# Package ‘excessmort’

December 5, 2025

**Type** Package

**Title** Excess Mortality

**Version** 0.8.2

**Depends** R (>= 3.5.0),

**Imports** stats, datasets, dplyr, tidyr, rlang, lubridate, splines,
ggplot2, scales

**Suggests** tidycensus, knitr, rmarkdown

**VignetteBuilder** knitr

**Description** Implementation of method for estimating excess mortality and other health related outcomes from weekly or daily count data described in Acosta and Irizarry (2021) ``A Flexible Statistical Framework for Estimating Excess Mortality''.

**License** Artistic-2.0

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** xz

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Rafael A. Irizarry [aut, cre],
Rolando Acosta [aut],
Mónica Robles-Fontán [aut],
Ferenci Tamás [ctb]

**Maintainer** Rafael A. Irizarry <rafael_irizarry@dfci.harvard.edu>

**Repository** CRAN

**Date/Publication** 2025-12-05 11:50:02 UTC

## Contents

approx_demographics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
cdc_state_counts . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
collapse_age_dist . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

compute_counts . . . 4
compute_expected . . . 5
cook_records . . . 7
excess_cumulative . . . 8
excess_model . . . 9
excess_plot . . . 12
excess_stats . . . 13
expected_diagnostic . . . 14
expected_plot . . . 15
fit_ar . . . 16
get_demographics . . . 16
group_age . . . 17
louisiana_counts . . . 17
new_jersey_counts . . . 18
noleap_yday . . . 18
puerto_rico_counts . . . 18
puerto_rico_icd . . . 19
world_counts . . . 19

**Index** **20**

---

`approx_demographics` *Interpolate demographic data*

---

## Description

Interpolate yearly population estimates so that a population estimate is provided for each day of the year. The function ‘approx‘ is used with ‘rule = 2‘ for extrapolation.

## Usage

```
approx_demographics(
  demo,
  first_day,
  last_day,
  by,
  extrapolation.type = c("linear", "constant", "none"),
  ...
)
```

## Arguments

| | |
|---|---|
| `demo` | A data frame with the yearly population estimates. Must have a numeric column named year and a numeric column named population. |
| `first_day` | First day to interpolate. If missing the first day of the first year in demo is used. |
| `last_day` | Last day to interpolate. If missing the last day of the last year in demo is used. |

`by` Vector of column names to group by, for example different demographic strata. If missing it will extrapolate within each strata. To collapse all strata, define as by = NULL.

`extrapolation.type`
Type of extrapolation. Either linear, constant, or none. If none is selected the NAs are returned.

`...` additional parameters sent to the function ‘approx‘.

## Value

A data frame with dates and population estimates.

---

`cdc_state_counts` *Weekly death counts for each USA state*

---

## Description

The Center for Disease Control (CDC) provides weekly estimated death counts for each state in the USA. This data frame includes these estimates for each state along with population sizes.

- state Name of USA state
- date Corresponding date of observation
- outcome Estimated number of deaths
- population Population estimate (from the Census)

## Usage

```
data(cdc_state_counts)
```

## Format

An object of class `"data.frame"`

---

`collapse_age_dist` *Callapse age groups into broader ones*

---

## Description

Collapse a count or demographic data frame into a broader age strata.

## Usage

```
collapse_age_dist(demo, breaks)

collapse_counts_by_age(counts, breaks)
```

### Arguments

| | |
|---|---|
| demo | A data frame with population sizes for different groups that will be collapsed. |
| breaks | The new age breaks for the new, broader, age strata. |
| counts | A data frame with counts and population sizes for different groups that will be collapsed |

### Value

A age groups represented in ‘demo‘ or ‘counts‘ are grouped using the new age breaks defined by breaks but containing the populations and counts, if applicable, for age groups defined by ‘breaks‘.

