Unmasking Seasonal Cycles in Human Fertility: How holiday sex and fertility cycles shape birth seasonality
Supplementary Material

Symul L., Hsieh P., Moreno C.R.C., Skene D.J., Holmes S., Martinez M.

This is the pdf-rendering of the analyses performed for the manuscript "Unmasking Seasonal Cycles in Human Fertility: How holiday sex and fertility cycles shape birth seasonality". The R markdown (.Rmd) file used to generate this pdf can be found on [link]. Each section of this pdf is a standalone Rmd file (also available on [link]) and can be run separately. The data for each section is available on the same github repository except for the sections "App data processing and filtering" and "App data aggregation" as the raw App data cannot be shared publicly to respect users' privacy and data ownership.

Contents

List of Tables

List of Figures

1 Summary of the material and methods

We tested three hypotheses on birth seasonality by using sexual activity data collected from over 500,000 individuals, representing 180 million days of active tracking data. The data were collected from Clue, a women's health mobile phone app (Fig 1b, main text). Since the app is specifically built for menstrual cycle tracking, we assume all users are female or menstruating individuals. We used data from individuals residing in the United States, United Kingdom, France, and Brazil, that represent a geographically and culturally diverse set of countries with a high number of users. We also collated time series of births for each country. Due to the size of the US and Brazil, US data were focused on two regions, California, which is a Western state, and the Northeast CDC Census Region, which includes 9 states on the Northeast Coast. Similarly, in Brazil, we focused on two culturally and geographically distinct Regions, the Central-West region, which includes the capital Brasilia, and the Northeast, which is on the Northern Atlantic Coast. Overall, we had six geographic locations for all analyses: UK, France, Central-West Brazil, Northeast Brazil, California, and the Northeastern U.S (Fig 1a, main text). The following are numbers of Clue users by location: UK (n = 133,387), France (n = 126,634), Central-West Brazil (n = 160,896), Northeast Brazil (n = 23,127), California (n = 39,330) and the Northeastern US (n = 51,780). Data were combined with mathematical models developed explicitly to test our hypotheses.

1.1 Clue Data

De-identified individual-level daily records were obtained from n = 535,154 Clue users. Each individual self-reported some combination of sexual activity, contraception use, menstruation and other sexual and reproductive health indicators daily. As with all tracking apps, however, there was variation in the fidelity with which users tracked. Overall, we obtained n = 55,110,476 daily records from July 2017- July 2019, representing a period of 180 million days of active tracking. Due to changes in tracking fidelity, we used a rolling window to identify periods of active tracking for each individual. Sexual activity was reported as three types: protected, unprotected or withdrawal sex.

For each geographic location and sex type, aggregated time-series of daily sexual activity were constructed. These time series represented the daily sex rate in the population as defined by the fraction of active users reporting a specific type of sex. Because these data are not ideal for estimating the absolute amount of sexual activity in a population (i.e., due to biases in app user demographics and variation in tracking fidelity), we de-trended the time series by removing the long-term moving average (Fig 1c, main text) which allowed us to focus on variation around the mean, particularly variation tied to day-of-week, holidays, and seasons.

To infer how sexual activity is impacted by holidays, weekends, and season, we fitted a statistical model to our data to measure relative sexual activity. This model allowed us to predict sexual activity for any location and any given calendar day in the year, based on the day-of-week, month, and its proximity to a local holiday (Fig 2, main text). Due to the large sample size, we were able to fit this statistical model for all combinations of sex type as well as birth control categories (e.g., contraceptive pills, condoms, etc.) (see section 5.3.3). Since sexual activity was similar among groupings, here we present results based on unprotected sex reported by users of any age and any birth control category.

Our model was structured as follows: $\text{sex}[i,j] = \sigma(\alpha_i \, \text{h}[i] + \beta_j \, \text{wdm}[j])$ where σ represents logistic function ($\sigma(x) = \frac{1}{1+e^{-x}}$) and each 365 days of the year is defined by a specific combination of i and j. The first term represents the contribution of the holiday to sexual activity and the second term represents the contribution of weekday and month. We assumed all holidays included in our models have a day off, with the exception of Valentine's Day and the Dia dos Namorados (i.e., Brazilian Valentine's Day). For holidays with a day off, we estimated a fixed amount of sexual activity in the first term and set the second term to its intercept value. Whereas, for Valentine's and Dia dos Namorados, was estimated for the holiday and combined with the second term for the weekday and month. This accounted for the fact that these holidays fall on different days-of-the-week each year and don't lead to a day off. In addition, we assumed there are three days off before and after Christmas and New Year and estimated a fixed amount of sexual activity on these days. Refer to section 5.3.3 for the holiday dichotomy (i.e., holiday categorization based on whether each holiday results in a day off of work/school).

1.2 Birth Data

Time series of monthly live births were obtained for each geographical area. The US data were obtained from the CDC Wonder database, and span 2007 to 2018 21. Brazilian birth data were obtained from DATASUS and span

the years 2000-2016 22. UK data are from the Office of National Statistics 23 and span 2000-2018; while data from France were retrieved from the INSEE database 24 for years 2005-2019. Due to variation in the number of days per calendar month, the time series were standardized to represent a typical 30-day month. We did this to ensure that seasonality in the time series was not an artifact of variation in days per month.

1.3   Mathematical Models

We translated each of the three birth seasonality hypotheses into a deterministic population-level mathematical model of conception and birth (Fig 3a, main text). In each model, daily conceptions were modeled as the product of daily sexual activity and daily fertility. Daily conception was then used to predict daily births (approx. 9 months into the future) using empirical distributions of gestation period for each country (Fig 3b). We searched existing literature for data on the mean gestation periods.(Jukic et al. 2013; Delnord et al. 2018; Delnord and Zeitlin 2019) Since the time series of births were monthly totals, we then aggregated simulated daily births to monthly. Parameters of each model were estimated by minimizing the sum of squared errors between simulations and data. For each model we calculated the likelihood and the AIC that includes a penalty for additional parameters. We then discriminated among models using AIC; depending on assumptions regarding seasonal fertility, models differed in parameters estimated.

Model A assumed variation in sexual activity entered as a time-varying variable in the model and constant fertility (Fig 3c, main text). The time-series of daily sexual activity were output from our statistical model fitted to the app data. Model B assumed constant sexual activity and seasonal variation in fertility expressed as a sinusoidal function with a period of 1 year (Fig 3c, main text). The phase and amplitude of this function was estimated for each geographical area. Model C assumed variation in sexual activity and seasonal variation in fertility (Fig 3c, main text). Once again, seasonal sexual activity was entered as a time-varying variable. However, in Model C we added a scaling factor for sexual activity that allowed for changes in amplitude. The sexual activity scaling factor, along with the phase and amplitude of seasonal fertility, was estimated for each geographical location for Model C. Lastly, the mean daily conception rate was calculated based on the long-term trend in the birth time series.

2 Birth Official Records: data processing

Monthly birth records were acquired for each geographical area of interest from national statistics offices (see below for details and links for each country). Here, we format the collected data so that they can be collated into a single table with the following attributes: source, country, area, lat, lon, year, month, month_num, date, births.

2.1 Brazil birth data

2.1.1 Data source

Data were downloaded on Feb 21, 2018 from

http://www2.datasus.gov.br/DATASUS/index.php?area=0205

The dataset holds monthly births for each Brazilian state.

```
BR_birth = read_csv(file = str_c(IO$birth_records_dir,"Brazil_monthly_births_by_states_2000_2017.csv"),
            col_types = cols(
              year = col_double(),
              location = col_character(),
              month = col_character(),
              month.number = col_double(),
              births = col_double(),
              location.type = col_character()
              )
)
```

2.1.2 Data aggregation per regions and formatting

The areas defined below correspond to the official Brazilian Regions.

```
brazil.dict.list = list(
  "North" = c("Acre", "Amapá", "Amazonas", "Pará", "Rondônia", "Roraima", "Tocantins"),
  "Northeast" = c("Alagoas", "Bahia", "Ceará", "Maranhão", "Paraíba", "Pernambuco", "Piauí", "Rio Grande do Norte", "Sergipe"),
  "Central-West" = c("Goiás", "Mato Grosso", "Mato Grosso do Sul", "Distrito Federal" ),
  "Southeast" = c("Espírito Santo", "Minas Gerais", "Rio de Janeiro", "São Paulo"),
  "South" = c("Paraná", "Rio Grande do Sul", "Santa Catarina")
)

brazil.dict = data.frame(region = rep(names(brazil.dict.list), lengths(brazil.dict.list)),
                states = unlist(brazil.dict.list),
                stringsAsFactors = FALSE)

brazil.dict.geo = rbind(
  data.frame(region = "North", lat = -3.129167, lon = -60.021389),
  data.frame(region = "Northeast", lat = -12.966667, lon = -38.516667),
  data.frame(region = "Central-West", lat = -15.779722, lon = -47.930556 ),
  data.frame(region = "Southeast", lat = -23.55, lon = -46.633333 ),
  data.frame(region = "South", lat = -25.433333, lon = -49.266667)
)

BR_birth = BR_birth %>% mutate(
  source = "svs.aids.gov.br",
  country = "Brazil",
  state = location,
  area = brazil.dict$region[match(location,brazil.dict$states)],
  month_num = month.number,
```

```
   month_short = month,
   month = months[month_num],
   date = as.Date(str_c(year,"-",month_num,"-01")),
   lat = brazil.dict.geo$lat[match(area, brazil.dict.geo$region)],
   lon = brazil.dict.geo$lon[match(area, brazil.dict.geo$region)]
 ) %>%  dplyr::select(all_of(columns_from_each_source), state)


save(BR_birth, file = paste0(IO$out_Rdata,"BR_birth_data_by_states.Rdata"))

# aggregate by area

BR_birth = BR_birth %>%
  group_by(source, country, area, lat, lon, year, month, month_num, date) %>%
  dplyr::summarize(births = sum(births), .groups = "drop")%>%
  dplyr::select(all_of(columns_from_each_source))

save(BR_birth, file = paste0(IO$out_Rdata,"BR_birth_data.Rdata"))
```

2.2   US birth data

2.2.1   Data source

The US data per states were downloaded on Feb 20, 2020 from the CDC Wonder website:

https://wonder.cdc.gov/natality.html

The dataset holds monthly births for each American state.

```
US_birth = read_tsv(file = str_c(IO$birth_records_dir,"US_monthly_births_2007_2018.txt"),
              col_types = cols(
                notes = col_skip(),
                state = col_character(),
                state_code = col_skip(),
                year = col_integer(),
                year_code = col_skip(),
                month = col_character(),
                month_num = col_integer(),
                births = col_integer()
              ))
```

2.2.2   Data aggregation per regions and formatting

Areas defined below correspond to the US census regions, except for California, which is kept separate from the "West" region to match with the areas available in the Clue App dataset.

```
US.dict.list = list(
  "Northeast" = c("Connecticut", "Maine", "Massachusetts", "New Hampshire", "Rhode Island", "Vermont",
          "New Jersey", "New York","Pennsylvania"),
  "Midwest" = c("Indiana","Illinois","Michigan","Ohio","Wisconsin",
          "Iowa","Kansas","Minnesota","Missouri","Nebraska","North Dakota","South Dakota"),
  "South" = c("Delaware", "District of Columbia", "Florida", "Georgia", "Maryland",
        "North Carolina", "South Carolina", "Virginia", "West Virginia",
        "Alabama","Kentucky","Mississippi","Tennessee",
        "Arkansas","Louisiana","Oklahoma","Texas"),
  "West - without California" = c("Arizona","Colorado","Idaho","New Mexico",
              "Montana","Utah","Nevada","Wyoming",
              "Alaska","Hawaii","Oregon","Washington"),
```

```r
  "California" = c("California")
)

US.dict = data.frame(region = rep(names(US.dict.list), lengths(US.dict.list)),
          states = unlist(US.dict.list),
          stringsAsFactors = FALSE)

US.dict.geo = rbind(
  data.frame(region = "Northeast", lat = 42, lon = -73),
  data.frame(region = "Midwest", lat = 42, lon = -90),
  data.frame(region = "South", lat = 33, lon = -88 ),
  data.frame(region = "West - without California", lat = 40, lon = -113 ),
  data.frame(region = "California", lat = 37, lon = -120)
)
```

```r
US_birth = US_birth %>%
  mutate(source = "CDC",
      country = "United States",
      date = as.Date(str_c(year,"-",month_num,"-01")),
      area = US.dict$region[match(state, US.dict$state)],
      lat = US.dict.geo$lat[match(area, US.dict.geo$region)],
      lon = US.dict.geo$lon[match(area, US.dict.geo$region)]) %>%
  dplyr::select(all_of(columns_from_each_source), state)


save(US_birth, file = paste0(IO$out_Rdata,"US_birth_data_by_states.Rdata"))

# aggregate by area

US_birth = US_birth %>%
  group_by(source, country, area, lat, lon, year, month, month_num, date) %>%
  dplyr::summarize(births = sum(births)) %>%
  dplyr::select(all_of(columns_from_each_source))
```

`summarise()` regrouping output by 'source', 'country', 'area', 'lat', 'lon', 'year', 'month', 'month_num' (override with `.groups` argument)

```r
save(US_birth, file = paste0(IO$out_Rdata,"US_birth_data.Rdata"))
```

2.3 UK birth data

2.3.1 Data source

UK data were downloaded in April 2020 from

https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/livebirths/datasets/birth characteristicsinenglandandwales

The dataset holds monthly data for England and Wales from 2000 to 2018.

```r
UK_birth = read_csv(file = str_c(IO$birth_records_dir,"UK_monthly_births_england_and_wales.csv"),
          col_types = cols(
            Year = col_double(),
            Total = col_number(),
            January = col_number(),
            February = col_number(),
            March = col_number(),
            April = col_number(),
            May = col_number(),
            June = col_number(),
            July = col_number(),
```

```
              August = col_number(),
           September = col_number(),
            October = col_number(),
           November = col_number(),
           December = col_number()
        ))
```

2.3.2 Data processing

```
UK_birth = UK_birth %>% select(-Total) %>% pivot_longer(cols = -Year, names_to = "Month",values_to = "births")

UK_birth = UK_birth %>% mutate(
  source = "ons.gov.uk",
  country = "United Kingdom",
  area = "",
  lat = 52,
  lon = -1,
  year = Year,
  month = Month,
  month_num =  match(Month,levels(month(1:12, label = TRUE, abbr = FALSE))),
  date = as.Date(str_c(year,"-",month_num,"-01"))
) %>%  select(all_of(columns_from_each_source))


save(UK_birth, file = paste0(IO$out_Rdata,"UK_birth_data.Rdata"))
```

2.4   France birth data

2.4.1 Data source

The France births data were downloaded in April 2020 from the INSEE data portal (https://www.insee.fr/fr/stat istiques/serie/000436391?idbank=000436391#Telechargement)

The dataset holds monthly births.

```
FR_birth = read_delim(file = str_c(IO$birth_records_dir,"France_birth_data.csv"), delim = ";",
            col_types = cols(
              month = col_character(),
              births = col_double(),
              code = col_character()
              ))
```

2.4.2 Data processing

```
FR_birth = FR_birth %>%
  rename(date_month_chr = month)%>%
  mutate(
    source = "INSEE",
    country = "France",
    area = "",
    lat = 46.62012,
    lon = 2.452757,
    year = str_sub(date_month_chr, 1, 4) %>% as.numeric(),
    month_num =  str_sub(date_month_chr, 6, 7) %>% as.numeric(),
    month = months[month_num],
```

```
      date = as.Date(str_c(year,"-",month_num,"-01"))
  ) %>%
  select(all_of(columns_from_each_source)) %>%
  arrange(year, month_num)

save(FR_birth, file = paste0(IO$out_Rdata,"FR_birth_data.Rdata"))
```

2.5 Collating datasets

```
birth = bind_rows(
  BR_birth,
  US_birth,
  UK_birth,
  FR_birth
)

# add country_area + arrange country by latitude (North to South)
birth = birth %>%
  ungroup() %>%
  dplyr::mutate(country_area = str_c(country, ifelse(area == "","",str_c(" - ",area)))) %>%
  arrange(desc(lat), country, year, month) %>%
  dplyr::mutate(country = factor(country))

save(birth, file = paste0(IO$out_Rdata,"birth_data.Rdata"))

birth_summary = birth %>%
  group_by(source, country, country_area) %>%
  summarize(start_date = min(date),
        end_date = max(date),
        n = n()) %>%
  arrange(country) %>%
  as.data.frame()
```

```
## `summarise()` regrouping output by 'source', 'country' (override with `.groups` argument)
```

```
kable(birth_summary, format = "pandoc", caption = "Summary of the official birth record dataset. 'n' is the number of data-point, i.e. of monthly birth record for each location.
```

Table 2.1: Summary of the official birth record dataset. 'n' is the number of data-point, i.e. of monthly birth record for each location.

source	country	country_area	start_date	end_date	n
svs.aids.gov.br	Brazil	Brazil - Central-West	2000-01-01	2016-06-01	198
svs.aids.gov.br	Brazil	Brazil - North	2000-01-01	2016-06-01	198
svs.aids.gov.br	Brazil	Brazil - Northeast	2000-01-01	2016-06-01	198
svs.aids.gov.br	Brazil	Brazil - South	2000-01-01	2016-06-01	198
svs.aids.gov.br	Brazil	Brazil - Southeast	2000-01-01	2016-06-01	198
INSEE	France	France	2005-01-01	2019-12-01	180
ons.gov.uk	United Kingdom	United Kingdom	2000-01-01	2018-12-01	228
CDC	United States	United States - California	2007-01-01	2018-12-01	144
CDC	United States	United States - Midwest	2007-01-01	2018-12-01	144
CDC	United States	United States - Northeast	2007-01-01	2018-12-01	144
CDC	United States	United States - South	2007-01-01	2018-12-01	144
CDC	United States	United States - West - without California	2007-01-01	2018-12-01	144

3 App (Clue) data processing and filtering

3.1 Dataset structure

The Clue dataset was provided in four tables:

- users: basic demographic information about the users.
- birth_control: users' birth control method as self-reported by users in their app profile. The table holds the type of birth control and the date at which they changed this option if they changed it. If they did not change their birth control in their app profile, the table provides the birth control self-reported by users when their first registered their app profile.
- cycles : cycles starts and ends. Provided in 20 files to keep each file at a reasonable size.
- tracking : time-series of the features tracked by the users. Provided in 20 files to keep each file at a reasonable size.

These tables were provided by Clue in csv files and we transformed into feather files (see package feather) to accelerate reading/writing operations.

3.2 Filtering for countries/areas of interest.

Each of the four tables are filtered so that they only contain the data for users of the six countries/areas of interest:

Brazil - Central-West, Brazil - Northeast, United States - California, United States - Northeast, France, United Kingdom

```
subset_folder = paste0(IO$r_Data,"Clue_US_BR_EU/input/")
if(!dir.exists(subset_folder)){dir.create(subset_folder, recursive = TRUE)}
```

```
selected_country_areas = dict$country_area$country_area
```

Filtering users table

```
users = read_feather(path = paste0(IO$r_Data,"Clue/input/users.feather"))
n_users_tot = nrow(users)
```

```
users = users %>% filter(country_area %in% selected_country_areas)
n_users_selected = nrow(users)
```

```
n_users_tot
```

```
## [1] 1256541
```

```
n_users_selected
```

```
## [1] 535154
```

```
(n_users_selected/n_users_tot) %>% round(.,2)
```

```
## [1] 0.43
```

```
write_feather(users, path = paste0(subset_folder, "users.feather"))
```

Filtering birth control table

```
BC = read_feather(path = paste0(IO$r_Data,"Clue/input/birth_control.feather"))
dim(BC)
```

```
## [1] 1474662       6
```

```
BC = BC %>% filter(user_id %in% users$user_id)
dim(BC)
```

```
## [1] 653452      6
```

```r
write_feather(BC, path = paste0(subset_folder, "birth_control.feather"))
```

Filtering cycles table

```r
input_folder = paste0(IO$r_Data,"Clue/input/cycles/")
files = list.files(input_folder)

output_folder = paste0(subset_folder, "cycles/")
if(!dir.exists(output_folder)){dir.create(output_folder)}

ok = foreach(file = files) %do% {
  cat(file %>% str_remove(.,"cycles00") %>%  str_remove(.,".feather")," | ")
  cycle = read_feather(path = str_c(input_folder,file))
  dim(cycle)
  cycle = cycle %>% filter(user_id %in% users$user_id)
  dim(cycle)
  write_feather(cycle, path = paste0(output_folder, file))
}
```

```
## 00  | 01  | 02  | 03  | 04  | 05  | 06  | 07  | 08  | 09  | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  |
```

Filtering tracking table

```r
input_folder = paste0(IO$r_Data,"Clue/input/tracking/")
files = list.files(input_folder)

output_folder = paste0(subset_folder, "tracking/")
if(!dir.exists(output_folder)){

  dir.create(output_folder)

  ok = foreach(file = files) %do% {
    cat(file,"\n")
    tracking = read_feather(path = str_c(input_folder,file))
    dim(tracking)
    tracking = tracking %>% filter(user_id %in% users$user_id)
    dim(tracking)
    write_feather(tracking, path = paste0(output_folder, file))
  }

}else{warning("Clue data: tracking table files filtered for country_area already existed. Filtering was NOT done at this execution.")}
```

```
## Warning: Clue data: tracking table files filtered for country_area already
## existed. Filtering was NOT done at this execution.
```

3.3 Dataset overview

User table

```
users = read_feather(path = paste0(IO$input_clue,"users.feather"))
```

The users table has 535154 rows (*i.e.* users) and 7 columns: user_id, birth_year_bin, country_area, height_bin, weight_bin, latest_birth_control, csv_file

```
df = table(country_area = users$country_area) %>% sort(decreasing = TRUE) %>% as.data.frame()
knitr::kable(df, format = "pandoc", caption = "Total number of app users in each area.")
```

Table 3.1: Total number of app users in each area.

country_area	Freq
Brazil - Central-West	160896
United Kingdom	133387
France	126634
United States - Northeast	51780
United States - California	39330
Brazil - Northeast	23127

```
df = df %>% mutate(
  country_area_col = dict$country_area$country_area_col[match(country_area, dict$country_area$country_area)],
  country_area_wrapped = str_replace(country_area," - ","\n") %>% factor(.,levels = str_replace(dict$country_area$country_area," - ","\n"))
)
ggplot(df, aes(x = country_area_wrapped, y = Freq, fill = country_area_col))+
  geom_bar(stat = "identity")+
  xlab("")+ylab("Total # of users")+
  scale_fill_identity()
```
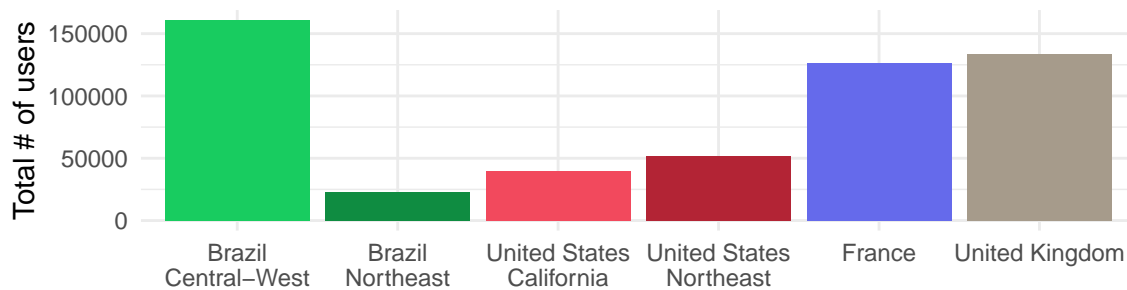

Figure 3.1: Total number of users in each considered area

```
kable(table(birth_year_bin = users$birth_year_bin) %>% as.data.frame(),
      format = "pandoc", caption = "Total number of app users in each birth year bin")
```

Table 3.2: Total number of app users in each birth year bin

birth_year_bin	Freq
1965-1969	138
1970-1974	4108
1975-1979	11335
1980-1984	26346
1985-1989	56642
1990-1994	97890
1995-1999	203964
2000-2004	134731

Birth control table

```
birth_control = read_feather(path = paste0(IO$input_clue,"birth_control.feather"))
```

The birth control table (birth_control) has 653452 rows (i.e. birth control at on-boarding and changes in birth control as declared by the users in their app profile) and 6 columns:

user_id, date, birth_control, pill_type, pill_regiment, csv_file

```
table(birth_control = birth_control$birth_control) %>% sort(decreasing = TRUE) %>% as.data.frame() %>%
  kable(., format = "pandoc", caption = "Number of times users declared the following birth control in their app profile")
```

Table 3.3: Number of times users declared the following birth control in their app profile

birth_control	Freq
none	269275
pill	259114
condoms	97774
IUD	14466
injection	4388
implant	4188
other	1575
fertility_awareness_method	465
vaginal_ring	166
patch	111

Cycle table

The cycles table is split in several files due to their large size. Here, we load one the these files and describe the dataset based on this file.

```
cycles_00 = read_feather(path = paste0(IO$input_clue,"cycles/cycles0000.feather"))
```

The cycles tables have the following 11 columns:

user_id, cycle_nb, cycle_start, cycle_end, cycle_length, period_start, period_end, period_length, neg_preg_test, pos_preg_test, latest_preg_test

This cycles file has a total of 287684 cycles for 206001 users.

If we extrapolate from this number, we expect the total number of cycles to be 20 x 287684 = 5753680.

```
ggplot(cycles_00, aes(x = cycle_length))+
  geom_histogram(binwidth = 1)+ xlab("cycle length")+
  scale_x_continuous(breaks = seq(0,98, by = 7), limits = c(0,98))
```

Warning: Removed 7954 rows containing non-finite values (stat_bin).

Warning: Removed 2 rows containing missing values (geom_bar).
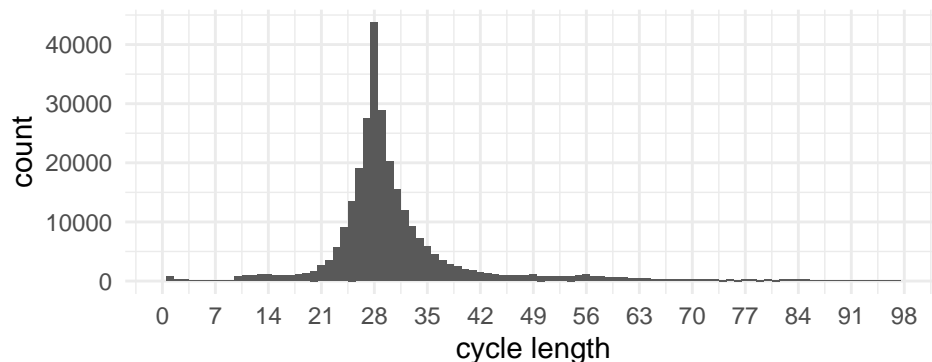
Figure 3.2: Histogram of the cycle lengths

Note that the cycles table is not further used in this analysis as the sex logs were aggregated at the population level, independent of the cycles in which they were logged.

Tracking table

The tracking table holds the daily logs of the app users.

```
tracking_folder = paste0(IO$input_clue,"tracking/")
files = list.files(tracking_folder)

tracking = read_feather(path = paste0(tracking_folder,files[1]))
```

The tracking table has 5 columns:

user_id, date, category, type, number.

Similarly to the cycles table, the tracking table is split into 20 files to keep each file at a reasonable size.

This particular files has 9988623 rows. If we extrapolate for the number of files, that means 1.9977246×10^8 total logs.

The tracking features are grouped into categories (each category has max 4 items, which are displayed together on the logging screen of the App).

Based on this particular file, the frequency of each of these category goes as follow:

Table 3.4: Number of times each category has been logged by users (based on 1/20 file of the tracking table)

category	Freq
ANY	2757502
period	1322117
pill_hbc	1127752
emotion	910480
pain	660806
sleep	628364
energy	620070
mental	466308
sex	363837
social	340289
digestion	293181
poop	235594
fluid	122796
exercise	62794
weight	39027
medication	20782
ailment	11469
iud	2630

16

category	Freq
injection_hbc	1289
bbt	1267
test	260
patch_hbc	9

Given that Clue did not provide data for all the tracking option they offer in the app, they additionally provided a "ANY" category, which tells us if anything was tracked in the app on that day. Note that for the categories displayed in the table above, all tracking data were provided, but some categories of the App, such as "doctor appointment" or "party" were not included in the dataset provided to the research team.

In section 3.10, we provide examples of users' time-series.

3.4   Data processing and filtering: WORKFLOW

- users table data augmentation (1)
  - split Country and Area into two distinct columns
  - estimate BMI from the weight and height bins
- Defining user batches: the tracking files provided by Clue did not ensure that the whole time-series of a user was held in a single file. Each user was thus assigned to a batch and the tracking table files are re-organized such that each file would contain the full time-series of users of a given batch. The users' features are also added to the tracking table at this step.
- users table data augmentation (2) : information from the tracking table is aggregated at the user level and added to the users table.
  - Dates of their first and last log in the app
  - Number of days they logged features in the app
  - Total number of logs
  - Total number of sex logs
- Label user's time-series (tracking table) with their **birth control** (from the birth_control table).
- Label user's time-series (tracking table) as **active** or **not active**. A user was considered active if they logged any feature in the app in a 42-day window. If that was the case, they were considered active for the whole 42-day window. 42 was chosen from 42 = 1.5*28, with 28 being the mode of cycle length distribution.

3.5   Users table data augmentation (1)

```
countries_areas = users$country_area %>% str_split_fixed(., " - ", n = 2)
users$country = countries_areas[,1]
users$area = countries_areas[,2]

write_feather(users, path = paste0(IO$output_clue, "users.feather"))

users$height_bin = factor(users$height_bin, levels = dict$height$bin)
users$weight_bin = factor(users$weight_bin, levels = dict$weight$bin)
heigh_mean = dict$height$mean[match(users$height_bin,dict$height$bin)]
weight_mean = dict$weight$mean[match(users$weight_bin,dict$weight$bin)]
users$est_mean_bmi = weight_mean/((heigh_mean/100)^2)

write_feather(users, path = paste0(IO$output_clue, "users.feather"))
```

```r
ggplot(users, aes(x = est_mean_bmi))+
 geom_histogram(binwidth = 2)+
 facet_grid(country_area ~ ., scale = "free_y")+
 theme(strip.text.y = element_text(angle = 0, hjust = 0),
      strip.background.y = element_rect(fill = "gray90", color = NA))
```

Warning: Removed 153517 rows containing non-finite values (stat_bin).
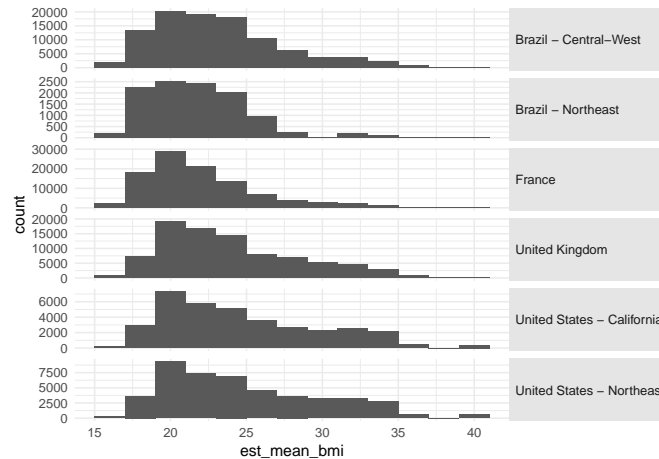
Figure 3.3: Histogram of the user's estimated BMI for each country/area. For each user, their BMI is estimated based on their 5kg-weight and 5cm-height bins.

3.6   Defining user batches

```r
# maximum number of users per batch
max_batch_size = 5000


# computing the number of batches given the total number of users and the minimum number of batches.
n_batch = max(par$min_n_batches, ceiling(nrow(users)/max_batch_size))

# effective batch size
batch_size = ceiling(nrow(users)/n_batch)


users$batch = rep(1:n_batch, each = batch_size)[1:nrow(users)] # assigning a batch to each user
write_feather(users, path = paste0(IO$output_clue, "users.feather"))
ok = file.copy(from = paste0(IO$output_clue, "users.feather"), to = paste0(IO$tmp_clue, "users_with_batches.feather"))
```

```r
# for each tracking file:
#   we read it,
#   assign each user to its batch
#   for each batch present in this file, we split the original tracking file and write them separately on disk
# for each user, we keep in the user table, the names of the original files in which its data were contained.
# Most users only have 1 file, but some can have many.



tracking_folder = paste0(IO$input_clue,"tracking/")
files = list.files(tracking_folder)


tmp_folder = paste0(IO$tmp_clue,"tracking_split_in_batches/")
tmp_final_folder = paste0(IO$tmp_clue,"tracking_in_batches/")



# Given its long execution time, we only execute this code if the folder in which the splitted files by batch are stored does NOT exist.
```

```r
# To reset, one needs to "manually" delete the folder.
if((!file.exists(tmp_final_folder)) & (!file.exists(tmp_folder))){
 dir.create(tmp_folder,recursive = TRUE)

 cl = makeCluster(par$n_cores)
 registerDoParallel(cl)

 tic()
 users_original_file_ids = foreach(file = files, .combine = rbind, .packages = c("feather","stringr")) %dopar%
 {
  tracking = read_feather(path = paste0(tracking_folder, file))
  dim(tracking)
  tracking$batch = users$batch[match(tracking$user_id, users$user_id)]
  if(nrow(tracking)>0){
   for(b in unique(tracking$batch)){
    tracking_batch = tracking[which(tracking$batch == b),]
    new_file_name = gsub(".feather",paste0("_batch_",b,".feather"),file)
    write_feather(tracking_batch, path = paste0(tmp_folder, new_file_name ))
   }
   users_original_file_ids = data.frame(user_id = unique(tracking$user_id), original_file_id = str_extract(file,"\\d{4}"))
  }else{  users_original_file_ids = data.frame(user_id = character(), original_file_id = character())}
  return(users_original_file_ids)
 }
 toc()
 stopImplicitCluster()

 users_o_f = aggregate(original_file_id ~ user_id ,users_original_file_ids, function(x)  paste0(sort(x), collapse = ","))
 colnames(users_o_f) = c("user_id","original_tracking_files")
 write_feather(users_o_f, path = paste0(IO$tmp_clue,"original_tracking_files_per_users.feather"))
}else{
 warning("App data processing: spliting in batches is NOT done at this execution. Re-using results from a previous execution.")
}
```

```
## Warning: App data processing: spliting in batches is NOT done at this execution.
## Re-using results from a previous execution.
```

```r
input_folder = paste0(IO$tmp_clue,"tracking_split_in_batches/")
output_folder = paste0(IO$output_clue,"tracking/")

# Similarly, we only execute this code if the folder does not exist.
# To re-execute this code, one needs to "manually" delete the folder.
if(!dir.exists(tmp_final_folder)){
 dir.create(tmp_final_folder,recursive = TRUE)

 # First we delete any existing tracking files in the output folder to avoid mess.
 if(file.exists(output_folder)){unlink(output_folder, recursive = TRUE)} ;dir.create(output_folder,recursive = TRUE)

 batches = unique(users$batch)
 ok = foreach(b = batches) %do% {
  cat("\t",b,"||\t")
  all_files = list.files(input_folder)
  files = all_files[grep(paste0("_batch_",b,"\\."),all_files)]

  cl = makeCluster(par$n_cores)
  registerDoParallel(cl)
  tracking = foreach(file  = files, .combine = rbind, .packages = "feather") %dopar%{tracking = read_feather(path = paste0(input_folder, file));return(tracking)}
  stopImplicitCluster()

  cols_to_add = c("birth_year_bin","country_area","height_bin","weight_bin", "est_mean_bmi")
```

```r
    m = match(tracking$user_id, users$user_id)
    for(col_to_add in cols_to_add){
      eval(parse(text = paste0("tracking$",col_to_add," = users$",col_to_add,"[m]")))
    }
    o = order(tracking$user_id, tracking$date, tracking$category, tracking$type, tracking$number)
    tracking = tracking[o,]

    new_file_name = paste0("tracking_",b,".feather")
    write_feather(tracking, path = paste0(output_folder, new_file_name ))
    ok = file.copy(from = paste0(output_folder, new_file_name ), to = paste0(tmp_final_folder, new_file_name), overwrite = TRUE)
  }
}else{
  warning("App data processing: re-assembling in batches is NOT done at this execution. Re-using results from a previous execution.")
}
```

Warning: App data processing: re-assembling in batches is NOT done at this
execution. Re-using results from a previous execution.

