 41.
 ↪71959, "lon": 11.294761},
                           opacity=0.5,
                           labels={color_column:label}
                          )
    fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
```

```
    fig.show()
```

[58]:
```
test['q_cases'] = [i for i in range(1, 6)]
```

[59]:
```
test
```

[59]:

	numero_superamenti	totale_casi_per_provincia	q_cases
0	1.400000	48.100000	1
1	1.650000	145.900000	2
2	2.421053	275.631579	3
3	5.000000	619.150000	4
4	11.900000	2251.600000	5

[60]:
```
drawing = superamenti_totale.copy()
drawing = drawing.drop(columns=['numero_superamenti',␣
 ↪'totale_casi_per_provincia'])
drawing = drawing.merge(test)
drawing.rename(columns={'sigla_provincia':'prov_acr'},␣
 ↪inplace=True)
```

3.1 Una visione geografica del fenomeno

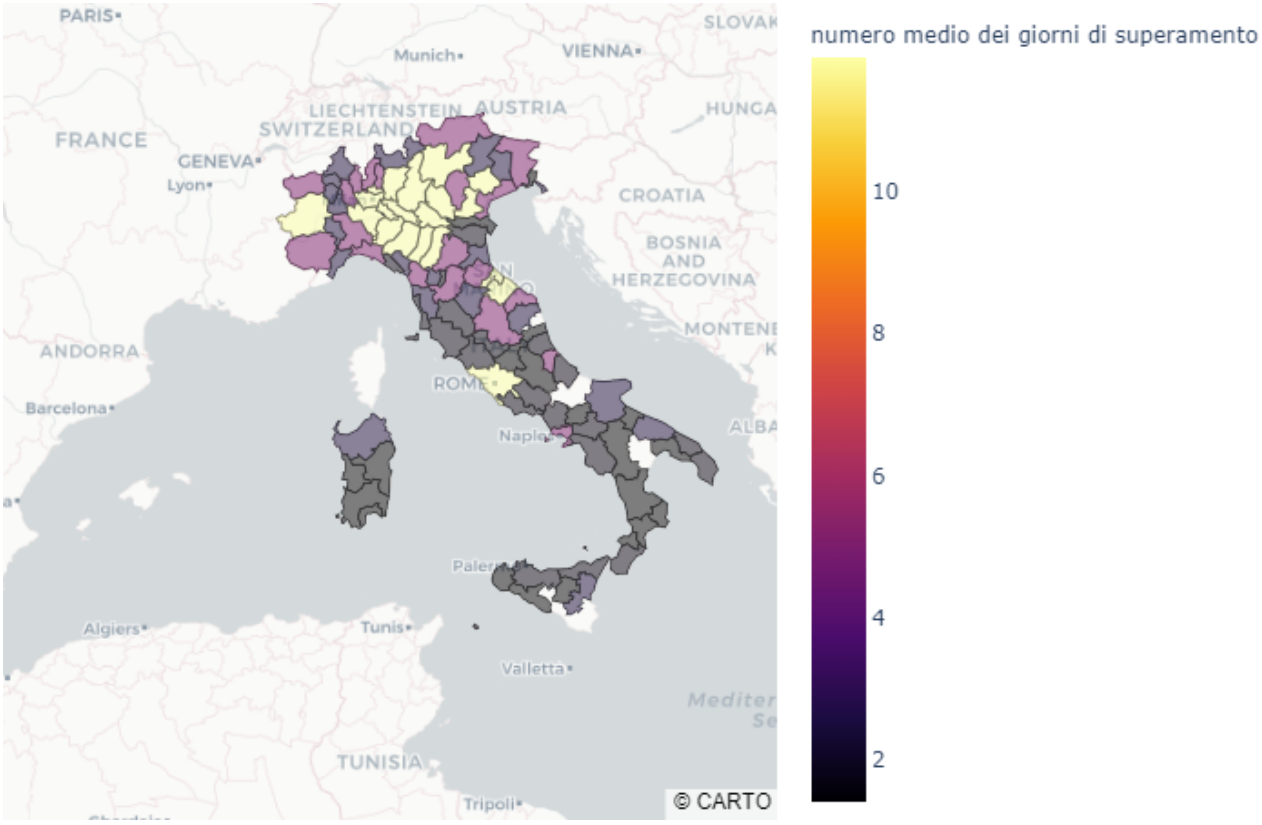
Le figure che seguono ci mostrano esattamente quanto abbiamo scoperto dalla precedente analisi:

- La Pianura Padana è maglia nera a livello europeo per l'inquinamento dell'aria. La concentrazione di ossidi dell'azoto e di polveri sottili, anche in tutta la zona circostante, è assolutamente preoccupante: nel Nord Italia, infatti, c'è l'area più inquinata d'Europa.

- La relazione tra i casi di COVID-19 e PM10 suggerisce un'interessante riflessione sul fatto che la concentrazione dei maggiori focolai si è registrata proprio in Pianura Padana mentre minori casi di infezione si sono registrati in altre zone d'Italia.

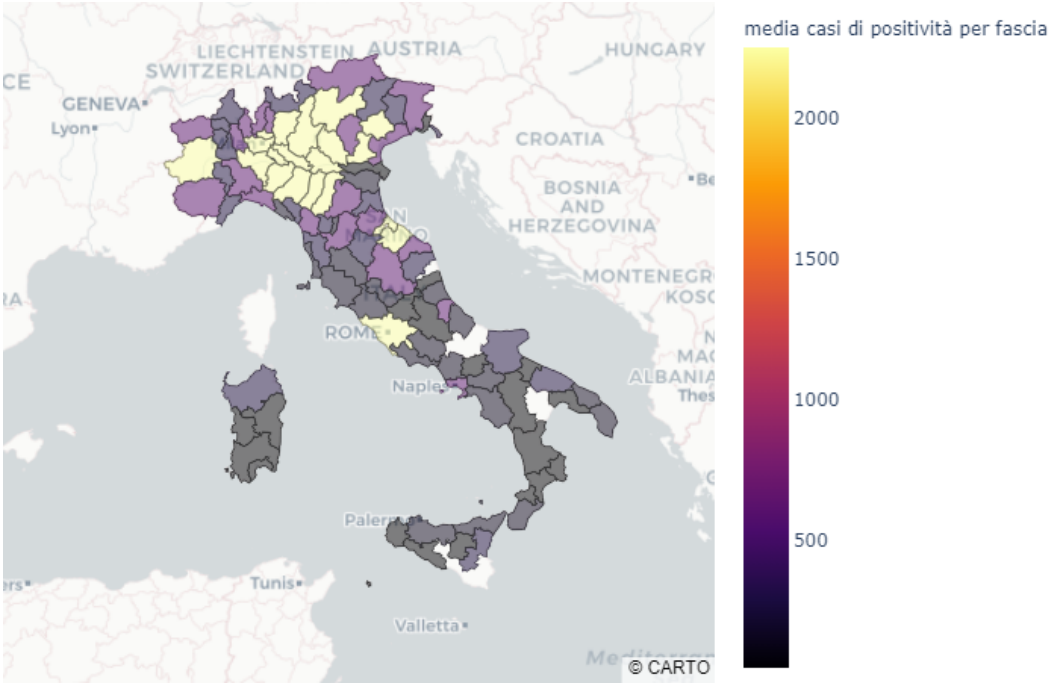
3.1.1 Numero medio dei giorni di superamento dei limiti per le fasce di contagio

[61]:
```
draw_map(drawing, 'numero_superamenti', 'numero medio dei␣
↪giorni di superamento')
```



3.1.2 Numero medio di positivi per fasce di contagio

[62]:
```
draw_map(drawing, 'totale_casi_per_provincia', 'media casi di␣
↪positività per fascia')
```



[63]:
```
lista_regioni = ['Lazio', 'Lombardia', 'Emilia-Romagna',␣
↪'Veneto', 'Piemonte', 'Campania', 'Puglia']
```

- La funzione *calculate_ delay()* conteggia la data a distanza di 11 giorni da quella passata in input

[64]:
```
import datetime as dt