### Examples

```
library(lubridate)
data(cook_records)
## define smaller subset for example
cook_demographics_subset <- cook_demographics[year(cook_demographics$date)==2021, ]
demo <- collapse_age_dist(cook_demographics_subset,
                          breaks = c(0, 20, 40, 60, 80, Inf))
```

---

## compute_counts *Compute counts*

---

### Description

Compute counts for groups from individual records

### Usage

```
compute_counts(
  dat,
  by = NULL,
  demo = NULL,
  date = "date",
  age = "age",
  agegroup = "agegroup",
  breaks = NULL
)
```

### Arguments

| | |
|---|---|
| dat | The data frame with the individual records |
| by | A character vector with the column names the define the groups for which we will compute counts |
| demo | A data frame with population sizes for each time point for each of the groups for which we will compute counts |

| | |
|---|---|
| `date` | The column name of the column that contains dates |
| `age` | The column name of the column that contains age |
| `agegroup` | The column name of the column that contains agegroup |
| `breaks` | The ages that define the agegroups |

## Details

This is helper function that helps convert individual records data, in which each death is a row, to a count data frame where each row is a date. It is particulalry helpful for defining agegroups. If you provide the 'breaks' argument it will automaticall divided the data and provide the counts for each age strata. You can also select variables to group by using the 'by' argument. One can provide a data frame with demogrpahic inform through the 'demo' argument. This tabe must have the population size for each group for each data.

## Value

A data frame with counts for each group for each date with population sizes, if demo was provided.

## Examples

```
library(lubridate)
data("cook_records")
the_breaks <- c(0, 20, 40, 60, 75, Inf)

## take subset for example
cook_records_subset <- cook_records[year(cook_records$date)==2021, ]
cook_demographics_subset <- cook_demographics[year(cook_demographics$date)==2021, ]

cook_counts <- compute_counts(cook_records_subset,
                       demo = cook_demographics_subset,
                       by = c("agegroup", "race", "sex"),
                       breaks = the_breaks)
```

---

# `compute_expected` *Compute expected counts for each day*

---

## Description

Compute the expected death count for each unit of time. We assume counts are over-dispersed Poisson distributed with a trend that accounts for slow, year-to-year, changes in death rate across time and a seasonal effect. The function takes a data frame with dates and counts and returns the data frame with the expected counts as a new column. It also returns a logical column that is 'TRUE' if that entry was used in the estimation procedure.

### Usage

```
compute_expected(
  counts,
  exclude = NULL,
  include.trend = TRUE,
  trend.knots.per.year = 1/7,
  extrapolate = TRUE,
  harmonics = 2,
  frequency = NULL,
  weekday.effect = FALSE,
  keep.components = TRUE,
  verbose = TRUE
)
```

### Arguments

| | |
|---|---|
| `counts` | A data frame with dates, counts, and population size. |
| `exclude` | A list of dates to exclude when fitting the model. This is typically the period for which you will later estimate excess counts. |
| `include.trend` | Logical that determines if a slow trend is included in the model. |
| `trend.knots.per.year` | Number of knots per year used for the time trend |
| `extrapolate` | Logical that determines if the slow trend is extrapolated past the range of data used to fit. This |
| `harmonics` | Number of harmonics to include in the seasonal effect |
| `frequency` | Number of data points per year. If not provided, the function attempts to estimate it |
| `weekday.effect` | A logical that determines if a day of the week effect is included in the model |
| `keep.components` | A logical that if 'TRUE' forces the function to return the estimated trend, seasonal model, and weekday effect, if included in the model. |
| `verbose` | A logical that if 'TRUE' makes function prints out updates on the estimation procedure |

### Details

Periods for which excess deaths will be estimated should be excluded when estimating expected counts. These can be supplied via the 'exclude' argument. Note that If 'extrapolate' is 'TRUE', the default, the time trend will be extrapolated following the estimated trend. If 'extrapolate' is 'FALSE' the trend is assumed to be a constant equal to the estimate on the last day before extrapolation. If the period for which excess deaths are estimated is long, extrapolation should be used with caution. We highly recommend exploring the estimated expected counts with the 'expected_diagnostics' function.

### Value

The ‘counts‘ data.frame with two columns added: ‘expected‘ and ‘excluded‘. The ‘expected‘ column is the estimated expected value of the counts for that date. The ‘excluded‘ column is a logical vector denoting if that date was excluded when estimating the expected value.

If the argument ‘keep.components‘ is ‘TRUE‘ a list is returned with ‘counts‘ data.frame in the first component, the estimated trend in the second, the estimated seasonal effect in the third and the estimated weekday effects in the fourth.