3.7 Users table data augmentation (2)

Information from the tracking table is aggregated at the user level and added to the users table:

* Dates of their first and last log in the app

* Number of days they logged features in the app

* Total number of logs

* Total number of sex logs

```r
if(!file.exists(paste0(IO$tmp_clue, "users_agg.feather"))){

  tracking_folder = paste0(IO$output_clue,"tracking/")
  files = list.files(tracking_folder)

  cl = makeCluster(par$n_cores)
  registerDoParallel(cl)

  tic()
  users_agg = foreach(file = files, .combine = rbind, .packages = c("feather","stringr", "plyr")) %dopar%
  {
    tracking = read_feather(path = paste0(tracking_folder, file))
    users_agg = ddply(tracking,
              .(user_id),
              summarize,
              first_obs = min(date),
              last_obs = max(date),
              n_obs_day = length(unique(date)),
              n_obs = sum(category != "ANY"),
              n_sex = sum(category == "sex"),
              n_mucus =  sum(category == "fluid"),
              n_temp = sum(category == "bbt")
    )
    return(users_agg)
  }
  toc()
  stopImplicitCluster()

  write_feather(users_agg, path = paste0(IO$tmp_clue, "users_agg.feather"))
```

```
}else{
  warning("App data processing: tracking aggregation at the user level is NOT done at this iteration. Re-using results from a previous execution.")
}
```

Warning: App data processing: tracking aggregation at the user level is NOT done
at this iteration. Re-using results from a previous execution.

```
users_agg = read_feather(path = paste0(IO$tmp_clue, "users_agg.feather"))

cols_to_add = setdiff(colnames(users_agg),"user_id")
m = match(users$user_id, users_agg$user_id)
for(col_to_add in cols_to_add){eval(parse(text = paste0("users$",col_to_add," = users_agg$",col_to_add,"[m]")))}
users$n_days = as.numeric(users$last_obs - users$first_obs)

write_feather(users, path = paste0(IO$output_clue, "users.feather"))
ok = file.copy(from = paste0(IO$output_clue, "users.feather"), to = paste0(IO$tmp_clue, "users_augmented.feather"))
```

3.8   Users birth control

The user's time-series (tracking table) are labelled with the **birth control** the user declared in their app profile (from the birth_control table). If a user changed birth control (and thus have multiple entries at different dates in the birth_control table), their time-series are labelled with these different birth control.

```
BC = birth_control
BC = BC %>%
  arrange(user_id, date) %>%
  mutate(birth_control = birth_control %>% replace_na("undefined"))

write_feather(BC, path = paste0(IO$output_clue, "birth_control.feather"))
```

```
BC = read_feather(path = paste0(IO$output_clue,"birth_control.feather"))

input_folder = paste0(IO$tmp_clue,"tracking_in_batches/")
output_folder = paste0(IO$output_clue,"tracking/")
tmp_folder = paste0(IO$tmp_clue, "tracking_with_user_defined_birth_control/")
if(!dir.exists(tmp_folder)){

  dir.create(tmp_folder)

  files = list.files(input_folder)

  cl = makeCluster(par$n_cores)
  registerDoParallel(cl)

  tic()
  catch =
    foreach(file = files,
        .combine = rbind,
        .packages = c("feather","stringr", "plyr","tidyverse")
    ) %dopar% {
      tracking = read_feather(path = paste0(input_folder, file))
      # adding the relative date to each user time-series
      agg = aggregate(date ~ user_id, tracking, min)
      min_date = agg$date[match(tracking$user_id, agg$user_id)]
      tracking = tracking %>%  mutate(rel_date = as.numeric(date - min_date + 1))

      # creating a table (u_tracking) that has one row per user and date
      u_tracking = tracking %>%
```

```r
    select(user_id, date) %>% unique() %>%
    mutate(birth_control = NA,pill_type = NA, pill_regiment = NA)
    # filtering the BC table to include only the users of this tracking table
    u_BC = BC %>% select(-csv_file) %>% filter(user_id %in% unique(tracking$user_id)) %>% unique()
    BC_exp = rbind(u_tracking, u_BC) %>% arrange(., user_id, date)

    # we replace the NAs with the latest value in the file
    BC_exp = BC_exp %>% group_by(user_id) %>% mutate(
      birth_control = replace_NAs_with_latest_value(birth_control) %>% replace_na("undefined"),
      pill_type = replace_NAs_with_latest_value(pill_type),
      pill_regiment = replace_NAs_with_latest_value(pill_regiment)
    )

    tracking_key = str_c(tracking$user_id, "_",tracking$date)
    BC_key = str_c(BC_exp$user_id,"_",BC_exp$date)
    m = match(tracking_key,BC_key)
    tracking$birth_control_ud = BC_exp$birth_control[m]
    tracking$pill_type_ud = BC_exp$pill_type[m]
    tracking$pill_regiment_ud = BC_exp$pill_regiment[m]


    write_feather(tracking, path = paste0(output_folder,file))
    file.copy(from = paste0(output_folder,file), to = paste0(tmp_folder,file), overwrite = TRUE)
  }
  toc()
  stopImplicitCluster()
}else{
  warning("App data processing: adding birth control to tracking table is NOT done at this execution. Re-using results from a previous execution.")
}
```

```
## Warning: App data processing: adding birth control to tracking table is NOT done
## at this execution. Re-using results from a previous execution.
```

3.9   Identifying active use of the app

User's time-series (tracking table) are labelled with **active** or **not active** labels. A user was considered active if they logged any feature in the app in a 42-day window. If that was the case, they were considered active for the whole 42-day window. 42 was chosen from 42 = 1.5*28, with 28 being the mode of cycle length distribution. A user can transition several times between "active" and "inactive" use of the App.

```r
active_tracking_filter = function(x, N = 28*1.5){
  res = stats::filter(c(rep(0,N-1),x), filter = rep(1,N), sides = 1) %>% pmin(1)
  res = na.omit(res) %>% as.vector()
  return(res)
}
```

```r
input_folder = paste0(IO$output_clue,"tracking/")
files = list.files(input_folder)

output_folder = paste0(IO$tmp_clue, "active_tracking/")
if(!dir.exists(output_folder)){

  dir.create(output_folder)

time_vec = seq(min(users$first_obs), max(users$last_obs), by = 1)

cl = makeCluster(par$n_cores)
registerDoParallel(cl)
```

```r
tic()
ok =
  foreach(file = files,
         .packages = c("dplyr", "feather")) %dopar% {
          tracking = read_feather(path = paste0(input_folder, file)) # we load the tracking table

          # we only keep the dates at which users logged smth in the app, discard all other info.
          any_tracking = tracking %>% filter(. , category == "ANY") %>%
            select(c(user_id, date, birth_control_ud))
          any_tracking$tracking_days = 1 # will be used later to match with the tmp df (see below)

          # we create a data.frame that has one row for each calendar day from the first to the last log
          tmp = data.frame(
            user_id = rep(unique(tracking$user_id), each = length(time_vec)),
            date = rep(time_vec, lu(tracking$user_id)))

          # we join this tmp table with the table that has active tracking.
          active_tracking = dplyr::full_join(tmp,any_tracking, by = c("user_id","date"))
          # tracking_days is zero when there were no entries in the tracking table
          active_tracking$tracking_days[is.na(active_tracking$tracking_days)] = 0
          # we keep the BC info
          active_tracking$birth_control_ud = ave(active_tracking$birth_control_ud,
                                      by =  active_tracking$user_id,
                                      FUN = replace_NAs_with_latest_value)
          # we check for any active tracking in 42 days window
          active_tracking = active_tracking %>%  group_by(user_id) %>%
            mutate(., tracking = active_tracking_filter(tracking_days))
          # only keep the necessary columns.
          active_tracking = active_tracking %>% select(user_id, date,birth_control_ud, tracking)

          ###
          # compressed table (we compress the active tracking table to speed up IO operations)
          active_tracking  = active_tracking %>%  group_by(user_id, birth_control_ud) %>%
            mutate(transition = diff(c(0, tracking)))
          active_tracking  = active_tracking %>%  group_by(user_id) %>%
            mutate(stretch_num = cumsum(transition == 1))
          active_tracking  = active_tracking %>%
            group_by(user_id, birth_control_ud, stretch_num, tracking) %>%
            mutate(stretch_length = sum(tracking))
          compressed_tracking = active_tracking %>% filter(transition == 1) %>%
            select(user_id, date,birth_control_ud, stretch_num, stretch_length) %>%
            rename(start_date = date)

          file_name = paste0("active_",file)
          write_feather(compressed_tracking, path = paste0(output_folder,file_name))
        }
  toc()
  stopImplicitCluster()

}else{
  warning("App data processing: identification of active use is NOT done at this execution. Re-using results from a previous execution.")
}
```

```
## Warning: App data processing: identification of active use is NOT done at this
## execution. Re-using results from a previous execution.
```

3.10 Examples of individual users time-series

```
input_folder = paste0(IO$output_clue,"tracking/")
files = list.files(input_folder)
file = files[1]

tracking = read_feather(path = paste0(input_folder, file)) # we load the tracking table
```

```
plot_user_timeseries = function(tracking, u){

  dat = tracking %>% filter(user_id == u) %>%
    mutate(rel_date = (date - min(date)) %>% as.numeric()) %>%
    arrange(rel_date) %>%
    mutate(category = factor(category, levels = c("ANY", unique(feature_dict$category) %>% as.character())),
        type_col = feature_dict$color[match(type, feature_dict$type)] %>% replace_na("black"))

  g = ggplot(dat, aes(x = rel_date, y = type, col = type_col))
  g = g +
    geom_point()+
    xlab("days from 1st log in the App")+ ylab("")+
    scale_color_identity()+
    facet_grid(category ~ ., scale = "free_y", space = "free", switch = "y")+
    theme(strip.placement = "outside",
        strip.background.y = element_rect(fill = "gray80", color = NA),
        strip.text.y.left = element_text(angle = 0, hjust = 1))
  g

}
```

```
u = c("dd1de61df75ded8f4dbbb0be30f27d2b8c658b4f")
```

```
plot_user_timeseries(tracking = tracking, u = u)
```
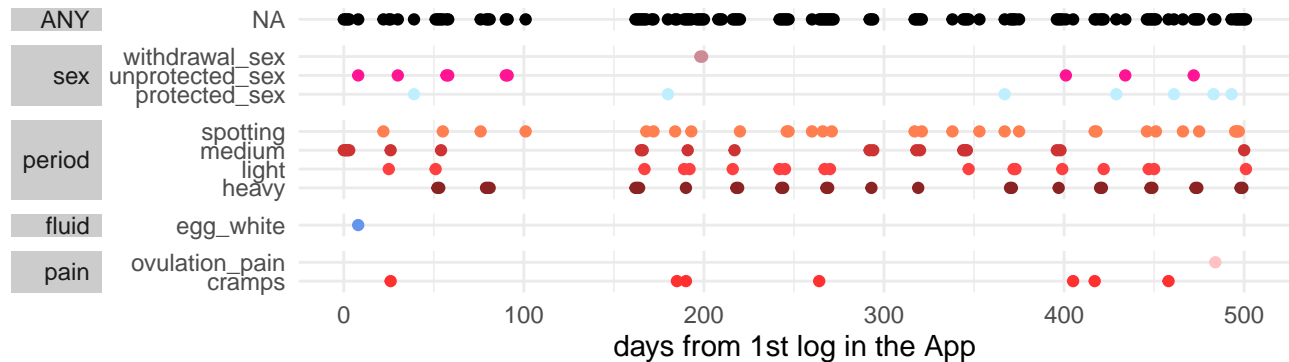

Figure 3.4: Example of a user tracking time-series (1)

```
# u = sample(tracking$user_id, 1)
# u = "b2f86085f009a8d1bc9691b782b6ffc1308871af"
u = "e12d958dc494c3c2a3aa2d42be6683f96bdd90e7"
```

```
plot_user_timeseries(tracking = tracking, u = u)
```
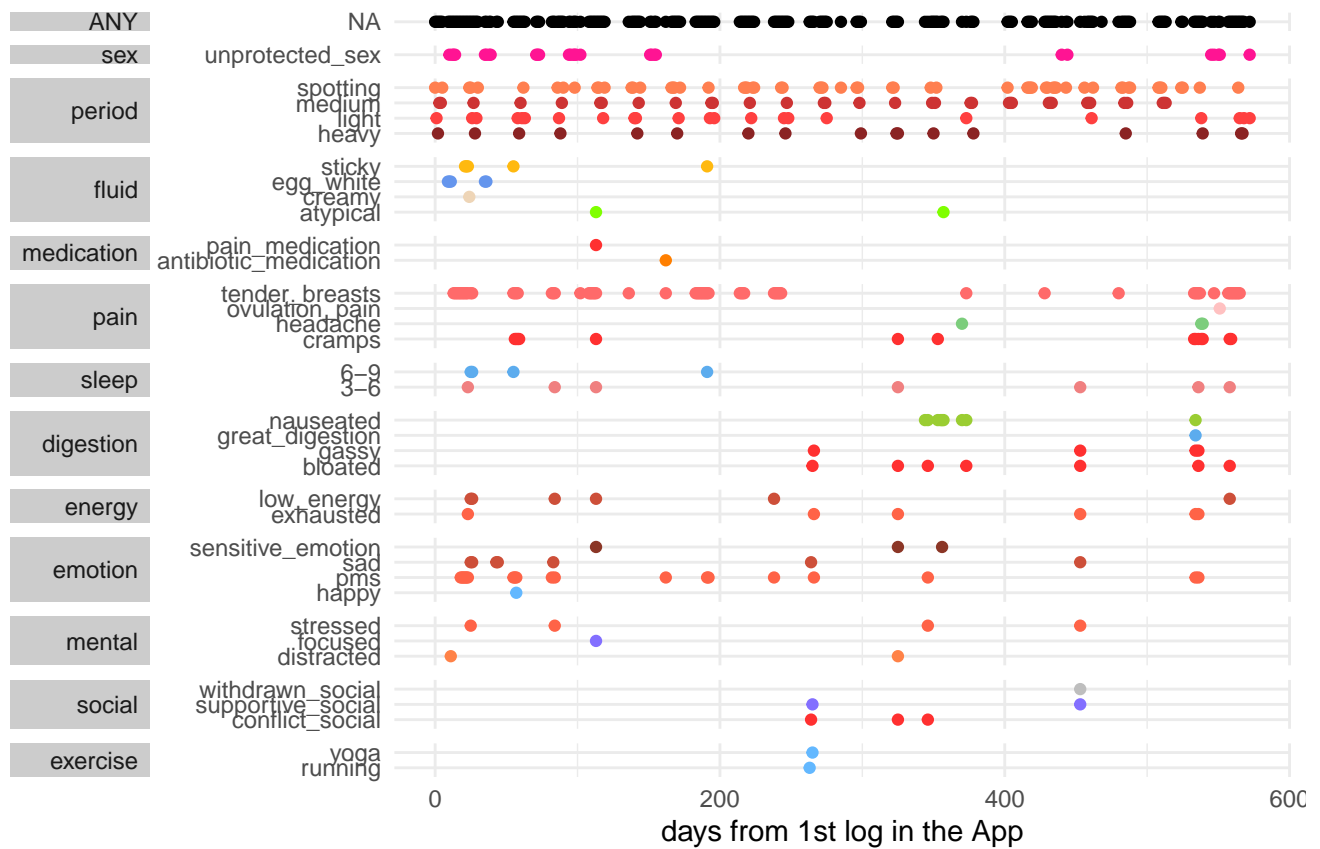
Figure 3.5: Example of a user tracking time-series (2)

3.11 Dataset characteristics

```r
# Directory with the tracking tables
input_folder = paste0(IO$output_clue,"tracking/")
files = list.files(input_folder)

# Directory with the active tracking tables
input_active_tracking = paste0(IO$tmp_clue,"active_tracking/")

# file on which the tracking_duration_stats are saved
tds_file = str_c(IO$tmp_clue, "tracking_duration_stats.feather")

if(!file.exists(tds_file)){

  tracking_duration_stats = purrr::map_dfr(
    files,
    .f = function(file){
      tracking = read_feather(str_c(input_folder, file))
      active_tracking =  read_feather(str_c(input_active_tracking, "active_", file))
      data.frame(file = file,
              number_of_days_of_observation = tracking %>% select(user_id, date) %>% distinct() %>% nrow(),
              total_duration_of_active_tracking = active_tracking$stretch_length %>% sum())
  })
  write_feather(tracking_duration_stats, path = tds_file)
}else{
  tracking_duration_stats = read_feather(path = tds_file)
}
```

Table 3.5: Total duration of active tracking and number of days tracked in the app.

number_of_days_of_observation	55110476.0
total_duration_of_active_tracking_in_days	180721915.0
number_of_years_of_observation	150987.6
total_duration_of_active_tracking_in_years	495128.5

```r
tracking_duration_stats_summary = tracking_duration_stats %>%
  summarise(number_of_days_of_observation = sum(number_of_days_of_observation),
            total_duration_of_active_tracking_in_days = sum(total_duration_of_active_tracking)) %>%
  mutate(number_of_years_of_observation = number_of_days_of_observation/365,
         total_duration_of_active_tracking_in_years = total_duration_of_active_tracking_in_days/365)
```

```r
kable(t(tracking_duration_stats_summary),
      format = "latex",
      booktabs = T,
      caption = "Total duration of active tracking and number of days tracked in the app.")
```

4 App data aggregation: constructing population-wide time-series

In this section, the App users logs are aggregated in population-wide time-series.

```
# Loading the users table
users = read_feather(path = paste0(IO$output_clue, "users.feather"))
```

```
# defining the date-range for which we will build the aggregated table.
# starts with the first observation of the first user in this dataset and ends with the last day a feature was logged by any user.
time_vec = seq(min(users$first_obs), max(users$last_obs), by = 1)
range(time_vec)
```

[1] "2016-01-01" "2019-06-30"

4.1 Aggregating variables

We will aggregate at the following levels:

- **country/area** (6): Brazil - Central-West, Brazil - Northeast, United States - California, United States - Northeast, France, United Kingdom

- **age**. Users are grouped into 2 age categories: the younger users (<= 23 years old) and the older (> 23 years old). This age limit (23 years old) is approximately the median age of the app users (we do not have an exact median value given that the birth year of users was given in 5-year bins).

- **birth control**. Birth control methods are grouped into two categories:

  - **F** (for potentially **f**ertile) for birth control methods, such as "condoms" or "fertility awareness method", where unprotected sex could lead to a pregnancy.

  - **I** (for **i**nfertile) for birth control methods, such as IUDs or the pill, where unprotected sex does not potentially lead to a pregnancy.

Table 4.1: Types of birth control available to users in the Clue profile and their classification into F or I type.

birth_control	type	type_descr
none	F	Potentially fertile, and thus fecundable with unprotected sex
fertility_awareness_method	F	Potentially fertile, and thus fecundable with unprotected sex
condoms	F	Potentially fertile, and thus fecundable with unprotected sex
pill	I	Infertile due to contaceptive
vaginal_ring	I	Infertile due to contaceptive
IUD	I	Infertile due to contaceptive
patch	I	Infertile due to contaceptive
injection	I	Infertile due to contaceptive
implant	I	Infertile due to contaceptive
vaginal_ring	I	Infertile due to contaceptive
other	?	NA
undefined	?	NA

4.2 Aggregation

We aggregate and build time-series counting:

- the number of active users

- the number of sex logs: protected sex, unprotected sex, all sex

- the number of logs for each control feature (see section 6): medium flow bleeding (period), exercise, tender breasts (pain), sleeping more than 9h.

```r
# Directory with the tracking tables
input_folder = paste0(IO$output_clue,"tracking/")
files = list.files(input_folder)

# Directory with the active tracking tables
input_active_tracking = paste0(IO$tmp_clue,"active_tracking/")

# Directory where the aggregated table will be stored
indicators_folder = paste0(IO$output_clue, "pop_indicators/")
if(!dir.exists(indicators_folder)){dir.create(indicators_folder)}

# We only run this code if the file containing the aggregated table does not exists.
if(!file.exists(paste0(indicators_folder, "tracking_pop_agg.feather"))){

 tic()
 tracking_pop_agg = foreach(file = files[1:3], .combine = rbind, .packages = c("dplyr","tidyverse")) %do% {
  cat("\t",file,"\t||")

  # tracking table
  tracking = read_feather(path = paste0(input_folder, file))
  # aggregating variables
  tracking$BC = dict$BC_all$type[match(tracking$birth_control_ud, dict$BC_all$birth_control)]
  tracking = tracking %>% filter(BC %in% c("F","I"))
  birth_year_bins = data.frame(birth_year_bin = unique(tracking$birth_year_bin))
  birth_year_bins = birth_year_bins %>%
   mutate(birth_year_bin_mid = (as.numeric(str_sub(birth_year_bin,1,4)) + as.numeric(str_sub(birth_year_bin,6,9)))/2)
  tracking = tracking %>% mutate(
   birth_year_bin_mid = birth_year_bins$birth_year_bin_mid[match(birth_year_bin, birth_year_bins$birth_year_bin)],
   age = round_date(date - lubridate::years(birth_year_bin_mid)) %>% year(),
   age_cat = cut(age, breaks = c(-Inf, 23, Inf), labels = c("<= 23", "> 23"))
  )

  # variables aggregates
  tracking_pop_agg_this_file = tracking %>%
   group_by(date, country_area, BC, age_cat) %>%
   dplyr::summarise(
    n_prot_sex = sum((category == "sex") & (type == "protected_sex"), na.rm = TRUE),
    n_unprot_sex = sum((category == "sex") & (type == "unprotected_sex"), na.rm = TRUE),
    n_wd_sex = sum((category == "sex") & (type == "withdrawal_sex"), na.rm = TRUE),
    n_exercise = sum(category == "exercise", na.rm = TRUE),
    n_long_sleep = sum((category == "sleep")&(type == ">9"), na.rm = TRUE),
    n_bleeding = sum((category == "period")&(type != "spotting"), na.rm = TRUE),
    n_medium_bleeding = sum((category == "period") & (type == "medium"), na.rm = TRUE),
    n_breast_pain = sum((category == "pain") & (type == "tender_breasts"), na.rm = TRUE),
    n_pill_taken = sum((category == "pill_hbc") & (type == "taken"), na.rm = TRUE)
   )

  # adding total sex
  tracking_pop_agg_this_file = tracking_pop_agg_this_file %>% mutate(n_sex = n_prot_sex + n_unprot_sex + n_wd_sex)

  ###
  # active tracking
  active_tracking_compressed = read_feather(path = paste0(input_active_tracking,"active_",file))
  active_tracking = expand_compressed_tracking(active_tracking_compressed)
  # aggregating variables
  active_tracking$BC = dict$BC_all$type[match(active_tracking$birth_control_ud, dict$BC_all$birth_control)]
  active_tracking =  active_tracking %>% filter(BC %in% c("F","I"))
  active_tracking = active_tracking %>% mutate(
```

```r
    birth_year_bin = users$birth_year_bin[match(user_id, users$user_id)],
    birth_year_bin_mid = birth_year_bins$birth_year_bin_mid[match(birth_year_bin, birth_year_bins$birth_year_bin)],
    age = round_date(date - lubridate::years(birth_year_bin_mid)) %>% year(),
    age_cat = cut(age, breaks = c(-Inf, 23, Inf), labels = c("<= 23", "> 23"))
  )
  active_tracking$country_area = tracking$country_area[match(active_tracking$user_id, tracking$user_id)]


  # total number of users
  active_tracking_agg = active_tracking %>%
    group_by(date, country_area, BC, age_cat) %>%
    summarize(n_users = sum(tracking, na.rm = TRUE))

  # We then need to join the feature aggregation with the # of active users
  tmp = dplyr::full_join(x = active_tracking_agg , y = tracking_pop_agg_this_file, by = c("date","country_area","BC", "age_cat")) %>%
    arrange(country_area, BC, age_cat, date) %>%
    replace_na(list(n_prot_sex = 0, n_unprot_sex = 0, n_wd_sex= 0, n_sex = 0,
              n_party = 0, n_bleeding = 0, n_medium_bleeding = 0, n_breast_pain = 0, n_pill_taken = 0))

  return(tmp)
}
toc()


tic()
# sum the results from all files
tmp = tracking_pop_agg %>% group_by(date, country_area, BC, age_cat) %>%
  summarise_each(.,sum) %>%  arrange(country_area, BC, age_cat, date)
# expand to have one row per possible date
tmp2 = expand.grid(date = time_vec, country_area = unique(tmp$country_area), BC = unique(tmp$BC), age_cat = unique(tmp$age_cat))
tmp3 = dplyr::full_join(tmp, tmp2, by = c("date","country_area","BC","age_cat")) %>%
  arrange(country_area, BC, date) %>%
  replace_na(., list(n_users = 0,n_prot_sex = 0,n_unprot_sex = 0, n_wd_sex= 0, n_sex = 0,
              n_exercise = 0, n_long_sleep = 0, n_bleeding = 0, n_medium_bleeding = 0, n_breast_pain = 0, n_pill_taken = 0))
# final table
tracking_pop_agg = tmp3

# save the results
write_feather(tracking_pop_agg, path = paste0(indicators_folder, "tracking_pop_agg.feather"))
toc()

}else{warning("The aggregation was not executed at this rendering. Loading data from a previous execution.")}
```

Warning: The aggregation was not executed at this rendering. Loading data from a
previous execution.

4.3   Filtering for last two years of data

As the number of users increased in time as the App became more popular, we only keep the last two years of
data to ensure a sufficient number of users.

```r
tracking_pop_agg = read_feather(path = paste0(indicators_folder, "tracking_pop_agg.feather"))

tracking_pop_agg = tracking_pop_agg %>%
 filter(date >= as.Date("2017-07-01"),
     date < as.Date("2019-07-01"),
     !is.na(age_cat))

nrow(tracking_pop_agg) == 730*2*2*6
```

```
## [1] TRUE
```

```
write_feather(tracking_pop_agg, path = paste0(indicators_folder, "tracking_pop_agg_last_two_years.feather"))
```

4.4 Additional categories

In addition to the 2x2 age and BC categories, a 3rd category (all) is created for both of these variables to include all users.

4.4.1 BC = all

```
tracking_pop_agg_BC_all = tracking_pop_agg %>%
  select(-BC) %>%
  group_by(country_area, age_cat, date) %>%
  summarize_each(funs = sum) %>%
  mutate(BC = "all") %>%
  select(all_of(colnames(tracking_pop_agg)))
```

```
## Warning: `summarise_each_()` is deprecated as of dplyr 0.7.0.
## Please use `across()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
tmp = bind_rows(tracking_pop_agg, tracking_pop_agg_BC_all)
```

```
tracking_pop_agg = tmp
```

4.4.2 Age = all

```
tracking_pop_agg_age_all = tracking_pop_agg %>%
  select(-age_cat) %>%
  group_by(country_area, BC, date) %>%
  summarize_each(funs = sum) %>%
  mutate(age_cat = "all") %>%
  select(all_of(colnames(tracking_pop_agg)))
```

```
tmp = bind_rows(tracking_pop_agg, tracking_pop_agg_age_all)
```

```
tracking_pop_agg = tmp
```

We check that all categories have 2 years (2*365 = 730) of data

```
tracking_pop_agg %>%
  group_by(country_area, BC, age_cat) %>%
  summarize(n = n()) %>%
  as.data.frame() %>%
  select(n) %>% unique()
```

```
## `summarise()` regrouping output by 'country_area', 'BC' (override with `.groups` argument)
```

n
730

```
write_feather(tracking_pop_agg, path = paste0(indicators_folder, "tracking_pop_agg_last_two_years_with_all_BC_and_age.feather"))
```

Finally, we check which categories have sufficient numbers of users at all time-points.

```
df = tracking_pop_agg %>%
  group_by(country_area, age_cat, BC) %>%
  summarize(min_n = min(n_users),
        max_n = max(n_users),
        median_n = median(n_users)) %>%
  ungroup() %>%
  mutate(age_cat = age_cat %>% factor(., levels = c("all","<= 23","> 23")),
      BC = BC %>% factor(., levels = c("all", "F","I"))) %>%
  filter(!is.na(age_cat)) %>%
  arrange(country_area, age_cat, BC)
```

`summarise()` regrouping output by 'country_area', 'age_cat' (override with `.groups` argument)

```
kable(df, format = "pandoc", caption = "Minimum, maximum and median number of users in each category")
```

Table 4.2: Minimum, maximum and median number of users in each category

country_area	age_cat	BC	min_n	max_n	median_n
Brazil - Central-West	all	all	6362	66825	59501.0
Brazil - Central-West	all	F	2735	37173	29331.5
Brazil - Central-West	all	I	3627	32411	28472.5
Brazil - Central-West	<= 23	all	5408	46681	41437.5
Brazil - Central-West	<= 23	F	1781	23866	19238.5
Brazil - Central-West	<= 23	I	3627	24678	21385.5
Brazil - Central-West	> 23	all	954	20185	17756.5
Brazil - Central-West	> 23	F	954	13360	10088.5
Brazil - Central-West	> 23	I	0	8650	6891.5
Brazil - Northeast	all	all	674	9472	8174.5
Brazil - Northeast	all	F	323	5680	4396.0
Brazil - Northeast	all	I	351	4150	3538.0
Brazil - Northeast	<= 23	all	581	7710	6612.0
Brazil - Northeast	<= 23	F	277	4740	3721.0
Brazil - Northeast	<= 23	I	304	3081	2723.5
Brazil - Northeast	> 23	all	93	1789	1544.5
Brazil - Northeast	> 23	F	46	940	671.0
Brazil - Northeast	> 23	I	47	1144	814.0
France	all	all	7235	66127	50506.0
France	all	F	2425	40759	28999.5
France	all	I	4810	25578	21467.5
France	<= 23	all	5724	43621	33167.5
France	<= 23	F	1686	24290	17000.5
France	<= 23	I	4038	19475	16083.5
France	> 23	all	1511	22554	17362.0
France	> 23	F	739	16491	11997.5
France	> 23	I	772	6148	5390.0
United Kingdom	all	all	7532	64377	56898.0
United Kingdom	all	F	4260	50745	43489.0
United Kingdom	all	I	3272	13984	13200.5
United Kingdom	<= 23	all	4817	34402	31122.5
United Kingdom	<= 23	F	2394	25053	22273.0
United Kingdom	<= 23	I	2423	9527	8924.0
United Kingdom	> 23	all	2715	30073	25586.5
United Kingdom	> 23	F	1866	25733	21228.5
United Kingdom	> 23	I	849	4515	4278.5
United States - California	all	all	1773	20009	16314.5
United States - California	all	F	1182	17203	13833.0
United States - California	all	I	591	2864	2494.0

country_area	age_cat	BC	min_n	max_n	median_n
United States - California	<= 23	all	1077	10992	8952.5
United States - California	<= 23	F	684	9104	7348.0
United States - California	<= 23	I	393	1914	1610.0
United States - California	> 23	all	696	9040	7360.5
United States - California	> 23	F	498	8115	6486.0
United States - California	> 23	I	0	966	868.0
United States - Northeast	all	all	2132	26105	21511.0
United States - Northeast	all	F	1311	21014	16867.0
United States - Northeast	all	I	325	5231	4600.5
United States - Northeast	<= 23	all	1043	14204	11935.5
United States - Northeast	<= 23	F	764	10793	8836.5
United States - Northeast	<= 23	I	0	3545	3029.5
United States - Northeast	> 23	all	848	11926	9584.0
United States - Northeast	> 23	F	547	10239	8031.5
United States - Northeast	> 23	I	301	1722	1553.0

4.5 Formatting to long format

Given that the aggregated time-series with sex and control features logs will be used in different files, we save them separately. Additionally, we transform both tables to a "long format" such that the type of sex (protected, unprotected, all) or the type of control feature becomes a column on which we can aggregate or group.