def calculate_delay(data):
    return data + dt.timedelta(days=11)
```

[65]:
```
cov_prov =␣
↪Covid_province[(Covid_province['denominazione_regione'].
↪isin(lista_regioni)) & (~Covid_province['sigla_provincia'].
↪isna())]
```

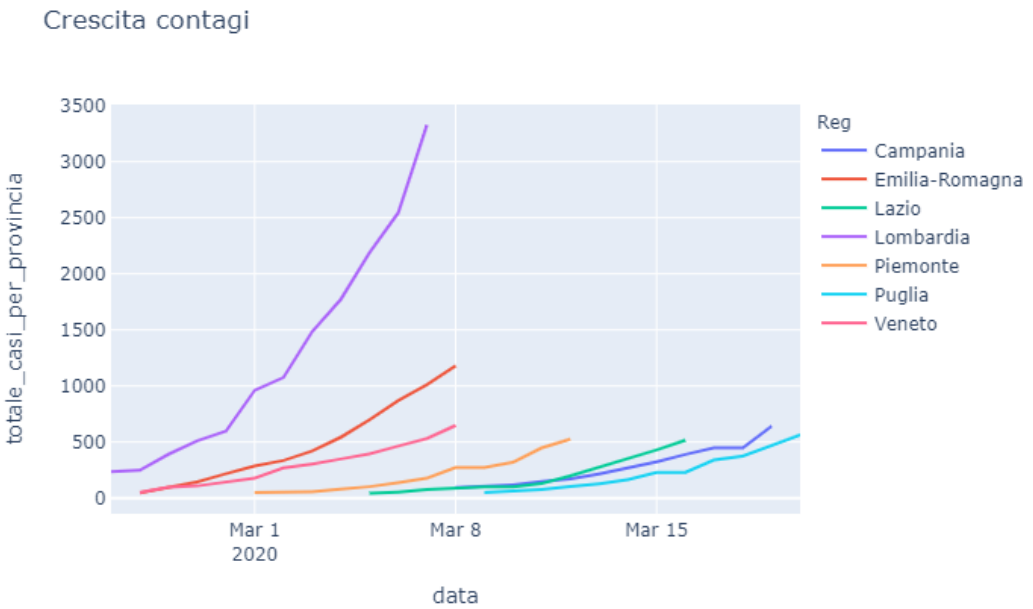
[66]:
```
cov_prov = cov_prov.groupby(by=['denominazione_regione',␣
↪'data'], as_index=False).agg({'totale_casi_per_provincia':
↪'sum'})
```

[67]:
```
data_regioni = pd.DataFrame()
```

```
for r in cov_prov['denominazione_regione'].unique():
    data_inizio = cov_prov[(cov_prov['denominazione_regione']␣
 ↪== r) & (cov_prov['totale_casi_per_provincia'] >␣
 ↪40)]['data'].min()
    data_end = calculate_delay(data_inizio)
    df = cov_prov[(cov_prov['data'] >= data_inizio) &␣
 ↪(cov_prov['data'] <= data_end) &␣
 ↪(cov_prov['denominazione_regione'] == r)]
    data_regioni = data_regioni.append(df)

data_regioni.rename(columns={'denominazione_regione':'Reg'},␣
 ↪inplace=True)
```

Le curve di espansione dell'infezione nelle regioni presentano andamenti perfettamente compatibili con i modelli epidemici, tipici di una trasmissione persona - persona per le regioni del sud Italia, mentre mostrano accelerazioni anomale proprio per quelle ubicate in Pianura Padana in cui i focolai risultano particolarmente virulenti e lasciano ragionevolmente ipotizzare ad una diffusione mediata da carrier ovvero da un veicolante.

[68]:
```
plot_storical_averages('data', 'totale_casi_per_provincia',␣
 ↪data_regioni, 'Crescita contagi', multi=True)
```



Tali analisi sembrano quindi dimostrare che, in relazione al periodo 10 Febbraio - 10 Marzo, concentrazioni elevate superiori al limite di PM10 in alcune Province del Nord Italia possano aver esercitato un'azione di boost, cioè di impulso alla diffusione virulenta dell'epidemia in Pianura Padana che non si è osservata in altre zone d'Italia che presentavano casi di contagi nello stesso periodo.

Oltre alle concentrazioni di particolato atmosferico, come fattore veicolante del virus, in alcune zone territoriali possono inoltre aver influito condizioni ambientali sfavorevoli al tasso di inattivazione virale.

3.2 Dati raccolti a livello regionale

Proprio per investigare anche su altri fattori, si è deciso di procedere anche con un'analisi a livello regionale, essendo disponibili a questo dettaglio ulteriori statistiche in grado di irrobustire le precedenti argomentazioni:

[69]:
```
#funzione per il bar plot dei dati di tutti i prossimi dati
def bar_plot(title,x, y, df):
  fig = px.bar(df, x=x, y=y, title=title)
  fig.show()
```

3.2.1 Fattori di rischio per la salute dei cittadini: l'obesità

La tabella *Obesità* descrive le abitudini alimentari degl'italiani per gli anni fra il 2001 e il 2019: in particolare, il dato considerato è la percentuale di over-18 obesi.

Fonte: Istat, Istituto nazionale di statistica

[70]:
```
obesità = pd.read_csv('https://bitbucket.org/christianderrico/
 ↪file_condivisi/raw/e7ca96d11691a602feafeb6847325b0bec81d781/
 ↪Obesity_2001-2019.csv').drop(columns=['Unnamed: 0',␣
 ↪'Tipo_dato'])
obesità.head(5)
```

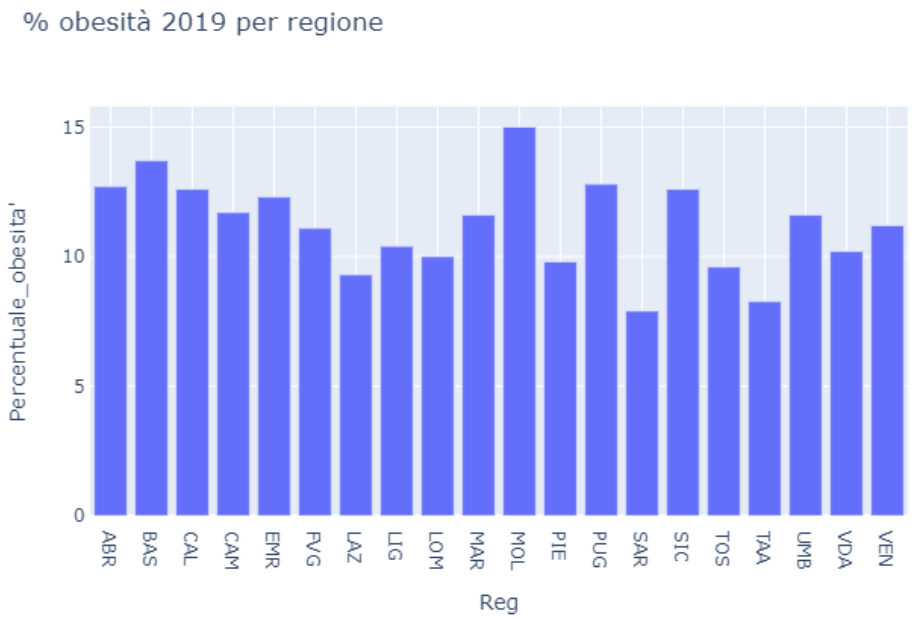
[70]:

	Anno	Territorio	Percentuale_obesita'
0	2001	ABR	9.4
1	2001	BAS	12.3
2	2001	CAL	8.0
3	2001	CAM	9.8

