Analysis of seasonal macrolide use and simulated GISP data

Scott Olesen

Antibiotic use

Macrolide use data are in the `use_data/` folder.

```
use_year_data = read_tsv('use_data/use_by_year.tsv')
use_age_data = read_tsv('use_data/use_by_age.tsv')
use_region_data = read_tsv('use_data/use_by_region.tsv')

use_year_data
```

```
## # A tibble: 60 x 3
##     year fill_month claims_per_1k_member
##    <int>      <int>                <dbl>
##  1  2011          0                 19.2
##  2  2011          1                 19.3
##  3  2011          2                 18.1
##  4  2011          3                 13.8
##  5  2011          4                 12.9
##  6  2011          5                 10.4
##  7  2011          6                  7.91
##  8  2011          7                  9.10
##  9  2011          8                 12.2
## 10  2011          9                 14.1
## # ... with 50 more rows
```

We fit:

- one model to all the data
- one model to each year
- one model for each age group
- one model for each Census region

The fit is

$$y \sim A\cos\left[\omega(t-P)\right] + C,$$

where y is the MIC (2-fold dilution), A is the amplitude, $\omega = 2\pi/(12 \text{ months})$, t is the month of the year, P is the phase delay, and C is the year-round average use.

```
omega = 2 * pi / 12

# function to run the non-linear fit
use_nls_f = function(df) {
  nls(claims_per_1k_member ~ A * cos(omega * (fill_month - phase)) + offset,
      start = list(A = 5.0, phase = 0.0, offset = 10.0),
      data = df)
}

# run non-linear fits on subsets of the data
use_model_f = function(dat, ...) {
```

```
  groups = quos(...)

  dat %>%
    group_by_at(groups) %>%
    nest() %>%
    mutate(model = map(data, use_nls_f),
           result = map(model, ~ tidy(., conf.int = TRUE)))
}

use_overall = use_model_f(use_year_data)
use_year = use_model_f(use_year_data, year)
use_age = use_model_f(use_age_data, age_group)
use_region = use_model_f(use_region_data, region)
```

All the results are in this table, which recapitulates the paper's Supplemental Table.