4.5.1 For sexual behavior

```
clue_sex_agg = tracking_pop_agg %>%
  select(country_area, BC, age_cat, date, n_users, n_sex, n_prot_sex, n_unprot_sex) %>%
  rename(n_all_sex = n_sex) %>%
  pivot_longer(cols = c(n_all_sex, n_prot_sex, n_unprot_sex), names_to = "sex_type", values_to = "n") %>%
  mutate(sex_type = sex_type %>% str_remove("n_"))

head(clue_sex_agg)
```

country_area	BC	age_cat	date	n_users	sex_type	n
Brazil - Central-West	F	<= 23	2017-07-01	1781	all_sex	114
Brazil - Central-West	F	<= 23	2017-07-01	1781	prot_sex	47
Brazil - Central-West	F	<= 23	2017-07-01	1781	unprot_sex	37
Brazil - Central-West	F	> 23	2017-07-01	954	all_sex	89
Brazil - Central-West	F	> 23	2017-07-01	954	prot_sex	23
Brazil - Central-West	F	> 23	2017-07-01	954	unprot_sex	43

```
nrow(clue_sex_agg) == nrow(tracking_pop_agg)*3
```

[1] TRUE

4.5.2 For control features

```
control_features_agg = tracking_pop_agg %>%
  select(country_area, BC, age_cat, date, n_users, n_exercise, n_long_sleep, n_medium_bleeding, n_breast_pain) %>%
  pivot_longer(cols = c(n_exercise, n_long_sleep, n_medium_bleeding, n_breast_pain), names_to = "control_features", values_to = "n") %>%
  mutate(control_features = control_features %>% str_remove("n_"))



head(control_features_agg)
```

country_area	BC	age_cat	date	n_users	control_features	n
Brazil - Central-West	F	<= 23	2017-07-01	1781	exercise	0
Brazil - Central-West	F	<= 23	2017-07-01	1781	long_sleep	0
Brazil - Central-West	F	<= 23	2017-07-01	1781	medium_bleeding	76
Brazil - Central-West	F	<= 23	2017-07-01	1781	breast_pain	63
Brazil - Central-West	F	> 23	2017-07-01	954	exercise	0
Brazil - Central-West	F	> 23	2017-07-01	954	long_sleep	0

```
nrow(control_features_agg) == nrow(tracking_pop_agg)*4
```

[1] TRUE

4.6   Visualizations of the aggregated time-series

We visualize here the aggregated time-series for each country-area and for users of any birth-control (BC = "all") and of any age (age_cat = "all").

```
A = clue_sex_agg %>%
 filter(BC == "all",
       age_cat == "all",
       sex_type == "all_sex") %>%
 mutate(country_area_col = dict$country_area$country_area_col[match(country_area, dict$country_area$country_area)])
```

```
ggplot(A, aes(x = date, y = n_users, col = country_area_col))+
 geom_line()+
 scale_color_identity(
  "Location",
  guide = "legend",
  breaks = dict$country_area$country_area_col,
  labels = dict$country_area$country_area)+
 ylab("# of active users")
```
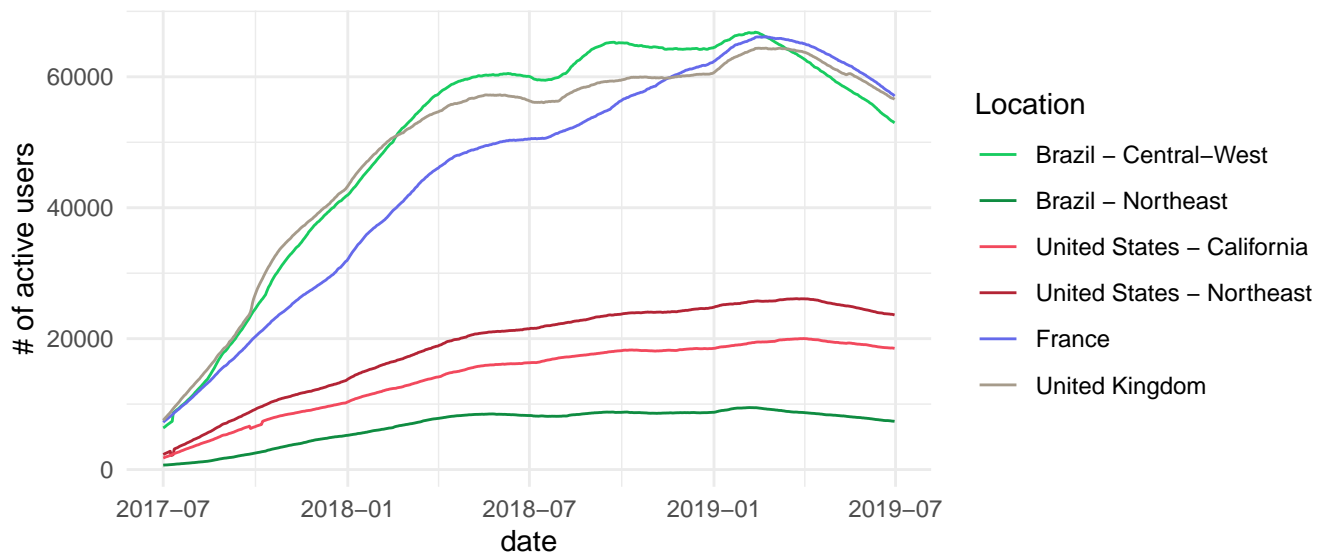

Figure 4.1: Number of active users at each time-point for each location

```
ggplot(A, aes(x = date, y = n, col = country_area_col))+
 geom_line()+
 scale_color_identity(
  "Location",
  guide = "legend",
  breaks = dict$country_area$country_area_col,
  labels = dict$country_area$country_area) +
```

```
guides(col = FALSE)+
ylab("# of sexual intercourses logged")
```
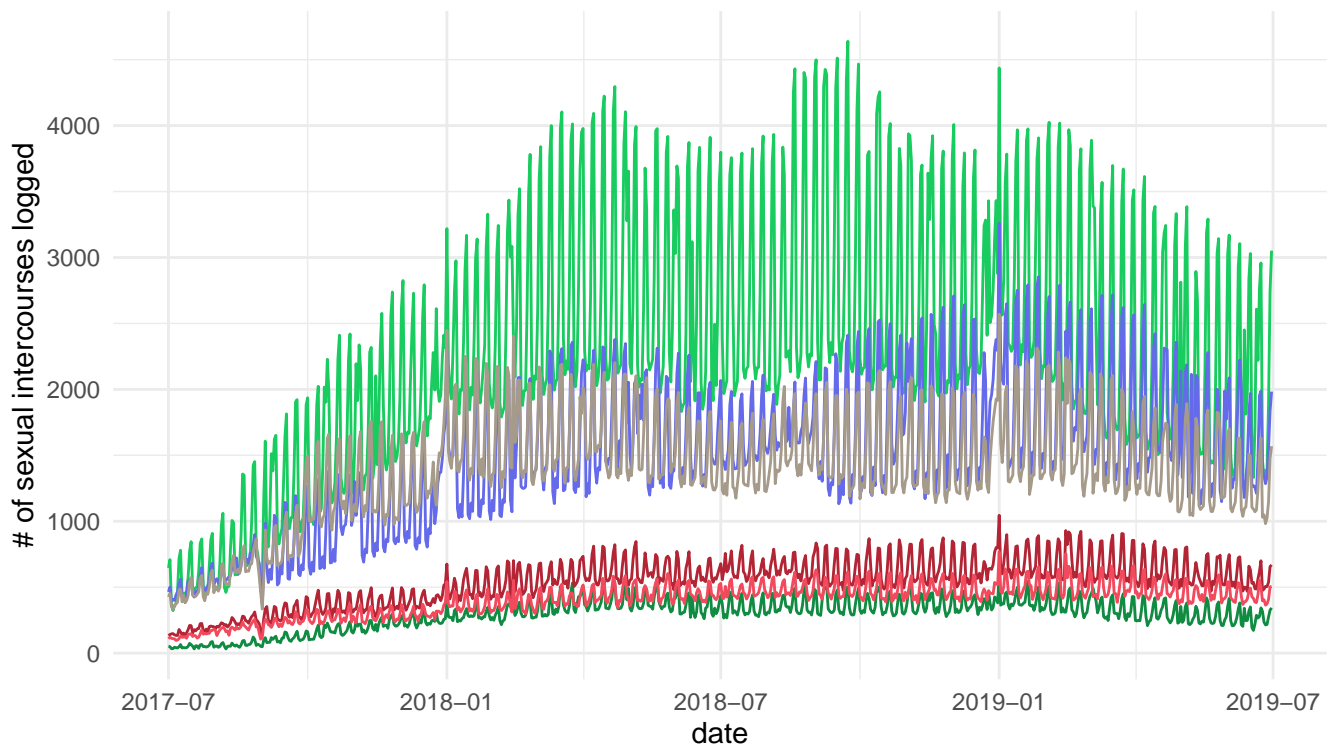

Figure 4.2: Number of sex logs at each time-point

```
ggplot(A, aes(x = date, y = n/n_users, col = country_area_col))+
geom_line()+
scale_color_identity(
  "Location",
  guide = "legend",
  breaks = dict$country_area$country_area_col,
  labels = dict$country_area$country_area) + guides(col = FALSE)+
ylab("# of sex / # of active users")
```
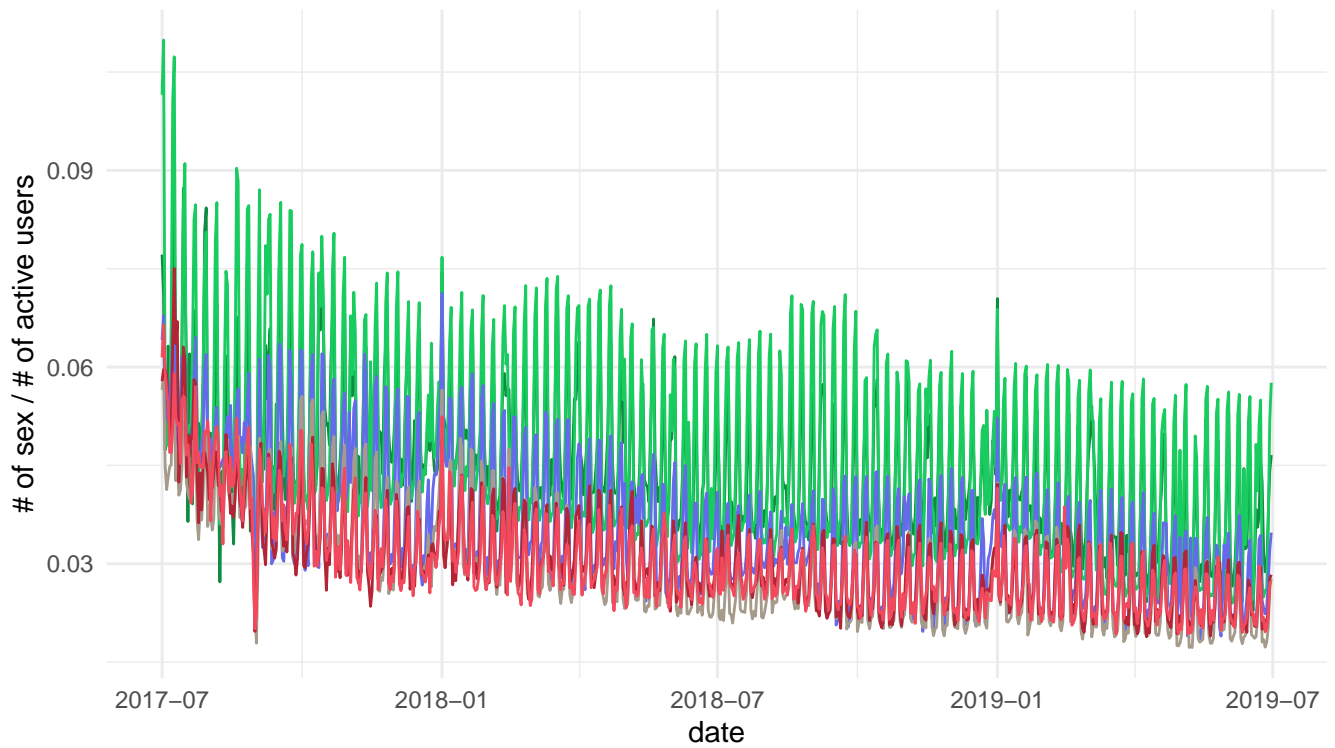
Figure 4.3: Relative number of sex logs at each time-point

```
write_feather(clue_sex_agg, path = str_c(IO$out_Rdata,"aggregated_sex_counts_clue_July2017-June2019_incl.feather"))

write_feather(control_features_agg, path = str_c(IO$out_Rdata,"aggregated_control_features_counts_clue_July2017-June2019_incl.feather"))
```

5   Birth Models

5.1   Mathematical Models

Mathematical model for number of births:

$$C(t) = f(t)S(t) \tag{1}$$

$$B(t) = \int_{t-max(G)}^{t-min(G)} d^G(\tau)\,(1 - l(\tau))\,C(\tau)d\tau \tag{2}$$

where

$C(t)$ = number of conceptions at time t;

$f(t)$ is the fertility at time t (= the odds of a conception from a sexual intercourse);

$S(t)$ the number of sexual intercourses at time t;

$B(t)$ is the number of births at time t;

G is the gestation duration (i.e. the duration of a pregnancy)

$d^G(t)$ is the probability density of the gestation duration and;

$l(t)$ is the pregnancy loss rate at time t (= the fraction of pregnancies ending up in a loss).

This model can be rewritten as

$$B(t) = \int_{t-max(G)}^{t-min(G)} d^G(\tau)\,(1 - l(\tau))\,f(\tau)\,S(\tau)d\tau \tag{3}$$

$$\Longrightarrow B(t) = \int_{t-max(G)}^{t-min(G)} d^G(\tau)\,F(\tau)\,S(\tau)\,d\tau \tag{4}$$

Where $F(\tau)$ is a function combining fertility and loss rate.

In discrete time-steps, the model becomes:

$$B_t = \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau\,F_\tau\,S_\tau \tag{5}$$

We will assume that the gestation duration follows a normal distribution around the average distribution duration $\bar{G}$ of standard deviation = σ_G days:

$$G \sim \mathcal{N}(\bar{G}, \sigma_G) = d_\tau \tag{6}$$

5.1.1   Comparing actual births with simulated births records from the sex reports of the app users.

In this study, we will compare the actual monthly birth records B_t^m (m stands for *measured*) with simulated births B_t^e (e stands from *estimated*) from the sexual behavior of the app users (S_t) while making different assumptions on the fertility F_t (see Models A, B & C below).

5.1.2 Births, fertility and sexual behavior expressed as relative changes around their average value.

Because we have data for a small fraction of the population (the app users) and because we don't know how many of the app users are never reporting sexual intercourses, we cannot extrapolate the absolute number of births from the number of sexual intercourses reported in the app. However, we can compare the relative changes throughout the year in reported sexual intercourse with the relative variations in births.

Two approaches are possible to compare the relative changes:

1. expressing B_t, S_t and F_t as a variation around a mean value.

I.e. $X_t = \bar{X} (1 + \tilde{X}_t)$

The model becomes

$$B_t = \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, F_\tau \, S_\tau \tag{7}$$

$$= \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, \bar{F} \, (1 + \tilde{F}_\tau) \, \bar{S} \, (1 + \tilde{S}_\tau) \tag{8}$$

$$= \bar{F}\bar{S} \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, (1 + \tilde{F}_\tau) \, (1 + \tilde{S}_\tau) \tag{9}$$

$$= E \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, (1 + \tilde{F}_\tau) \, (1 + \tilde{S}_\tau) \tag{10}$$

where E is a scaling factor that reflects how the relative variations in sexual intercourse translates into births.

2. expressing B_t and S_t as the fraction of the total number of yearly births or sexual intercourses: $X_t = \mathbf{X} \, x(t)$ where $\mathbf{X}$ is the total number of births or sexual intercourses in the population.

In this case, the model becomes

$$B_t = \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, F_\tau \, S_\tau \tag{11}$$

$$= \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, \mathbf{F} \, F_\tau \, \mathbf{S} \, s_\tau \tag{12}$$

$$= \mathbf{S} \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, F_\tau \, s_\tau \tag{13}$$

$$= E_2 \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, F_\tau \, s_\tau \tag{14}$$

where $E_2 = \mathbf{S}$ is a scaling factor.

Note that we have the equivalence: $(1 + \tilde{S}_t) = 365 * s_t$ over a non-leap year (for leap year, the multiplicative factor is 366).

In our implementation, we chose the first approach:

$$B_t = E \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, (1 + \tilde{F}_\tau) \, (1 + \tilde{S}_\tau) \tag{15}$$

5.1.3   Model A: Constant Fertility, Varying Sexual Behavior.

In **model A**, we assume that f and l (and thus F) are constant (not seasonal): seasonal patterns in births are driven by seasonal patterns in sexual intercourses.

We can thus simplify to:

$$B_t = E \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, (1 + \tilde{S}_\tau) \tag{16}$$

5.1.4   Model B: Varying fertility, Constant Sexual Behavior

In **model B**, we assume that seasonal variations in birth is only driven by varying fertility and that variations in sex does not contribute to the seasonal variations ($S_t = \bar{S}$):

$$B_t = E \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, (1 + \tilde{F}_\tau) \tag{17}$$

$$\text{with } \tilde{F}_\tau = \alpha \, \cos(\omega \, (\tau - T)) \tag{18}$$

$$\text{and } \omega = \frac{2\pi}{P} \tag{19}$$

Where

α is the relative amplitude of the fertility variation,

T is the time of peak fertility,

ω is the frequency of the sine curve and

P the period (i.e. 12 months or 365 days).

5.1.5   Model C: Varying fertility, Varying Sexual Behavior

In **model C**, we assume that both varying fertility and sexual behavior drive births rhythms.

$$B_t = E \sum_{\tau=t-max(G)}^{t-min(G)} d_\tau \, (1 + \tilde{F}_\tau) \, (1 + \beta \tilde{S}_\tau) \tag{20}$$

$$\text{with } \tilde{F} = \alpha \, \cos(\omega \, (t - T)) \tag{21}$$

$$\text{and } \omega = \frac{2\pi}{P} \tag{22}$$

Where

α is the relative amplitude of the fertility variation,

β is the relative amplitude of the variations in sexual activity,

T is the time of the year with peak fertility,

ω is the frequency of the sine curve and

P the period (i.e. 12 months or 365 days).

Note that this model does not make assumptions on the underlying biological causes of fertility. The variations in fertility may originate from variation in female fertility (either in the conception rate or the loss rate) or in male fertility or both.

5.1.6 Model parameters

Each model has some parameters which need to be fixed or estimated.

5.1.6.1 Scaling factor
E, which is the average daily birth rate, is computed from the births records themselves:

- First the monthly births are normalized by the number of days in each month (we re-scale the births as if each month had a duration of 30 days).

- Then the moving average of these normalized monthly births is computed over a period of 12 months

- Finally, this moving average is divided by 30 to obtain the average daily births and values are extrapolated to obtain a smooth variation of the daily births over the years with available data.

5.1.6.2 Fertility and sexual activity parameters
There are two fertility parameters α and T

α is the relative amplitude of the fertility cosine curve. It can take values between 0 and 1 as a relative amplitude larger than one would lead to negative values ($\tilde{F} = \bar{F}\,(1 + \alpha \cos(\omega\,(t - T)))$. T is the peak time of fertility, i.e. when the cos takes its maximal value. Time is expressed as "fraction of the year", to T also takes its value in [0,1].

There is one parameter to adjust the relative amplitude of the sexual activity: β.

These three parameters are estimated so that they minimize the sum of square of the residuals (SSR) between the simulated births and the actual birth records:

$$\text{SSR} = \sum_t (B_t^m - B_t^e)^2$$

Where B_t^m (m for measured) are the birth records at time t and B_t^e (e for estimated) are the simulated births at time t.

We use the function optim with method = "L-BFGS-B" for the optimization of these parameters.

5.1.6.3 Gestation duration.

5.1.6.3.1 Average gestational duration
While the average gestational age at birth (gestational duration) is of approximately 9 months (38 weeks), differences in gestational age at birth were found in different countries (Delnord et al. 2018).

In (Delnord et al. 2018), gestational age at birth in France was of 39.4 weeks, of 39.6 weeks in the UK and of 39 weeks in the US (California excluded).

These differences can be explained by different preterm birth rates, consistently found higher in the US than in most European countries (Delnord and Zeitlin 2019; MacDorman et al. 2014), or by delivery choices (e.g. planned C-sections) and most common method for the evaluation of the gestational age (e.g. estimation by ultrasounds or based on the last menstrual period).

Here we fixed the average gestation duration ($\bar{G}$) to a given value for each considered country based on the existing literature, when available, or, for Brazil, based on the alignment of the high-frequency fluctuations in the birth curve with the sexual behavior peaks (See section 5.8).

```
G_table = dict$country_area %>% mutate(G = c(37.5,37.5,37.5,37.5,38,38.5)) %>% select(country_area, G)

write_feather(G_table, path = str_c(IO$out_Rdata,"Gestation_par_table.feather"))

kable(G_table, format = "pandoc", caption = "Average gestation duration (in weeks) for each considered area")
```

Table 5.1: Average gestation duration (in weeks) for each considered area

country_area	G
Brazil - Central-West	37.5
Brazil - Northeast	37.5
United States - California	37.5
United States - Northeast	37.5
France	38.0
United Kingdom	38.5

5.1.6.3.2   Variability of gestational duration   In normal, non pre-term births, the natural gestational age at birth was found to show important variation (Jukic et al. 2013). Following the data presented in (Jukic et al. 2013), the standard deviation of the gestational duration was set to $\sigma_G = 10$ days.

5.2 Workflow

For each user category (i.e. country/area, age group and birth control type)

and each type of sex (protected, unprotected, sum of both sex type):

1. **Sexual behavior**

    a. Compute the relative changes in sexual intercourse from the app data

    b. Model the sexual behavior as a combination of a weekly trend, a seasonal trend and a holiday response

2. **Overall births trend (average daily birth)**

    a. Correct for the number of days in each month

    b. Compute the average daily birth for each country/area (the scaling factor E)

3. **Optimize the model parameters** (α, β and T) for each birth model (A, B or C); i.e. minimize the SSR from several initial values of the parameters by:

    a. Predicting the daily sexual behavior from the weekdays-seasons-holidays model for the time-window which have births records available.

    b. Simulating the daily births

    c. Aggregating as a monthly time-series

    d. SSR: Computing the difference (the residuals) between the simulated and actual births.

4. **Simulate monthly births** with the optimized parameters

5. **Determine best birth model (A, B or C)**

    a. By comparing the SSR and the AIC (Akaike Information Criteria) for each model

    b. By performing a **seasonal decomposition** to compare the seasonal trends of the actual and simulated births.

5.3 Sexual behavior: modelling relative changes

```
clue_sex_agg = read_feather(path =str_c(IO$out_Rdata,"aggregated_sex_counts_clue_July2017-June2019_incl.feather"))
str(clue_sex_agg)
```

```
## tibble [118,260 x 7] (S3: tbl_df/tbl/data.frame)
##  $ country_area: chr [1:118260] "Brazil - Central-West" "Brazil - Central-West" "Brazil - Central-West" "Brazil - Central-West" ...
##  $ BC      : chr [1:118260] "F" "F" "F" "F" ...
##  $ age_cat   : chr [1:118260] "<= 23" "<= 23" "<= 23" "> 23" ...
##  $ date    : Date[1:118260], format: "2017-07-01" "2017-07-01" ...
##  $ n_users   : num [1:118260] 1781 1781 1781 954 954 ...
##  $ sex_type  : chr [1:118260] "all_sex" "prot_sex" "unprot_sex" "all_sex" ...
##  $ n      : num [1:118260] 114 47 37 89 23 43 135 53 40 92 ...
```

These data are daily sex counts and sex rates (i.e. sex counts divided by the number of active users) aggregated by geographic area and birth control type from the sex logs of the users of the menstrual cycle tracking app Clue.

The preparation of these data from the raw logs can be found in section 3 (file 2_app_data_processing_and_filtering.Rmd) and the aggregation has been done in section 4 (file 3_app_data_aggregation.Rmd).

5.3.1 Relative changes in sexual intercourses computed from the app data

For each day, each country/area and BC, we have the total number of logs for sexual intercourses (protected, unprotected and combined) X as well as the number of active users U.

We can compute the ratio between these two and have the relative count of sexual intercourses: $r = \frac{X}{U}$, for every day, country/area, age category, BC and sex type ($r_t^{\text{geo,age cat,BC,sex type}}$ which we can note r_t^{cat} for clarity).

```
clue_sex_agg = clue_sex_agg %>%
 mutate(r = n/n_users)
```

```
# if n_users is 0, this leads to infinite values, which we will replace by the max of that time-series
clue_sex_agg = clue_sex_agg %>%
 group_by(country_area, BC, age_cat, sex_type) %>%
 mutate(r = ifelse(is.infinite(r),sort(unique(r), decreasing = TRUE)[2],r))
```

```
ggplot(clue_sex_agg %>% dplyr::filter(BC == "all", age_cat == "all"), aes(x = date, y = r, col = sex_type))+
 geom_line()+
 facet_grid(country_area ~ .)+
 theme(legend.position = "bottom",
     strip.text.y = element_text(angle = 0, hjust = 0))
```
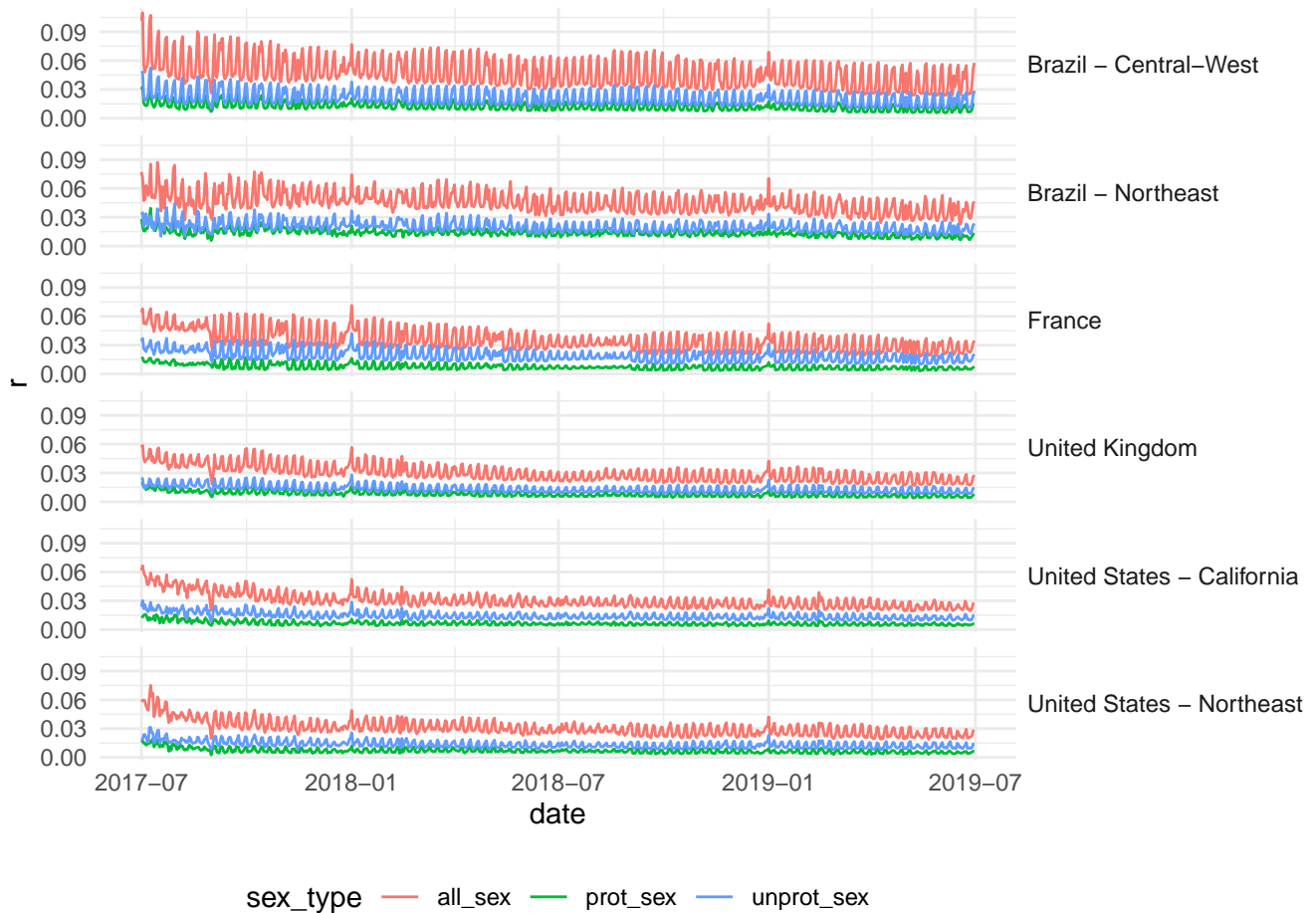
Figure 5.1: Number of sex logs divided by the number of active users (users from all age and on any birth control type)

This time-series has a visible decreasing trend as the number of people using the app increases over time and the amplitude of the variations decreases.

We compute the trend.

```
tmp = clue_sex_agg %>%
  arrange(country_area, BC, age_cat, sex_type, date) %>%
  group_by(country_area, BC, age_cat, sex_type) %>%
  mutate(t = row_number(),
      trend = predict(loess(r ~ t))) %>%
  ungroup() %>% select(-t)


clue_sex_agg = tmp
```

```
ggplot(clue_sex_agg %>% dplyr::filter(BC == "all", age_cat == "all"), aes(x = date, y = r, col = sex_type))+
  geom_line(alpha = 0.5)+
  geom_line(aes(y = trend))+
  facet_grid(country_area ~ ., scale = "free_y")+
  theme(legend.position = "bottom",
      strip.text.y = element_text(angle = 0, hjust = 0))
```
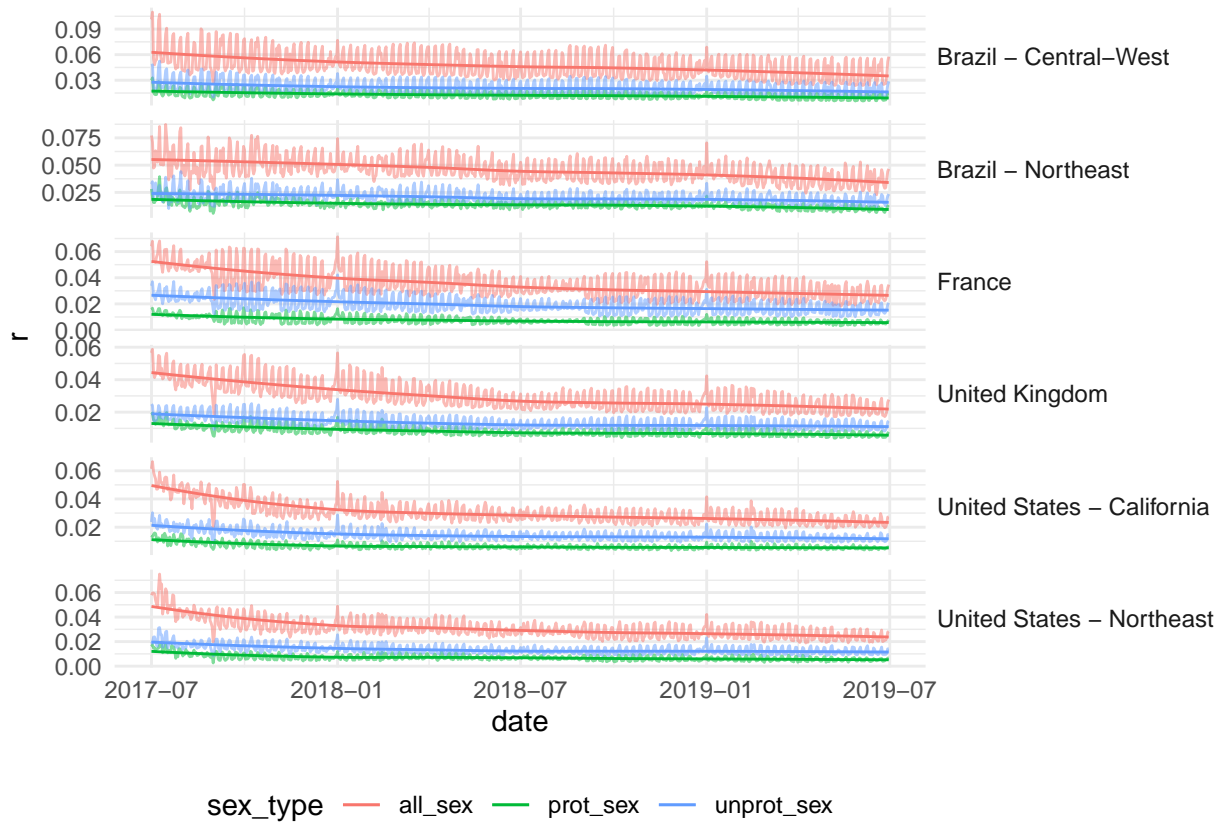
Figure 5.2: Relative sex counts (users from all age and on any birth control type) and trend in relative sex counts

```
tmp = tmp %>% mutate(country_area == country_area %>% factor(.,levels = levels(dict$country_area$country_area)),
          country_area_col = dict$country_area$country_area_col[match(country_area, dict$country_area$country_area)])

ggplot(tmp %>% dplyr::filter(BC == "all", age_cat == "all", sex_type == "all_sex"), aes(x = date, y = r, col = country_area_col))+
 geom_line(alpha = 0.5)+
 geom_line(aes(y = trend))+
 scale_color_identity()+
 guides(col = FALSE)+
 facet_grid(country_area ~ ., scale = "free_y")+
 theme(strip.text.y = element_text(angle = 0, hjust = 0))
```
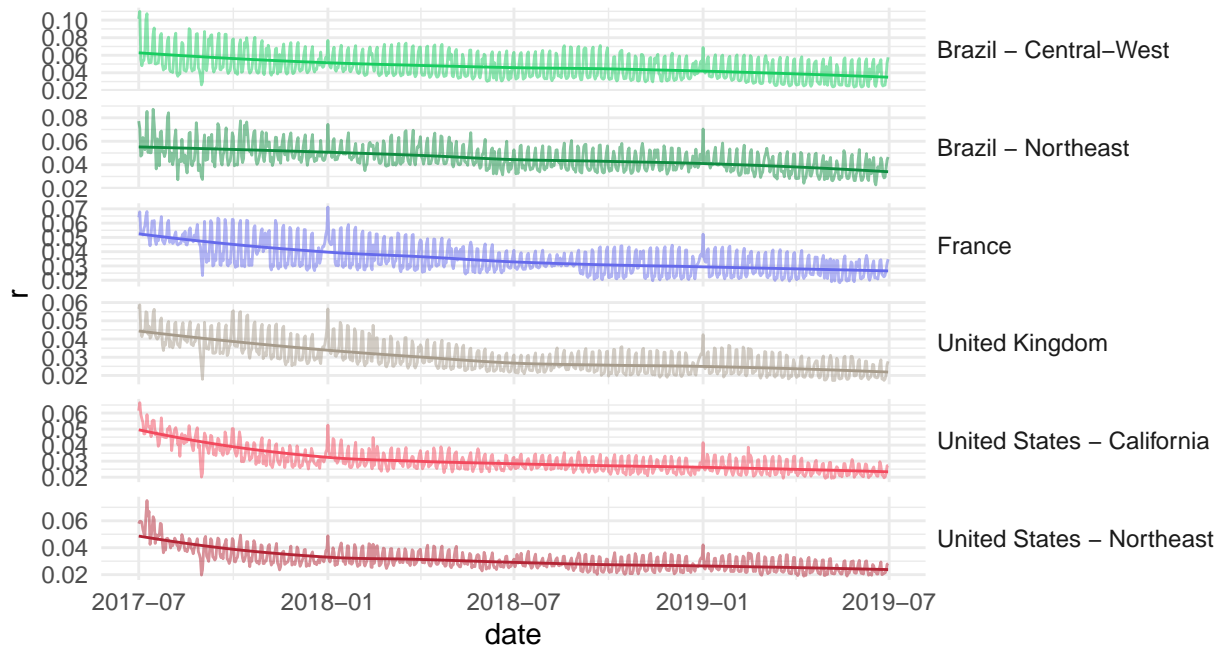
Figure 5.3: Relative sex counts (users from all age and on any birth control type, any sex log) and trend in relative sex counts

```
rm(tmp)
```

[http://onlinestatbook.com/2/sampling_distributions/samp_dist_mean.html]

We need to perform a mutliplicative de-trending ($r(d) = t(d) * x(d)$)

```
g1 = ggplot(clue_sex_agg %>% dplyr::filter(BC == "all", age_cat == "all"),
    aes(x = date, y = r / trend)) +
  geom_line() +
  facet_grid(country_area ~ .)+
  ggtitle("multiplicative decomposition")

g2 = ggplot(clue_sex_agg %>% dplyr::filter(BC == "all", age_cat == "all"),
    aes(x = date, y = r - trend)) +
  geom_line() +
  facet_grid(country_area ~ .)+
  ggtitle("additive decomposition")

cowplot::plot_grid(g1, g2, ncol = 2, align = "h")
```
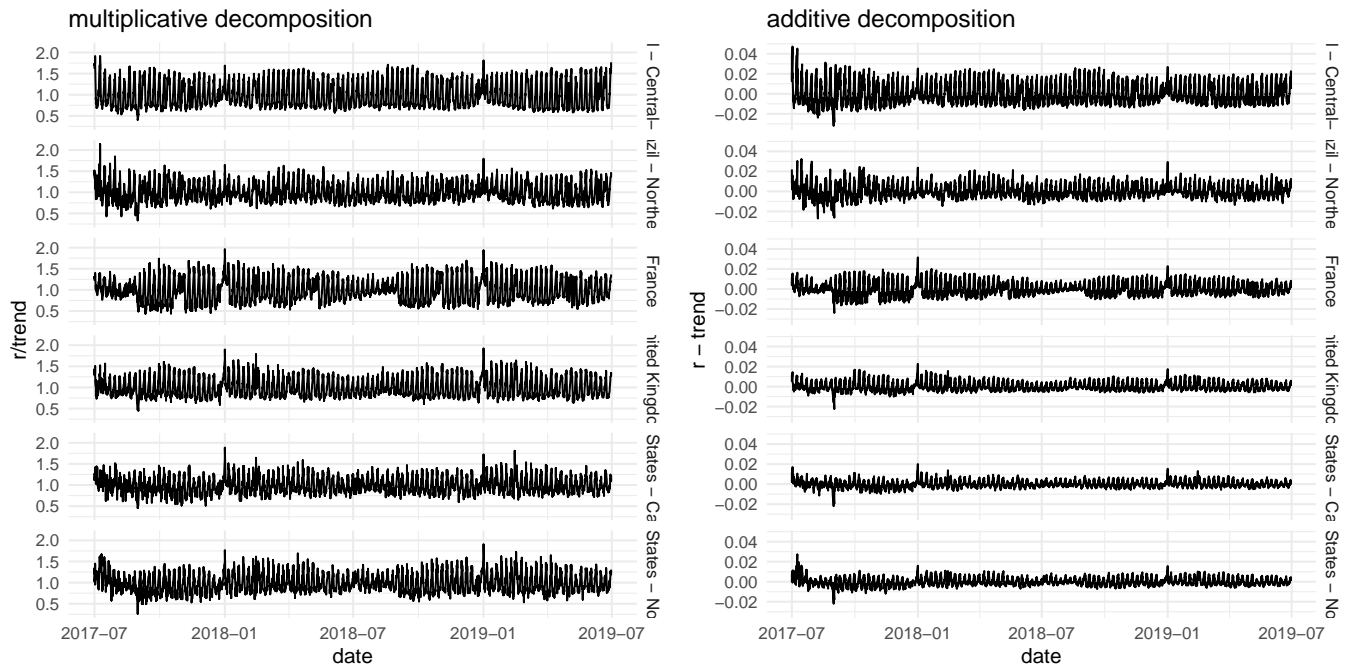
Figure 5.4: Comparison of mutliplicative (accounts for variation of the variance with the number of samples) vs additive (does not account) detrending.