```
4  2001        EMR                    9.3
```

[71]:
```
obesità_2019 = obesità[obesità['Anno'] == 2019].
↪rename(columns={'Territorio':'Reg'}).drop(columns='Anno')
```

[72]:
```
bar_plot('% obesità 2019 per regione', 'Reg',␣
↪'Percentuale_obesita\'', obesità_2019)
```



3.2.2 Fattori di rischio per la salute dei cittadini: il fumo

La tabella *Fumatori* racconta l'abitudine al fumo degl'italiani per gli anni fra il 2001 e il 2019: in particolare, la tabella riporta la percentuale della popolazione con il vizio della sigaretta.

Fonte: Istat, Istituto nazionale di statistica

[73]:
```
fumatori = pd.read_csv('https://bitbucket.org/christianderrico/
↪file_condivisi/raw/fc448b6fd8ffc5d0103b5f01f8027b099c8a6f25/
↪Fumatori_italiani_2001-2019.csv').drop(columns='Unnamed: 0')
```

[74]:
```
fumatori.head(5)
```

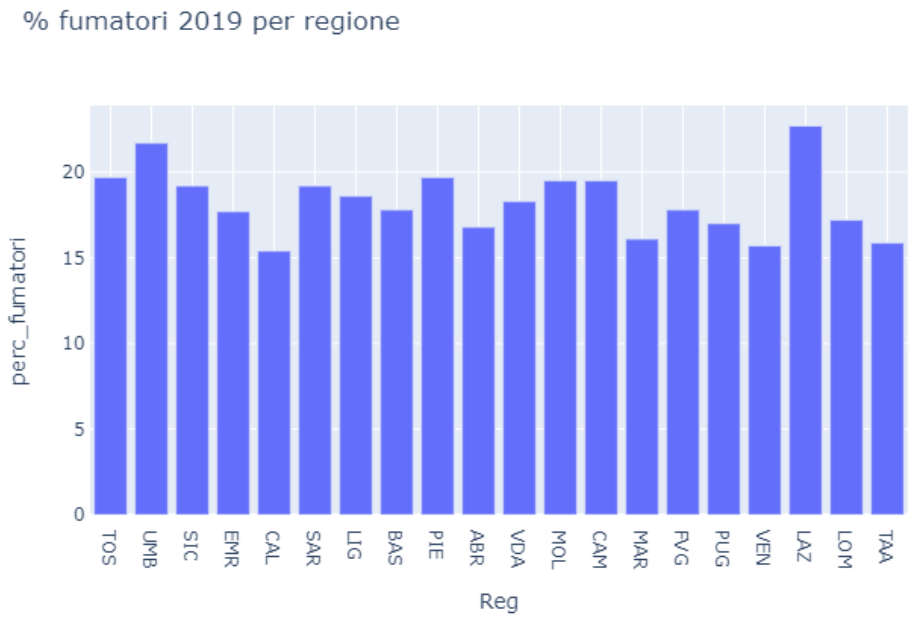
[74]:

	Reg	Anno	% fumatori
0	TOS	2001	24.9
1	TOS	2002	23.1
2	TOS	2003	23.2
3	TOS	2005	22.1
4	TOS	2006	22.6

[75]:
```
fumatori.rename(columns={'% fumatori':'perc_fumatori'},␣
↪inplace=True)
```

[76]:
```
fumatori_2019 = fumatori[fumatori['Anno'] == 2019].
↪drop(columns='Anno')
```

[77]:
```
bar_plot('% fumatori 2019 per regione', 'Reg',␣
↪'perc_fumatori', fumatori_2019)
```



[78]:
```
rischi = obesità_2019.rename(columns={'Territorio':'Reg'}).
↪merge(fumatori_2019)
```

3.2.3 Dati ambientali: dati meteo storici italiani

Dati meteo storici della penisola italiana; in particolare sono presenti le temperature estive e invernali medie per le regioni italiane nel periodo che va fra l'anno 2012 e l'anno 2018

Fonte: Wheather Spark

[79]:
```
Italia_meteo = pd.read_csv('https://bitbucket.org/
↪christianderrico/file_condivisi/raw/
↪70a7694bd7173450d3807954c608538007d0beb1/
↪Temperature_medie_regionali_2012-2018.csv').
↪drop(columns='Unnamed: 0')
```

[80]:
```
Italia_meteo.head(5)
```

[80]:

	Reg	Anno	t_estiva	t_invernale
0	BAS	2014	296.102502	282.678735
1	BAS	2015	297.634655	281.198988
2	BAS	2016	296.410545	282.102284
3	BAS	2017	298.424478	280.365530
4	BAS	2018	296.875907	281.616902

3.2.4 Dati ambientali: medie annuali di PM10 e PM2.5 per le regioni italiane

Sono stati raccolti i dati sulle medie annuali di PM10 e PM2.5 per le regioni italiane nel periodo compreso fra l'anno 2012 e l'anno 2018

Fonte: Istat, Istituto nazionale di statistica

[81]:
```
inquinamento_regioni = pd.read_csv('https://bitbucket.org/
↪christianderrico/file_condivisi/raw/
↪fc448b6fd8ffc5d0103b5f01f8027b099c8a6f25/
↪Inquinamento_2012-2018.csv').drop(columns='Unnamed: 0')
```

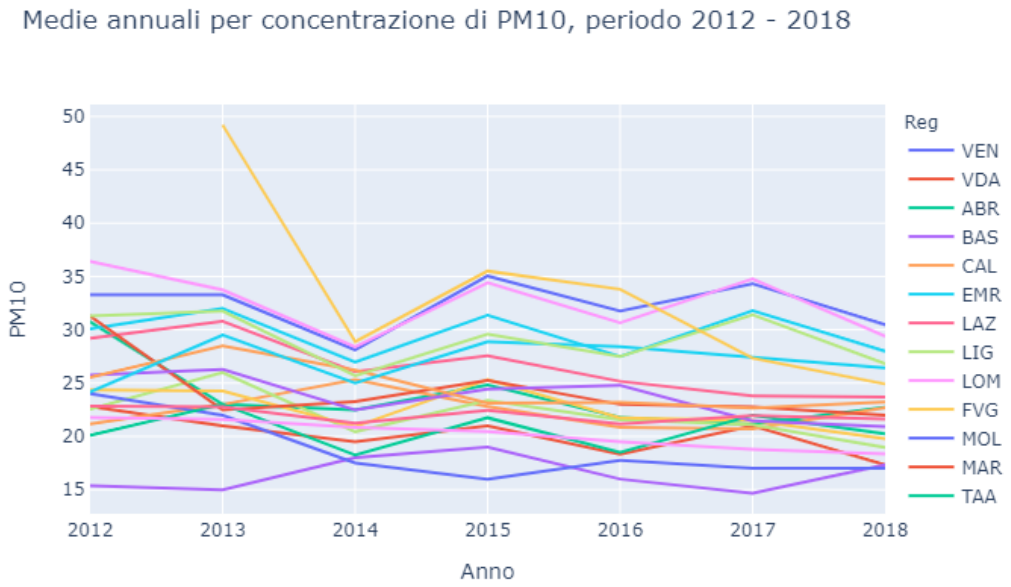
[82]:
```
for v in inquinamento_regioni['Anno'].unique():
  val_na_pm25 =␣
↪len(inquinamento_regioni[(inquinamento_regioni['Anno'] ==␣
↪v) & (inquinamento_regioni['PM2.5'].isna())])
```

```
 val_na_pm10 =␣
↪len(inquinamento_regioni[(inquinamento_regioni['Anno'] ==␣
↪v) & (inquinamento_regioni['PM10'].isna())])
 print('Anno: ' + str(v) + ' % valori Pm2.5 nulli: ' +␣
↪str((val_na_pm25 / 20) * 100))
 print('Anno: ' + str(v) + ' % valori Pm10 nulli: ' +␣
↪str((val_na_pm10 / 20) * 100))
 print("\n")
```