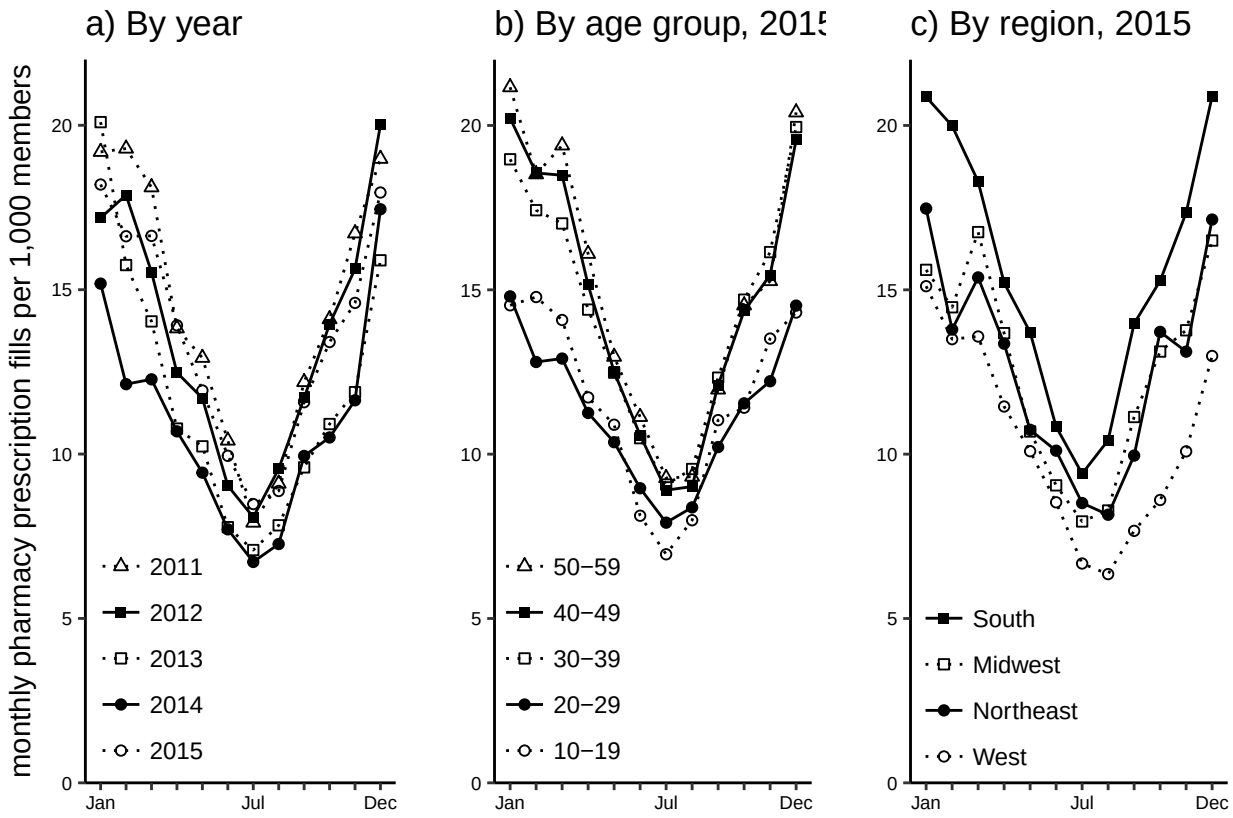
```
bind_rows(
  use_overall %>% mutate(group_name = 'overall', group = 'overall'),
  use_year %>% mutate(group_name = 'year', group = as.character(year)),
  use_age %>% mutate(group_name = 'age', group = age_group),
  use_region %>% mutate(group_name = 'region', group = region)
) %>%
  select(group_name, group, result) %>%
  unnest() %>%
  mutate(group_name = factor(group_name, levels=c('overall', 'year', 'age', 'region')),
         sig = if_else(p.value < 0.05, '*', ' '),
         range = sprintf('%.2f (%.2f to %.2f) %s', estimate, conf.low, conf.high, sig)) %>%
  select(group_name, group, term, range) %>%
  spread(term, range) %>%
  select(group_name, group, A, phase, offset) %>%
  arrange(group_name, group) %>%
  kable(caption = 'Macrolide use fits')
```

Table 1: Macrolide use fits

group_name	group	A	phase	offset
overall	overall	4.75 (4.14 to 5.37) *	0.03 (-0.22 to 0.28)	12.84 (12.40 to 13.28) *
year	2011	5.41 (4.73 to 6.09) *	0.14 (-0.10 to 0.38)	14.40 (13.92 to 14.88) *
year	2012	4.97 (4.10 to 5.84) *	-0.19 (-0.52 to 0.15)	13.56 (12.95 to 14.18) *
year	2013	5.00 (3.65 to 6.36) *	0.14 (-0.38 to 0.66)	11.82 (10.87 to 12.78) *
year	2014	3.91 (2.63 to 5.20) *	-0.17 (-0.81 to 0.47)	10.91 (10.00 to 11.82) *
year	2015	4.56 (3.96 to 5.15) *	0.17 (-0.08 to 0.42)	13.51 (13.09 to 13.93) *
age	10-19	3.60 (2.99 to 4.21) *	-0.00 (-0.33 to 0.32)	10.99 (10.56 to 11.42) *
age	20-29	3.29 (2.71 to 3.88) *	-0.07 (-0.41 to 0.27)	11.01 (10.59 to 11.42) *
age	30-39	5.37 (4.65 to 6.08) *	-0.05 (-0.31 to 0.20)	14.16 (13.66 to 14.67) *
age	40-49	5.54 (4.90 to 6.18) *	0.06 (-0.16 to 0.28)	13.74 (13.29 to 14.19) *
age	50-59	5.53 (4.85 to 6.21) *	0.11 (-0.12 to 0.35)	13.79 (13.31 to 14.27) *
region	Midwest	4.50 (3.84 to 5.16) *	-0.02 (-0.30 to 0.27)	12.47 (12.00 to 12.93) *
region	Northeast	4.28 (3.57 to 5.00) *	0.10 (-0.22 to 0.42)	11.96 (11.45 to 12.46) *
region	South	5.77 (5.10 to 6.44) *	-0.14 (-0.36 to 0.08)	15.14 (14.67 to 15.61) *
region	West	3.73 (3.16 to 4.30) *	0.50 (0.21 to 0.80) *	9.71 (9.31 to 10.11) *