For comparison, we also show the additive detrending which does not remove the decreasing trend in variance.

```
clue_sex_agg = clue_sex_agg %>% dplyr::mutate(x = r/trend)
```

These multiplicatively detrended time-series ($x(d)$) correspond to $(1 + \tilde{S(t)})$ in the model described above.

```
A = clue_sex_agg %>%
  filter(BC == "all",
         age_cat == "all",
         sex_type == "all_sex") %>%
  mutate(country_area_col = dict$country_area$country_area_col[match(country_area, dict$country_area$country_area)])


ggplot(A, aes(x = date, y = x, col = country_area_col))+
  geom_line()+
  scale_color_identity()+
  guides(col = FALSE)+
  ylab("detrended relative sex")+
  facet_grid(country_area ~ .)+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
```

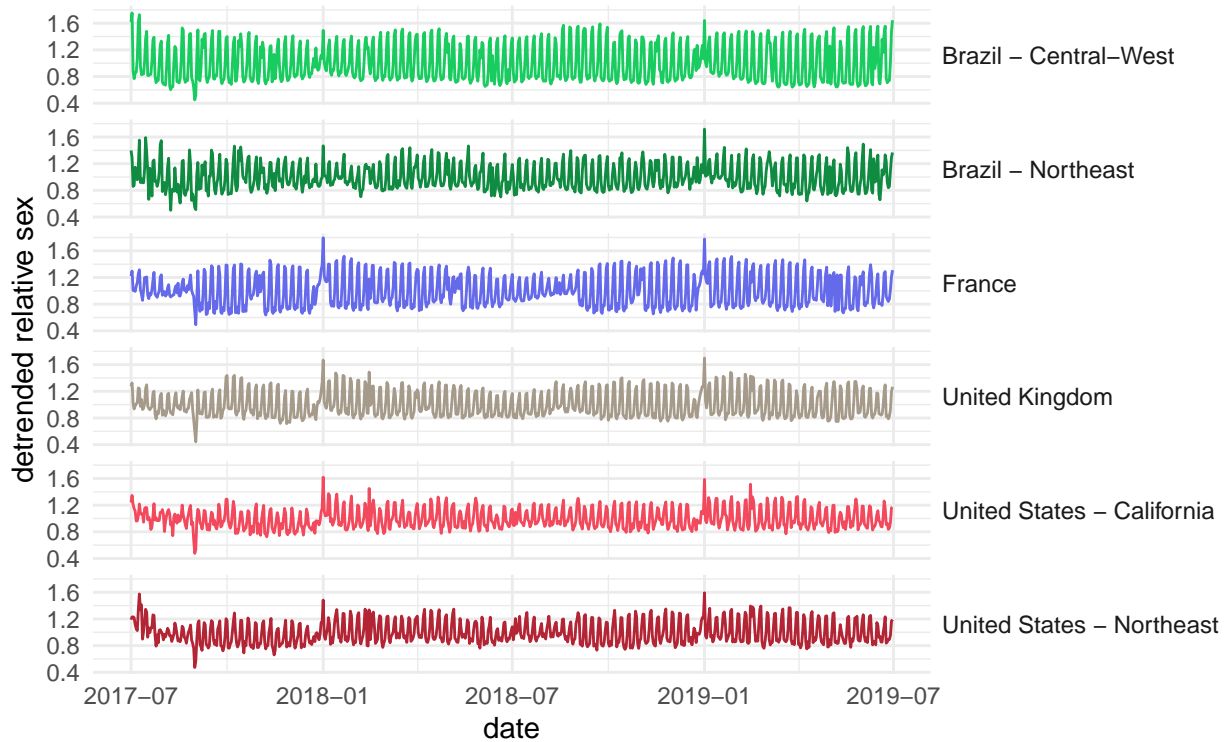Figure 5.5: Detrended relative changes in sexual behavior (all users of all age group on any birth control type, all sex).

5.3.2 Comparing sexual activity by sex type.

Here we visualize the relative sexual activity and compare by sex type for each location.

```
g = ggplot(clue_sex_agg %>% filter(BC == "all", age_cat == "all"), aes(x = date, y = x, col = sex_type)) +
    geom_line()+
    facet_grid(country_area ~ .)


g
```
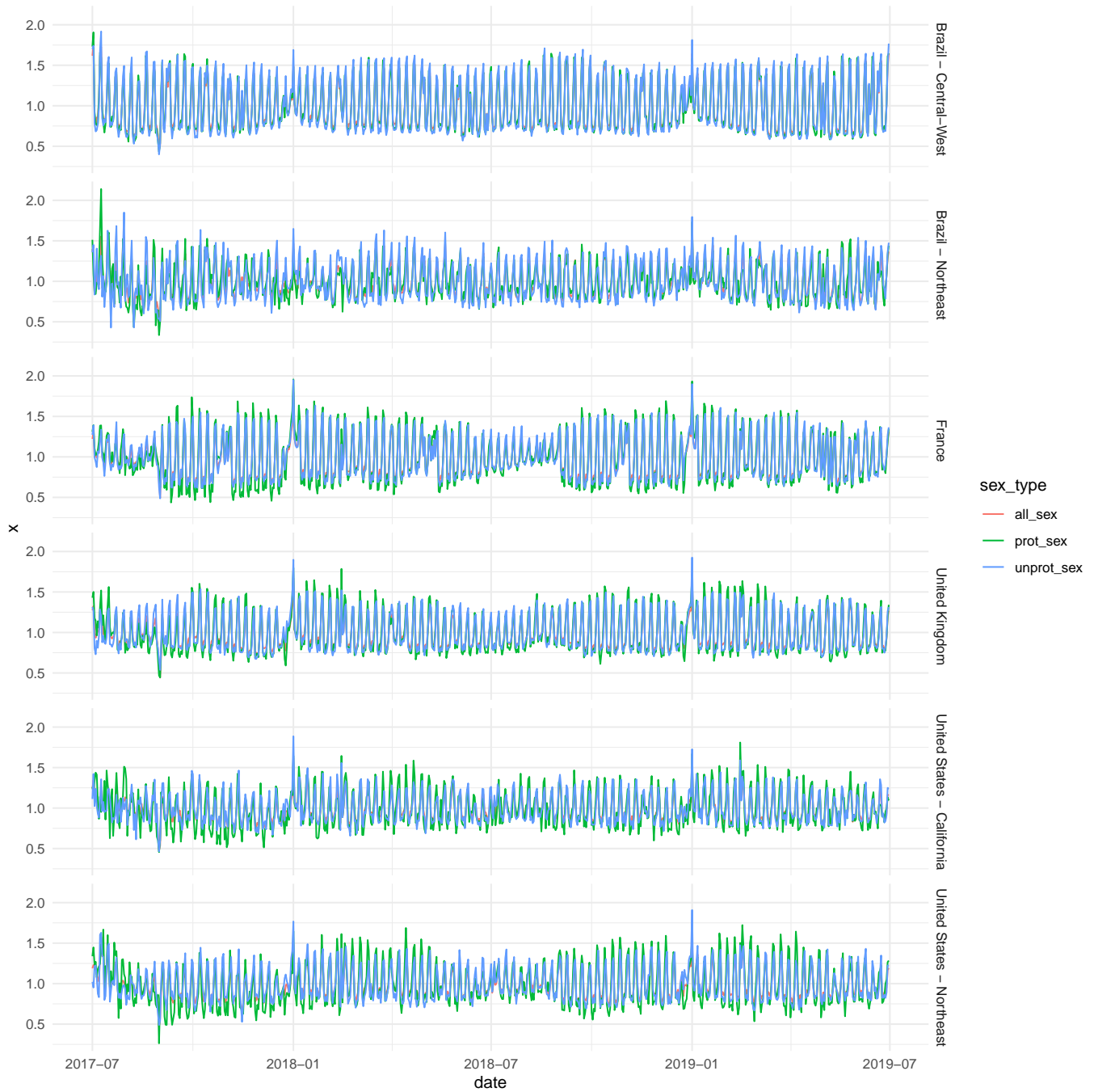
Figure 5.6: Comparing sexual activity by sex type for all users of each location

```
for(ca in unique(clue_sex_agg$country_area)){
  g = ggplot(clue_sex_agg %>% filter(country_area == ca), aes(x = date, y = x, col = sex_type)) +
    geom_line()+
    facet_grid(BC + age_cat ~ .) +
    ggtitle(ca)

  print(g)
}
```

Figure 5.7: Comparing sexual activity by sex type for each location and user group.

Figure 5.8: Comparing sexual activity by sex type for each location and user group.

Figure 5.9: Comparing sexual activity by sex type for each location and user group.
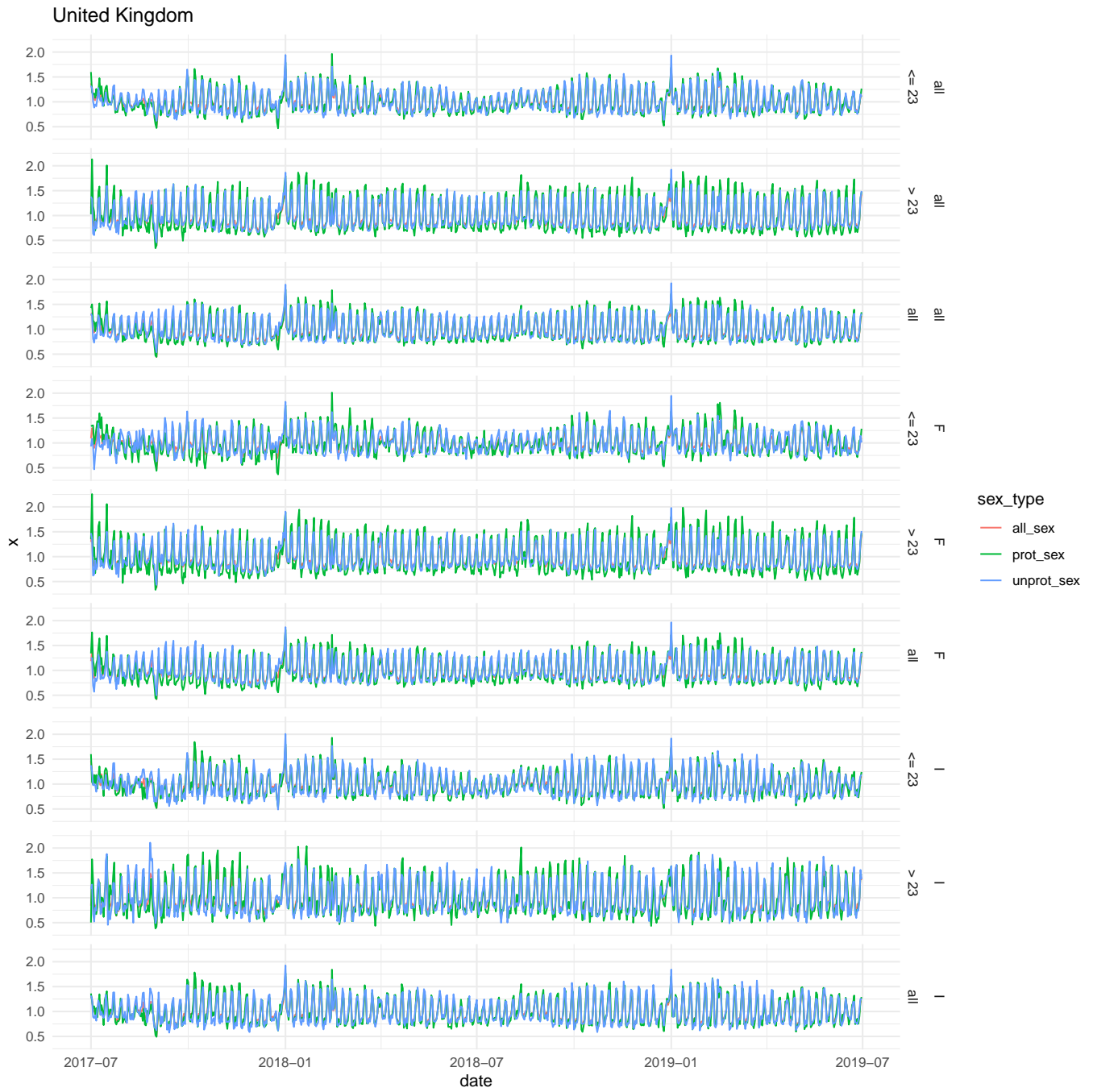
Figure 5.10: Comparing sexual activity by sex type for each location and user group.
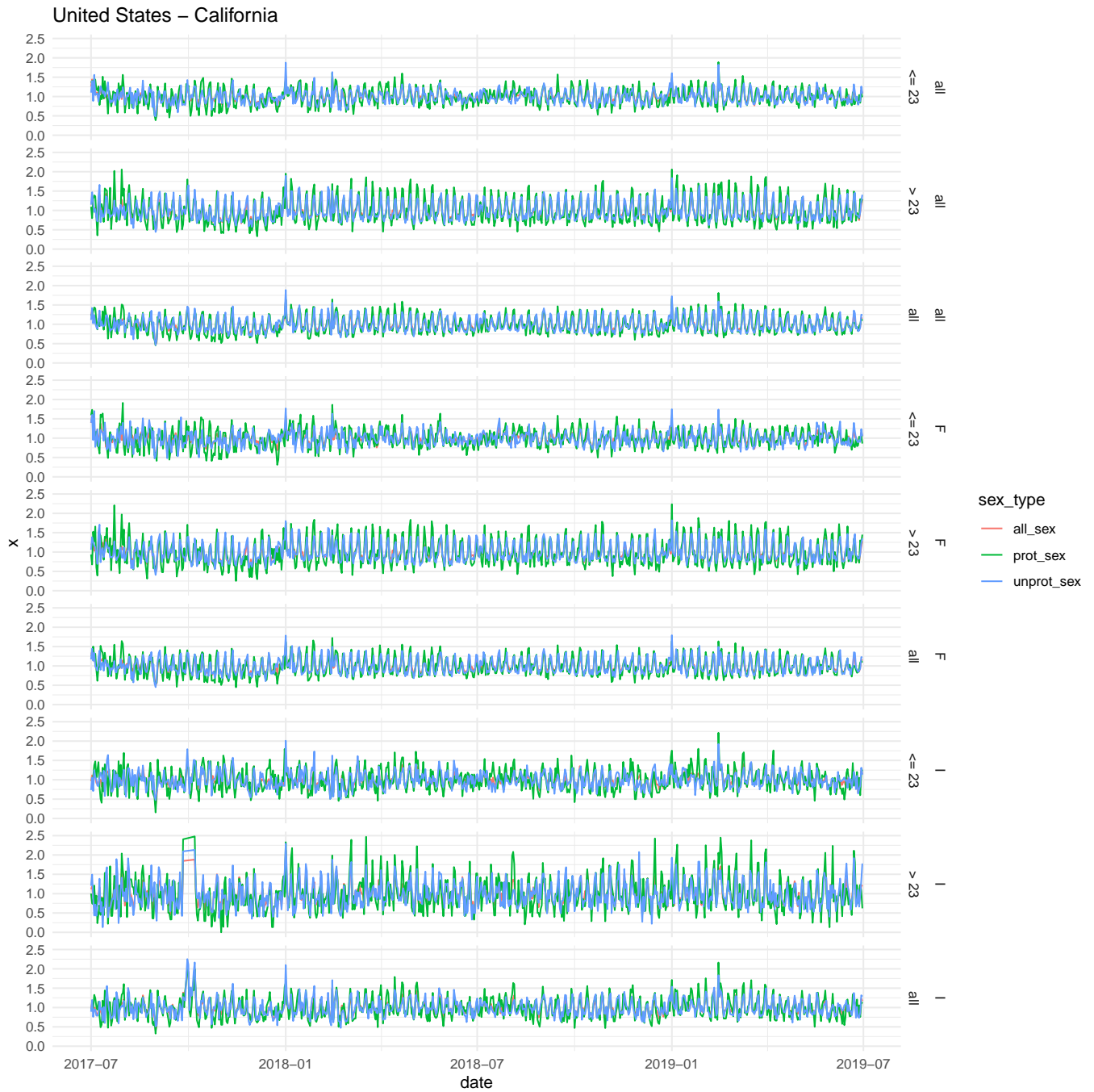
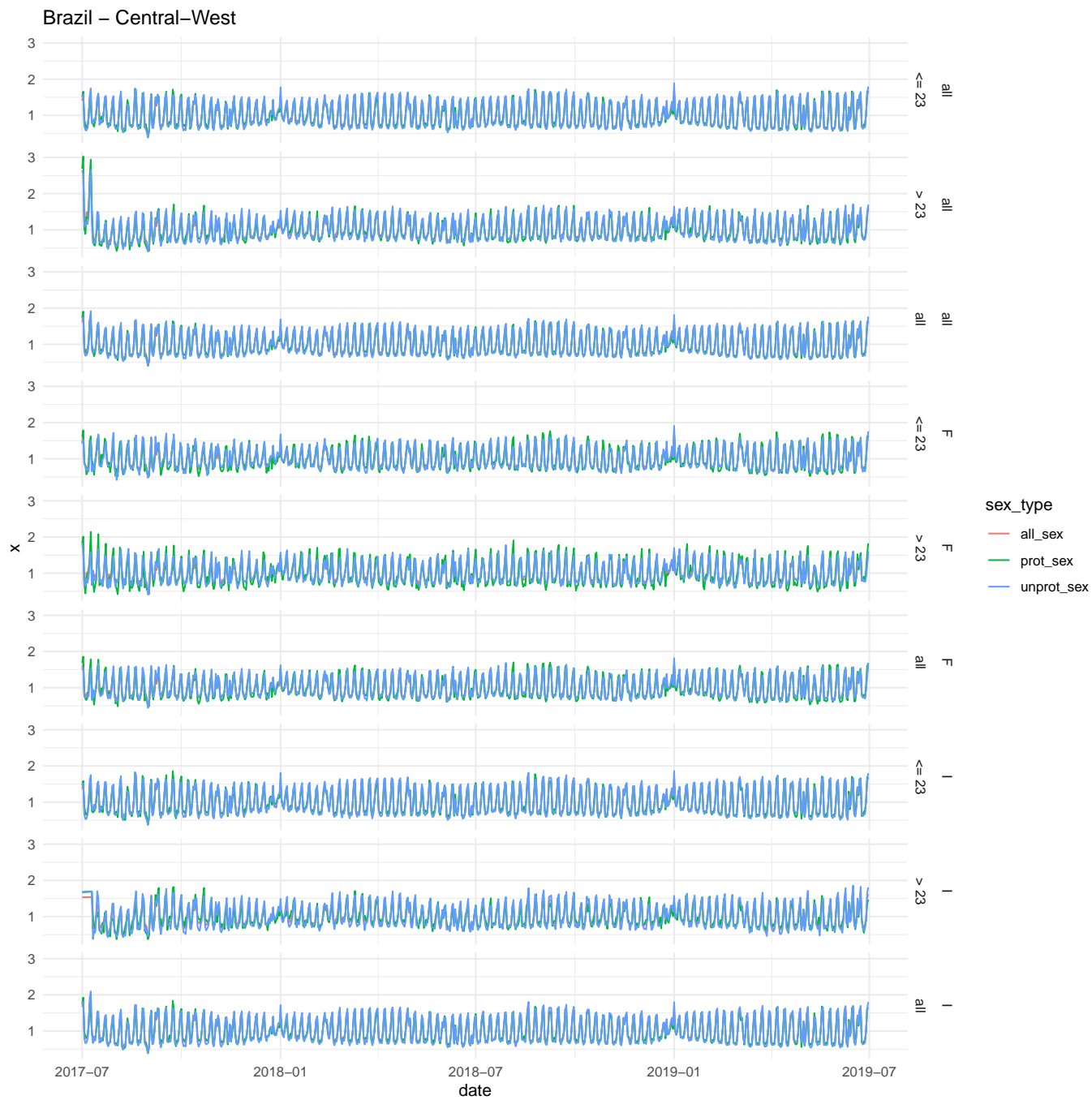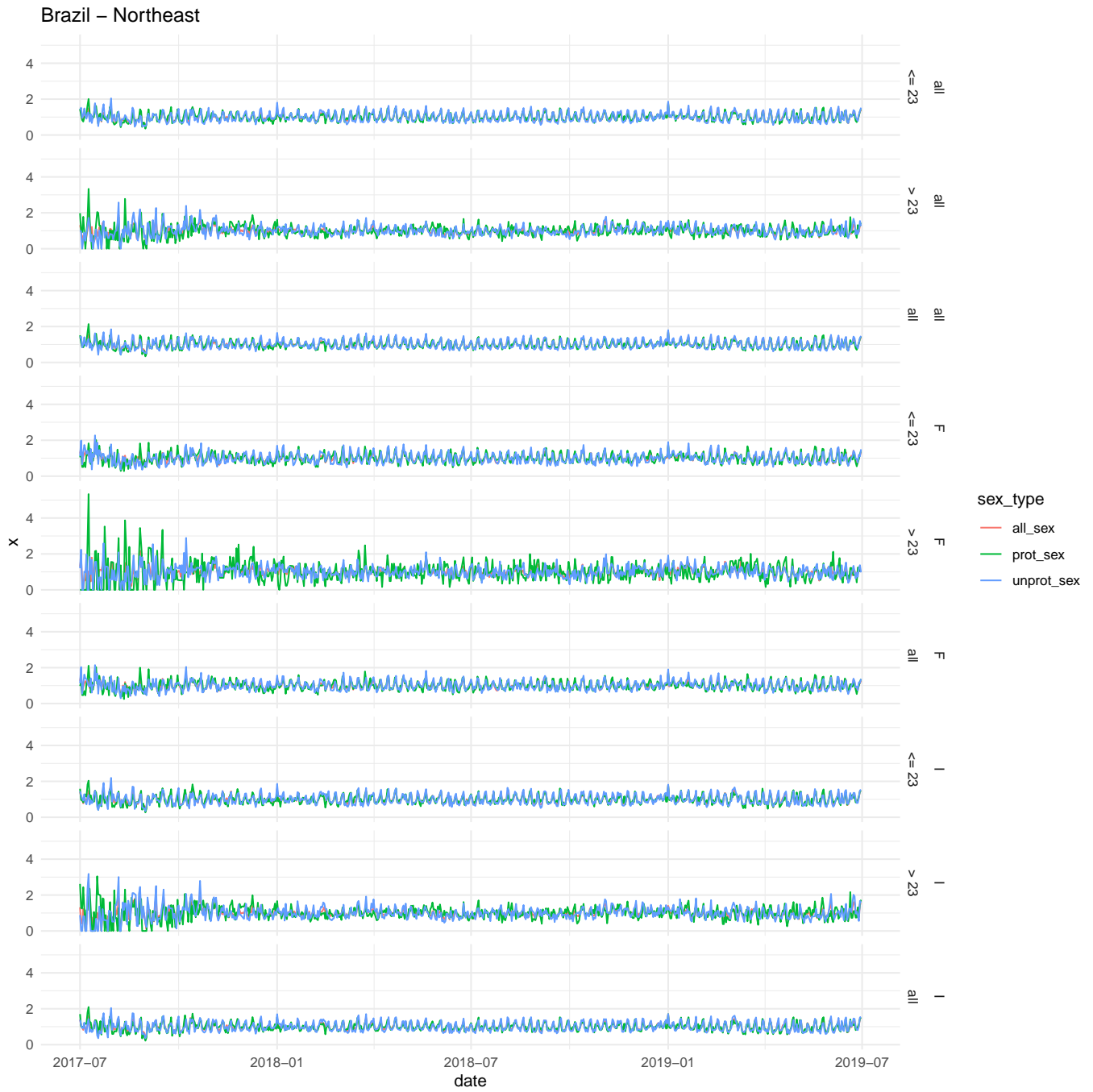Figure 5.11: Comparing sexual activity by sex type for each location and user group.
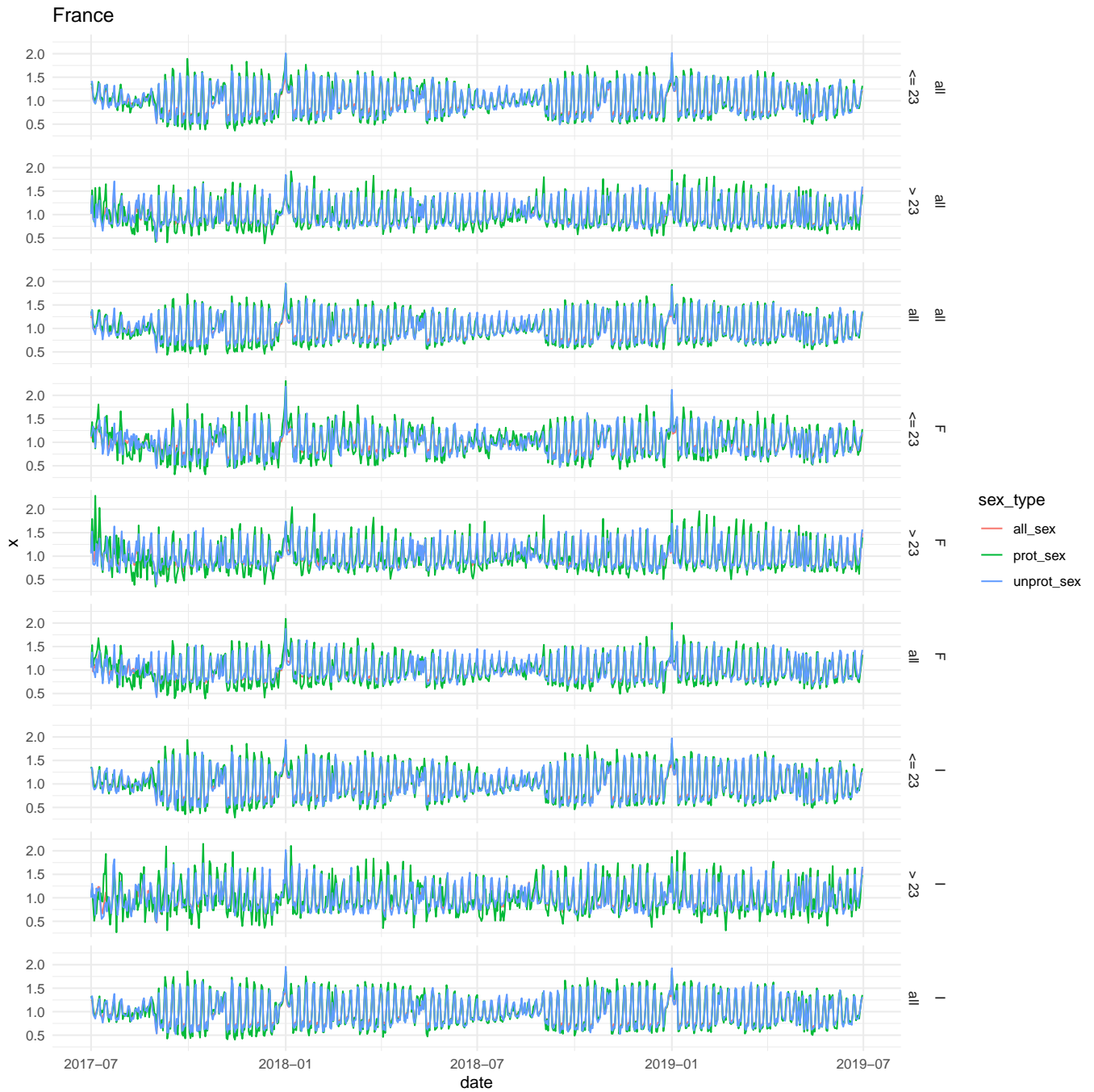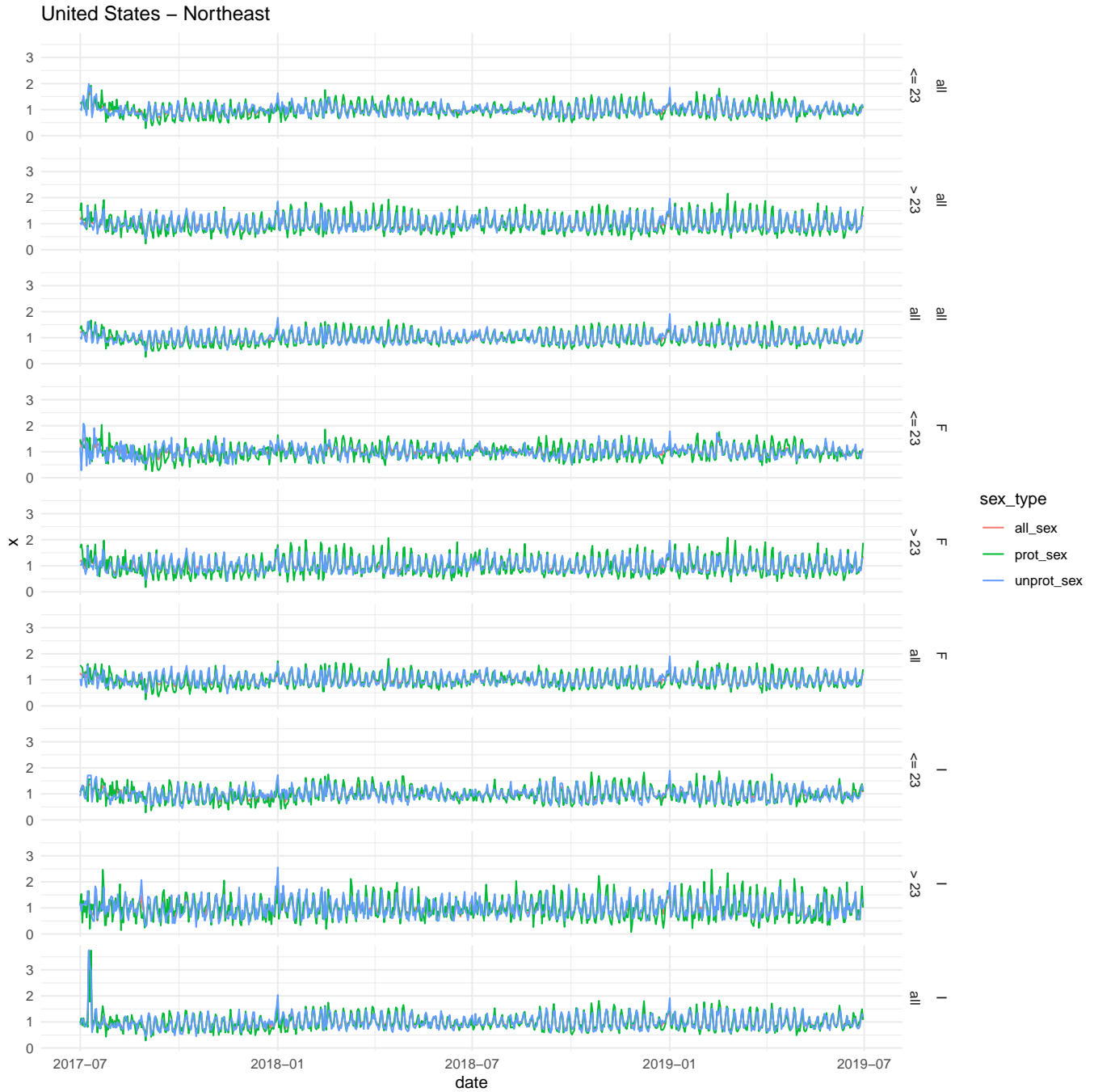
Figure 5.12: Comparing sexual activity by sex type for each location and user group.

5.3.3 Sexual activity model (weekdays-seasons-holidays)

In figure 5.5, one can observe a strong weekly trend, a mild seasonal trend (e.g. stronger in France) and peaks on holidays.

Consequently, sexual activity is here modelled as a combination of these trends. A generalized linear model (glm) is used to model and predict the relative sexual behavior from the two following categorical input variables:

- **Weekly-monthly trend**: because the seasonal trend is weak and seems to primarily affect the amplitude of the weekly trend, the weekly and seasonal trends are modelled as an interaction between the weekdays and the month. The input corresponding to this trend is a categorical variable with 7x12 levels for the combination of each weekday (Mon-Sun) with each month (Jan - Dec). In addition to these 7x12 levels, a "reference day" (ref_day) is added as the reference (first) level of this categorical variable. A synthetic

datapoint, in which the relative sexual frequency (output) is set to 1, is added to the training data ensuring that the model intercept value is 1, the average sexual frequency.

- **Holiday peaks**: the effects of holidays on the sexual frequency is modelled as a peak response for each holiday. The input variable is also a categorical variable in which the reference (and most common) level is normal_day, i.e. days in which there is no holiday or celebration. The other levels of this categorical variable are the holiday/celebration names (e.g. Labor day or National day). Additionally, each holiday is padded by 3 days so that we can account for the effect of holidays on the surrounding days (e.g. Christmas +1,2,3). These padding days are distinct levels of the holiday variable.

Finally, for some holidays, the "context", i.e. the day of the week at which they happened mattered, while for others, it didn't make a difference. The main consequence is that some holidays, such as Valentine's day, impact the sexual frequency *additively* to the weekday variation, while other holidays, such as New Year, lead to an increased absolute level in sexual frequency that is the always the same, independent of the weekday. The reason for this is that, at New Year for example, most people are already on a lighter schedule at their work/school around the New Year and the exact day the New Year happens does not matter. For other such holidays, it does not matter because the weekday is always the same (e.g. Thanksgiving is always on Thursday).