```
Anno: 2012 % valori Pm2.5 nulli: 95.0
Anno: 2012 % valori Pm10 nulli: 0.0


Anno: 2013 % valori Pm2.5 nulli: 10.0
Anno: 2013 % valori Pm10 nulli: 0.0


Anno: 2014 % valori Pm2.5 nulli: 10.0
Anno: 2014 % valori Pm10 nulli: 0.0


Anno: 2015 % valori Pm2.5 nulli: 10.0
Anno: 2015 % valori Pm10 nulli: 0.0


Anno: 2016 % valori Pm2.5 nulli: 10.0
Anno: 2016 % valori Pm10 nulli: 0.0


Anno: 2017 % valori Pm2.5 nulli: 10.0
Anno: 2017 % valori Pm10 nulli: 0.0


Anno: 2018 % valori Pm2.5 nulli: 10.0
Anno: 2018 % valori Pm10 nulli: 0.0
```

- Questo l'andamento delle medie di concentrazione per il PM10, agente inquinante di cui abbiamo le informazioni più precise.

[83]:
```
plot_storical_averages('Anno', 'PM10', inquinamento_regioni,␣
↪"Medie annuali per concentrazione di PM10, periodo 2012 -␣
↪2018", multi=True)
```



3.2.5 Dati socio-economici: popolazione per regione 2012-2020

Ecco i dati sulla popolazione regionale fra il 2012 e il 2020: - totale popolazione residente - densità abitativa - totale stranieri residenti in Italia

Presenti anche informazioni sull'estensione per km2 delle regioni, dato necessario al calcolo della densità di popolazione.

Fonte: Istat, Istituto nazionale di statistica

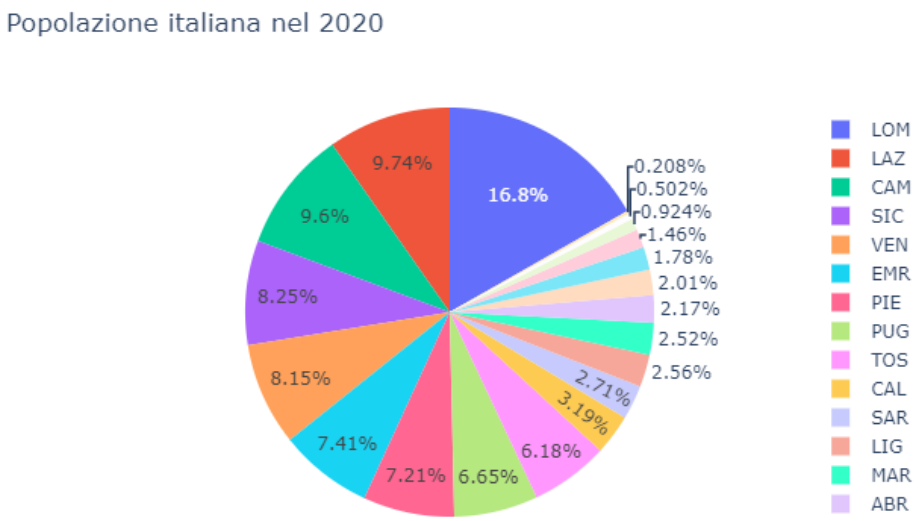
[84]:
```
popo = pd.read_csv("https://bitbucket.org/christianderrico/
↪file_condivisi/raw/ba560ce38a449801a9ac7a195c10e2b832d12ca6/
↪Regioni_italiane_popolazione_2012-2020_compresi_straneri.
↪csv").drop(columns=["Unnamed: 0", 'Percentuale_stranieri',␣
↪'Stranieri'])
```

[85]:
```
popo.head(5)
```

[85]:
```
   Anno  Reg  Popolazione_residente  Superficie_territoriale  ␣
↪  Densita
0  2012  ABR                1306416.0                10831.4964  ␣
↪120.612698
1  2012  BAS                 577562.0                10073.1105  ␣
↪57.337006
2  2012  CAL                1958418.0                15221.6148  ␣
↪128.660331
3  2012  CAM                5764424.0                13670.5980  ␣
↪421.665826
4  2012  EMR                4341240.0                22452.2072  ␣
↪193.354709
```

[86]:
```
popo_2020 = popo[popo['Anno'] == 2020]
```

[87]:
```
px.pie(popo_2020, values='Popolazione_residente', names='Reg',␣
↪title='Popolazione italiana nel 2020')
```



3.2.6 Dati sanitari: Posti letto

Sono stati collezionati i dati sui posti letto disponibili presso tutte le strutture sanitarie italiane, private e pubbliche, nell'anno 2018

Fonte: ISS, Istituto superiore di sanità

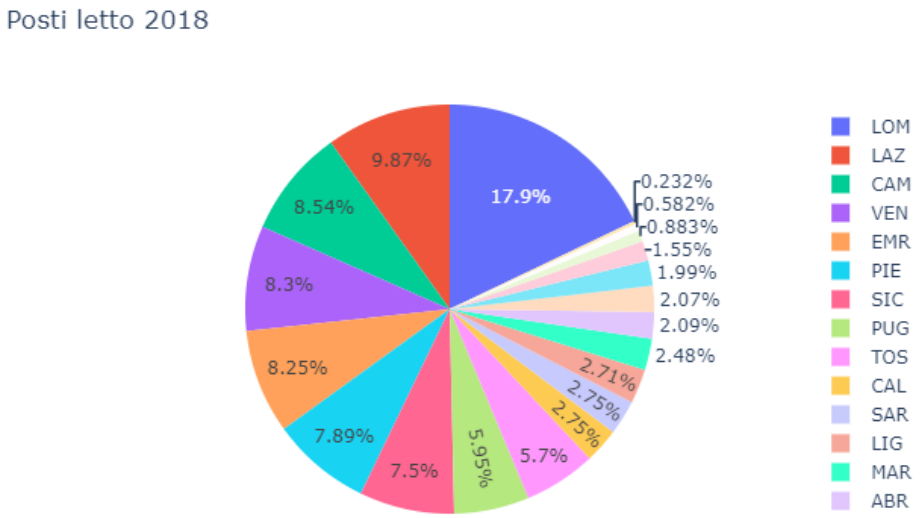
[88]:
```
posti_letto = pd.read_csv("https://bitbucket.org/
↪christianderrico/file_condivisi/raw/
↪ba560ce38a449801a9ac7a195c10e2b832d12ca6/
↪Posti_letto_Italia_2018.csv").drop(columns='Unnamed: 0')
```

[89]:
```
posti_letto.head(5)
```

[89]:

	Reg	Anno	Totale posti letto
0	ABR	2018	4415
1	BAS	2018	1862
2	CAL	2018	5796
3	CAM	2018	18003
4	EMR	2018	17395

[90]:
```
px.pie(posti_letto, values='Totale posti letto', names='Reg',␣
↪title='Posti letto 2018')
```



3.2.7 Dati sanitari: Patologie croniche pregresse

Raccolti anche i dati che mostrano la concentrazione regionale percentuale di persone con almeno una patologia cronica pregressa nell'anno 2019

Fonte: Istat, Istituto nazionale di statistica

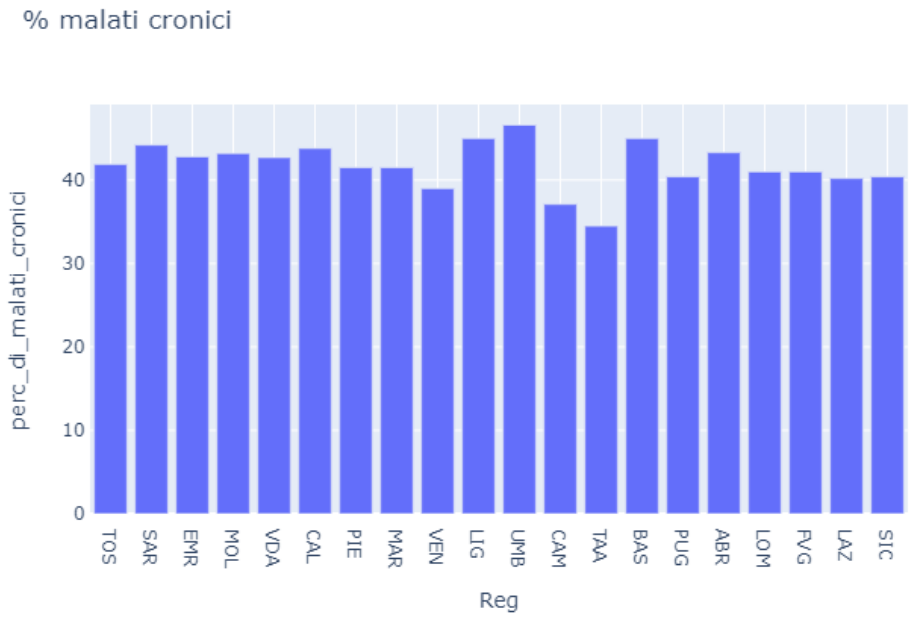
[91]:
```
pregresse = pd.read_csv('https://bitbucket.org/
↪christianderrico/file_condivisi/raw/
↪c0f7487b2137cd1d5b58e3abd0b147bae9c7e360/pregresse.csv').
↪drop(columns='Unnamed: 0')
```

[92]:
```
pregresse.head(5)
```

[92]:

	Reg	perc_di_malati_cronici
0	TOS	41.9
1	SAR	44.2
2	EMR	42.8
3	MOL	43.2
4	VDA	42.7

[93]:
```
bar_plot('% malati cronici', 'Reg', 'perc_di_malati_cronici',␣
↪pregresse)
```



3.2.8 Dati su mortalità per regione

Sono stati raccolti anche i dati sulla mortalità per regione, per il periodo compreso fra il 2012 e il 2018.

Il livello di dettaglio di queste informazioni include anche la divisione in fasce di età:

- mortalità 0-14 anni
- mortalità 15-44 anni
- mortalità 45-64 anni
- mortalità over 65 anni

Oltre al numero di decessi e alla totalità della popolazione, sono presenti due ulteriori metriche:

- Il tasso di mortalità grezzo, definito come "il tasso di mortalità per tutte le cause di morte per una popolazione", calcolato come il numero totale di decessi durante un dato intervallo di tempo diviso per la popolazione, per 1.000 o 100.000.
- Il tasso di mortalità standardizzato per età è una media ponderata dei tassi di mortalità specifici per età per 100.000 persone, dove i pesi sono le proporzioni delle persone nei gruppi di età corrispondenti della popolazione standard selezionata.

Fonte: Istat, Istituto nazionale di statistica

[94]:
```
mort_base = pd.read_csv("https://bitbucket.org/
 ↪christianderrico/file_condivisi/raw/
 ↪ba560ce38a449801a9ac7a195c10e2b832d12ca6/
 ↪Mortalit%C3%A0_base_2012-2018.csv").drop(columns='Unnamed:␣
 ↪0')
mort_014 = pd.read_csv("https://bitbucket.org/christianderrico/
 ↪file_condivisi/raw/ba560ce38a449801a9ac7a195c10e2b832d12ca6/
 ↪Mortalit%C3%A0_0-14.csv").drop(columns='Unnamed: 0')
mort_1444 = pd.read_csv("https://bitbucket.org/
 ↪christianderrico/file_condivisi/raw/
 ↪3d636001dcffcd635a5db953b0787a2460a5fe04/
 ↪Mortalit%C3%A0_15-44.csv").drop(columns='Unnamed: 0')
```