### Examples

```
data(new_jersey_counts)
exclude_dates <- as.Date("2012-10-29") + 0:180
counts <- compute_expected(new_jersey_counts, exclude = exclude_dates, weekday.effect = TRUE)
library(ggplot2)
expected_plot(counts)
```

---

## cook_records *Cook County Medical Examiner Records*

---

### Description

A data frame with each row represening a death. The data includes death date and demographic information. When you load this dataset you also load ‘cook_demographics‘ which includes the population size for each demographice group by date.

### Usage

```
data("cook_records")
```

### Format

A data frame with these columns

**sex** Sex of the deceased

**age** Age of the deceased

**race** Race of the deceased

**residenceplace** Residence placed of the deceased

**date** Date of the death

**cause_1** Reported cause of death

**type_of_death** Type of death

## excess_cumulative *Compute cumulative excess deaths*

### Description

This function takes the output of the ‘excess_model‘ function, a start date, and end date and calculates excess mortality and standard errors.

### Usage

```
excess_cumulative(fit, start, end)
```

### Arguments

| | |
|---|---|
| fit | The output of ‘excess_model‘ |
| start | The start date |
| end | The end date |

### Value

A data frame with the following columns

**date** The date

**observed** The observed excess mortality,which is the sum of observed minus expected until that date

**sd** The standard deviation for excess mortality for that date if year is typical

**fitted** The estimated of excess mortality based on the estimated smooth event effect curve

**se** The standard error for ‘fitted‘

### Examples

```
data(new_jersey_counts)
exclude_dates <- as.Date("2012-10-29") + 0:180
control_dates <- seq(min(new_jersey_counts$date), min(exclude_dates) - 1, by = "day")
f <- excess_model(new_jersey_counts,
start = as.Date("2012-09-01"),
end = as.Date("2013-09-01"),
exclude = exclude_dates,
model = "correlated",
weekday.effect = TRUE,
control.dates = control_dates)

excess_cumulative(f,
start = as.Date("2017-12-15"),
end = as.Date("2017-12-21") )
```

excess_model *Fit excess count model*

## Description

This function estimates the increase in the rate for a count time series relative to the rate for a typical year. Two options are available: 1 - model the rate increase as a smooth function and estimate this function or 2 - estimate the total excess in intervals. For 1 an ‘event‘ date can be provided and a discontinuity included in the model. You can do either 1 or 2 or both.