To account for these differences in holiday, we replaced the value of the weekday_month variable by ref_day when the holiday impact on sexual frequency is not additive to the weekday variation.

```r
clue_sex_agg = clue_sex_agg %>%
  mutate(cat = interaction(country_area, BC, age_cat, sex_type))


# for each category of users
sex_models =
  purrr::map(.x = unique(clue_sex_agg$cat), .f = function(category){
  # cat(category %>% as.character(),"\n")
  # retrieve and augment the data
  this_cat_sex_data = clue_sex_agg %>% filter(cat == category)
  this_cat_sex_data = augment_with_weekdays_months_and_holidays(this_cat_sex_data, verbose = FALSE) # see Scripts/00_functions.R
  # the above function adds

  # add a fictive reference day to ensure the intercept = 1
  # and thus that the coefficients of the models are comparable between categories
  ref_day = this_cat_sex_data[1,] %>%
    mutate(
      weekday_month = factor("ref_day", levels = levels(this_cat_sex_data$weekday_month)),
      weekday_month_x = factor("ref_day", levels = levels(this_cat_sex_data$weekday_month_x)),
      holiday_ID  = factor("normal day", levels = levels(this_cat_sex_data$holiday_ID)),
      x = 1)
  this_cat_sex_data = bind_rows(ref_day, this_cat_sex_data)

  # fit the model without contextual holidays
  formula = "x ~ weekday_month + holiday_ID"
  glm_sex_behavior_1 = glm(data = this_cat_sex_data, formula = formula, family = "gaussian")

  # fit the model with contextual holidays (performs better, see below)
  formula = "x ~ weekday_month_x + holiday_ID"
  glm_sex_behavior = glm(data = this_cat_sex_data,
              formula = formula,
              family = "gaussian")

  glm_sex_behavior = reduce_storage_size_of_glm_model(model = glm_sex_behavior)

  # results
  res = list(country_area = unique(this_cat_sex_data$country_area),
        BC = unique(this_cat_sex_data$BC),
        age_cat = unique(this_cat_sex_data$age_cat),
        sex_type = unique(this_cat_sex_data$sex_type),
        model = glm_sex_behavior,
```

```
        intercept = glm_sex_behavior$coefficients[1],
        ssr = sum(glm_sex_behavior$residuals^2),
        ssr_not_contextual = sum(glm_sex_behavior_1$residuals^2))

  res
 }
)


# to retrieve the models based on the values of the list elements, we can use:
# sapply(sex_models, "[[","country_area")
# sapply(sex_models, "[[","BC")
# sapply(sex_models, "[[","sex_type")

sex_models_df = data.frame(
  country_area = sapply(sex_models, "[[","country_area"),
  BC = sapply(sex_models, "[[","BC"),
  age_cat = sapply(sex_models, "[[","age_cat"),
  sex_type = sapply(sex_models, "[[","sex_type"),
  SSR = sapply(sex_models, "[[","ssr"),
  SSR_1 = sapply(sex_models, "[[","ssr_not_contextual"),
  intercept = sapply(sex_models, "[[","intercept")
)

sex_models_df = sex_models_df %>%
  left_join(.,dict$country_area, by = "country_area") %>%
  left_join(.,dict$BC, by = "BC")

for(ca in unique(sex_models_df$country_area)){
  j = which(sex_models_df$country_area == ca)
  sex_models_this_location = list()
  for(i in 1:length(j)){sex_models_this_location[[i]] = sex_models[[j[i]]]}
  save(sex_models_this_location, file = str_c(IO$out_Rdata,"sex_models_",ca,".Rdata"))
}

save(sex_models_df, file = str_c(IO$out_Rdata,"sex_models_df.Rdata"))
```

First, looking at the values of the residuals on the training set (no test set was used here because we only had two years of data and preferred to use both years rather than using one year as the training set and the other year as the test/validation set), we observe large differences between categories, despite the fact that each time-series has the same number of time-point (i.e. 2 years of data = 730 data-point).

```
ggplot(sex_models_df, aes(x = BC, y = SSR, fill = BC_col))+
  geom_bar(stat = "identity", aes(y = SSR_1), fill = "red", alpha = 0.5)+
  geom_bar(stat = "identity")+
  scale_fill_identity()+
  facet_grid(country_area ~ age_cat + sex_type)+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
```
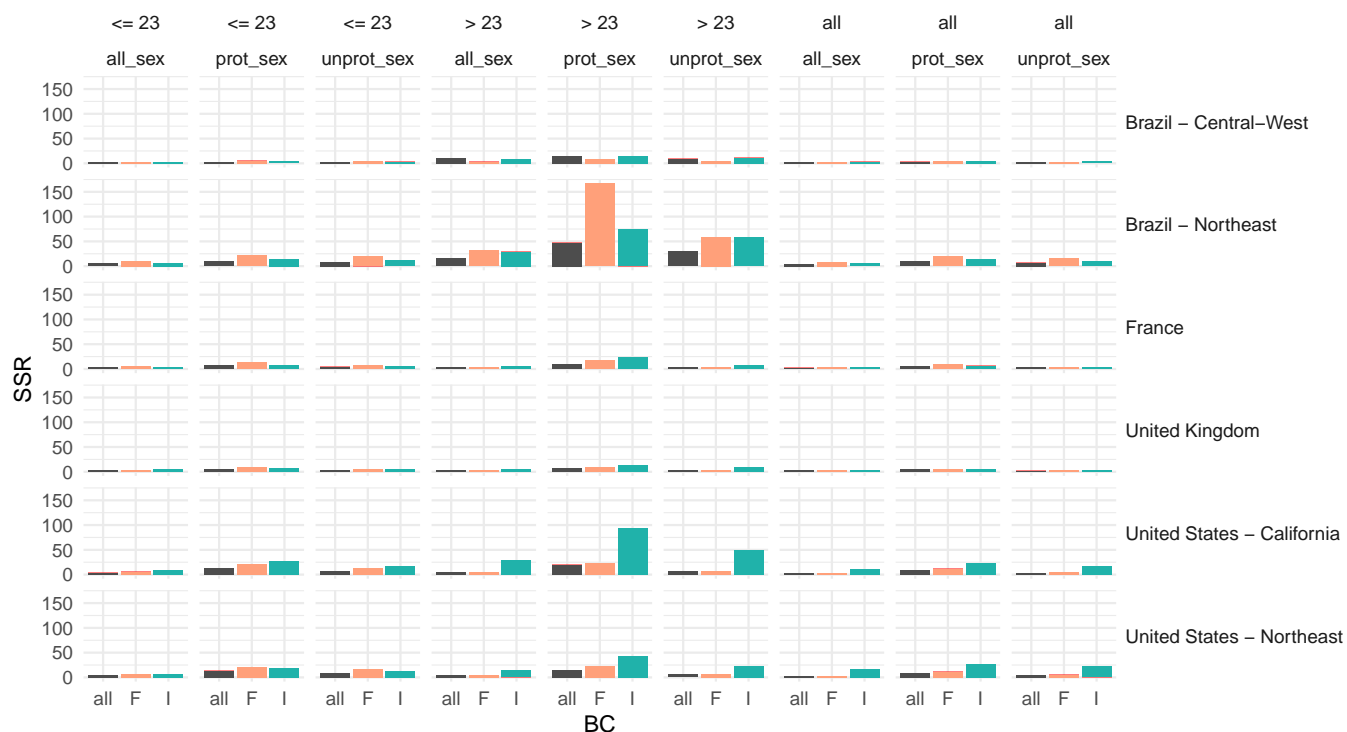
Figure 5.13: Sex models residuals on the training data

These differences can mostly be explained by the number of users that contributed to the aggregated time-series.

```
df_agg = clue_sex_agg %>%
  group_by(cat) %>%
  summarize(min_n_users = min(n_users),
        max_n_users = max(n_users),
        median_n_users = median(n_users))
```

`summarise()` ungrouping output (override with `.groups` argument)

```
tmp = full_join(sex_models_df %>% mutate(cat = interaction(country_area, BC, age_cat, sex_type)), df_agg, by = "cat")

ggplot(tmp, aes(x = median_n_users, y = SSR, col = country_area))+
  geom_point()+ scale_x_log10()+
  xlab("median # of users (log scale)")
```
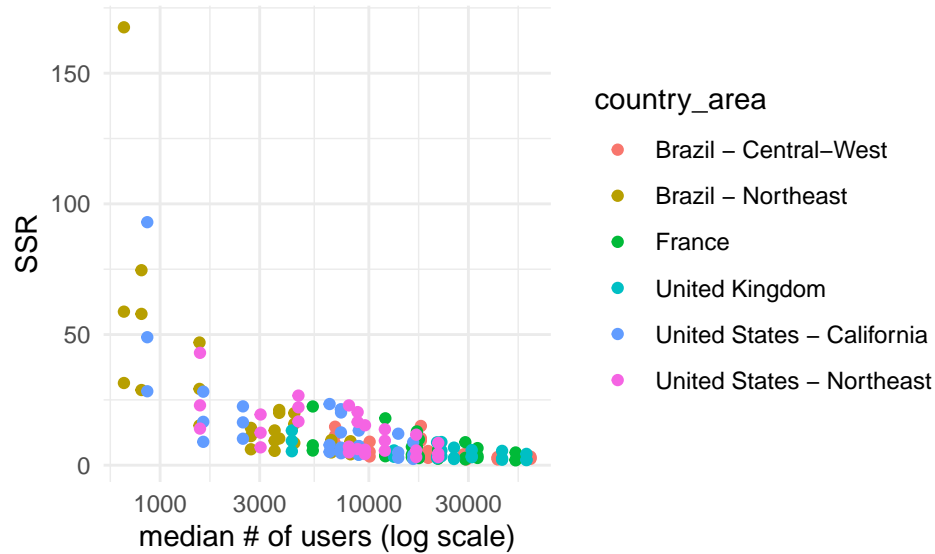
Figure 5.14: Residuals as a function of the median number of users that contributed to the aggregated time-series

The more users that were part of a category, the more the sexual behavior is accurately modeled as a weekly-seasonal trend + a holiday response. Categories with less users to build the time-series are more sensitive to individual variations in reported sexual intercourse.

```
ggplot(sex_models_df, aes(x = BC, y = (SSR - SSR_1)/SSR*100, fill = BC_col))+
  geom_hline(yintercept = 0)+
  geom_bar(stat = "identity")+
  scale_fill_identity()+
  ylab("% change in SSR (SSR_context - SSR_additive)")+
  facet_grid(country_area ~ age_cat + sex_type)+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
```


Figure 5.15: Percent change in residuals from sex models with contextual holidays vs strictly additive model

In figure 5.15 the residuals of the "contextual holiday model" are compared with the residuals of a strictly additive model (see explanations earlier). In general, the "contextual holiday model" performs better and is used for the sexual behavior predictions from now on.

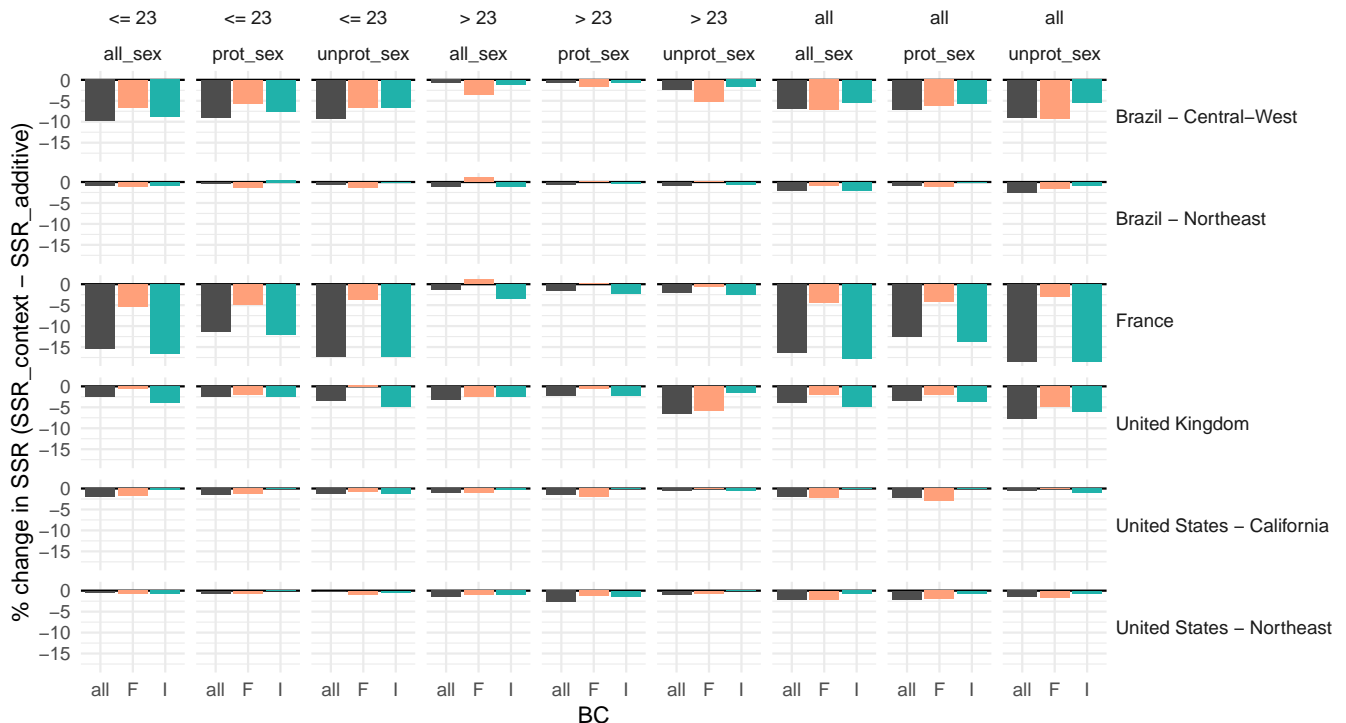**5.3.3.1 Visualization of the model coefficient of the residuals**   We show here the coefficient of the glm model and the fitted time-series as well as the residuals for the 6 countries/areas of interest. For each country/area, we display these coefficients and residuals for the timeseries of users from any age group (age_cat = "all"), any birth-control type (BC = "all") and for the sum of all sexual intercourse (sex_type = "all").

The models are saved on the github repo so these plots can be reproduced for any other category.

```r
ok = foreach(ca = unique(sex_models_df$country_area)) %do% {

  # retrieving the model for this country and only plotting for BC all and sex_type all
  j = which(
    (sex_models_df$country_area == ca) &
      (sex_models_df$BC == "all") &
      (sex_models_df$sex_type == "all_sex") &
      (sex_models_df$age_cat == "all") )
  glm_sex_behavior = sex_models[[j]]$model

  # Visualization of the coefficient
  g_coef = ggplot_sex_activity_glm_coefficient(model = glm_sex_behavior, show_weekly_patterns = TRUE)
  g_coef = g_coef +
    ggtitle(str_c(ca," - BC: all - age_cat: all - sex type: all_sex"))

  # Visualization of the training + fitted time-series  # TO DO: plot the two years above each other.
  g_residuals = ggplot_sex_activity_data_fitted_and_residuals(model = glm_sex_behavior)

  g = suppressWarnings(cowplot::plot_grid(plotlist = list(g_coef, g_residuals), ncol = 1, heights = c(1.2,1)))
  print(g)
  return()
}
```

Figure 5.16: (Top) Coefficients of the generalized linear models used to predict relative sexual behavior changes. (Bottom) Actual vs Fitted and Residuals (squared difference between the actual and fitted values) over the two years of data used as training set.
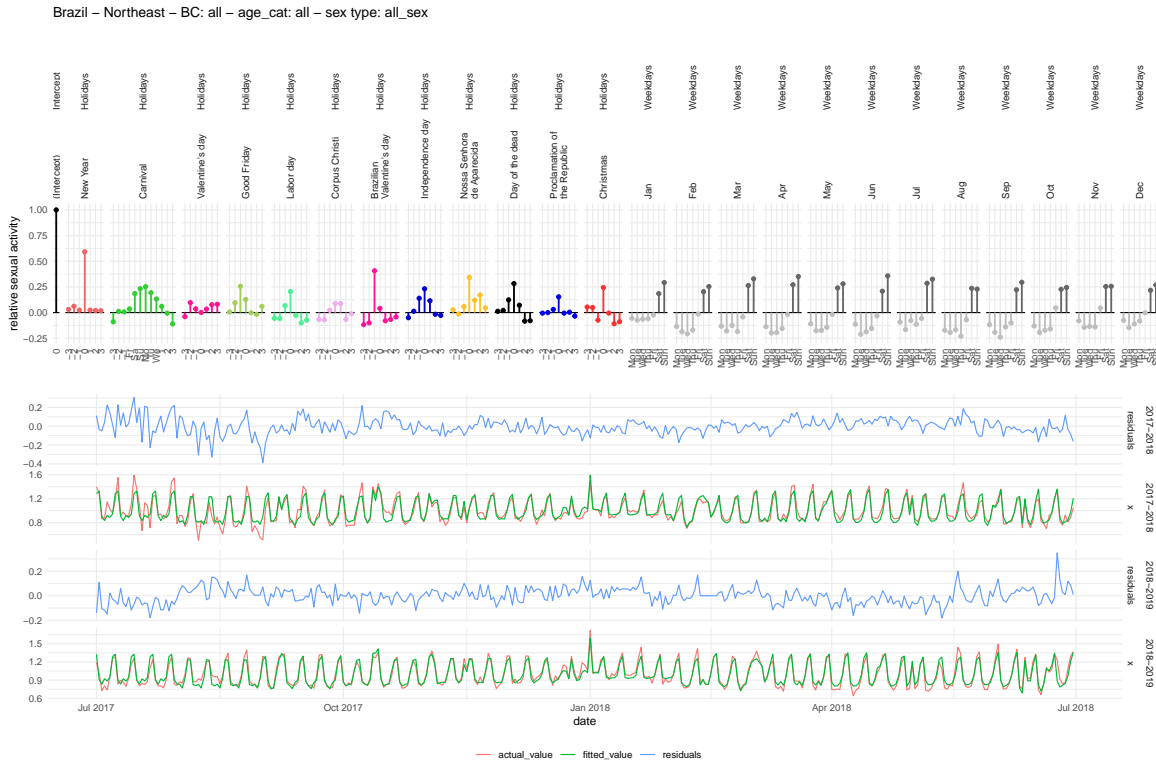
Figure 5.17: (Top) Coefficients of the generalized linear models used to predict relative sexual behavior changes. (Bottom) Actual vs Fitted and Residuals (squared difference between the actual and fitted values) over the two years of data used as training set.
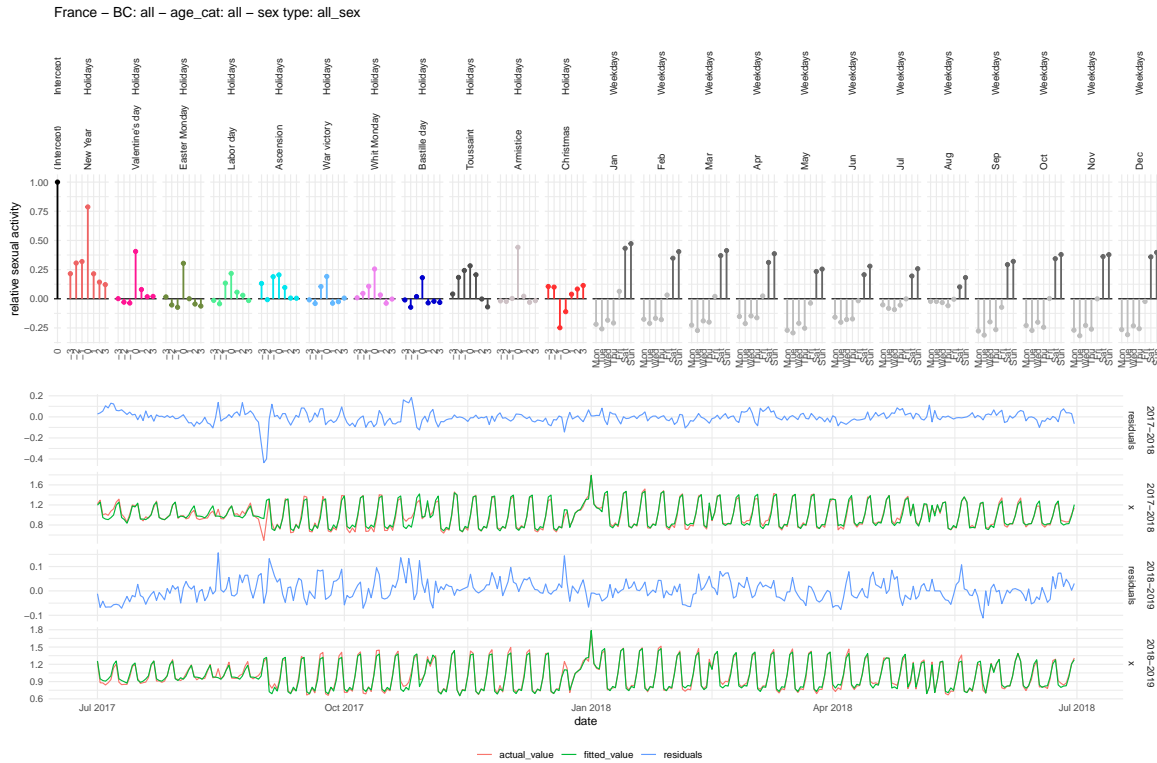
Figure 5.18: (Top) Coefficients of the generalized linear models used to predict relative sexual behavior changes. (Bottom) Actual vs Fitted and Residuals (squared difference between the actual and fitted values) over the two years of data used as training set.
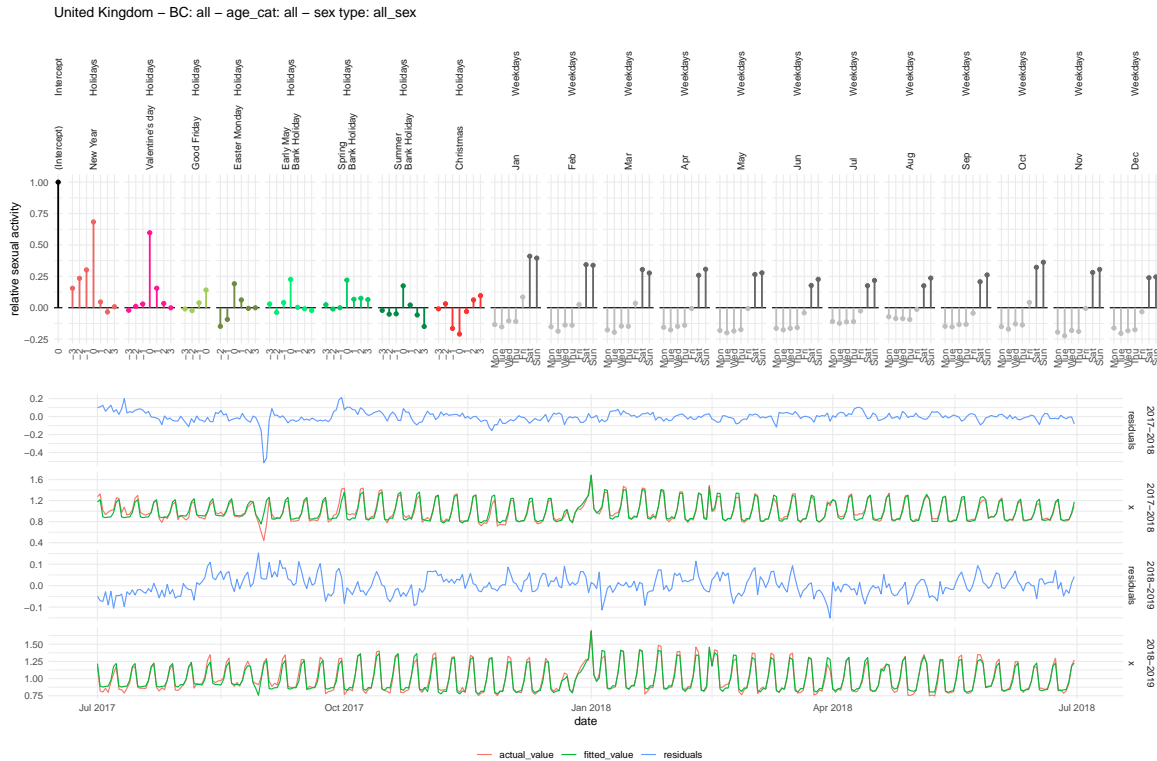
Figure 5.19: (Top) Coefficients of the generalized linear models used to predict relative sexual behavior changes. (Bottom) Actual vs Fitted and Residuals (squared difference between the actual and fitted values) over the two years of data used as training set.

Figure 5.20: (Top) Coefficients of the generalized linear models used to predict relative sexual behavior changes. (Bottom) Actual vs Fitted and Residuals (squared difference between the actual and fitted values) over the two years of data used as training set.
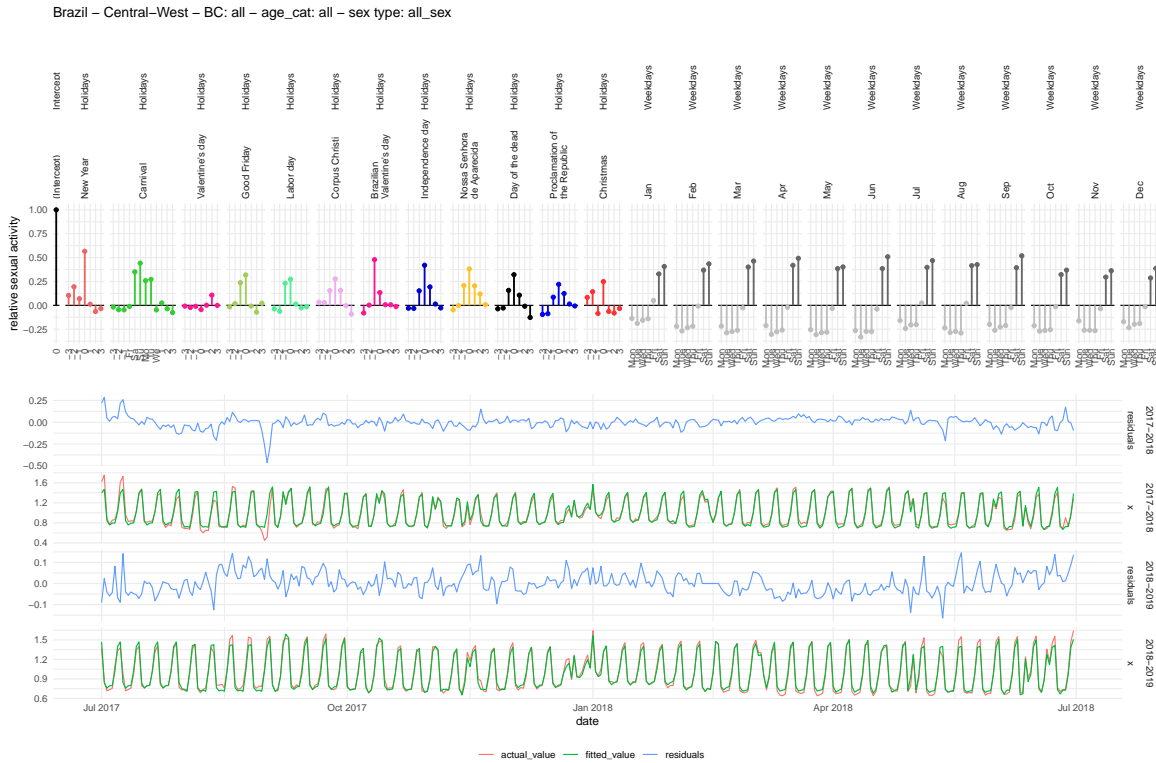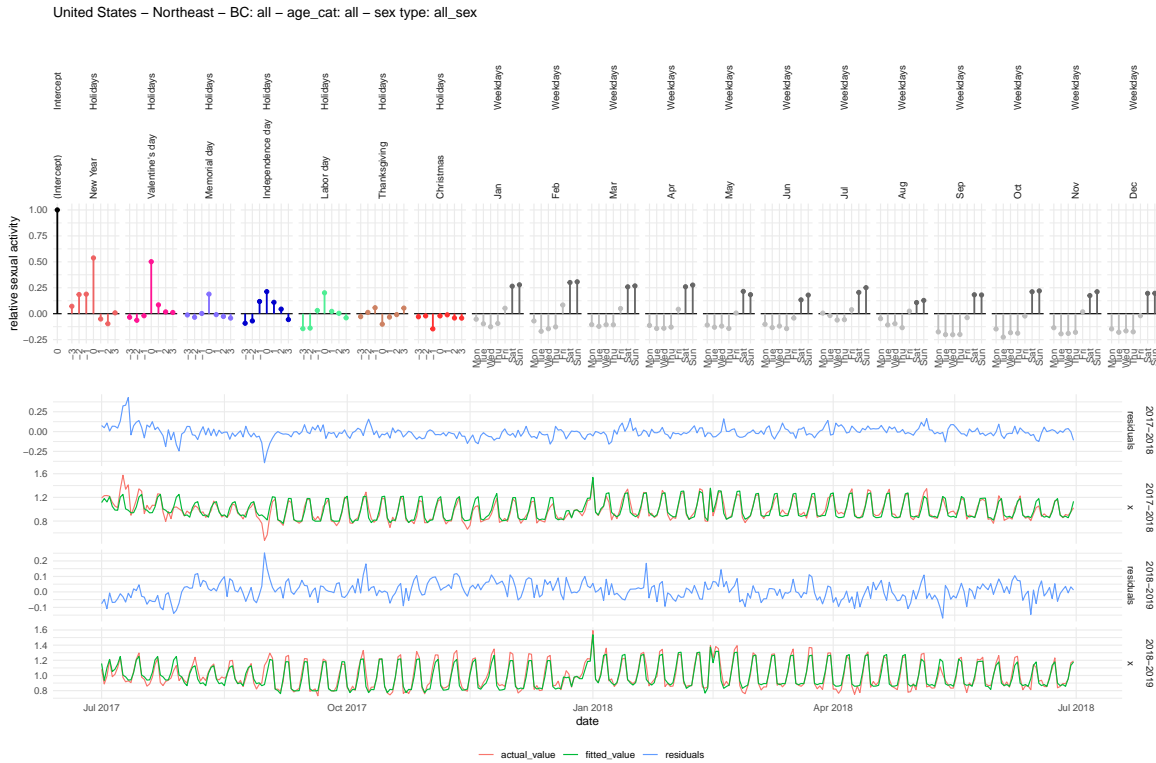

Figure 5.21: (Top) Coefficients of the generalized linear models used to predict relative sexual behavior changes. (Bottom) Actual vs Fitted and Residuals (squared difference between the actual and fitted values) over the two years of data used as training set.
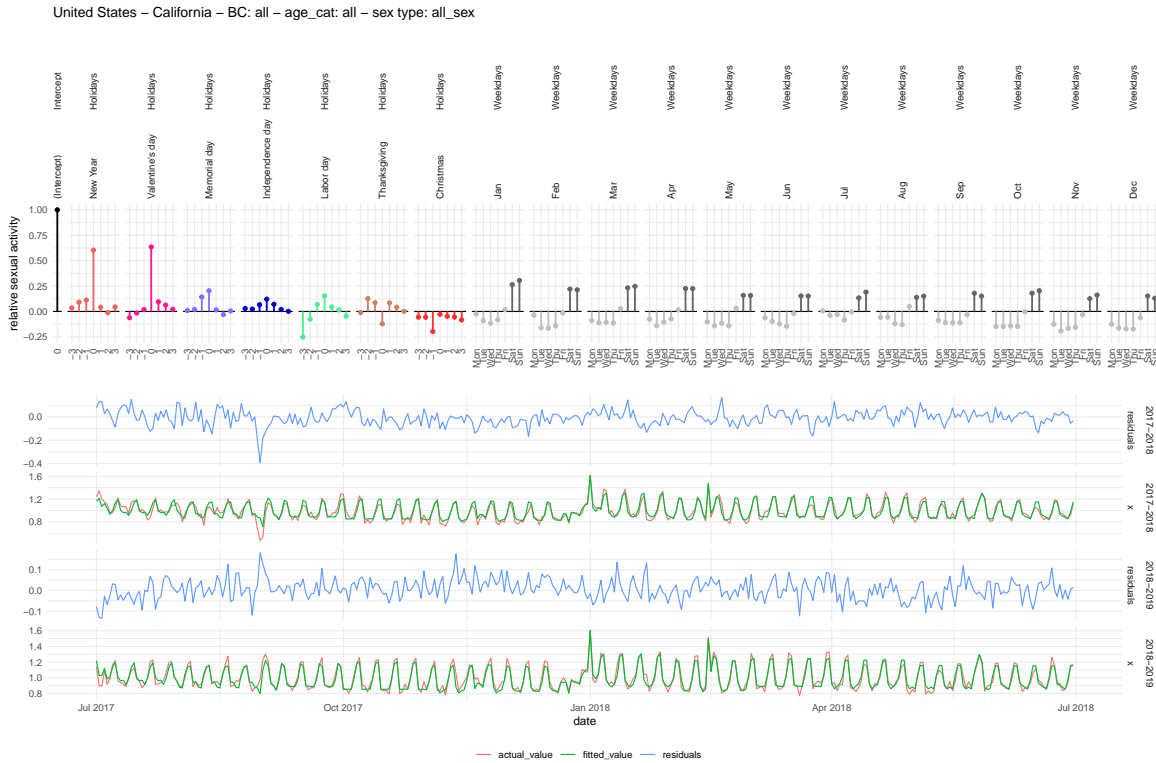
5.4 Official birth records

```
load(str_c(IO$out_Rdata,"birth_data.Rdata"), verbose = TRUE)
```

```
## Loading objects:
##   birth
```

```
official_birth_records = birth %>% dplyr::filter(country_area %in% clue_sex_agg$country_area)
rm(birth)
```

```
official_birth_records = official_birth_records %>%
  mutate(country_area = factor(country_area, levels = dict$country_area$country_area))
```

```
official_birth_records = official_birth_records %>%
  left_join(., dict$country_area %>% dplyr::select(country_area, country_area_col), by = "country_area")
```

```
ggplot(official_birth_records, aes(x = date, y = births, col = country_area_col))+
  geom_line()+
  scale_color_identity()+
  facet_grid(country_area ~ ., scale = "free_y")+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
```


Figure 5.22: Official birth records - raw monthly data

5.4.1 Correcting for the number of days in each months

```
official_birth_records = official_birth_records %>%
  mutate(births_original_numbers = births,
         year_month = year + (month_num-1)/12)
```

```
# preparing a data.frame that has the number of day in each month
date_seq = seq(min(official_birth_records$date), max(official_birth_records$date) + days(365), by = 1)
date_seq = data.frame(date = date_seq, year = year(date_seq), month_num = month(date_seq))
date_seq = date_seq %>% mutate(year_month = year + (month_num-1)/12)
n_days_per_month = date_seq %>% group_by(year_month) %>% dplyr::summarise(n_days = n(), .groups = "drop")
```

```
# joining with births table
official_birth_records = left_join(official_birth_records, n_days_per_month, by = "year_month")

# correcting
official_birth_records = official_birth_records %>%
  mutate(births = births_original_numbers/n_days*30)
```

```
# visualization
ggplot(official_birth_records, aes(x = date, y = births, col = country_area_col))+
  # uncorrected births
  #geom_point(aes(y = births_original_numbers), size = 0.5,  col = "gray")+
  geom_line(aes(y = births_original_numbers), col = "gray")+
  #corrected births
  #geom_point(size = 0.5)+
  geom_line()+
  # settings
  scale_color_identity()+
  facet_grid(country_area ~., scale = "free")+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
```
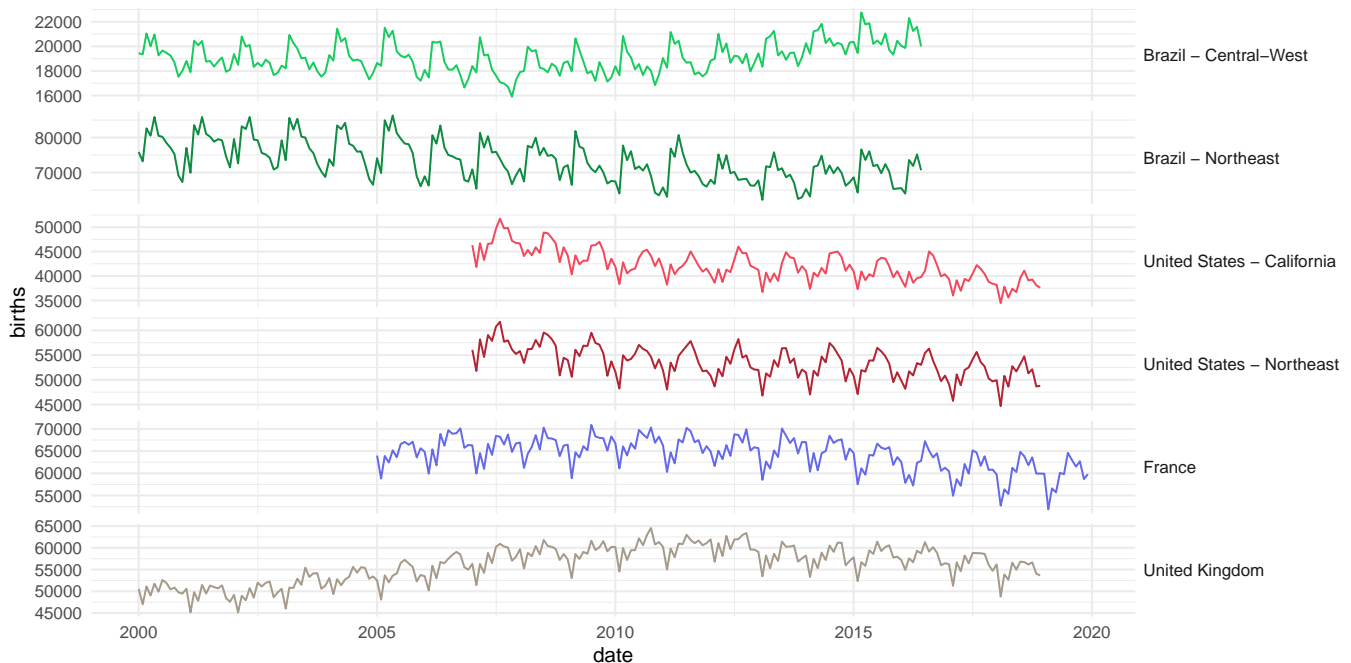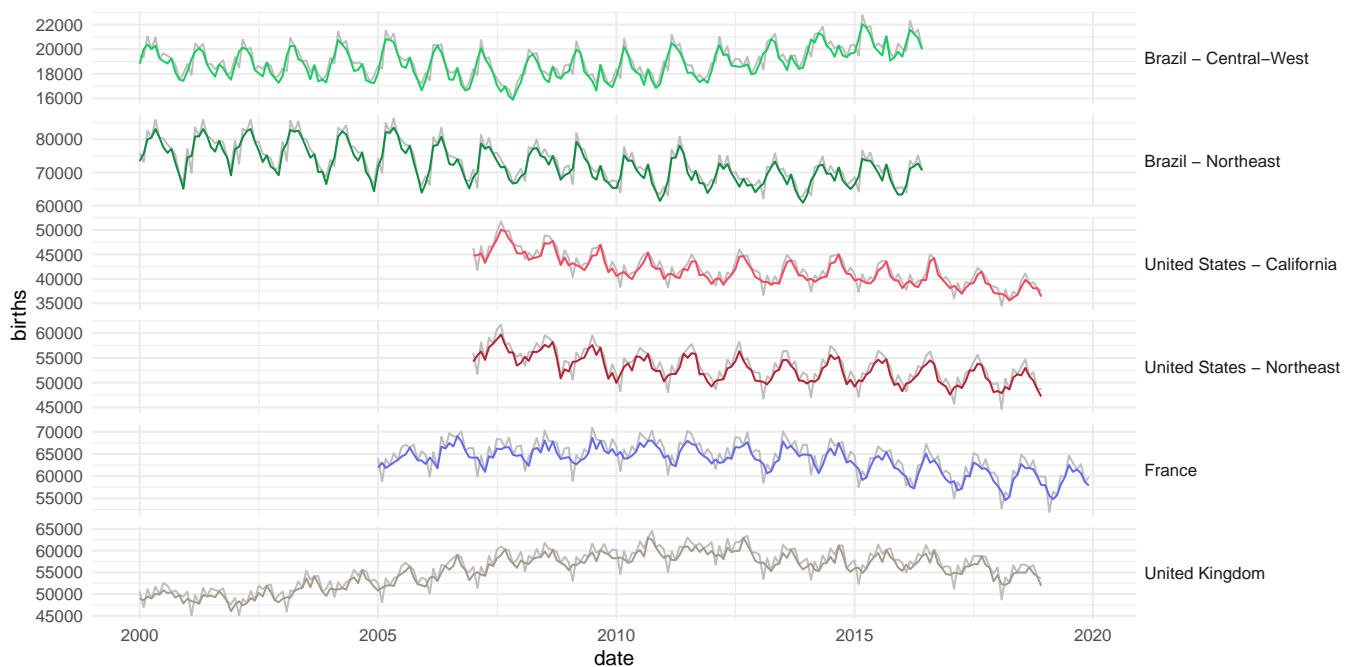

Figure 5.23: Official birth records: corrected for the number of days in each month (colored lines) and raw (light gray lines) monthly data

The gray lines under the colored lines are the actual (non-corrected) birth records. We can see that February is often associated with a local minima, which disappear when we correct for the month duration.

```
write_feather(official_birth_records, path = str_c(IO$out_Rdata,"official_birth_records.feather"))
```

5.4.2   Average daily births

```
average_daily_births_df = foreach(ca = unique(official_birth_records$country_area), .combine = bind_rows) %do%{
  this_ca_births = official_birth_records %>% filter(country_area == ca)

  date_seq = seq(min(this_ca_births$date), max(this_ca_births$date) + months(1), by = 1)
  date_seq = data.frame(date = date_seq, year = year(date_seq), month_num = month(date_seq))
```

```r
date_seq = date_seq %>%
  mutate(
    country_area = ca,
    year_month = year + (month_num-1)/12
  )

this_ca_ave_daily_births = date_seq %>%
  full_join(.,
        this_ca_births %>% select(country_area, date, births),
        by = c("country_area","date"))

this_ca_ave_daily_births = this_ca_ave_daily_births %>%
  mutate(m_ave_daily_births = births/30)
this_ca_ave_daily_births$m_ave_daily_births[this_ca_ave_daily_births$date == max(date_seq$date)] =
  this_ca_ave_daily_births$m_ave_daily_births[this_ca_ave_daily_births$date == max(this_ca_births$date)]

this_ca_ave_daily_births = this_ca_ave_daily_births %>%
  mutate(daily_births = na.spline(m_ave_daily_births, method = "natural"),
       t = row_number())
this_ca_ave_daily_births$ave_daily_births = predict(loess(daily_births ~ t, data = this_ca_ave_daily_births, span = 0.5))

g = ggplot(this_ca_ave_daily_births, aes(x = date))+
  geom_point(aes(y = m_ave_daily_births), col = "black")+
  geom_line(aes(y = daily_births), col = "gray")+
  geom_line(aes(y = ave_daily_births), col = "blue")+
  ggtitle(ca)
print(g)

res = this_ca_ave_daily_births %>% select(country_area, date, ave_daily_births)
return(res)
}
```

Warning: Removed 6712 rows containing missing values (geom_point).