```
mort_4564 = pd.read_csv("https://bitbucket.org/
 ↪christianderrico/file_condivisi/raw/
 ↪ba560ce38a449801a9ac7a195c10e2b832d12ca6/
 ↪Mortalit%C3%A0_45-64.csv").drop(columns='Unnamed: 0')
mort_old = pd.read_csv("https://bitbucket.org/christianderrico/
 ↪file_condivisi/raw/3d636001dcffcd635a5db953b0787a2460a5fe04/
 ↪Mortalit%C3%A0_over_64.csv").drop(columns='Unnamed: 0')
```

[95]:
```
mort_base.head(5)
```

[95]:
```
  Territorio  Tasso_std_per_età  Decessi  Popolazione ␣
 ↪Tasso_grezzo
0        ABR             1063.2   103394      9248392       ␣
 ↪1118.0
1        BAS             1016.4    43290      3995007       ␣
 ↪1083.6
2        CAL              935.4   137473     13766156        ␣
 ↪998.6
3        CAM              855.8   376294     40782462        ␣
 ↪922.7
4        EMR             1085.3   346541     30965205       ␣
 ↪1119.1
```

[96]:
```
mort_014.head(5)
```

[96]:
```
  Territorio  Decessi  Popolazione_per_età  Crude rate
0        ABR      303              1184392   25.582746
1        BAS      151               485825   31.081151
2        CAL      632              1894907   33.352560
3        CAM     1772              6302402   28.116264
4        EMR     1032              4159953   24.807973
```

[97]:
```
mort_1444.head(5)
```

[97]:
```
  Territorio  Decessi  Popolazione_per_età  Crude rate
0        ABR     1701              3327122   51.125267
1        BAS      768              1490889   51.512889
2        CAL     2733              5280361   51.757825
3        CAM     8678             16193119   53.590664
4        EMR     5075             10710485   47.383475
```

[98]:
```
mort_4564.head(5)
```

[98]:

	Territorio	Decessi	Popolazione_per_età	Crude rate
0	ABR	9121	2643274	345.064492
1	BAS	4038	1151482	350.678517
2	CAL	13912	3809509	365.191420
3	CAM	47085	11132595	422.947210
4	EMR	27832	8854445	314.328001

[99]:
```
mort_old.head(5)
```

[99]:

	Territorio	Decessi	Popolazione_per_età	Crude rate
0	ABR	92269	2093604	4407.184931
1	BAS	38333	866811	4422.301978
2	CAL	120196	2781379	4321.453495
3	CAM	318759	7154346	4455.459660
4	EMR	312602	7240322	4317.515160

[100]:
```
mort_014.rename(columns={'Popolazione_per_età':'pop_0_14'},␣
↪inplace=True)
mort_1444.rename(columns={'Popolazione_per_età':'pop_15_44'},␣
↪inplace=True)
mort_4564.rename(columns={'Popolazione_per_età':'pop_45_64'},␣
↪inplace=True)
mort_old.rename(columns={'Popolazione_per_età':'old_pop'},␣
↪inplace=True)
```

[101]:
```
mort_base = pd.merge(mort_base, mort_014.iloc[:,[0,2]], on =␣
↪"Territorio")
mort_base = pd.merge(mort_base, mort_1444.iloc[:,[0,2]], on =␣
↪"Territorio")
mort_base = pd.merge(mort_base, mort_4564.iloc[:,[0,2]], on =␣
↪"Territorio")
mort_base = pd.merge(mort_base, mort_old.iloc[:,[0,2]], on =␣
↪"Territorio")
```

[102]:
```
mort_base['old_pecent'] = mort_base['old_pop'] /␣
↪mort_base['Popolazione']
mort_base['0_14_pecent'] = mort_base['pop_0_14'] /␣
↪mort_base['Popolazione']
```

```
mort_base['15_44_pecent'] = mort_base['pop_15_44'] /␣
 ↪mort_base['Popolazione']
mort_base['45_64_pecent'] = mort_base['pop_45_64'] /␣
 ↪mort_base['Popolazione']
```

[103]:
```
mort_base.head(5)
```

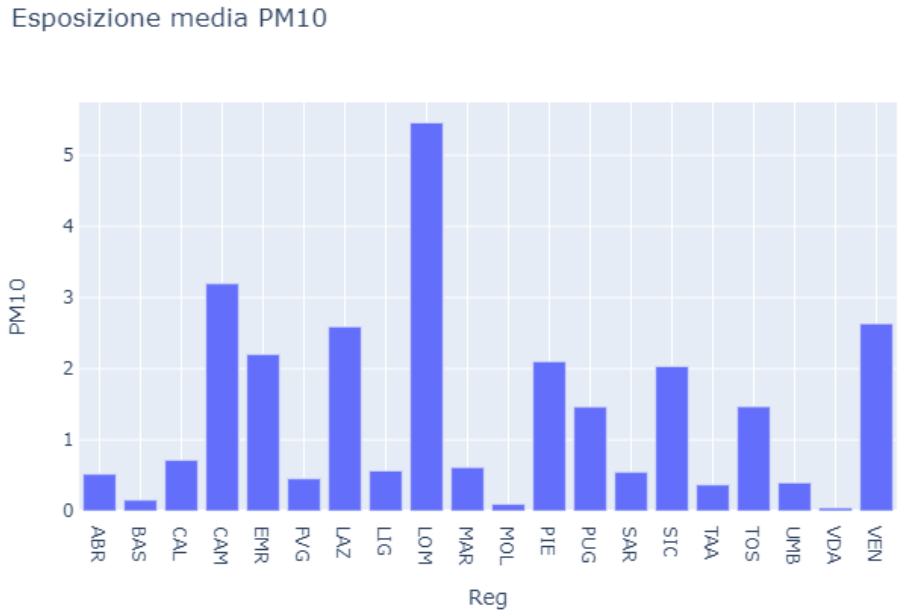
3.3 Analisi regionale

La data scelta per lo studio è quella di ieri:

[104]:
```
Covid_regioni['data'] = [pd.to_datetime(str(v)[:10]) for v in␣
 ↪Covid_regioni['data']]
Covid_analisi_regioni = Covid_regioni[Covid_regioni['data'] ==␣
 ↪str(yesterday)]
```

Viene calcolata l'esposizione media alle polveri sottili di diametro 10: tale fattore è pesato per la popolazione residente nella regione.

[105]:
```
esposizione_media = inquinamento_regioni.groupby(by='Reg',␣
 ↪as_index=False).agg({'PM10':'mean', 'PM2.5':'mean'})
esposizione_media = esposizione_media.merge(popo_2020).
 ↪drop(columns='PM2.5')
esposizione_media['PM10'] = esposizione_media['PM10'] *␣
 ↪(esposizione_media['Popolazione_residente'] /␣
 ↪esposizione_media['Popolazione_residente'].sum())