To recapitulate the paper's Figure 1:

```
x_axis_labels = c('Jan', '', '', '', '', '', 'Jul', '' ,'', '', '', 'Dec')

# function to plot the three macrolide use subfigures
use_plot_f = function(df, group_, shape_values, linetype_values, title,
                      legend_x = 0.21, legend_y = 0.18) {
  group_ = enquo(group_)

  p = ggplot(df, aes_(~fill_month, ~claims_per_1k_member,
                 group = group_, shape = group_, linetype = group_)) +
    geom_point() +
    geom_line() +
    scale_shape_manual(values = shape_values) +
    scale_linetype_manual(values = linetype_values) +
    scale_x_discrete('', labels = x_axis_labels) +
    ylab('') +
    scale_y_continuous(limits = c(0, 22), expand = c(0, 0)) +
    theme_classic() +
    theme(legend.position = c(legend_x, legend_y),
        axis.text = element_text(color = 'black', size = 7),
        legend.title = element_blank(),
        legend.background = element_blank()) +
    ggtitle(title)

  p
}

p1 = use_year_data %>%
  mutate_at(vars(fill_month, year), factor) %>%
  use_plot_f(year, c(2, 15, 0, 19, 1), rep(c(3, 1), 5), 'a) By year') +
  ylab('monthly pharmacy prescription fills per 1,000 members')

p2 = use_age_data %>%
  filter(year == 2015) %>%
  mutate(fill_month = factor(fill_month), age = fct_rev(factor(age_group))) %>%
  use_plot_f(age, c(2, 15, 0, 19, 1), rep(c(3, 1), 5), 'b) By age group, 2015')

p3 = use_region_data %>%
  filter(year == 2015) %>%
  mutate(region = factor(region, levels = c('South', 'Midwest', 'Northeast', 'West')),
         fill_month = factor(fill_month)) %>%
  use_plot_f(region, c(15, 0, 19, 1), rep(c(1, 3), 4), 'c) By region, 2015', 0.28, 0.14)

# show a single combined figure
grid.arrange(
  p1, p2, p3, nrow=1, padding=unit(0.1, 'line')
)
```



Resistance

The simulated resistance data are in the `data/` folder. The columns show:

- the clinic (of 40 total) that the isolate was collected from
- the year of isolation
- the month of isolation (0 = January, 11 = December)
- the 2-fold dilution `y`
- the months since the start of the dataset `t`
- a combined clinic/year label