Figure 5.24: Average daily births (birth long-term trend).

Warning: Removed 5298 rows containing missing values (geom_point).

Figure 5.25: Average daily births (birth long-term trend).

Warning: Removed 4239 rows containing missing values (geom_point).

Figure 5.26: Average daily births (birth long-term trend).

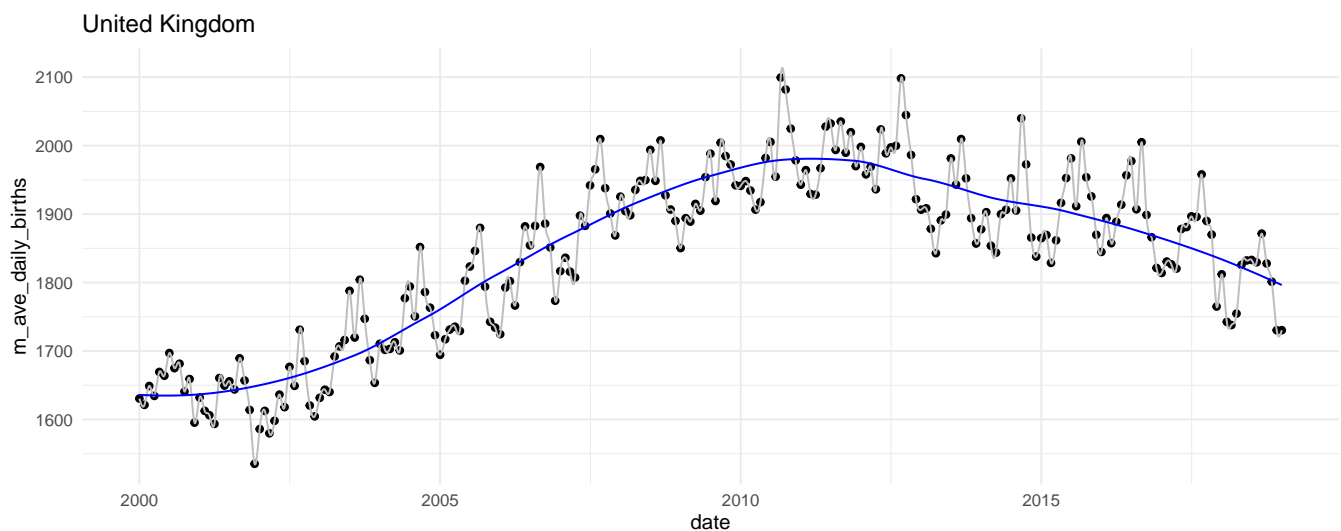## Warning: Removed 4239 rows containing missing values (geom_point).

Figure 5.27: Average daily births (birth long-term trend).

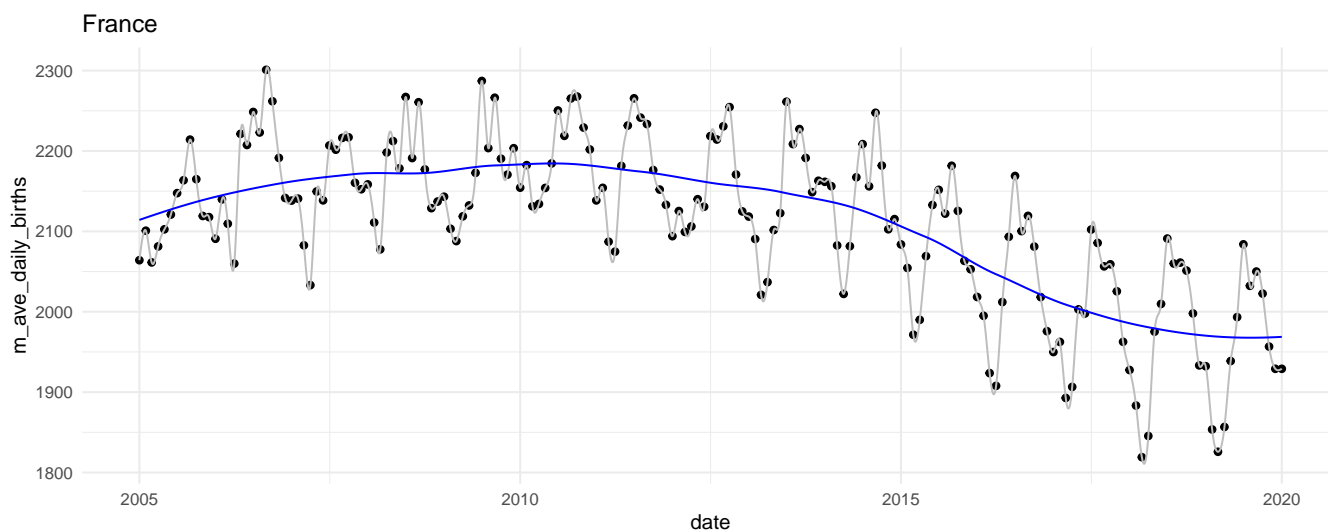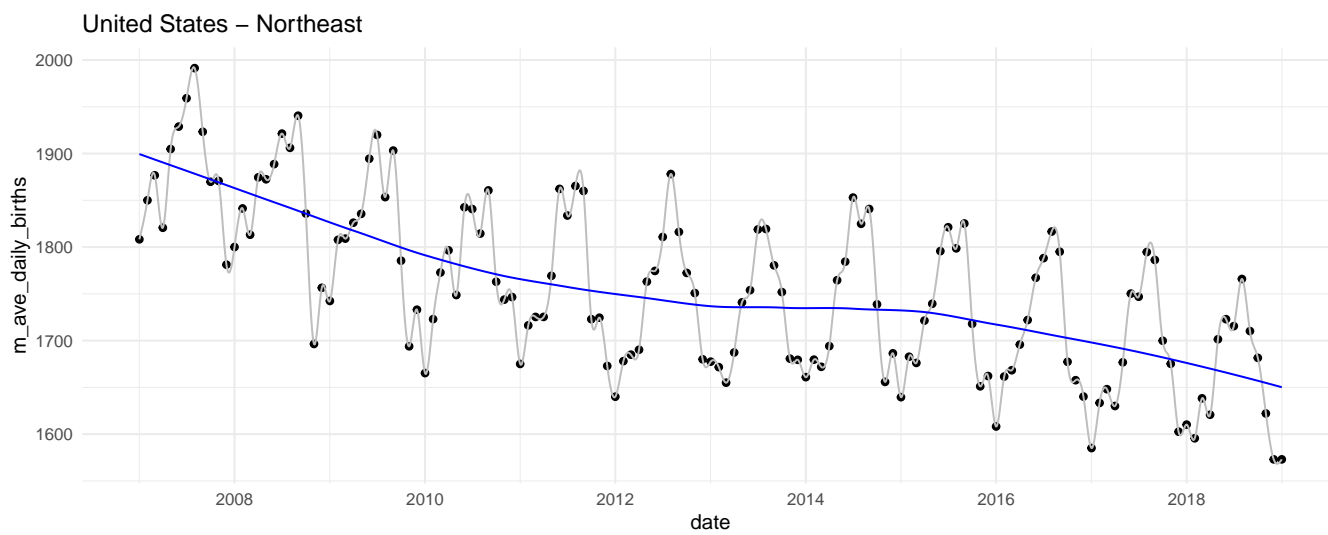## Warning: Removed 5828 rows containing missing values (geom_point).
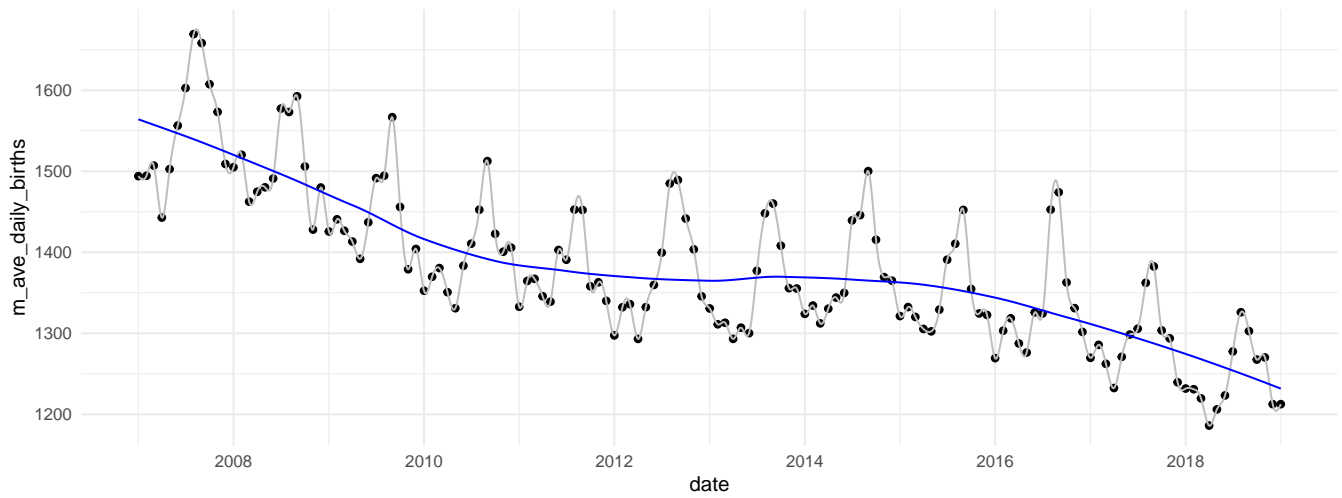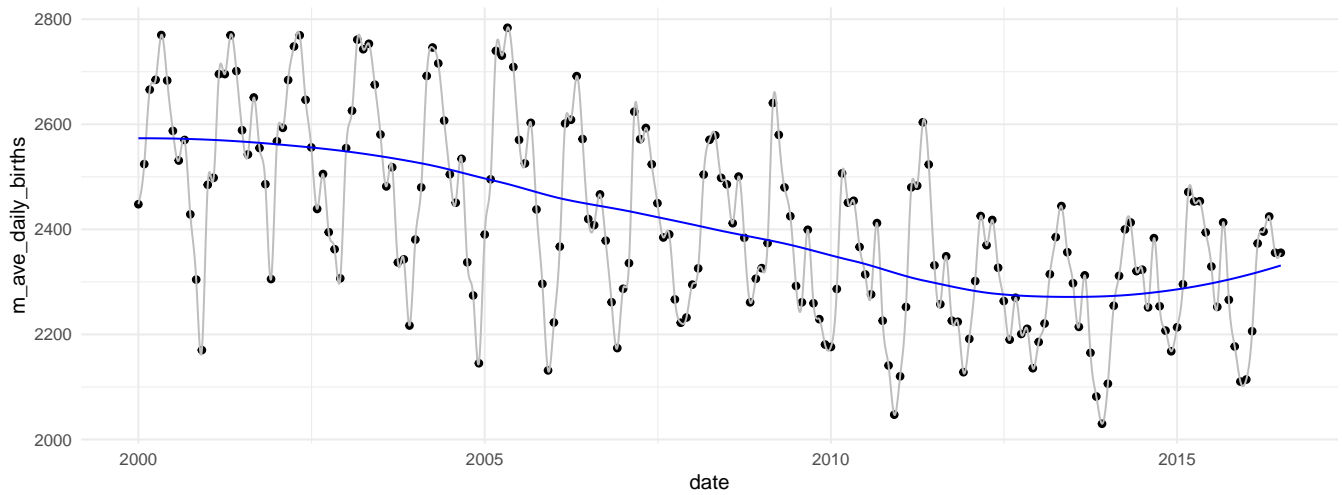

Figure 5.28: Average daily births (birth long-term trend).

Warning: Removed 5828 rows containing missing values (geom_point).
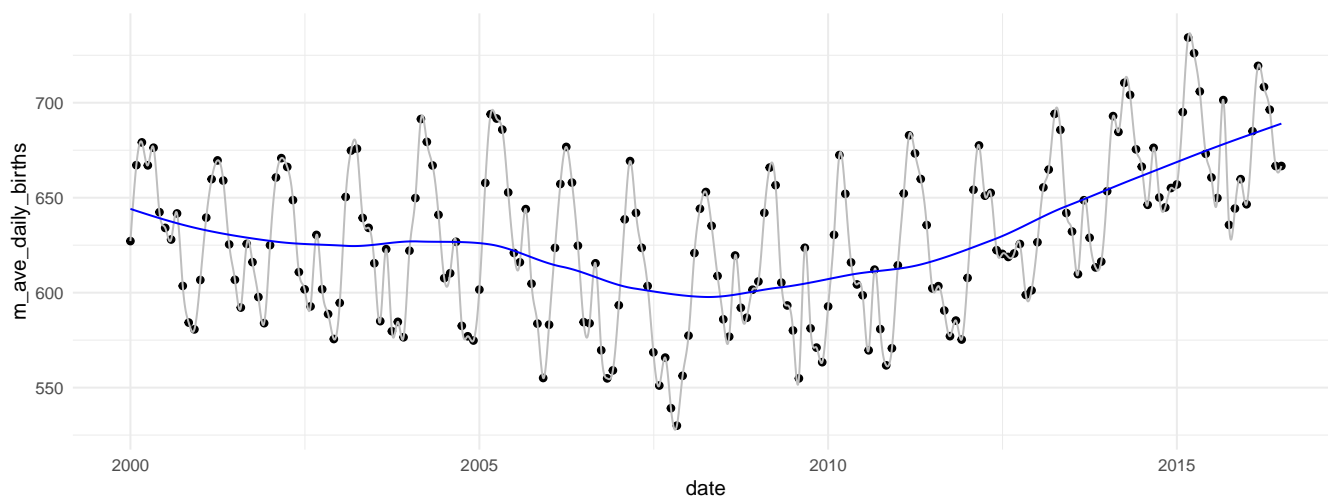
Figure 5.29: Average daily births (birth long-term trend).

5.5 Model parameters optimization

In this section, the fertility parameters α (relative amplitude) and T (peak fertility time) are optimized for model B and C and for each category independently.

5.5.1 Optimization functions

We first define the functions needed to optimize the parameters and simulate the daily and monthly births.

```r
optimize_birth_param = function(varying_par = c(),
                    varying_par_prior = list(),
                    fixed_par = list(),
                    n_init = 10,
                    sex_df,
                    ave_daily_birth_df,
                    actual_monthly_birth_df,
                    verbose = FALSE){

 par_names = c("alpha","Tp", "beta","G","Gsd")
 # alpha is the fertility amplitude,
 # Tp is the peak time of fertility
 # beta is the amplitude (i.e the difference between the highest value and the mean, divided by the mean) of the sex curve
 # G is the average gestation period
 # Gsd is the standard deviation of the gestation period

 if((length(varying_par) == 0) & (length(fixed_par) == 0)){
   warning("varying_par and fixed_par are empty. All parameters will be optimized.")
   varying_par = par_names
 }

 if(! all(varying_par %in% par_names)) stop(paste("Parameters must be any of :",par_names))
 if(! all(names(fixed_par) %in% par_names)) stop(paste("Parameters must be any of :",par_names))

 if(length(varying_par) == 0){
   if(verbose) cat("nothing to optimize\n")
   # we don't need to optimize anything; we just compute the residuals
   par = c(0,0)

   SSR = residuals_simulated_vs_actual_monthly_birth(par = c(),
                             sex_df = sex_df,
                             ave_daily_birth_df= ave_daily_birth_df,
                             actual_monthly_birth_df = actual_monthly_birth_df,
                             alpha = 0, Tp = 0, beta = fixed_par[["beta"]], G = fixed_par[["G"]], Gsd = fixed_par[["Gsd"]])

   optimized_par = data.frame(SSR = SSR,
                   alpha = fixed_par[["alpha"]], Tp = fixed_par[["Tp"]],
                   beta = fixed_par[["beta"]],
                   G = fixed_par[["G"]],  Gsd = fixed_par[["Gsd"]],
                   stringsAsFactors = FALSE)
 }else{
   if(verbose) cat("Optimizing ",varying_par,"\n")

   # we run n_init optimization with different initial conditions
   res = foreach(i = 1:n_init, .combine = bind_rows)%do%{
     if(verbose) cat("\t",i,"\n")


     # initial values of the parameters
```

```r
  par_init =
    c(runif(1, min = 0.01, max = 0.1), # alpha
      i/n_init, # Tp
      runif(1,min = 0.01, max = 1), # beta
      G0 + sample(-Gsd0:Gsd0,1), # G
      runif(1, min =  Gsd0/2, max = 2*Gsd0) # Gsd
    )
  lower = c(0, 0, 0, G0-Gsd0, Gsd0/2);
  upper = c(1, 1, 2, G0+Gsd0, Gsd0*2);

  if(length(varying_par_prior) > 0){
    for(par_name in par_names){
      if(par_name %in% names(varying_par_prior)){
        par_init[which(par_name == par_names)] = sample(varying_par_prior[[par_name]],1)
      }
    }
  }

  ix = c()
  for(par_name in par_names){
    if(par_name %in% varying_par){
      ix = c(ix, which(par_names == par_name))
      eval(parse(text = paste0(par_name," = NULL")))
    }else{
      if(! par_name %in% names(fixed_par))
        stop(paste("If",par_name,"is not varying, its value needs to be specified in 'fixed_par"))
      eval(parse(text = paste0(par_name," = fixed_par[['",par_name,"']]")))
    }
  }
  par_init = par_init[ix]
  lower = lower[ix]
  upper = upper[ix]

  if(verbose) cat(par_init, "\n")
  # optimization
  optimized_par = optim(par = par_init, lower = lower, upper = upper,
                    fn = residuals_simulated_vs_actual_monthly_birth,
                    sex_df = sex_df,
                    ave_daily_birth_df = ave_daily_birth_df,
                    actual_monthly_birth_df = actual_monthly_birth_df,
                    alpha = alpha, Tp = Tp, beta = beta, G = G, Gsd = Gsd,
                    method = "L-BFGS-B")

  opt_alpha = ifelse("alpha" %in% varying_par, optimized_par$par[which(ix == 1)],alpha)
  opt_Tp = ifelse("Tp" %in% varying_par, optimized_par$par[which(ix == 2)], Tp)
  opt_beta = ifelse("beta" %in% varying_par, optimized_par$par[which(ix == 3)], beta)
  opt_G = ifelse("G" %in% varying_par, optimized_par$par[which(ix == 4)],G)
  opt_Gsd = ifelse("Gsd" %in% varying_par, optimized_par$par[which(ix == 5)], Gsd)
  df = data.frame(value = optimized_par$value,
            alpha = opt_alpha, Tp = opt_Tp,
            beta = opt_beta,
            G = opt_G, Gsd = opt_Gsd)

  return(df)
}
opt = which.min(res$value)
optimized_par = data.frame(SSR = res$value[opt],
                  alpha = res$alpha[opt], Tp = res$Tp[opt],
```

```r
                    beta = res$beta[opt], G = res$G[opt], Gsd = res$Gsd[opt],
                    stringsAsFactors = FALSE)
  }

  return(optimized_par)
}




residuals_simulated_vs_actual_monthly_birth = function(
  par = c(alpha, Tp, beta, G, Gsd),
  sex_df,
  ave_daily_birth_df,
  actual_monthly_birth_df,
  alpha = NULL, Tp = NULL, beta = NULL, G = NULL, Gsd = NULL
){

  if(is.null(alpha)) alpha = par[1]
  if(is.null(Tp)) Tp = par[2]
  if(is.null(beta)) beta = par[3]
  if(is.null(G)) G = par[4]
  if(is.null(Gsd)) Gsd = par[5]

  df = simulated_vs_actual_monthly_birth(alpha = alpha, Tp = Tp, beta = beta, G = G, Gsd = Gsd,
                            sex_df = sex_df,
                            ave_daily_birth_df = ave_daily_birth_df,
                            actual_monthly_birth_df = actual_monthly_birth_df)
  SSR = sum(df$sq_residuals)
  return(SSR)
}




simulated_vs_actual_monthly_birth = function(alpha, Tp, beta , G, Gsd,
                            sex_df,
                            ave_daily_birth_df,
                            actual_monthly_birth_df){

  simulated_daily_births = simulate_daily_births(alpha = alpha, Tp = Tp, beta = beta, G = G, Gsd = Gsd,
                              sex_df = sex_df,
                              ave_daily_birth_df = ave_daily_birth_df )
  simulated_monthly_births = aggregate_monthly_births(simulated_daily_births)
  df = inner_join(simulated_monthly_births %>% select(year_month, births) %>% rename(sim_births = births),
          actual_monthly_birth_df %>%  select(year_month, births),
          by = "year_month")
  df = df %>% mutate(
    residuals = sim_births - births,
    sq_residuals = residuals^2
  )
  # we remove the first and last 6 months of data
  df = df %>% arrange(year_month) %>%
    head(.,-6) %>% tail(.,-6)
  return(df)
}




aggregate_monthly_births = function(daily_births){
```

```r
  df = daily_births %>%
    mutate(year_month = year(date_births) + (month(date_births)-1)/12) %>%
    group_by(year_month) %>%
    summarize(n_days = n(),
          uncorrected_births = sum(births),
          births = uncorrected_births/n_days*30,
          .groups = "drop")
  return(df)
}




simulate_daily_births = function(alpha, Tp, beta, G, Gsd,
                    sex_df,
                    ave_daily_birth_df){

  beta_o = max(sex_df$sex - 1)
  if(beta_o != 0) sex_df = sex_df %>% mutate(sex_o = sex,
                          rel_sex = (sex-1)/max(sex-1),
                          sex = 1 + beta * rel_sex,
                          sex = sex %>% pmax(.,0))

  df = inner_join(sex_df %>% select(date, sex),
          ave_daily_birth_df , by = "date")

  df = df %>%
    mutate(
      year_day = year(date)+(yday(date)-1)/ifelse(leap_year(date),366,365),
      fertility = 1 + alpha*cos( (year_day - Tp) * (2 * pi)),
      conceptions = sex * fertility,
      smoothed_conceptions = smooth_conceptions(conceptions, sd_gestation = Gsd),
      date_conceptions = date,
      date_births = date + days(round(G)),
      births = ave_daily_births * smoothed_conceptions)

  df = df %>%
    select(country_area, date, date_births,
        ave_daily_births, sex, fertility,
        conceptions, smoothed_conceptions,
        births) %>%
    rename(date_conceptions = date)

  return(df)
}




smooth_conceptions = function(conceptions, sd_gestation = 9){
  x = conceptions
  if(all(is.na(x))){stop("conceptions can't be only NAs")}
  N = (sd_gestation*3.5) %>% round()
  gestation_distribution = dnorm(-N:N,mean = 0, sd = sd_gestation)
  xx = c(rev(x[1:N]),x, rev(x[(length(x)-N+1):length(x)]))
  y = stats::filter(x = xx, filter = gestation_distribution)
  y = y[!is.na(y)]
  return(y)
}
```

```
predict_daily_sex_behavior = function(model , date_range, country_area){
  date_seq = seq(min(date_range), max(date_range), by = 1)
  df = data.frame(date = date_seq)
  df = df %>% mutate(country_area = country_area)

  df = augment_with_weekdays_months_and_holidays(df)
  df$sex = predict(object = model, newdata = df, type = "response")
  return(df)
}
```

5.5.2   Parameters optimization

```
official_birth_records = official_birth_records %>%
  arrange(country_area, date) %>%
  group_by(country_area) %>%
  mutate(trend = predict(loess(births ~ year_month, span = 0.6)),
      detrended_births = births / trend,
      mean_detrended_birth = mean(detrended_births),
      centered_detrended_births = detrended_births - mean_detrended_birth)%>%
  ungroup()

fourier_transform = official_birth_records %>%
  group_by(country_area,year) %>%
  mutate(n_months = n()) %>%
  filter(n_months == 12) %>%
  ungroup() %>%
  group_by(country_area) %>%
  mutate(ft = fft(centered_detrended_births))

model_B_param_rough_estimates = fourier_transform %>%
  group_by(country_area) %>%
  mutate(
    i = row_number(),
    n_tp = n(),
    f_index = (n_tp %/% 12) + 1,
    amplitude_raw = max(ifelse(i == f_index, abs(ft), 0)),
    amplitude = amplitude_raw/n_tp * 2,
    angle = max(ifelse(i == f_index, -atan2(Im(ft), Re(ft)), -Inf)) %% (2*pi),
    birth_peak = angle / (2*pi),
    fertility_peak = (birth_peak + 1/3.5) %% 1)  %>%
  ungroup() %>%
  select(country_area, amplitude, birth_peak, fertility_peak) %>% distinct()

optimize_params_for_each_model = function(location, categories){
  cat(location,"\n")

  # optimization for model A: needs to be done for each sub-category in that location
  cat("\t model A\n")
  optimized_params_A =
    purrr::map_dfr(
      .x = categories$cat[categories$country_area == location],
      .f = optimize_params_for_model_A,
      categories = categories)
```

```r
# optimization for model B: only needs to be done once for the location
cat("\t model B\n")
optimized_params_B = optimize_params_for_model_B(location, categories = categories)

# optimization for model C: needs to be done for each sub-category in that location
cat("\t model C\n\t")
optimized_params_C =
  purrr::map_dfr(
    .x = categories$cat[categories$country_area == location],
    .f = optimize_params_for_model_C,
    categories = categories,
    optimized_params_B = optimized_params_B[1,])
cat("\n\t done\n")

optimized_par_this_loc = bind_rows(optimized_params_A, optimized_params_B, optimized_params_C)
optimized_par_this_loc
}


source("Scripts/00_functions_optim_each_model.R")
```

The parameters are now being optimized for each category of users (combination of location, birth control and age group) and sex type.

```r
G_table = read_feather(path = str_c(IO$out_Rdata,"Gestation_par_table.feather"))
categories = clue_sex_agg %>% select(country_area, BC, age_cat, sex_type, cat) %>% distinct()

if(!file.exists(str_c(IO$out_Rdata,"optimized_fertility_parameters.feather"))){

  tic()
  opt_par_df = purrr::map_dfr(.x = unique(categories$country_area), .f = optimize_params_for_each_model, categories = categories)
  toc()

  write_feather(opt_par_df, path = str_c(IO$out_Rdata,"optimized_fertility_parameters.feather"))
}else{
  warning("Fertility parameters were not optimized at this execution.\nLoading values from a previous execution.")
  opt_par_df = read_feather(path = str_c(IO$out_Rdata,"optimized_fertility_parameters.feather"))
}
```

Warning: Fertility parameters were not optimized at this execution.
Loading values from a previous execution.

```r
# opt_par_df_Brazil_Northeast = optimize_params_for_each_model(location = "Brazil - Northeast", categories = categories)
```

Computing beta_eff

```r
beta_eff = purrr::map_dfr(.x = unique(categories$country_area), .f = function(location){

  j = which((sex_models_df$country_area == location) & (sex_models_df$BC == "all") & (sex_models_df$age_cat == "all") & (sex_models_df$sex_type == "unprot_sex"))
  this_cat_predicted_sex = predict_daily_sex_behavior(model = sex_models[[j]]$model, date_range = c(as.Date("2000-01-01"),as.Date("2015-12-31")), country_area = lo

  beta_C = opt_par_df %>% ungroup() %>% filter(model == "C", country_area == location, BC == "all", age_cat == "all", sex_type == "unprot_sex") %>% select(beta) %>%

  this_cat_monthly_sex = this_cat_predicted_sex %>%
    mutate(
      sex_A = sex,
      beta_A = max(sex_A - 1),
      beta_C = beta_C,
      sex_C = 1 + beta_C * (sex_A - 1)/beta_A,
      year = year(date)) %>%
    group_by(month, year, beta_A, beta_C) %>%
    summarize(sex_A = sum(sex_A),
```

```
          sex_C = sum(sex_C),
          n = n(),
          sex_A = sex_A/n*30,
          sex_C = sex_C/n*30,
          .groups = "drop") %>%
   filter(n > 20) %>%
   arrange(year, month) %>%
   mutate(beta_eff_A = (max(sex_A)-mean(sex_A))/mean(sex_A),
        beta_eff_C = (max(sex_C)-mean(sex_C))/mean(sex_C),
        country_area = location
   )

 betas = this_cat_monthly_sex %>%
   select(country_area, beta_A, beta_C, beta_eff_A, beta_eff_C) %>%
   distinct() %>%
   pivot_longer(cols = starts_with("beta"), names_to = c(".value","model"), names_pattern =  "(.*)_([AC])")

 betas

})

opt_par_df %>% ungroup() %>%  filter(BC == "all", age_cat == "all", sex_type == "unprot_sex") %>%
 select(country_area, model, alpha) %>%
 left_join(., beta_eff, by = c("country_area", "model")) %>%
 mutate(beta = beta %>% replace_na(0),
      beta_eff = beta_eff %>% replace_na(0))
```

5.5.3   Optimized parameters visualization

```
opt_par_df = opt_par_df %>%
 mutate(country_area_col = dict$country_area$country_area_col[match(country_area, dict$country_area$country_area)])


ggplot(opt_par_df %>% filter( ( (model %in% c("A","B")) & (BC=="all") & (age_cat=="all") & (sex_type == "all_sex") ) | (model == "C") ) ,
     aes(y = SSR, x = interaction(age_cat, BC, sex_type), fill = country_area_col)) +
 geom_bar(stat = "identity", col = "white")+
 xlab("")+
 scale_fill_identity()+
 facet_grid(country_area ~  model , scale = "free", space = "free_x")+
 theme(axis.text.x = element_text(angle = 90, hjust = 1),
     strip.text.y = element_text(angle = 0, hjust = 0))
```
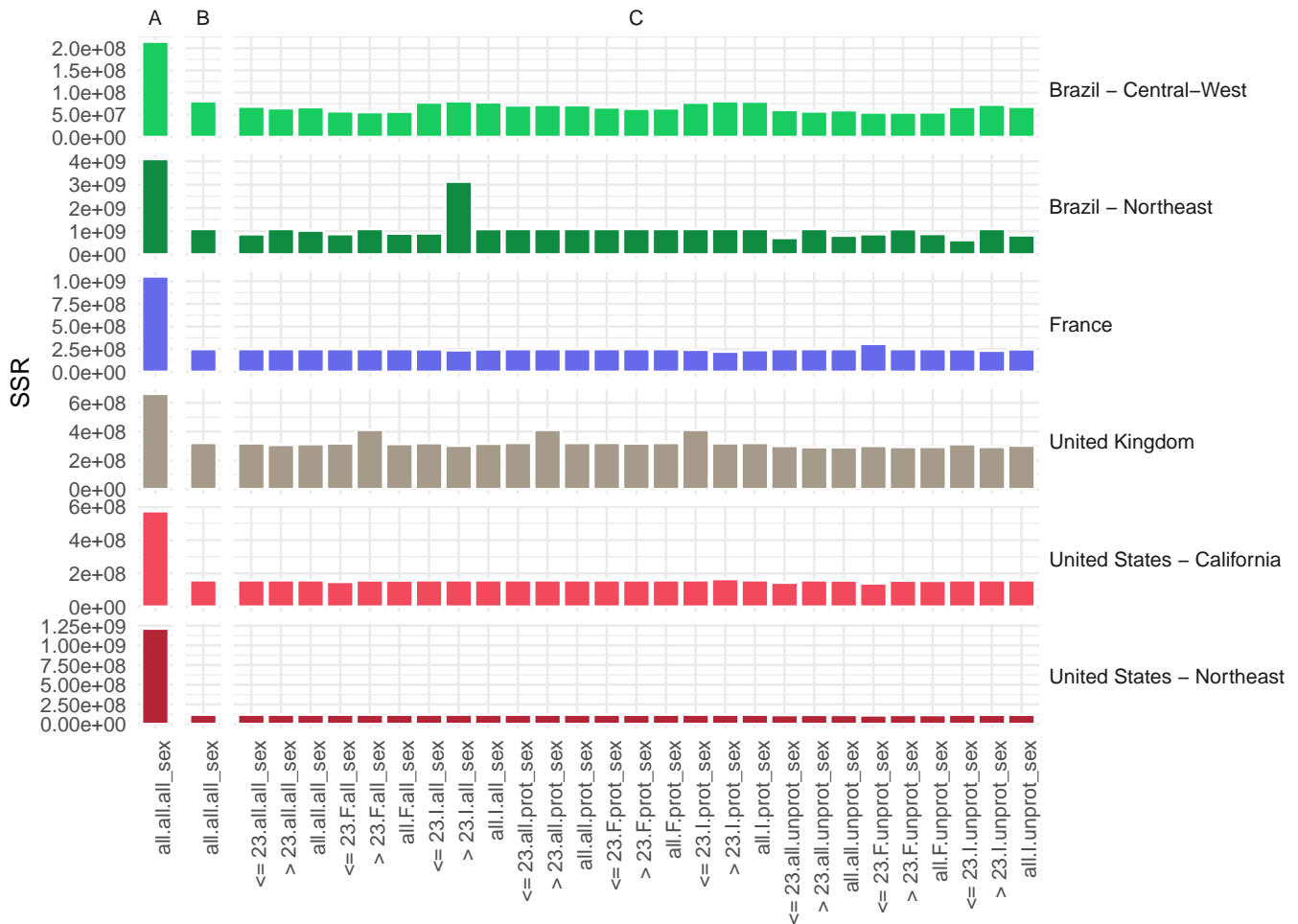
Figure 5.30: Comparison of the residuals for each model and each user category (model C)

```r
opt_par_df = opt_par_df %>%
  mutate(country_area_wrapped =
      country_area %>%
      str_replace(.," - ","\n") %>%
      factor(., levels = dict$country_area$country_area  %>% str_replace(.," - ","\n")))

ggplot(opt_par_df %>% filter(model != "A", ((model == "B")&(BC=="all")&(age_cat=="all")&(sex_type == "all_sex")) | (model == "C") ) ,
    aes(y = alpha, x = interaction(BC, sex_type, age_cat), fill = country_area_col)) +
  geom_bar(stat = "identity", col = NA)+
  scale_fill_identity()+
  xlab("")+
  facet_grid(. ~ country_area_wrapped + model , scale = "free", space = "free")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
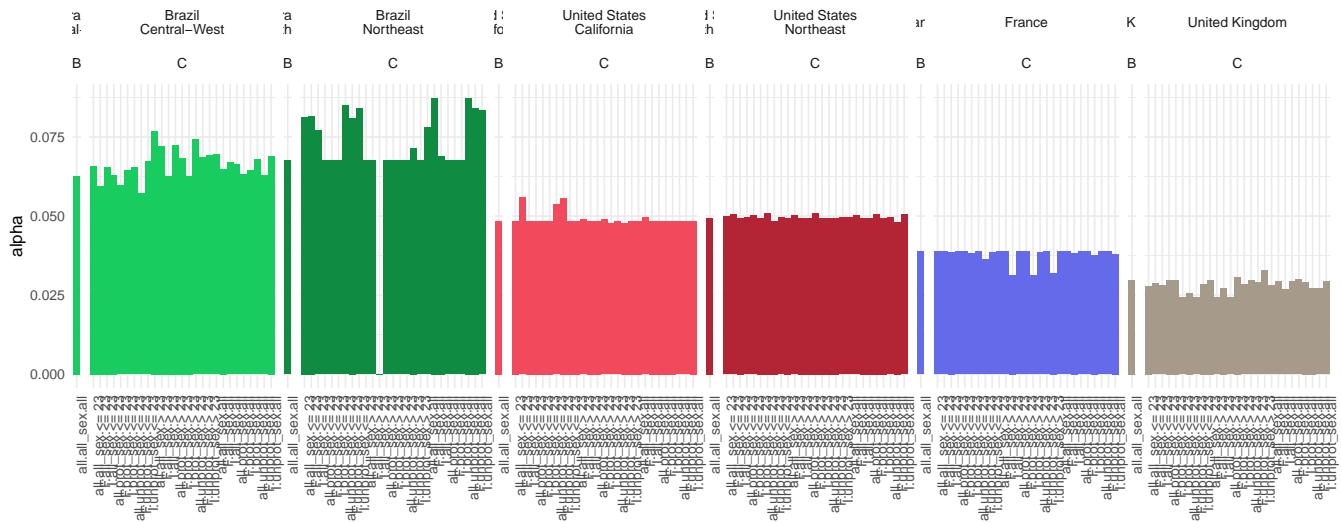
Figure 5.31: Relative amplitude of the fertility curve (optimized value)

```
ggplot(opt_par_df %>% filter(model != "A", ((model == "B")&(BC=="all")&(age_cat=="all")&(sex_type == "all_sex")) | (model == "C") ) ,
    aes(y = 12*(Tp %% 1), x = interaction(BC, sex_type, age_cat), col = country_area_col)) +
geom_point()+
scale_color_identity()+
scale_y_continuous(breaks = seq(0,12,by = 2),limits = c(0,12))+
ylab("Peak time of fertility [months]")+
facet_grid(. ~ country_area_wrapped + model , scale = "free", space = "free")+
theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
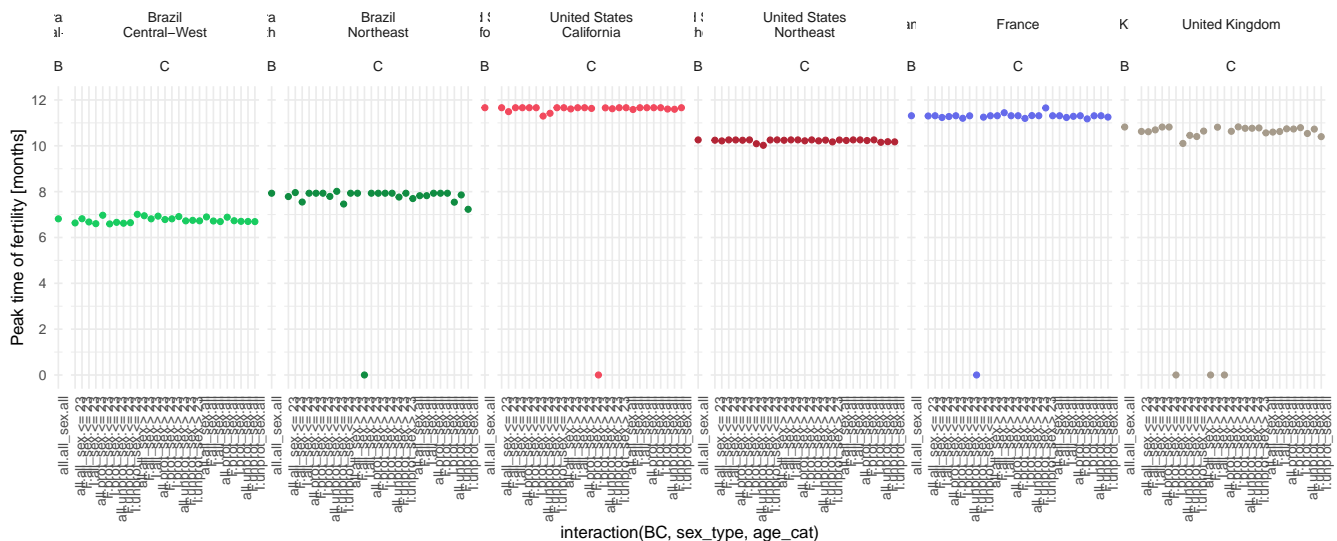

Figure 5.32: Peak time of fertility (optimized value). The y-axis shows the time of the year, expressed in month, at which fertility is the highest. A value of 0 (or 12) corresponds to January 1st.