## Usage

```
excess_model(
  counts,
  start = NULL,
  end = NULL,
  knots.per.year = 12,
  event = NULL,
  intervals = NULL,
  discontinuity = TRUE,
  model = c("quasipoisson", "poisson", "correlated"),
  exclude = NULL,
  include.trend = TRUE,
  trend.knots.per.year = 1/7,
  harmonics = 2,
  frequency = NULL,
  weekday.effect = FALSE,
  control.dates = NULL,
  max.control = 5000,
  order.max = 14,
  aic = TRUE,
  maxit = 25,
  epsilon = 1e-08,
  alpha = 0.05,
  min.rate = 1e-04,
  keep.counts = FALSE,
  keep.components = TRUE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `counts` | A data frame with date, count and population columns. |
| `start` | First day of interval to which model will be fit |
| `end` | Last day of interval to which model will be fit |
| `knots.per.year` | Number of knots per year used for the fitted smooth function |

| | |
|---|---|
| event | If modeling a discontinuity is desired, this is the day in which it happens |
| intervals | Instead of ‘start‘ and ‘end‘ a list of time intervals can be provided and excess is computed in each one |
| discontinuity | Logical that determines if discontinuity is allowed at ‘event‘ |
| model | Which version of the model to fit |
| exclude | Dates to exclude when computing expected counts |
| include.trend | Logical that determines if a slow trend is included in the model. |
| trend.knots.per.year | Number of knots per year used by ‘compute_expected‘ to estimate the trend for the expected counts |
| harmonics | Number of harmonics used by ‘compute_expected‘ to estimate seasonal trend |
| frequency | Number of observations per year. If not provided an attempt is made to calculate it |
| weekday.effect | Logical that determins if a day of the week effects is included in the model. Should be ‘FALSE‘ for weekly or monthly data. |
| control.dates | When ‘model‘ is set to ‘correlated‘, these dates are used to estimate the covariance matrix. The larger this is the slower the function runs. |
| max.control | If the length of ‘control.dates‘ is larger than ‘max.control‘ the function stops. |
| order.max | Larges order for the Autoregressive process used to model the covariance structure |
| aic | A logical that determines if the AIC criterion is used to selected the order of the AR process |
| maxit | Maxium number of iterations for the IRLS algorithm used when ‘model‘ is ‘correlated‘ |
| epsilon | Difference in deviance requried to declare covergenace of IRLS |
| alpha | Percentile used to define what is outside the normal range |
| min.rate | The estimated expected rate is not permited to go below this value |
| keep.counts | A logical that if ‘TRUE‘ forces the function to include the data used to fit the expected count model. |
| keep.components | A logical that if ‘TRUE‘ forces the function to return the estimated trend, seasonal model, and weekday effect, if included in the model. Ignored if expected counts already provided or ‘keep.counts‘ is ‘FALSE‘. |
| verbose | Logical that determines if messages are displayed |

## Details

Three versions of the model are available: 1 - Assume counts are Poisson distributed, 2 - assume counts are overdispersed Poisson, or 3 - assume a mixed model with correlated errors. The second is the default and recommended for weekly count data. For daily counts we often find evidence of correlation and recommend the third along with setting ‘weekday.effect = TRUE‘.

If the ‘counts‘ object includes a ‘expected‘ column produced by ‘compute_expected‘ these are used as the expected counts. If not, then these are computed.

### Value

If only ‘intervals‘ are provided a data frame with excess estimates described below for ‘excess‘. if ‘start‘ and ‘end‘ are provided the a list with the following components are included:

**date** The dates for which the estimate was computed

**observed** The observed counts

**expected** The expected counts

**fitted** The fitted curve for excess counts

**se** The point-wise standard error for the fitted curve

**population** The population size

**sd** The standard deviation for observed counts on a typical year

**cov** The estimated covariance matrix for the observed counts

**x** The design matrix used for the fit

**betacov** The covariance matrix for the estimated coefficients

**dispersion** The estimated overdispersion parameter

**detected_intervals** Time intervals for which the 1 - ‘alpha‘ confidence interval does not include 0

**ar** The estimated coefficients for the autoregressive process

**excess** A data frame with information for the time intervals provided in ‘itervals‘. This includes start, end, observed death rate (per 1,000 per year), expected death rate, standard deviation for the death rate, observed counts, expected counts, excess counts, standard deviation

### Examples

```
data(cdc_state_counts)
counts <- cdc_state_counts[cdc_state_counts$state == "Massachusetts", ]
exclude_dates <- c(seq(as.Date("2017-12-16"), as.Date("2018-01-16"), by = "day"),
seq(as.Date("2020-01-01"), max(cdc_state_counts$date), by = "day"))
f <- excess_model(counts,
exclude = exclude_dates,
start = min(counts$date),
end = max(counts$date),
knots.per.year = 12)
data(new_jersey_counts)
exclude_dates <- as.Date("2012-10-29") + 0:180
control_dates <- seq(min(new_jersey_counts$date), min(exclude_dates) - 1, by = "day")
f <- excess_model(new_jersey_counts,
start = as.Date("2012-09-01"),
end = as.Date("2013-09-01"),
exclude = exclude_dates,
model = "correlated",
weekday.effect = TRUE,
control.dates = control_dates)
```

excess_plot *Plot results from fitted excess count model*

## Description

Plot results from fitted excess count model

## Usage

```
excess_plot(fit, title = "", ylim = NULL, show.data = TRUE, alpha = 0.05)
```

## Arguments

| | |
|---|---|
| fit | The output from ‘excess_model‘ |
| title | A title to add to plot |
| ylim | A vector with two numbers that determines the kimits for the y-axis |
| show.data | A logical that determines if the observed percent changes are shown |
| alpha | 1 - ‘alpha‘ confidence intervals are shown |

## Value

An ggplot object containing the plot.

## Examples

```
data(new_jersey_counts)
exclude_dates <- as.Date("2012-10-29") + 0:180
control_dates <- seq(min(new_jersey_counts$date), min(exclude_dates) - 1, by = "day")
f <- excess_model(new_jersey_counts,
start = as.Date("2012-09-01"),
end = as.Date("2013-09-01"),
exclude = exclude_dates,
model = "correlated",
weekday.effect = TRUE,
control.dates = control_dates)

library(ggplot2)
excess_plot(f)
```

---

`excess_stats` *Excess counts in an interval*

---

## Description

Helper function to compute excess deaths statistics for a

## Usage