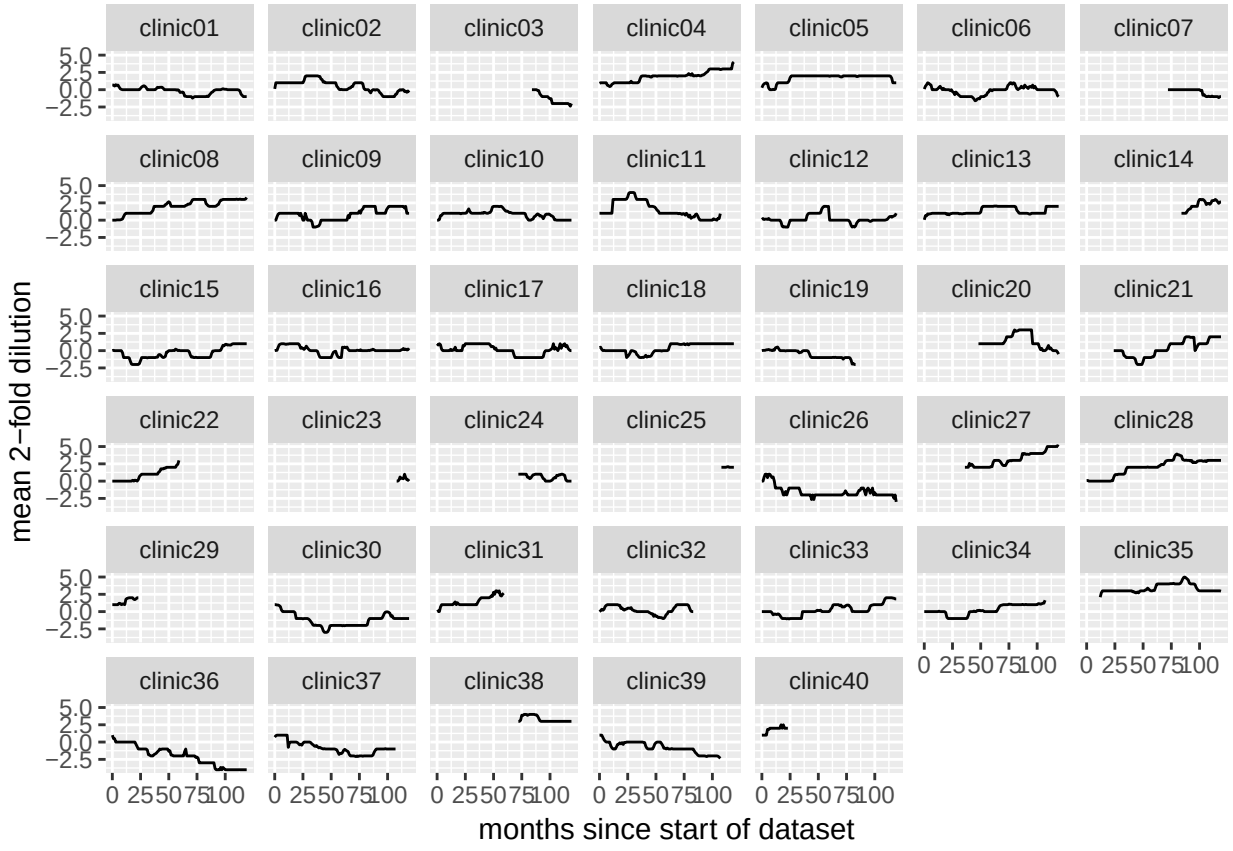
```
res_dat = read_tsv('resistance_data/simulated_gisp_data.tsv') %>%
  mutate(t = (year - min(year)) * 12 + month,
         clinic_year = str_c(clinic, '_', year))

res_dat
```

```
## # A tibble: 56,047 x 7
##    isolate_id clinic    year month     y     t clinic_year
##         <int> <chr>    <int> <int> <int> <dbl> <chr>
##  1          1 clinic01     1     0     1    0. clinic01_1
##  2          2 clinic01     1     0     0    0. clinic01_1
##  3          3 clinic01     1     0     0    0. clinic01_1
##  4          4 clinic01     1     0     1    0. clinic01_1
##  5          5 clinic01     1     0     1    0. clinic01_1
##  6          6 clinic01     1     0     1    0. clinic01_1
##  7          7 clinic01     1     0     1    0. clinic01_1
```

```
##  8          8 clinic01     1     0     1    0. clinic01_1
##  9          9 clinic01     1     0     1    0. clinic01_1
## 10         10 clinic01     1     0     1    0. clinic01_1
## # ... with 56,037 more rows
```