bar_plot('Esposizione media PM10', 'Reg', 'PM10',␣
 ↪esposizione_media)
```



Operazioni analoghe sono state effettuate anche con la temperatura media per il periodo 2012-2018, invernale ed estiva:

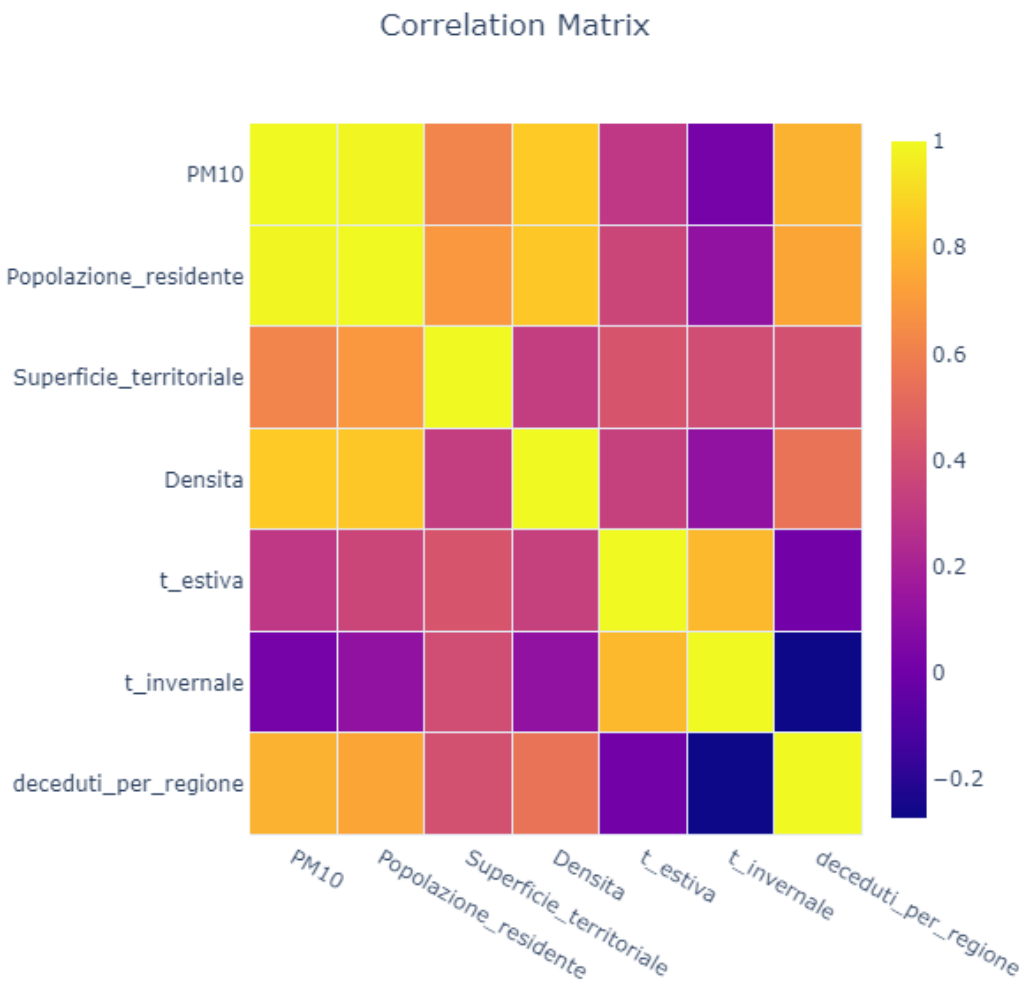
[106]:
```
ita_meteo = Italia_meteo.groupby(by='Reg', as_index=False).
↪agg({
    't_estiva':'mean',
    't_invernale':'mean'
})
```

[107]:
```
Covid_analisi_regioni['denominazione_regione'] = [r[:3].
↪upper().replace('TRE', 'TAA').replace('VAL', 'VDA').
↪replace('FRI', 'FVG').replace('EMI', 'EMR') for r in␣
↪Covid_analisi_regioni['denominazione_regione']]
Covid_analisi_regioni.rename(columns={'denominazione_regione':
↪'Reg'}, inplace=True)
```

[108]:
```
ultima_analisi = esposizione_media.merge(ita_meteo).
↪merge(Covid_analisi_regioni.loc[:, ['Reg',␣
↪'deceduti_per_regione']]).drop(columns='Anno')
```

Queste le correlazioni:

[109]:
```
draw_correlation_heat_map(ultima_analisi)
```



[110]:
```
"Correlazione fra PM10 e decessi da Covid-19: " +␣
↪str(ultima_analisi.corr().loc['PM10',␣
↪'deceduti_per_regione'])
```

[110]:
```
'Correlazione fra PM10 e decessi da Covid-19: 0.78'
```

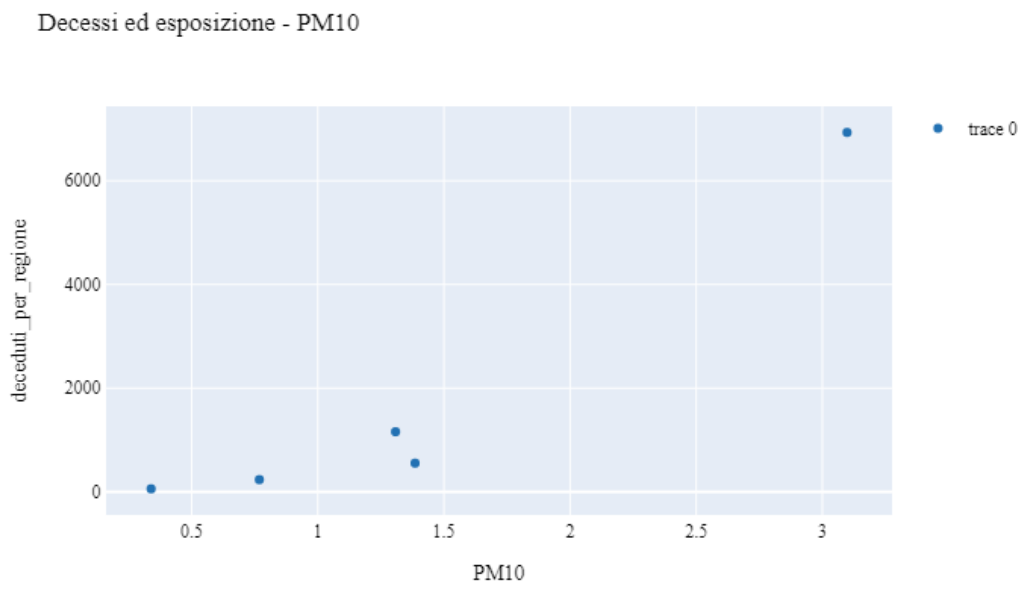
Ancora una volta l'output è stato suddiviso in 5 fasce sulla base della divisione in quantili della variabile dipendente:

[111]:
```
ultima_analisi = categorize_var(ultima_analisi,␣
↪'deceduti_per_regione', 'q_deaths')
```

[112]:
```
test = ultima_analisi.groupby(by='q_deaths', as_index=False).
↪mean()
deaths = test['q_deaths']
```

Il grafico mostra la correlazione fra le fasce di decessi per Covid-19 ed esposizione media al PM10:

[113]:
```
titolo = 'Decessi ed esposizione - PM10'
draw_scatter_plot([create_plot(test, 'PM10',␣
↪'deceduti_per_regione', 'q_deaths', titolo)])
```



[114]:
```
test.drop(columns='q_deaths', inplace=True)
test_2 = pd.DataFrame(scaler.fit_transform(test), columns=test.
↪columns)
```

Questi i risultati della regressione lineare che cerca di ricostruire l'andamento del fenomeno: con una precisione del 91%, la correlazione parrebbe essere attestata anche questa volta

[115]:
```
legend = get_statistics(lean_reg, test_2[['PM10']],␣
↪test_2['deceduti_per_regione'], test_2[['PM10']],␣
↪test_2['deceduti_per_regione'])
plot_paramethers(lean_reg, test_2[['PM10']])
```

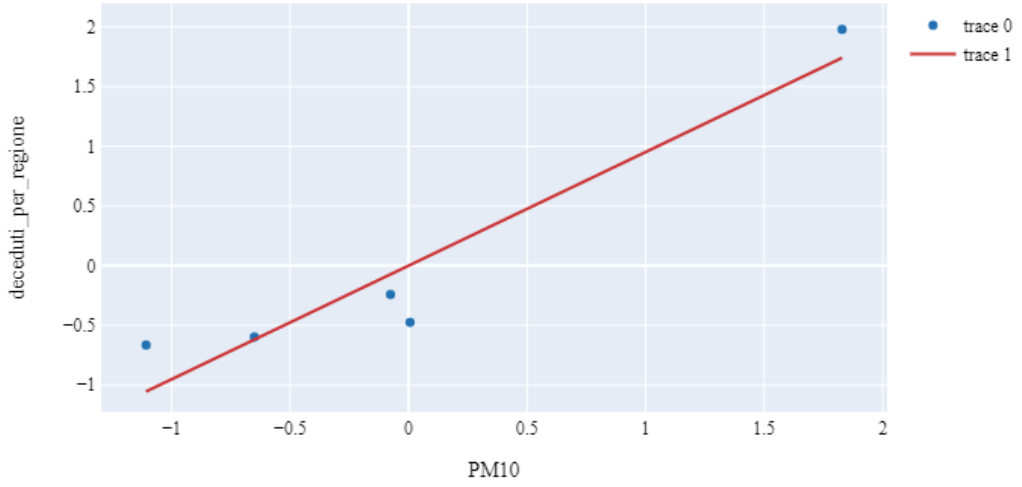
```
'Model accuracy (R^2): 0.91'


'Mean Squared Error: 0.1'


'Mean Absolute Percentage Error: 1720.2'
```

[115]:

	0
PM10	0.952019

[116]:
```
test_2['q_deaths'] = deaths
```

[117]:
```
calculate_line(lean_reg, test_2[['PM10']], test_2, 'PM10',␣
↪'deceduti_per_regione', 'q_deaths', line=False)
```



- Si può osservare che il terzo gruppo di regioni è stato esposto ad un livello di PM10 maggiore, ma ha avuto un numero di decessi inferiore rispetto

al quarto gruppo. Sicuramente tale dato è stato influenzato da ulteriori fattori.

- La densità media di popolazione per il quarto gruppo risulta leggermente più alta di quella del terzo gruppo: questo ovviamente ha portato più persone a contatto tra loro e causato più contagi in occasione dello scoppio di un focolaio. Tuttavia la Campania risulta la regione con una densità di popolazione maggiore in Italia, ma non occupa nemmeno la prima posizione all'interno del suo gruppo.

[118]:
```
display('Densità terzo gruppo: ' +␣
 ↪str(ultima_analisi[ultima_analisi['q_deaths'] ==␣
 ↪3]['Densita'].mean()))
display('Densità quarto gruppo: ' +␣
 ↪str(ultima_analisi[ultima_analisi['q_deaths'] ==␣
 ↪4]['Densita'].mean()))
```

```
'Densità terzo gruppo: 206.97944829397858'
```

```
'Densità quarto gruppo: 237.19051711207607'
```

[119]:
```
ultima_analisi[ultima_analisi['q_deaths'] == 3]
```

[119]:

	Reg	PM10	Popolazione_residente	Superficie_territoriale	Densita	t_estiva	t_invernale	deceduti_per_regione	q_deaths
0	ABR	0.516247	1305770.0	10831.4964	120.553057	297.737265	280.821289	477	3
3	CAM	3.195712	5785861.0	13670.5980	423.233936	297.002090	281.382258	457	3
12	PUG	1.463733	4008296.0	19540.5181	205.127417	299.295352	282.484764	583	3
15	TAA	0.366618	1074819.0	13604.7211	79.003384	293.941225	277.871404	697	3

[120]:
```
ultima_analisi[ultima_analisi['q_deaths'] == 4]
```

[120]:
```
    Reg      PM10  Popolazione_residente ␣
↪Superficie_territoriale     Densita  \
6   LAZ  2.590271              5865544.0               17231.
↪7227  340.392200
7   LIG  0.563478              1543127.0                5416.
↪1520  284.912056
9   MAR  0.612196              1518400.0                9401.
↪1839  161.511573
16  TOS  1.466802              3722729.0               22987.
↪4371  161.946240

      t_estiva  t_invernale  deceduti_per_regione  q_deaths
6   299.008323   281.354166                   902         4
7   295.922288   281.015898                  1594         4
9   294.425568   277.195220                   989         4
16  299.414541   282.453916                  1153         4
```

- Due regioni su quattro del quarto gruppo figurano fra le prime dieci regioni per percentuale di malati cronici: sono la Liguria seconda e la Toscana decima; nel loro gruppo si piazzano rispettivamente seconde e prime. Per il terzo gruppo figura solamente l'Abruzzo.

[121]:
```
sanity = pregresse.sort_values('perc_di_malati_cronici',␣
↪ascending=False).merge(mort_base.
↪rename(columns={'Territorio':'Reg'}), on='Reg')
```

[122]:
```
sanity.head(10).iloc[:, [0, 1]]
```

[122]:
```
   Reg  perc_di_malati_cronici
0  UMB                    46.6
1  LIG                    45.0
2  BAS                    45.0
3  SAR                    44.2
4  CAL                    43.8
5  ABR                    43.3
6  MOL                    43.2
7  EMR                    42.8
8  VDA                    42.7
```