5.6   Simulating monthly births

```
births = foreach(category = unique(clue_sex_agg$cat), .combine = bind_rows) %do%{

this_cat_sex_data = clue_sex_agg %>% filter(cat == category)
ca = unique(this_cat_sex_data$country_area)
bc = unique(this_cat_sex_data$BC)
ac = unique(this_cat_sex_data$age_cat)
```

77

```r
st = unique(this_cat_sex_data$sex_type)
#cat(ca," - BC: ",bc," - age_cat: ",ac," - ",st, "\n")


# average daily births
this_cat_ave_daily_births = average_daily_births_df %>% filter(country_area == ca)


# actual births
this_ca_births = official_birth_records %>% filter(country_area == ca)


# predicting sex
j = which((sex_models_df$country_area == ca) & (sex_models_df$BC == bc) & (sex_models_df$age_cat == ac) & (sex_models_df$sex_type == st))
this_cat_model = sex_models[[j]]$model
this_cat_predicted_sex = predict_daily_sex_behavior(model = this_cat_model, date_range = range(this_cat_ave_daily_births$date), country_area = ca)



# model A
this_cat_opt_par = opt_par_df %>% filter(model == "A", country_area == ca, age_cat == ac, BC == bc, sex_type == st)
beta = this_cat_opt_par$beta
G = this_cat_opt_par$G
Gsd = this_cat_opt_par$Gsd
monthly_births_A = simulated_vs_actual_monthly_birth(alpha = 0, Tp = 0, beta = beta, G = G, Gsd = Gsd,
                               sex_df = this_cat_predicted_sex,
                               ave_daily_birth_df = this_cat_ave_daily_births,
                               actual_monthly_birth_df = this_ca_births)
monthly_births_A$model = "A"



# model B
this_cat_opt_par = opt_par_df %>% filter(model == "B", country_area == ca, age_cat == ac, BC == bc, sex_type == st)
alpha = this_cat_opt_par$alpha
Tp = this_cat_opt_par$Tp
G = this_cat_opt_par$G
Gsd = this_cat_opt_par$Gsd

sex_model_B = this_cat_predicted_sex %>% mutate(sex = 1)

monthly_births_B = simulated_vs_actual_monthly_birth(alpha = alpha, Tp = Tp, beta = 0, G = G, Gsd = Gsd,
                               sex_df = sex_model_B,
                               ave_daily_birth_df = this_cat_ave_daily_births,
                               actual_monthly_birth_df = this_ca_births)
monthly_births_B$model = "B"



# model C
this_cat_opt_par = opt_par_df %>% filter(model == "C", age_cat == ac, country_area == ca, BC == bc, sex_type == st)
alpha = this_cat_opt_par$alpha
Tp = this_cat_opt_par$Tp
beta = this_cat_opt_par$beta
G = this_cat_opt_par$G
Gsd = this_cat_opt_par$Gsd
monthly_births_C = simulated_vs_actual_monthly_birth(alpha = alpha, Tp = Tp, beta = beta, G = G, Gsd = Gsd,
                               sex_df = this_cat_predicted_sex,
                               ave_daily_birth_df = this_cat_ave_daily_births,
                               actual_monthly_birth_df = this_ca_births)
monthly_births_C$model = "C"

# Putting them together
monthly_births = bind_rows(monthly_births_A,monthly_births_B, monthly_births_C)
```

```
  monthly_births = monthly_births %>%
    mutate(
      country_area = ca,
      BC = bc,
      age_cat = ac,
      sex_type = st,
      cat = category
    ) %>%
    select(country_area, BC, age_cat, sex_type, cat,model, year_month, sim_births, births, residuals, sq_residuals)
  return(monthly_births)
}
```

```
write_feather(births, path = str_c(IO$out_Rdata, "simulated_births.feather"))
```

5.6.1 Time-series visualization

Time series are visualized only for BC = "all" and sex_type = "unprot_sex". Data for all simulated time-series are available on the github repo and can be visualized using the same script.

```
#births = read_feather(path = str_c(IO$out_Rdata, "simulated_births.feather"))

ok = foreach(ca = unique(clue_sex_agg$country_area)) %do% {

 this_cat_births = births %>% filter(country_area == ca, BC == "all", sex_type == "unprot_sex")
 g = ggplot(this_cat_births, aes(x = year_month, y = sim_births/1000 , col = model))
 g = g+
   geom_line(aes(y = births/1000), col = "black", size = 1)+
   geom_line()+
   guides(col = FALSE)+
   ylab("Actual (black) and simulated (colored) births (thousands)")+
   xlab("date")+
   facet_grid(age_cat ~ model)+
   ggtitle(str_c(ca," | BC = all | sex = unprotected sex"))+
   theme(strip.background = element_rect(fill = "gray90", color = NA))
 print(g)

}
```
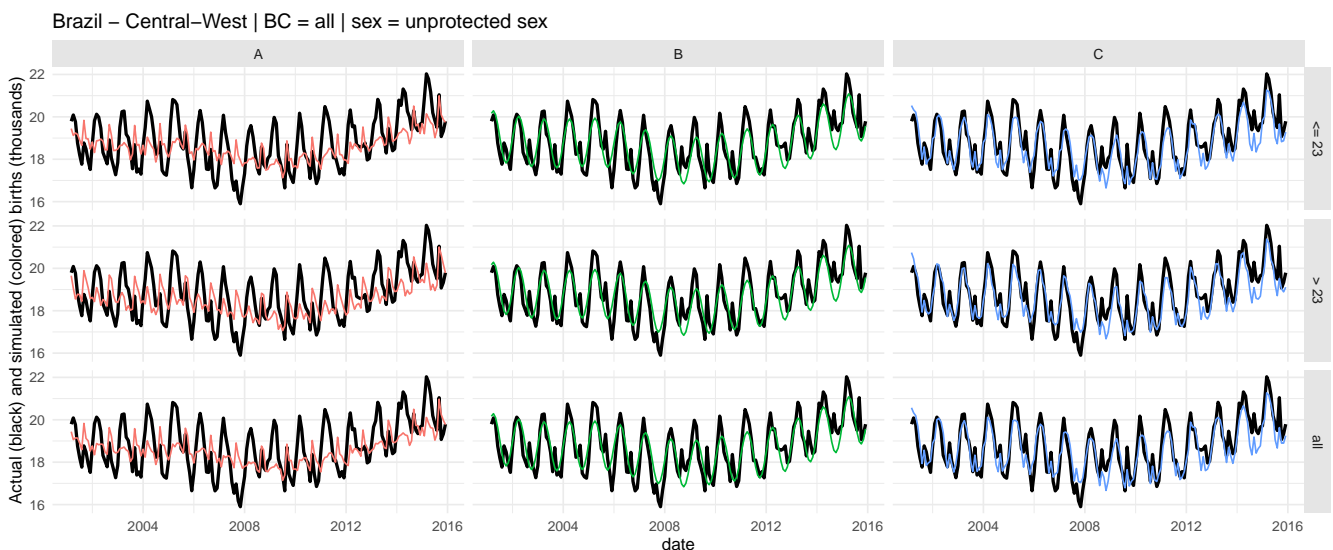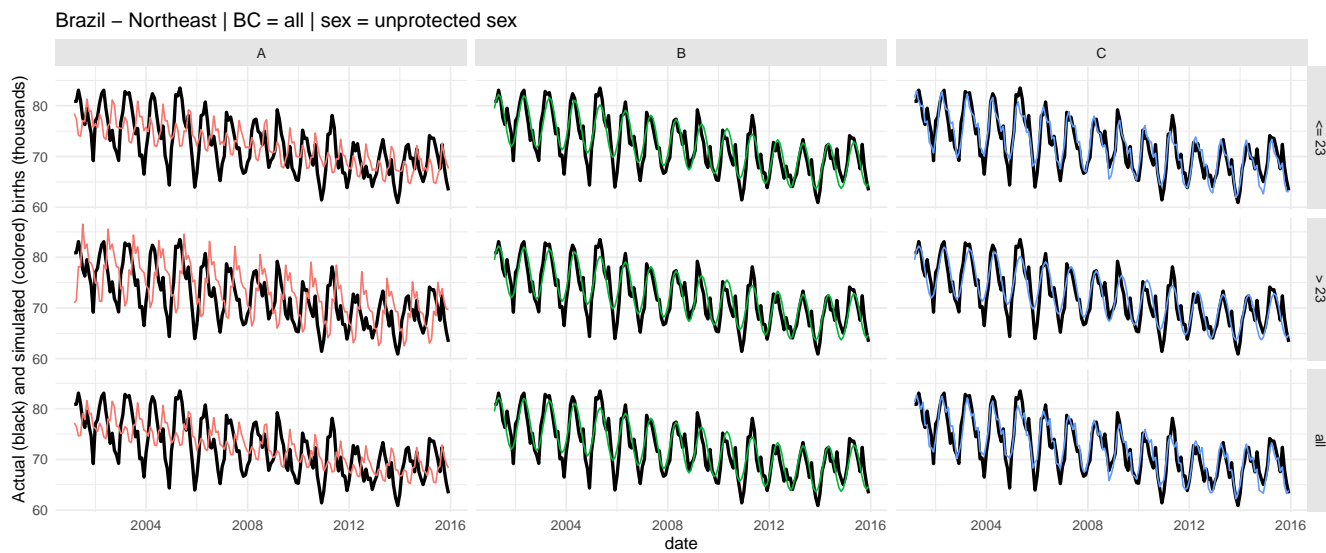

Figure 5.33: Simulated vs actual births.

Figure 5.34: Simulated vs actual births.
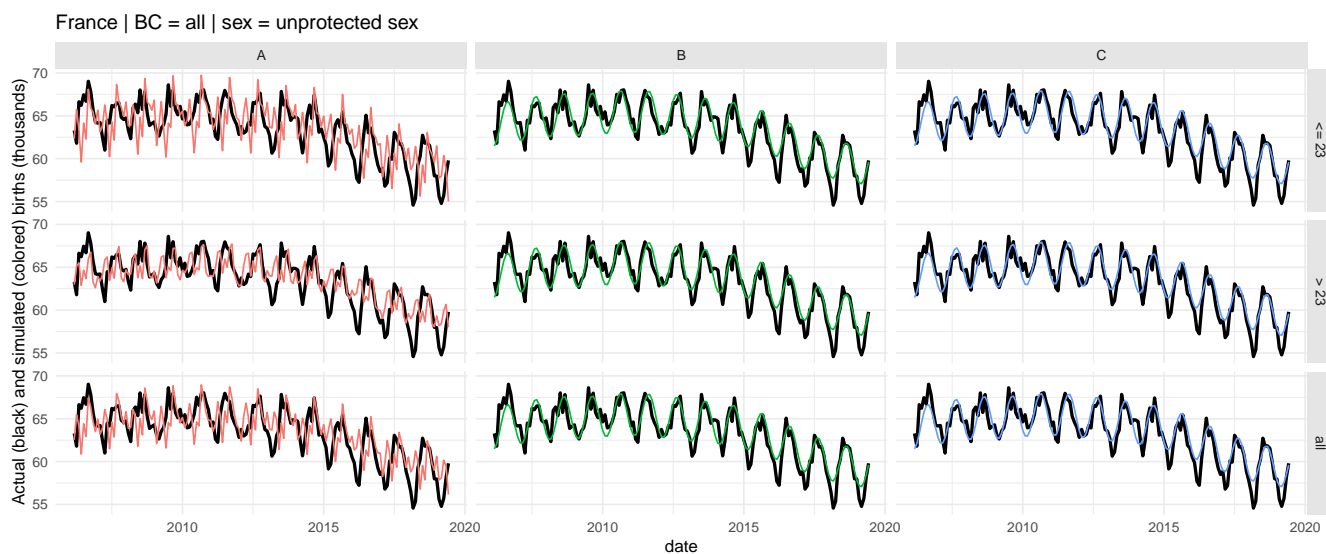
Figure 5.35: Simulated vs actual births.
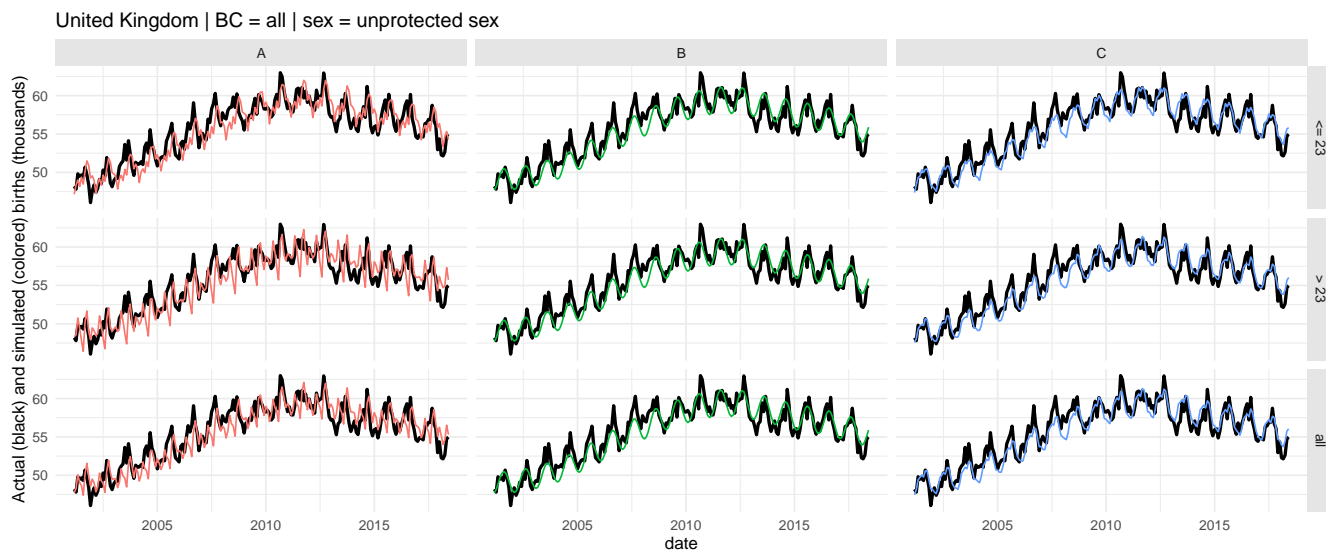
Figure 5.36: Simulated vs actual births.
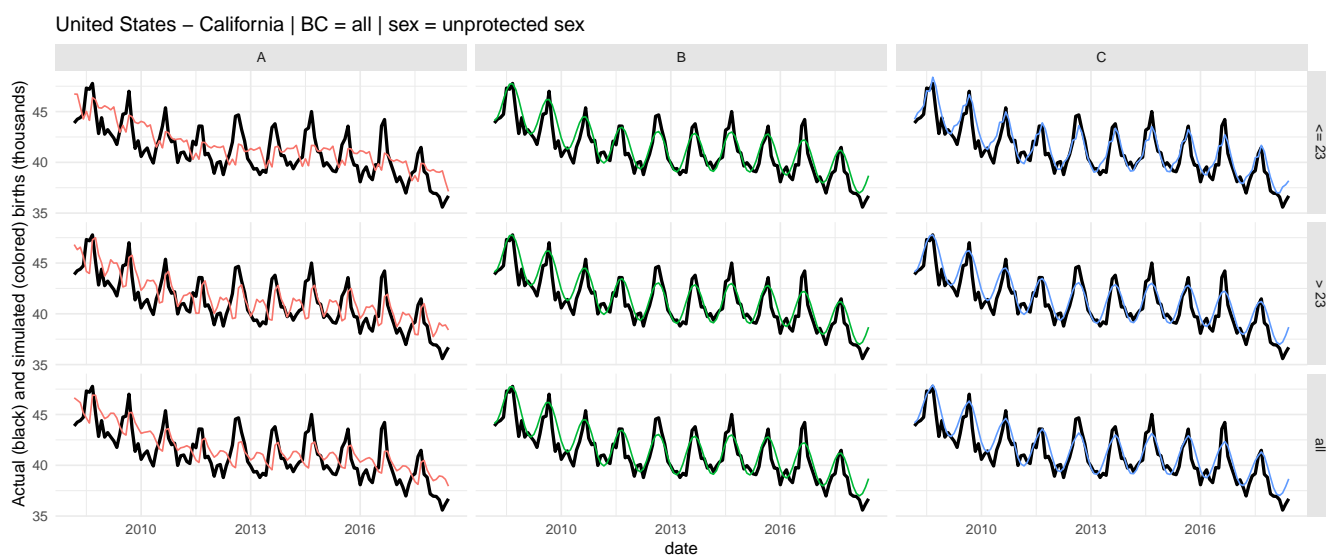
Figure 5.37: Simulated vs actual births.
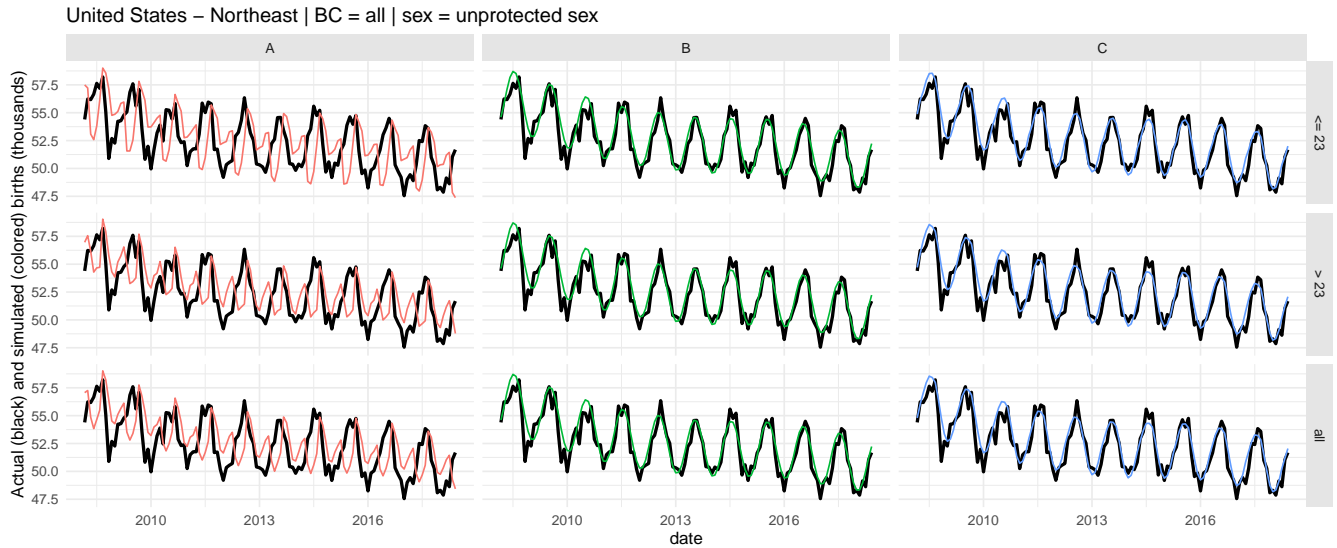
Figure 5.38: Simulated vs actual births.

5.7 Determining best model (A, B or C)

To understand if the seasonal sexual variations are (partially) driving seasonal birth patterns, the AIC (Akaike Information Criteria) is used to compare the three models.(Cavanaugh 1997)

5.7.1 Residuals and AIC

```
opt_par_df = opt_par_df %>% mutate(sex_type_short = sex_type %>% str_remove(.,"_sex"))

ggplot(opt_par_df, aes(x = model, y = SSR , fill = model))+ #
  geom_bar(stat = "identity")+
  guides(fill = FALSE)+
  facet_grid(country_area ~ age_cat + BC + sex_type_short, scale = "free_y")+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
```


Figure 5.39: Residuals for each model and each category of users.

We compute the AIC for each category of users.

```
# first we need to get the number of data point for each time-series
n_tp = births %>%
  group_by(country_area, age_cat, BC, sex_type, model) %>%
  summarize(n = n(),
        sigmasq = var(births), #residuals
        .groups = "drop")

opt_par_df = full_join(opt_par_df, n_tp, by = c("country_area", "BC","age_cat", "sex_type", "model"))

opt_par_df = opt_par_df %>%
  mutate(n_par = case_when(
    model == "A" ~ 0,
    model == "B" ~ 2,
    model == "C" ~ 3),
    lL = n * log(1/sqrt(2*pi*sigmasq)) - 1/(2*sigmasq)*SSR,
    AIC = 2*n_par - 2*lL
  )

ggplot(opt_par_df, aes(x = model, y = AIC , col = model))+ #
  geom_point(size = 3)+
  facet_grid(country_area ~ sex_type_short + BC + age_cat , scale = "free_y")+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
```

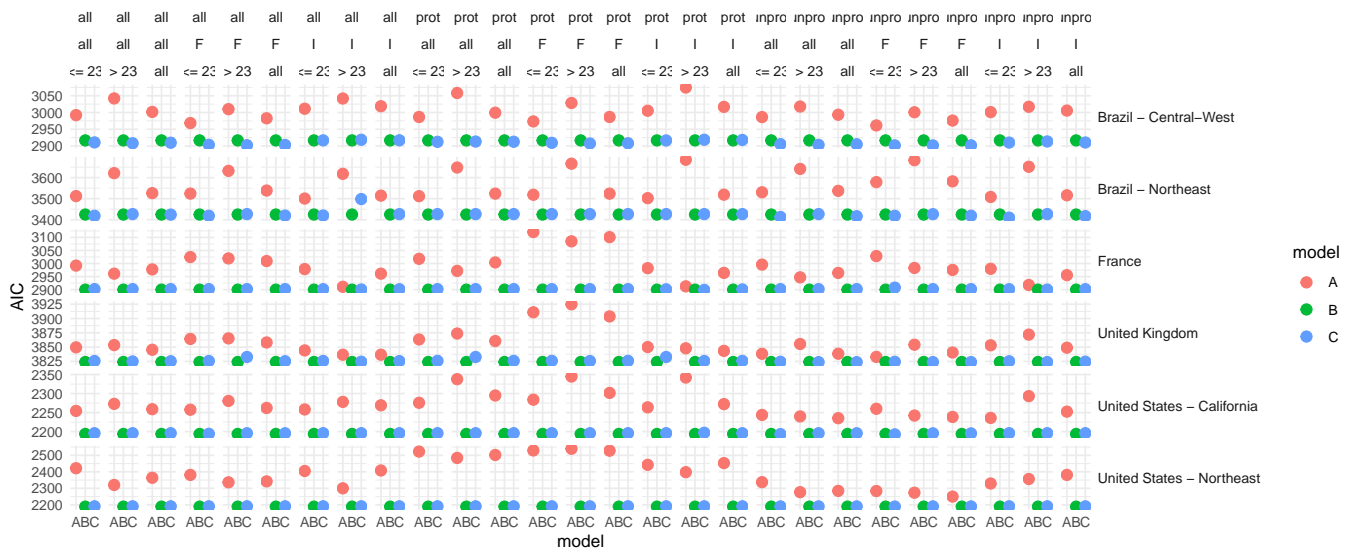

Figure 5.40: AIC for each model and each category of users.

```
ggplot(opt_par_df, aes(x = AIC, fill = model))+
  geom_histogram(position = "identity", bins = 50, aes(y = ..ndensity..))+
  facet_grid(model ~ country_area, scale = "free")+ # age_cat + BC + sex_type
  theme(legend.position = "bottom")
```
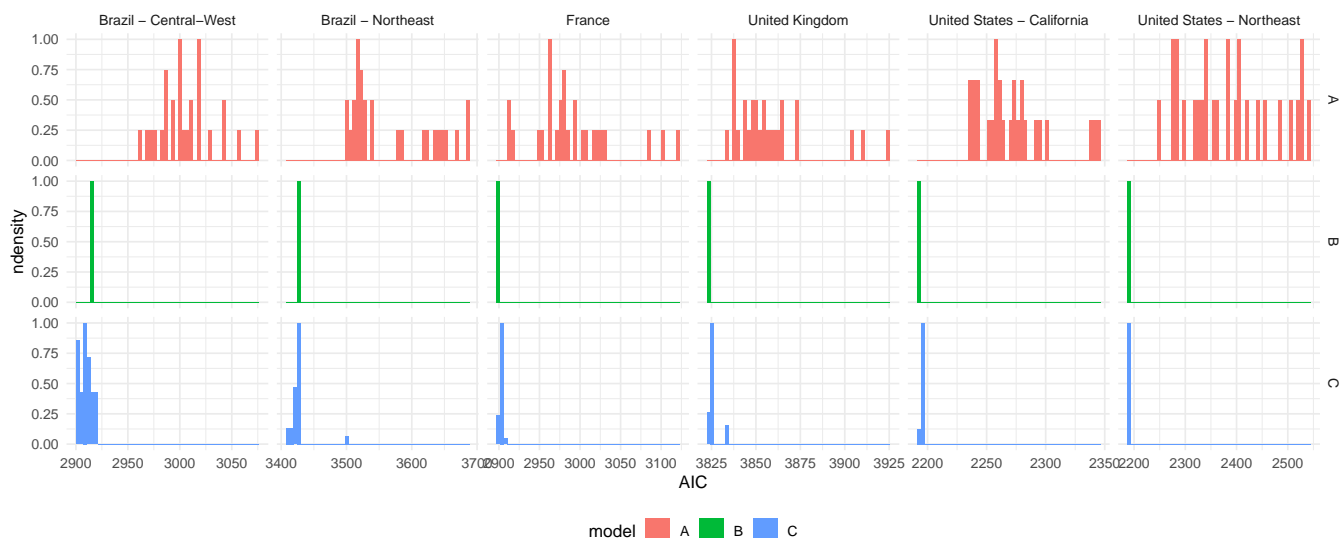
Figure 5.41: AIC distributions for each model and country/area.

The first observation, is that the AIC values of model A are always much higher than those of models B or C. The conclusion of this observation is that variations in sexual behavior alone do not explain seasonal births.

```r
AIC_table = opt_par_df %>%
  select(country_area, sex_type, BC, age_cat, model, AIC) %>%
  arrange(country_area, sex_type, BC, age_cat, model) %>%
  mutate(AIC = round(AIC, 1)) %>%
  pivot_wider(id_cols = c(country_area, BC, age_cat),
        values_from = AIC,
        names_from = c("sex_type","model"))


colnames(AIC_table) = colnames(AIC_table) %>%
  str_remove("all_sex_") %>% str_remove("unprot_sex_") %>% str_remove("prot_sex_")

kable(
  AIC_table,
  format = "latex",
  booktabs = "T",
  linesep = "",
  caption = "AIC values for each country/area, sex type and user category"
  ) %>%
  kable_styling(
    latex_options = c("striped","scale_down"),
    stripe_index = rep(1:3, 3)+ rep(c(0,6,12), each = 6),
    font_size = 8) %>%
add_header_above(c(" " = 3, "Total sex" = 3, "Protected sex" = 3, "Unrotected sex" = 3))

best_models = opt_par_df %>%
  arrange(country_area, BC, age_cat, sex_type, AIC) %>%
  group_by(country_area, BC, age_cat, sex_type) %>%
  top_n(n = 1, wt = desc(AIC)) %>%
  rename(best_model = model) %>%
  ungroup()

best_models_table = best_models %>%
  select(country_area, BC, age_cat , sex_type, best_model) %>%
  pivot_wider(names_from = c("BC", "sex_type"), values_from = "best_model", names_prefix = "BC: ", names_sep = " - ")
```

Table 5.2: AIC values for each country/area, sex type and user category

country_area	BC	age_cat	Total sex			Protected sex			Unrotected sex		
			A	B	C	A	B	C	A	B	C
Brazil - Central-West	all	<= 23	2992.2	2916.6	2910.6	2986.6	2916.6	2912.4	2986.3	2916.6	2905.5
Brazil - Central-West	all	> 23	3042.0	2916.6	2908.1	3058.2	2916.6	2913.2	3017.9	2916.6	2903.3
Brazil - Central-West	all	all	3001.8	2916.6	2909.7	2999.4	2916.6	2912.6	2993.4	2916.6	2905.1
Brazil - Central-West	F	<= 23	2968.5	2916.6	2903.5	2973.4	2916.6	2909.4	2961.4	2916.6	2901.8
Brazil - Central-West	F	> 23	3010.0	2916.6	2902.3	3028.5	2916.6	2907.4	3000.7	2916.6	2901.7
Brazil - Central-West	F	all	2982.7	2916.6	2903.0	2986.6	2916.6	2908.0	2976.0	2916.6	2901.9
Brazil - Central-West	I	<= 23	3011.3	2916.6	2916.7	3005.3	2916.6	2916.5	3001.2	2916.6	2910.2
Brazil - Central-West	I	> 23	3041.8	2916.6	2918.6	3074.7	2916.6	2918.6	3017.1	2916.6	2913.4
Brazil - Central-West	I	all	3018.7	2916.6	2916.9	3016.8	2916.6	2918.0	3005.8	2916.6	2910.3
Brazil - Northeast	all	<= 23	3512.3	3424.6	3418.6	3511.8	3424.6	3426.6	3530.4	3424.6	3413.0
Brazil - Northeast	all	> 23	3621.4	3424.6	3426.6	3648.1	3424.6	3426.6	3641.6	3424.6	3426.6
Brazil - Northeast	all	all	3526.3	3424.6	3424.2	3524.1	3424.6	3426.6	3537.1	3424.6	3416.5
Brazil - Northeast	F	<= 23	3524.0	3424.6	3418.8	3518.6	3424.6	3426.6	3578.6	3424.6	3418.7
Brazil - Northeast	F	> 23	3632.2	3424.6	3426.6	3666.3	3424.6	3426.6	3682.9	3424.6	3426.3
Brazil - Northeast	F	all	3538.7	3424.6	3419.6	3523.9	3424.6	3426.6	3582.2	3424.6	3419.0
Brazil - Northeast	I	<= 23	3500.7	3424.6	3420.0	3502.5	3424.6	3426.6	3508.1	3424.6	3409.8
Brazil - Northeast	I	> 23	3618.2	3424.6	3498.1	3685.1	3424.6	3426.6	3651.7	3424.6	3426.6
Brazil - Northeast	I	all	3514.5	3424.6	3426.4	3519.0	3424.6	3426.6	3516.0	3424.6	3417.0
France	all	<= 23	2992.1	2899.9	2901.9	3018.0	2899.9	2901.8	2995.8	2899.9	2901.9
France	all	> 23	2961.1	2899.9	2901.9	2971.7	2899.9	2901.9	2947.1	2899.9	2901.9
France	all	all	2977.4	2899.9	2901.9	3004.0	2899.9	2901.9	2964.0	2899.9	2901.9
France	F	<= 23	3024.7	2899.9	2901.9	3120.6	2899.9	2901.9	3028.4	2899.9	2907.7
France	F	> 23	3019.5	2899.9	2901.9	3084.9	2899.9	2901.9	2982.9	2899.9	2901.9
France	F	all	3009.4	2899.9	2901.9	3101.0	2899.9	2901.9	2975.4	2899.9	2901.9
France	I	<= 23	2978.6	2899.9	2901.6	2982.1	2899.9	2901.1	2979.6	2899.9	2901.7
France	I	> 23	2910.8	2899.9	2900.5	2912.5	2899.9	2899.0	2917.8	2899.9	2900.0
France	I	all	2961.1	2899.9	2901.5	2964.0	2899.9	2900.7	2955.4	2899.9	2901.6
United Kingdom	all	<= 23	3849.6	3824.0	3825.8	3863.4	3824.0	3826.0	3838.4	3824.0	3824.4
United Kingdom	all	> 23	3853.6	3824.0	3825.0	3873.8	3824.0	3832.7	3855.5	3824.0	3823.8
United Kingdom	all	all	3845.4	3824.0	3825.4	3860.6	3824.0	3826.0	3838.5	3824.0	3823.8
United Kingdom	F	<= 23	3864.3	3824.0	3825.7	3910.9	3824.0	3826.0	3833.0	3824.0	3824.5
United Kingdom	F	> 23	3865.2	3824.0	3832.7	3924.9	3824.0	3825.7	3854.2	3824.0	3823.9
United Kingdom	F	all	3858.1	3824.0	3825.4	3903.6	3824.0	3826.0	3840.6	3824.0	3823.9
United Kingdom	I	<= 23	3844.2	3824.0	3825.9	3850.1	3824.0	3832.7	3853.2	3824.0	3825.3
United Kingdom	I	> 23	3836.8	3824.0	3824.6	3848.0	3824.0	3825.8	3871.9	3824.0	3823.9
United Kingdom	I	all	3836.8	3824.0	3825.6	3843.4	3824.0	3826.0	3849.0	3824.0	3824.6
United States - California	all	<= 23	2254.4	2193.8	2195.8	2275.7	2193.8	2195.8	2243.7	2193.8	2193.4
United States - California	all	> 23	2272.9	2193.8	2195.8	2337.8	2193.8	2195.8	2239.9	2193.8	2195.7
United States - California	all	all	2258.8	2193.8	2195.8	2294.9	2193.8	2195.8	2235.2	2193.8	2195.6
United States - California	F	<= 23	2257.5	2193.8	2194.2	2284.0	2193.8	2195.8	2260.0	2193.8	2192.6
United States - California	F	> 23	2280.6	2193.8	2195.7	2344.8	2193.8	2195.7	2242.2	2193.8	2195.5
United States - California	F	all	2262.2	2193.8	2195.6	2301.7	2193.8	2195.8	2238.5	2193.8	2195.2
United States - California	I	<= 23	2258.1	2193.8	2195.8	2263.8	2193.8	2195.8	2235.9	2193.8	2195.8
United States - California	I	> 23	2278.1	2193.8	2195.8	2342.1	2193.8	2197.0	2293.2	2193.8	2195.8
United States - California	I	all	2269.1	2193.8	2195.8	2272.2	2193.8	2195.8	2252.4	2193.8	2195.8
United States - Northeast	all	<= 23	2421.8	2189.0	2191.0	2522.8	2189.0	2191.0	2337.4	2189.0	2190.2
United States - Northeast	all	> 23	2320.4	2189.0	2191.0	2484.3	2189.0	2191.0	2276.7	2189.0	2190.6
United States - Northeast	all	all	2364.0	2189.0	2191.0	2502.1	2189.0	2191.0	2284.6	2189.0	2190.5
United States - Northeast	F	<= 23	2381.3	2189.0	2191.0	2529.5	2189.0	2191.0	2283.6	2189.0	2189.6
United States - Northeast	F	> 23	2336.4	2189.0	2191.0	2539.8	2189.0	2190.9	2273.4	2189.0	2190.6
United States - Northeast	F	all	2342.0	2189.0	2191.0	2528.2	2189.0	2191.0	2249.9	2189.0	2190.3
United States - Northeast	I	<= 23	2405.1	2189.0	2191.0	2442.4	2189.0	2191.0	2329.3	2189.0	2191.0
United States - Northeast	I	> 23	2300.0	2189.0	2191.0	2397.9	2189.0	2191.0	2356.4	2189.0	2190.8
United States - Northeast	I	all	2408.2	2189.0	2191.0	2453.1	2189.0	2191.0	2381.4	2189.0	2190.8