```
excess_stats(
  start,
  end,
  obs,
  mu,
  cov,
  pop,
  frequency,
  fhat = NULL,
  X = NULL,
  betacov = NULL
)
```

## Arguments

| | |
|---|---|
| `start` | First day of interval |
| `end` | Last day of interval |
| `obs` | Observed counts |
| `mu` | Expected counts |
| `cov` | Covariance matrix for percent change |
| `pop` | Population size |
| `frequency` | Observations per year |
| `fhat` | Estimated percent increase |
| `X` | Design matrix used to estimate fhat |
| `betacov` | Covariance matrix for parameter estimates used to estimate fhat |

expected_diagnostic *Diagnostic Plots for Model Fit*

### Description

Check mean model fit via diagnostic figures of the model components

### Usage

```
expected_diagnostic(
  expected,
  start = NULL,
  end = NULL,
  color = "#D22B2B",
  alpha = 0.5
)
```

### Arguments

| | |
|---|---|
| expected | The output from ‘compute_expected‘ with ‘keep.components = TRUE‘ |
| start | First day to show |
| end | Last day to show |
| color | Color for the expected curve |
| alpha | alpha blending for points |

### Value

A list with six ggplot objects: ‘population‘ is a time series plot of the population. ‘seasonal‘ is a plot showing the estimated seasonal effect. ‘trend‘ is a time series plot showing the estimated trend. ‘weekday‘ is a plot of the estimated weekday effects if they were estimated. ‘expected‘is a time series plot of the expected values. ‘residual‘ is a time series plot of the residuals.

### Examples

```
library(dplyr)
library(lubridate)
library(ggplot2)

flu_season <- seq(make_date(2017, 12, 16), make_date(2018, 1, 16), by = "day")

exclude_dates <- c(flu_season, seq(make_date(2020, 1, 1), today(), by = "day"))

res  <- cdc_state_counts %>%
 filter(state == "Massachusetts") %>%
 compute_expected(exclude = exclude_dates,
                 keep.components = TRUE)
```

```
p <- expected_diagnostic(expected = res, alpha = 0.50)

p$population
p$seasonal
p$trend
p$expected
p$residual
```

---

`expected_plot` *Plot Expected Counts*

---

## Description

Check if expected counts fit data

## Usage

```
expected_plot(
  expected,
  title = "",
  start = NULL,
  end = NULL,
  ylim = NULL,
  weekly = FALSE,
  color = "#3366FF",
  alpha = 0.5
)
```

## Arguments

| | |
|---|---|
| `expected` | The output from ‘compute_expected‘ |
| `title` | A title to add to plot |
| `start` | First day to show |
| `end` | Last day to show |
| `ylim` | A vector with two numbers that determines the kimits for the y-axis |
| `weekly` | Logical that determines if data should be summarized into weekly counts |
| `color` | Color for the expected curve |
| `alpha` | alpha blending for points |

## Value

A ggplot object containing a plot of the original counts and the estimated expected values.

### Examples

```
data(new_jersey_counts)
exclude_dates <- as.Date("2012-10-29") + 0:180
e <- compute_expected(new_jersey_counts, exclude = exclude_dates, weekday.effect = TRUE)

library(ggplot2)
expected_plot(e, start = as.Date("2012-09-01"), end = as.Date("2013-09-01"))
```

---

## fit_ar *Fit an ar model to residuals from expected counts*

---

### Description

Helper function to estimate autoregressive mode

### Usage

```
fit_ar(counts, control.dates = NULL, order.max = 5, aic = FALSE, plot = FALSE)
```

### Arguments

| | |
|---|---|
| counts | Output from ‘compute_excpected‘ |
| control.dates | Dates to use to estimate covariance |
| order.max | Maximum order of autoregressive process |
| aic | Logical that determines if AIC is used |
| plot | logical that determines if an autocorrelation plot is generated for exploration purposes |

---

## get_demographics *Get demographic data from Census*

---

### Description

Get demographic data from Census

### Usage