```
9   TOS                    41.9
```

- Liguria e Toscana sono anche ai vertici della classifica delle regioni per percentuale di popolazione over-65, la categoria a rischio per conseguenze del contagio da Covid-19. Presenti anche le Marche, all'ultimo posto l'Abruzzo.

[123]:
```
sanity.sort_values('old_pecent', ascending=False).iloc[:, [0,␣
↪-4]].head(10)
```

[123]:
```
    Reg  old_pecent
1   LIG    0.279551
13  FVG    0.250365
9   TOS    0.246465
0   UMB    0.246221
8   VDA    0.245524
6   MOL    0.245196
11  PIE    0.244115
10  MAR    0.237063
7   EMR    0.233821
5   ABR    0.226375
```

- La Liguria inoltre è prima per tasso grezzo di mortalità, mentre la Toscana settima; a queste si aggiungono anche le Marche (ottave), sempre appartenenti al quarto gruppo. Per il terzo gruppo è presente ancora una volta l'Abruzzo, fra le ultime posizioni.

[124]:
```
sanity.sort_values('Tasso_grezzo', ascending=False).head(10).
↪iloc[:, [0, 5]]
```

[124]:
```
    Reg  Tasso_grezzo
1   LIG        1379.6
6   MOL        1243.2
8   VDA        1187.7
11  PIE        1178.1
13  FVG        1172.9
0   UMB        1170.1
9   TOS        1161.2
10  MAR        1128.9
7   EMR        1119.1
5   ABR        1118.0
```

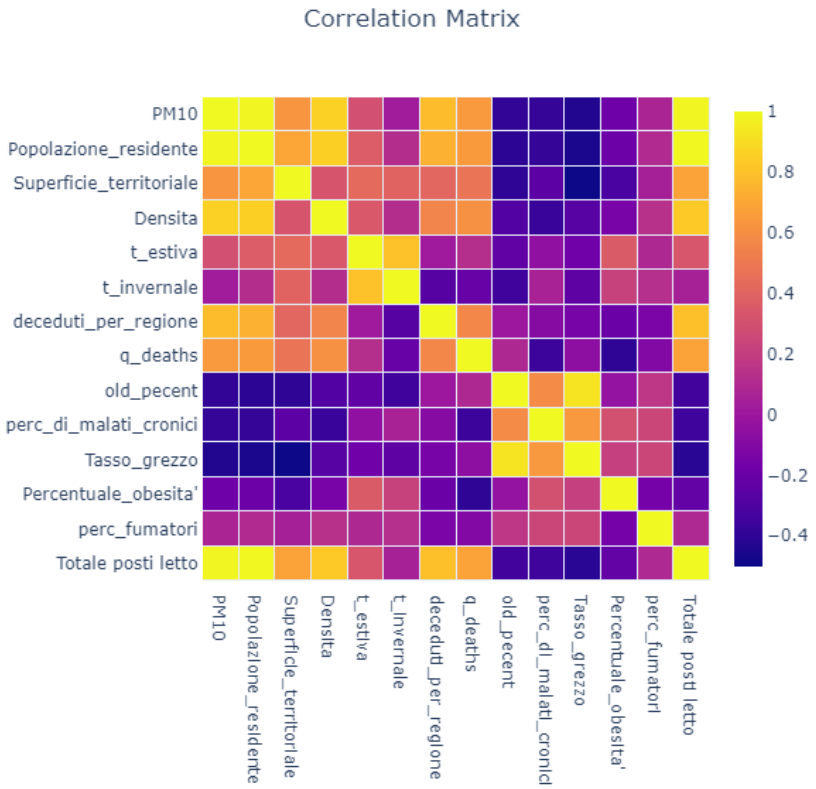
Sulla base di queste considerazioni, si effettua un'ulteriore analisi, questa volta considerando le singole 20 regioni italiane e le variabili oggetto delle precedenti riflessioni, unite ai dati sui posti letto a disposizione della popolazione e alle statistiche sul fumo.

L'obiettivo è cercare di capire come la combinazione di tutti questi fattori abbia agito sul virus.

[125]:
```
test = ultima_analisi.merge(sanity.loc[:, ['Reg',␣
↪'old_pecent', 'perc_di_malati_cronici', 'Tasso_grezzo']]).
↪merge(rischi, on='Reg').merge(posti_letto).
↪drop(columns='Anno')
```

Queste le correlazioni:

[126]:
```
draw_correlation_heat_map(test)
```



[127]:
```
reg = test['Reg']
test.drop(columns='Reg', inplace=True)
```

[128]:
```
test_2 = pd.DataFrame(scaler.fit_transform(test), columns=test.
↪columns).drop(columns='q_deaths')
```

[129]:
```
y = test_2['deceduti_per_regione']
X = test_2.iloc[:, [v for v in range(len(test_2.columns))if v !
↪= 6]]
```

Questi i punteggi ottenuti dal modello lineare:

[130]:
```
legend = get_statistics(lean_reg, X, y, X, y)
```