The data were simulated to reproduce some of the features of the original GISP dataset. Notably, there are secular trends in the data, which differ between clinics, and the seasonal pattern is not obvious at all to the eye. The plot shows the monthly average 2-fold dilutions in each clinic and month.

```
res_dat %>%
  group_by(clinic, t) %>%
  summarize(y = mean(y)) %>%
  ggplot(aes(t, y)) +
  facet_wrap(~ clinic) +
  geom_line() +
  xlab('months since start of dataset') +
  ylab('mean 2-fold dilution')
```



Note that these data are *not* intended to reflect the actual values in the GISP data. They are only for demonstration of the methods; *not* for showing how the precise results of the paper came about.

The nonlinear model uses R's `nls` function. There are many clinic/year combinations, so rather than enumerating them all in a formula passed to nls, we use `model.matrix` and matrix multiplication `%*%`, storing the slope and intercept estimates in vectors, rather than as individual numbers.

The slope values correspond to the $B_{c(i)}$ and the intercepts to the $C_{c(i)}$ terms in the paper.

```
# number of clinic/year combinations in the data
n_cys = length(unique(res_dat$clinic_year))
```

```
# create a model matrix: rows represent data rows, columns represent each of
# the clinic/years. Most entries are 0; 1 means that this data row is from the
# corresponding clinic/year.
model_matrix = model.matrix(~ 0 + clinic_year, res_dat) %>%
  set_colnames(str_replace(colnames(.), '^clinic_year', ''))

stopifnot(dim(model_matrix) == c(nrow(res_dat), n_cys))

# as a first guess for clinic/year intercepts, use the mean values
start_intercepts = res_dat %>%
  group_by(clinic_year) %>%
  summarize(y = mean(y)) %>%
  arrange(clinic_year) %T>%
  # check that the order of these values matches the model matrix columns
  { stopifnot(all(.$clinic_year == colnames(model_matrix))) } %>%
  pull(y)

# as a first guess for clinic/year slopes, just use zero
start_slopes = rep(0, n_cys)

# sinusoidal + linear fit function
omega = 2 * pi / 12
fit_f = function(month, A, phase, slope, intercept) {
  intercept_term = drop(model_matrix %*% intercept)
  slope_term = drop(model_matrix %*% slope) * month
  A * sin(omega * (month - phase)) + slope_term + intercept_term
}

# check if the model has been run previously. if it has, don't bother running
# it again, since it takes a few minutes.
model_fn = 'resistance_model_values.tsv'
if (file.exists(model_fn)) {
  res_model_values = read_tsv(model_fn)
} else {
  # fit the model
  res_model = nls(y ~ fit_f(month, A, phase, slope, intercept),
              start = list(A = 0.5, phase = 0.0, intercept = start_intercepts, slope = start_slopes),
              data = res_dat)

  # extract the interesting bits
  model_values1 = summary(res_model)$coefficients %>%
    set_colnames(c('estimate', 'std.error', 'statistic', 'p.value')) %>%
    as_tibble(rownames = 'term')

  model_values2 = confint.default(res_model) %>%
    set_colnames(c('conf.low', 'conf.high')) %>%
    as_tibble(rownames = 'term')

  res_model_values = left_join(model_values1, model_values2, by = 'term')

  # save those values
  write_tsv(res_model_values, model_fn)
}
```

We interrogate the `model` object to get the point estimates, standard errors, and confidence intervals for the amplitude A and phase terms. The data were generated with amplitude $A = 0.1$ and phase 1.

```
res_model_values %>%
  filter(term %in% c('A', 'phase')) %>%
  kable()
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
A	0.0938403	0.0021367	43.91917	0	0.0896525	0.0980281
phase	0.9535219	0.0428826	22.23565	0	0.8694736	1.0375702