```
get_demographics(
  geography = "state",
  state,
  county = NULL,
  years = 2018,
  vars = c("SEX", "AGEGROUP", "RACE", "HISP")
)
```

### Arguments

| | |
|---|---|
| `geography` | state or county |
| `state` | name of the state |
| `county` | name of the county |
| `years` | vector of years for which we obtain data |
| `vars` | names of variables that define strat of which we want population estimates |

---

## `group_age` *Assign age to group*

---

### Description

Assign age to group

### Usage

```
group_age(age, breaks)
```

### Arguments

| | |
|---|---|
| `age` | Vector of ages |
| `breaks` | Ages that define strata |

### Value

A factor that groups the ages into the age groups defined by ‘breaks‘.

---

## `louisiana_counts` *Louisiana daily mortality*

---

### Description

A data frame with Louisiana daily mortality counts from 2003-01-01 to 2016-12-31, which includes the day Katrina made landfall on 2015-08-29. The outcome column includes the number of deaths for that day.

### Usage

```
data("louisiana_counts")
```

### Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1461 rows and 3 columns.

## new_jersey_counts *New Jersey daily mortality*

### Description

A data frame with Louisiana daily mortality counts from 2007-01-01 to 2015-12-31 which includes the day Sandy made landfall on 2012-10-29. The outcome column includes the number of deaths for that day.

### Usage

```
data("new_jersey_counts")
```

### Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 3287 rows and 3 columns.

## noleap_yday *Compute year of the day ingoring Feb 29*

### Description

Compute year of the day ingoring Feb 29

### Usage

```
noleap_yday(x)
```

### Arguments

| | |
|---|---|
| `x` | date |

## puerto_rico_counts *Puerto Rico daily mortality*

### Description

A data frame with Puerto Rico daily mortality counts, stratified by agegroup, from 1985 to 2020. which includes the days hurricanes Hugo, Georges, and Maria made landfall on 1989-09-18, 1998-09-21, and 2017-09-20, respectively. The outcome column includes the number of deaths for that day. This data frame also includes population counts and estimates for the period. Population counts for 1985 to 2000 are interpolations of decennial census products since 1980 to 2000. For 2000 onward, we use interpolated population estimates (PEP).

## Usage

```
data("puerto_rico_counts")
```

## Format

An object of class `data.table` (inherits from `data.frame`) with 499644 rows and 5 columns.

---

`puerto_rico_icd` *Puerto Rico daily mortality by cause of death*

---

## Description

A data frame with Puerto Rico daily mortality counts, stratified by cause of death from 1999 to 2020. which includes the day hurricanes Maria made landfall on 2017-09-20. The outcome column includes the number of deaths for that day for that ICD-10 code. The object ‘icd‘ is included to show the description of each ICD-10 code.

## Usage

```
data("puerto_rico_icd")
```

## Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 262980 rows and 4 columns.

---

`world_counts` *Weekly death counts for several countries*

---

## Description

A data frame with weekly death counts from multiple countries that includes the year 2020. Each country has a different range of data and most countries have at least 5 years of pre-2020 data. The original data was collated by the Financial Times .

## Usage

```
data(world_counts)
```

## Format

An object of class `"data.frame"`

## Details

- country Name of the country
- date Corresponding date of observation
- outcome Estimated number of deaths
- population Population estimate (from the Census)

# Index


∗ **datasets**
    cdc_state_counts, 3
    cook_records, 7
    louisiana_counts, 17
    new_jersey_counts, 18
    puerto_rico_counts, 18
    puerto_rico_icd, 19
    world_counts, 19

approx_demographics, 2

cdc_state_counts, 3
collapse_age_dist, 3
collapse_counts_by_age
        (collapse_age_dist), 3
compute_counts, 4
compute_expected, 5
cook_demographics (cook_records), 7
cook_records, 7

excess_cumulative, 8
excess_model, 9
excess_plot, 12
excess_stats, 13
expected_diagnostic, 14
expected_plot, 15

fit_ar, 16

get_demographics, 16
group_age, 17

icd (puerto_rico_icd), 19

louisiana_counts, 17

new_jersey_counts, 18
noleap_yday, 18

puerto_rico_counts, 18
puerto_rico_icd, 19

world_counts, 19