```
'Model accuracy (R^2): 0.95'
```

```
'Mean Squared Error: 0.1'
```

```
'Mean Absolute Percentage Error: 52.5'
```

Questi i parametri:

[131]:
```
plot_paramethers(lean_reg, X)
```

[131]:

	0
PM10	0.958326
Popolazione_residente	-1.605102
Superficie_territoriale	-0.615145
Densita	-0.765045
t_estiva	-0.316341
t_invernale	0.481555
old_pecent	0.316166
perc_di_malati_cronici	0.068463
Tasso_grezzo	-0.043933
Percentuale_obesita'	-0.085976
perc_fumatori	-0.257200
Totale posti letto	2.694364

E i rispettivi p-values:

[132]:
```
plot_p_values(lean_reg, X, y)
```

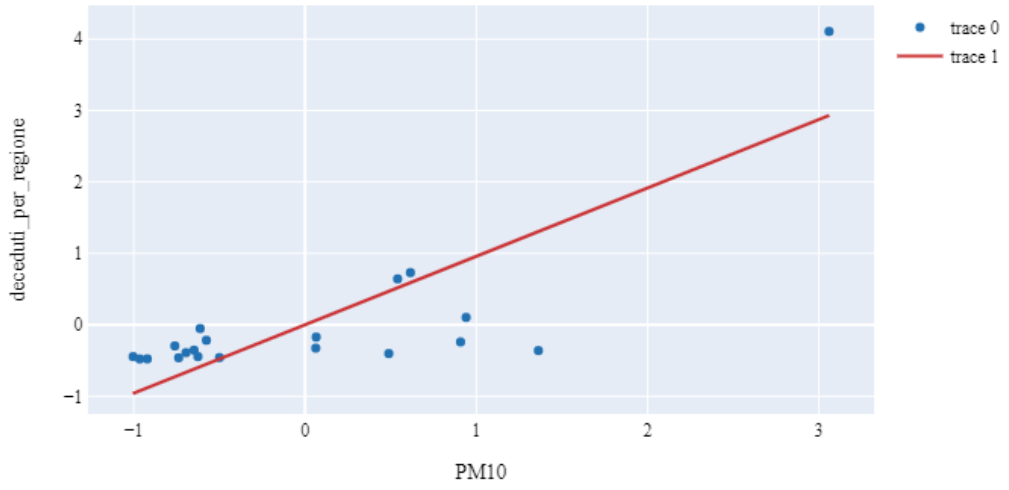
```
                                0
Intercept                1.000000
PM10                     0.036756
Popolazione_residente    0.041811
Superficie_territoriale  0.000198
Densita                  0.000249
t_estiva                 0.017101
t_invernale              0.001217
old_pecent               0.163049
perc_di_malati_cronici   0.386470
Tasso_grezzo             0.834505
Percentuale_obesita'     0.365077
perc_fumatori            0.000367
Totale posti letto       0.000557
```

- le regioni delle Pianura Padana confermano il trend già tracciato nello svolgimento dell'analisi provinciale, anche se i decessi per la Lombardia vanno ben oltre le previsioni del modello; discorso inverso per il Veneto, che grazie a delle adeguate misure di contenimento ha saputo fronteggiare la pandemia e ridurre il numero dei morti.

- per quanto riguarda le altre regioni, a pesare è stata la temperatura estiva, che probabilmente ha contribuito a ridurre la virulenza del Coronavirus, (per le regioni del sud, Puglia e Sicilia fra le altre) insieme alla lontananza dal focolare lombardo, il principale dell'epidemia.

- i picchi delle restanti regioni (fra cui la Liguria) sono da attribuire probabilmente alle caratteristiche demografiche della popolazione, in particolare alle abitudini rischiose per la salute, alla presenza di numerosi malati con patologie pregresse e in età considerata a rischio (over-65)

[133]:
```
test_2['Reg'] = reg
calculate_line(lean_reg, X, test_2, 'PM10',␣
 ↪'deceduti_per_regione', 'Reg', line=True)
```



3.4 Conclusioni

- Le analisi condotte hanno cercato di dimostrare delle relazioni fra inquinamento e la malattia causata dal nuovo Coronavirus.
- In due modi si è ipotizzato che tali relazioni si siano manifestate:
  - il particolato potrebbe aver fatto da *carrier*, vettore nella diffusione della pandemia, e aver avuto quindi un ruolo cruciale nella fase iniziale dell'emergenza sanitaria.
  - l'esposizione prolungata agl'agenti inquinanti della popolazione potrebbe averne indebolito le difese immunitarie, rendendola più "fragile" e più sensibile all'impatto del Covid.

Si è lavorato ad un livello di dettaglio provinciale e regionale:

- si è tenuto conto di quanto svolto dalla SIMA (Società Italiana Medicina Ambientale) nel lavoro "Relazione circa l'effetto dell'inquinamento da particolato atmosferico e la diffusione di virus nella popolazione", e si è cercato di riproporre la medisima analisi, estendendone però il periodo di osservazione e aumentando i dati:

  - il lavoro originale considera il periodo relativo alla seconda metà di febbraio, mentre questo studio ha esteso tale periodo fino all'inizio della quarantena.
  - la SIMA nel paper del proprio lavoro ha prima ammesso che sia il PM10 che il PM2.5 avessero avuto ruolo di carrier in epidemie passate, ma poi ha consderato solo i superamenti per il PM10 (essendo l'unico dei due fattori ad essere regolamentato per legge da un limite giornaliero), tralasciando i dati sul PM2.5, che invece sono stati considerati daquesta analisi, con l'adozione del limite giornaliero proposto dall'OMS (istituto mondiale della sanità) per tale agente inquinante.

- sulla base delle ipotesi del team di ricerca di Harvard guidato da Xiao Wu e Rachel C. Nethery, responsabili del lavoro "Exposure to air pollution and COVID-19 mortality in the United States", si è tentato di riproporre qualcosa di analogo in suolo italiano:
  - diversi tra i fattori utilizzati dal team di ricerca americano (temperatura, posti letto disponibili, caratteristiche demografiche..) sono stati recuperati ed utilizzati per dimostrare gli effetti dell'esposizione del PM10 sulla popolazione, in relazione ai decessi da Covid-19 registrati dalla protezione civile (che ha fornito i dati sulla pandemia).
  - soggetti dello studio sono state le regioni, per cui è stato possibile mettere insieme statistiche che fossero quanto più simili a quelle prodotti dagl'istituti di ricerca consultati dal team Harvard.

I risultati hanno effettivamente dimostrato come "la specificità della velocità di incremento dei casi di contagio che ha interessato in particolare alcune zone del Nord Italia potrebbe essere legata alle condizioni di inquinamento da particolato atmosferico che ha esercitato un'azione di carrier e di boost." come sostenuto dal gruppo di ricerca italiano della SIMA.

Inoltre, si è notato come le aree più esposte all'inquinamento siano state anche quelle più colpite dalle conseguenze della nuova patologia, qualora fattori climatici esterni o misure di contenimento adeguate non siano intervenuti a limare i danni.

Sulla base di tutto questo e citando ancora una volta il lavoro della SIMA, "si suggerisce di tenere conto di questo contributo sollecitando misure restrittive di contenimento dell'inquinamento", che sempre di più sta diventando il nemico numero uno per la soppravvivenza del genere umano sulla Terra.

Ringraziamenti

Il primo ringraziamento va al relatore di questo lavoro, il Prof. Gianluca Moro, che ha reso possibile tutto ciò e ha acceso il mio personale interesse nei confronti di questa splendida disciplina.

Grazie alla mia famiglia che mi ha sempre sostenuto, che ha esultato per le mie vittorie e mi ha consolato dopo le sconfitte. Grazie ai miei amici, quelli di sempre e quelli con i quali mi sono goduto questo cammino di 3 anni. Grazie a tutte le cose che mi sono capitate, grazie a chi ha tifato per me, a chi mi ha spinto a dare sempre di più, a chi mi ha preso col morale a terra e l'ha risollevato, a chi ha condiviso con me anche 10 secondi del suo tempo, grazie a chi mi ha aiutato, a chi mi ha ostacolato, grazie a chi c'è sempre stato e a chi ci sarà sempre.

Bibliografia

[1] Francesca Costabile Daniele Contini. Does air pollution influence covid-19 outbreaks? 2020.

[2] Il Fatto Quotidiano. Coronavirus, lo studio dell'università di harvard: "correlazione tra smog e aumento del tasso di mortalità per covid 19". 2020.

[3] Irma D'Aria. Coronavirus: l'inquinamento ha aperto la strada alla diffusione dell'infezione. 2020.

[4] Harvard T.H. Chan. Air pollution linked with higher covid-19 death rates. 2020.

[5] Francesca Dominici Xiao Wu, Rachel C. Nethery. *COVID-19 PM2.5. A national study on long-term exposure to air pollution and COVID-19 mortality in the United States*. Harvard University, 2020.

[6] Jianxi Luo. *When Will COVID-19 End? Data-Driven Prediction (as of 26 April 2020)*. SUTD Data-Driven Innovation Lab, 2020.

[7] Istat-Iss. Impatto dell'epidemia covid-19 sulla mortalitÀ totale della popolazione residente primo trimestre 2020. 2020.

[8] Il Fatto Quotidiano. Coronavirus, ecco come prendersi un raffreddore può aiutare a proteggersi dal covid. gli studi su science e cell. 2020.

[9] Sydney I. Ramirez Jose Mateus Jennifer M. Dan Carolyn Rydyznski Moderbacher Stephen A. Rawlings Aaron Sutherland Lakshmanane Premkumar Ramesh S. Jadi Daniel Marrama Aravinda M. de Silva April Frazier Aaron F. Carlin Jason A. Greenbaum Bjoern Peters Florian Krammer Davey M. Smith Shane Crotty Alessandro Sette Alba Grifoni, Daniela Weiskopf. *Targets of T Cell Responses to SARS-CoV-2 Coronavirus in Humans with COVID-19 Disease and Unexposed Individuals*. Cell, 2020.

[10] Rosemary J. Boyton Daniel M. Altmann1. *PERSPECTIVECORONAVIRUS SARS-CoV-2 T cell immunity: Specificity, function, durability, and role in protection.* Science, 2020.

[11] Gianluigi de Gennaro Alessia Di Gilio Jolanda Palmisani Paolo Buono Gianna Fornari Maria Grazia Perrone Andrea Piazzalunga Pierluigi Barbieri Emanuele Rizzo Alessandro Miani Leonardo Setti, Fabrizio Passarini. *Evaluation of the potential relationship between Particulate Matter (PM) pollution and COVID-19 infection spread in Italy.* SIMA - Società Italiana di Medicina Ambientale, 2020.

[12] Elisa Murgese. *Allevamenti intensivi, polveri sottili e Covid-19.* GreenPeace, 2020.

[13] Greg G. Wolff. *Influenza vaccination and respiratory virus interference among Department of Defense personnel during the 2017–2018 influenza season.* ScienceDirect, 2020.

[14] Vito Latora Rosario Le Moli Alessandro Pluchino Giovanni Russo Andrea Rapisarda Nadia Giuffrida Chiara Zappalà Alessio E. Biondo, Giuseppe Inturri. *A Novel Methodology for Epidemic Risk Assessment: the case of COVID-19 outbreak in Italy.* Università di Catania, 2020.

[15] Silvio Brusaferro Graziano Onder, Giovanni Rezza. *Case-Fatality Rate and Characteristics of Patients Dying in Relation to COVID-19 in Italy.* JAMA, 2020.

[16] Yu Hu Wen-hua Liang Chun-quan Ou Jian-xing He Lei Liu Hong Shan Chun-liang Lei David S.C. Hui Bin Du Lan-juan Li et al. Wei-jie Guan, Zheng-yi Ni. *Clinical Characteristics of Coronavirus Disease 2019 in China.* NEJM, 2020.

[17] ISTAT Istituto nazionale di statistica. I.stat - la banca dati degl'esperti: http://dati.istat.it/.

[18] ISPRA Istituto superiore per la protezione e la ricerca ambientale. La rete ispra - arpa - appa: le biblioteche e i centri di documentazione ambientale.

[19] Inc Cedar Lake Ventures. Wheather spark - le condizioni meteo tipiche ovunque sulla terra: https://it.weatherspark.com/.

[20] Gruppo idealista. Idealista: https://www.idealista.it/.

[21] Associazione Bancaria Italiana Osservatorio del Mercato Immobiliare dell'Agenzia delle Entrate (OMI). Rapporti immobiliari residenziali.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21]