Table 5.3: Best Models (i.e. with the lowest AIC) for each country/area, user category and sex type

country_area	age_cat	BC: all			BC: F			BC: I		
		all_sex	prot_sex	unprot_sex	all_sex	prot_sex	unprot_sex	all_sex	prot_sex	unprot_sex
Brazil - Central-West	<= 23	C	C	C	C	C	C	B	C	C
Brazil - Central-West	> 23	C	C	C	C	C	C	B	B	C
Brazil - Central-West	all	C	C	C	C	C	C	B	B	C
Brazil - Northeast	<= 23	C	B	C	C	B	C	C	B	C
Brazil - Northeast	> 23	B	B	B	B	B	B	B	B	B
Brazil - Northeast	all	C	B	C	C	B	C	B	B	C
France	<= 23	B	B	B	B	B	B	B	B	B
France	> 23	B	B	B	B	B	B	B	C	B
France	all	B	B	B	B	B	B	B	B	B
United Kingdom	<= 23	B	B	B	B	B	B	B	B	B
United Kingdom	> 23	B	B	C	B	B	C	B	B	C
United Kingdom	all	B	B	C	B	B	C	B	B	B
United States - California	<= 23	B	B	C	B	B	C	B	B	B
United States - California	> 23	B	B	B	B	B	B	B	B	B
United States - California	all	B	B	B	B	B	B	B	B	B
United States - Northeast	<= 23	B	B	B	B	B	B	B	B	B
United States - Northeast	> 23	B	B	B	B	B	B	B	B	B
United States - Northeast	all	B	B	B	B	B	B	B	B	B

```r
colnames(best_models_table) = colnames(best_models_table) %>%
  str_remove("BC: all") %>% str_remove("BC: F") %>% str_remove("BC: I") %>%
  str_remove(" - ")
```

```r
kable(best_models_table, format = "latex", booktabs = T, linesep = "",
    caption = "Best Models (i.e. with the lowest AIC) for each country/area, user category and sex type") %>%
  kable_styling(
    latex_options = c("striped","scale_down"),
    stripe_index = rep(1:3, 3)+ rep(c(0,6,12), each = 3),
    font_size = 8) %>%
add_header_above(c(" " = 2, "BC: all" = 3, "BC: F" = 3, "BC: I" = 3))
```

In table 5.3, model B is predominantly the best model for most countries/areas and users category. However, the differences in AIC are sometimes very mild, as shown in fig 5.42. In this figure, to understand if sexual variations contribute to seasonal birth together with seasonal fertility, we display the AIC difference between model B and model C.

```r
tmp = opt_par_df %>% filter(model != "A") %>%
  select(country_area, age_cat, BC, sex_type, model, AIC) %>%
  pivot_wider(names_from = model, values_from = AIC) %>%
  mutate(AIC_diff_B_minus_C = B-C,
      cat = interaction(country_area, BC, age_cat, sex_type))

tmp = full_join(tmp, df_agg, by = "cat") %>%
  mutate(sex_type_short = sex_type %>% str_remove(.,"_sex"))

age_df = purrr::map_dfr(.x = unique(tmp$country_area),
              .f = function(x) tmp %>% filter(country_area == x, age_cat == get_age_cat(x), BC == "all", sex_type == "unprot_sex"))

ggplot(tmp, aes(ymin = 0, ymax = median_n_users, xmin = 0, xmax = 1))+
  geom_rect(aes(fill = AIC_diff_B_minus_C))+
  geom_rect(data = age_df, fill = "transparent", color = "black")+
  scale_x_continuous(breaks = NULL) +  #scale_y_continuous(breaks = NULL) +
  xlab("") +
  ylab("Relative median number of users \n contributing to the aggregated time-series") +
  scale_fill_gradient2(name = "Difference in AIC:  AIC(B) - AIC(C)", low = "red", high = "blue", mid = "gray90", midpoint = 0)+
  facet_grid(country_area ~ age_cat + BC + sex_type_short , scale = "free_y")+
  theme(legend.position = "bottom",
```
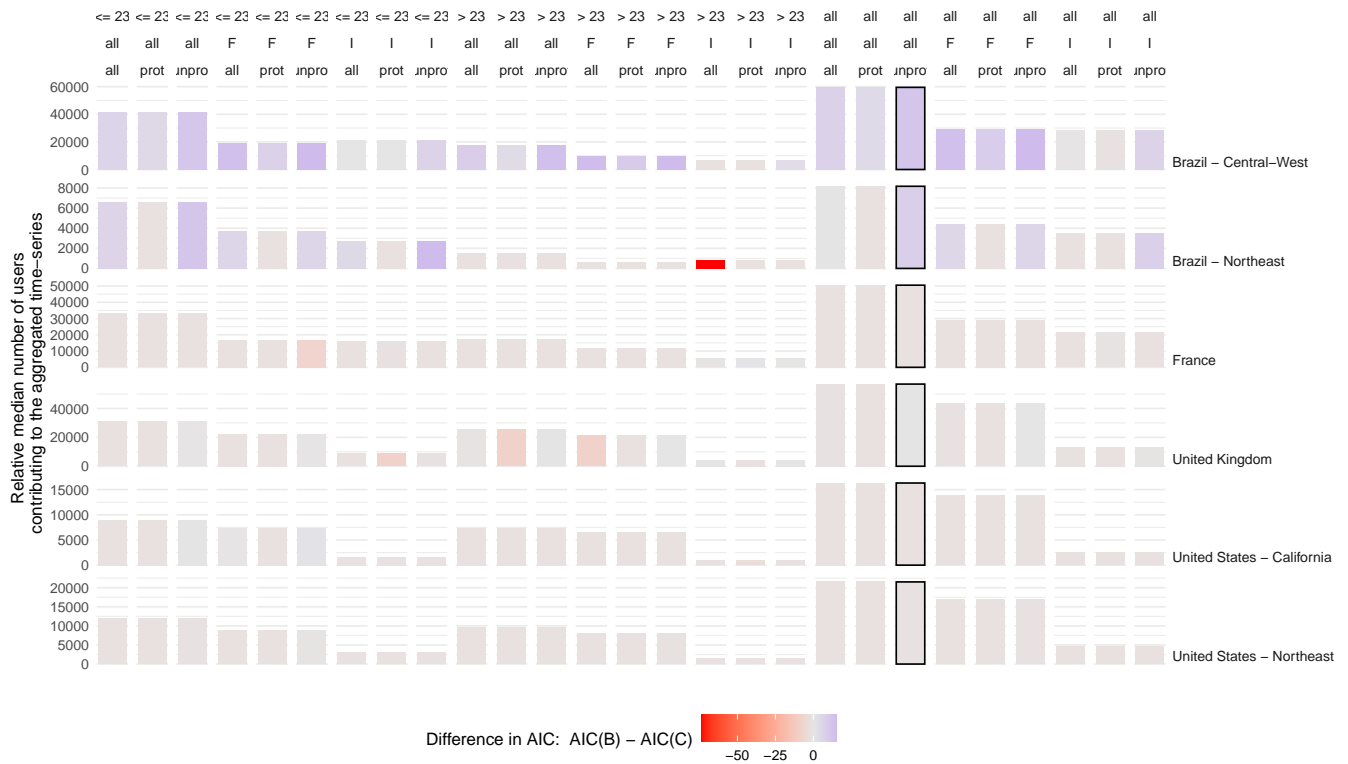
Figure 5.42: Difference in AIC between model B and model C. A positive (blue-ish) value indicated that sexual frequency variations contribute to explaining the seasonal birth patterns. Rectangle height is proportional to the median number of app users who contributed to the aggregated time-series of sexual frequency. The black rectangles indicates the categories of users used for the rest of the analysis (see text)

In fig 5.42, the black rectangles show the categories of users selected for the rest of the analysis. These categories correspond to sex_type = 'unprotected sex' (which is most likely to lead to a pregnancy), includes all users on any kind of birth control (incl. no birth control at all) and for most countries/areas, users of all age groups were considered.

write_feather(opt_par_df, path = **str_c**(IO**$**out_Rdata, "optimal_parameters_and_AIC.feather"))

87

5.7.2   Seasonal decomposition and seasonal trends comparisons

Before doing a seasonal decomposition of the actual vs simulated births, a visualization of the time-series with the months stacked on top of each other for each year of simulated data helps understand where the variability in high-frequency changes in births patterns and how model A and model C, by accounting for non-fixed holidays, correlates with these high-frequency changes or minor peaks.

```r
ok = foreach(ca = unique(clue_sex_agg$country_area)) %do% {

  ac = get_age_cat(country_area = ca)

  bc = "all"

  this_cat_births = births %>% filter(country_area == ca,  BC == bc, age_cat == ac, sex_type == "unprot_sex")
  aic = opt_par_df %>% filter(country_area == ca,  BC == bc, age_cat == ac, sex_type == "unprot_sex")
  best_model_df = best_models %>% filter(country_area == ca,  BC == bc, age_cat == ac, sex_type == "unprot_sex")

  g = ggplot(this_cat_births, aes(x = month, y = sim_births/1000 , col = model))
  g = g+
    geom_line(aes(y = births/1000), col = "black", size = 1.2)+
    geom_line()+
    guides(col = FALSE)+
    ylab("Births (thousands)")+
    facet_grid(year ~ model, scale = "free_y")+
    scale_x_continuous(breaks = seq(1,12,by = 3), labels = c("Jan","Apr","Jul","Oct"))+
    ggtitle(str_c(ca, " / BC : all / age_cat : ",ac," /sex : unprotected \n BEST MODEL = ",best_model_df$best_model))+
    theme(strip.text.y = element_text(angle = 0, hjust = 0))
  print(g)

}
```

Figure 5.43: Actual (black lines) and simulated (colored lines) births.

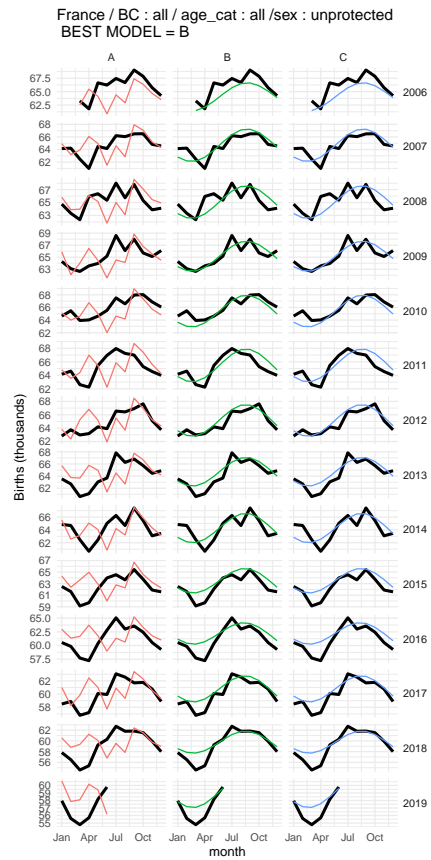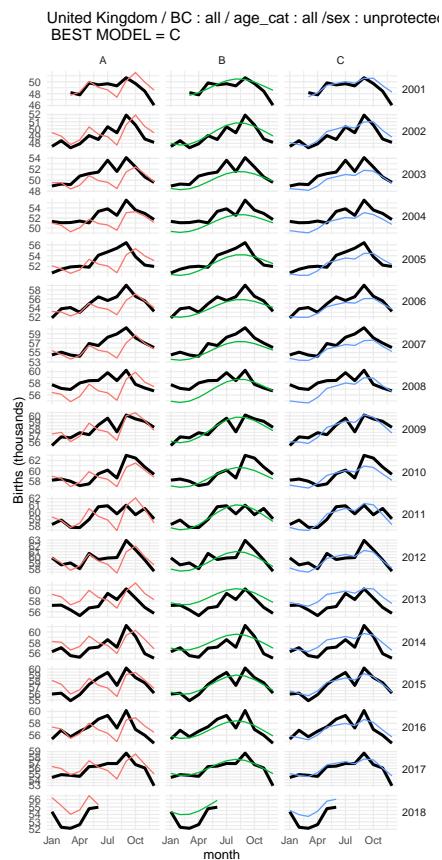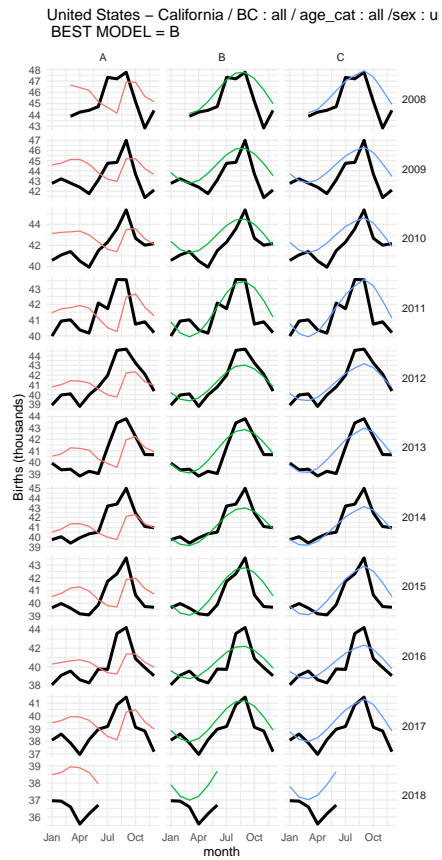Figure 5.44: Actual (black lines) and simulated (colored lines) births.

Figure 5.45: Actual (black lines) and simulated (colored lines) births.

Figure 5.46: Actual (black lines) and simulated (colored lines) births.

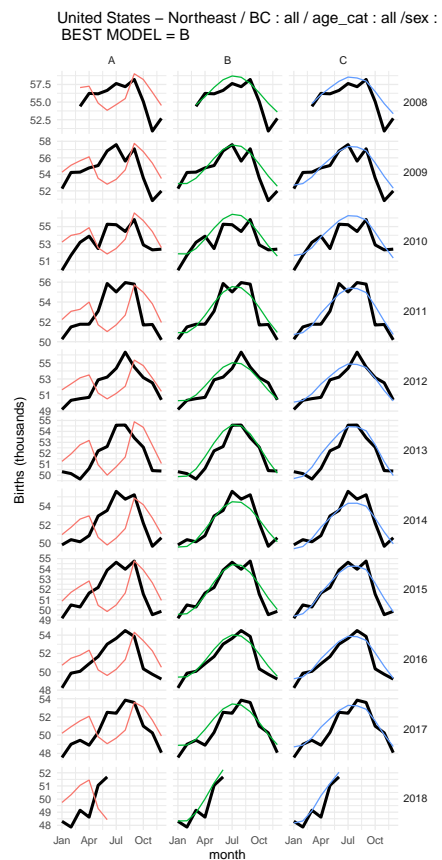Figure 5.47: Actual (black lines) and simulated (colored lines) births.

Figure 5.48: Actual (black lines) and simulated (colored lines) births.

```r
births_STL = foreach(ca = unique(births$country_area), .combine = function(x,y) suppressWarnings(bind_rows(x,y))) %do% {

  ac = get_age_cat(country_area = ca)

  this_ca_births = births %>% filter(country_area == ca, age_cat == ac, sex_type == "unprot_sex", BC == "all")

  ### seasonal decomposition sim births
  this_ca_births_STL = foreach(m = c("A","B","C"), .combine = function(x,y) suppressWarnings(bind_rows(x,y))) %do% {
    this_ca_births_M = this_ca_births %>% filter(model == m)
    sb_ts = ts(this_ca_births_M$sim_births,
          start = c(this_ca_births_M$year[1],this_ca_births_M$month[1]),
          end = c(last(this_ca_births_M$year),last(this_ca_births_M$month)),
          frequency = 12)
    sb_stl = stl(sb_ts, s.window = "periodic")
    this_ca_births_M = this_ca_births_M %>% mutate(
      sim_births_trend = sb_stl$time.series[,2],
      sim_births_seasonal = sb_stl$time.series[,1],
      sim_births_remainder = sb_stl$time.series[,3]
    )
    this_ca_births_M
  }

  ### seasonal decomposition births
  this_ca_births_ = this_ca_births %>% filter(model == "A")
  b_ts = ts(this_ca_births_$births,
        start = c(this_ca_births_$year[1],this_ca_births_$month[1]),
        end = c(last(this_ca_births_$year),last(this_ca_births_$month)),
        frequency = 12)
  b_stl = stl(b_ts, s.window = "periodic")
  this_ca_births_ = this_ca_births_ %>% mutate(
    births_trend = b_stl$time.series[,2],
    births_seasonal = b_stl$time.series[,1],
    births_remainder = b_stl$time.series[,3]
  ) %>% select(country_area, BC, sex_type, year_month, births_trend, births_seasonal, births_remainder)

  # joining sim births all models with measured births
  this_ca_births_STL = full_join(this_ca_births_STL, this_ca_births_, by = c("country_area", "BC", "sex_type", "year_month"))

  # return
  this_ca_births_STL
}

write_feather(births_STL, path = str_c(IO$out_Rdata, "simulated_births_seasonal_trends.feather"))

g = ggplot(births_STL, aes(x = month, y = sim_births_seasonal, col = model))
g = g +
  geom_hline(yintercept = 0, col = "gray80")+
  geom_line(aes(y = births_seasonal), col = "black", size = 1.2)+
  geom_line()+
  guides(col = FALSE)+
  scale_x_continuous(breaks = seq(0,12, by = 3))+
  facet_grid(country_area ~ model, scale = "free_y")+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
g
```
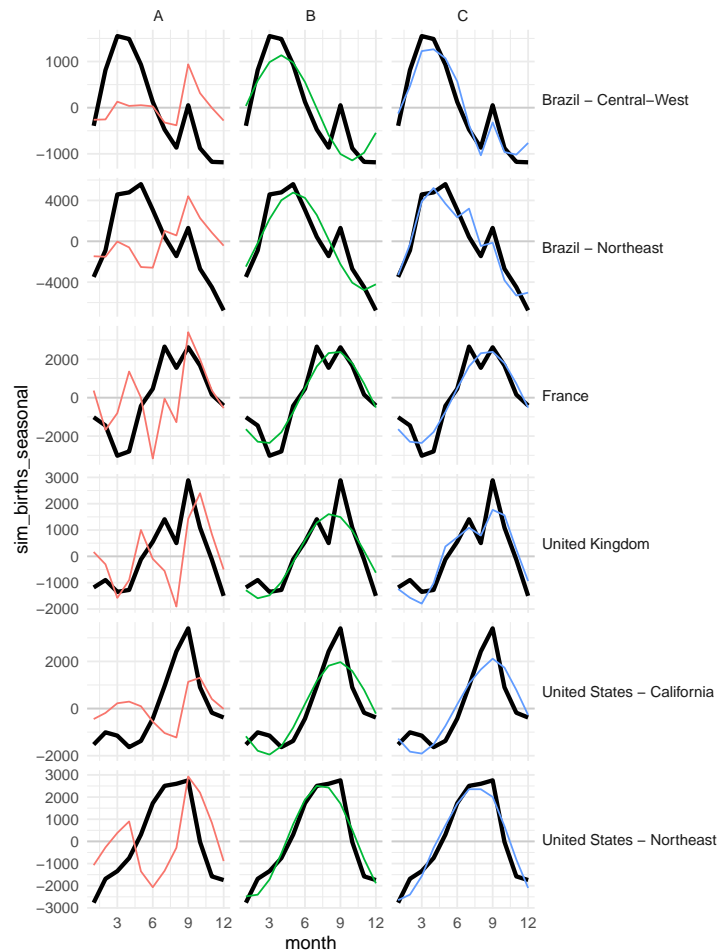
Figure 5.49: Seasonal trends of actual (black lines) and simulated (colored lines) births.

```r
g = ggplot(births_STL %>% filter(model != "A"), aes(x = year_month, y = sim_births_remainder, col = model))
g = g +
  geom_hline(yintercept = 0, col = "gray80")+
  geom_line(aes(y = births_remainder), col = "black")+
  geom_line()+
  guides(col = FALSE)+
  ylab("Remainders of the seasonal decompositions")+ xlab("date")+
  facet_grid(country_area ~ model, scale = "free", labeller = label_both)+
  ggtitle("Remainders")+
  theme(strip.text.y = element_text(angle = 0, hjust = 0))
g
```
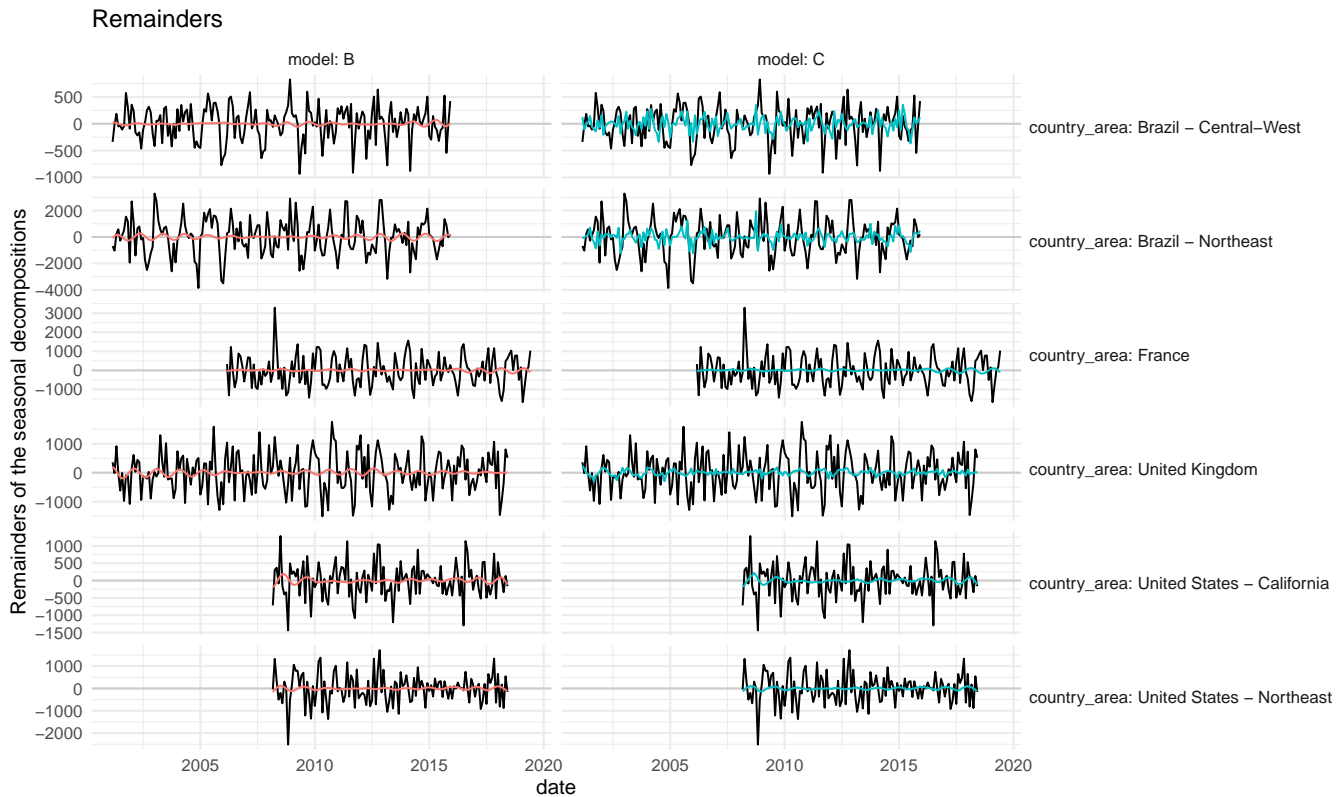
Remainders

Figure 5.50: Remainders of the seasonal decompositions on actual (black lines) and simulated (colored lines) births.

The Remainders of the actual births are a lot larger than the Remainders of the simulated births, which means that the variability in the actual births is larger than the variation in the simulated births. In our model, the year-to-year variability in the simulated birth is driven by the non-fixed holidays. This seems to not be a sufficient source of variation to explain the variability in the actual births.

5.7.3 Year-to-year variability in holiday day-of-year.

```
holidays_flex = get_extended_holidays(countries = unique(official_birth_records$country),
                        year_range = range(official_birth_records$year),
                        hdict = dict$holidays,
                        n_days = 0)

holidays_flex = holidays_flex %>%
  mutate(
    day_of_year = as.Date("2020-01-01") + days(date - floor_date(date, unit = "year")),
    color = dict$holidays$color[match(holiday_name, dict$holidays$holiday_name)])

g = ggplot(holidays_flex, aes(x = day_of_year, y = year(date), col = color))
g = g +
  geom_point()+
  ylab("Year")+ xlab("Holiday date")+
  scale_x_date(date_labels = "%b %d")+
  scale_color_identity(name = "", guide = "legend", labels = dict$holidays$holiday_name_wrapped, breaks = dict$holidays$color)+
  facet_grid(country ~ .)+
  theme(legend.position = "bottom")
g
```
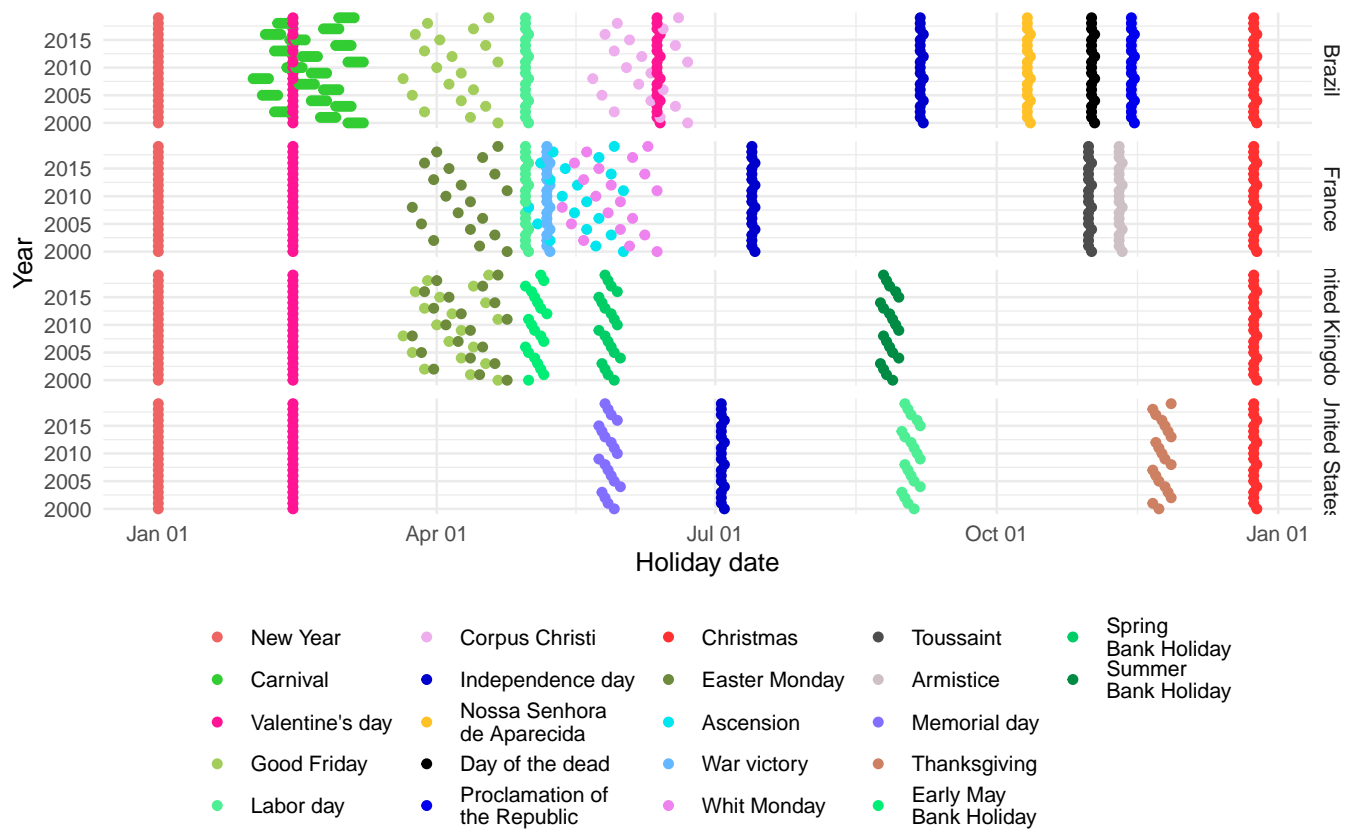
Figure 5.51: Holidays in considered countries

5.8 Varying average gestation duration and the spread of the gestation duration distribution

Here, we investigate the impact of changing the gestation duration distribution by changing the average and the standard deviation.

```r
Gs = c(37:39)*7
Gsds = c(5,10,20)

sim_births_G = foreach(ca = unique(clue_sex_agg$country_area), .combine = bind_rows) %do%{
 res = foreach(G = Gs, .combine = bind_rows) %do% {
  r = foreach(Gsd = Gsds, .combine = bind_rows) %do% {

    #cat(ca,"\n")
    #cat("\tG: ",G,"\n")
    #cat("\tGsd: ",Gsd,"\n")

    bc = "all"; st = "unprot_sex"; ac = get_age_cat(country_area = ca)

    # average daily births
    this_cat_ave_daily_births = average_daily_births_df %>% filter(country_area == ca)

    # actual births
    this_ca_births = official_birth_records %>% filter(country_area == ca) %>% arrange(date)

    # predicting sex
    j = which((sex_models_df$country_area == ca) & (sex_models_df$BC == bc) & (sex_models_df$age_cat == ac) & (sex_models_df$sex_type == st))
    this_cat_model = sex_models[[j]]$model
    this_cat_predicted_sex = predict_daily_sex_behavior(model = this_cat_model,
                                date_range = range(this_cat_ave_daily_births$date),
                                country_area = ca)



    #  model parameters
    pars = opt_par_df %>% filter(country_area == ca, BC == bc, age_cat == ac, sex_type == st, model == "C")
    alpha = pars$alpha;
    Tp = pars$Tp
    beta = pars$beta

    # simulating births
    monthly_births_C = simulated_vs_actual_monthly_birth(alpha = alpha, Tp = Tp, beta = beta, G = G, Gsd = Gsd,
                                sex_df = this_cat_predicted_sex,
                                ave_daily_birth_df = this_cat_ave_daily_births,
                                actual_monthly_birth_df = this_ca_births)
    monthly_births_C = monthly_births_C %>%
     mutate(model = "C",
         G = G,
         Gsd = Gsd,
         country_area = ca,
         sex_type = st,
         BC = bc)

    #SSR = monthly_births_C$sq_residuals %>% sum()
    #cat("\t\t\t",round(100*SSR/pars$SSR), "\n")

    return(monthly_births_C)
  }
 }
}
```

```
}
```

```
SSR_df = sim_births_G %>%
  group_by(country_area, BC, sex_type, model, G, Gsd ) %>%
  summarize(SSR = sum(sq_residuals)) %>%
  mutate(color = dict$country_area$country_area_col[match(country_area, dict$country_area$country_area)])
```

`summarise()` regrouping output by 'country_area', 'BC', 'sex_type', 'model', 'G' (override with `.groups` argument)

```
model_B_SSR = opt_par_df %>% filter(model == "B", BC == "all", sex_type == "unprot_sex") %>% select(country_area, SSR)
```

```
g = ggplot(SSR_df, aes(x = as.factor(G), y = SSR, fill = color))+
  scale_fill_identity()+
  geom_bar(stat = "identity")+
  geom_hline(data = model_B_SSR, aes(yintercept = SSR), col = "black")+
  facet_grid(country_area ~ Gsd, scale = "free_y")
print(g)
```
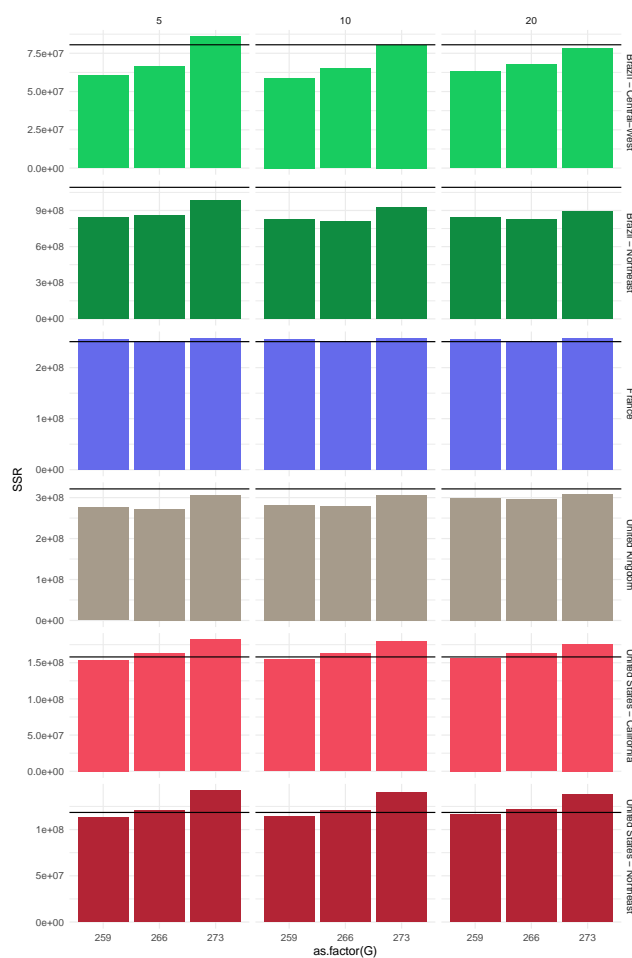

Figure 5.52: Model C residuals for various values of G (x axis) and Gsd (vertical panels). The black horizontal lines are at the values of the model B residuals.

```
for(ca in unique(SSR_df$country_area)){

  this_ca_births_G = sim_births_G %>% filter(country_area == ca, Gsd == 10)
  ref_line = min(this_ca_births_G$births)-max(this_ca_births_G$residuals)

  g = ggplot(this_ca_births_G , aes(x = year_month))
  g = g+
```

```r
  geom_segment(aes(y = ref_line, yend = ref_line + residuals, x = year_month, xend = year_month, col = sq_residuals), #
         size = 1.2)+
  geom_hline(yintercept = ref_line, col = "gray")+
  scale_color_gradient(low = "white", high = "red")+
  geom_line(aes(y = births), col = "black",size = 1.2)+
  geom_line(aes(y = sim_births), col = "indianred1")+
  facet_grid( G  ~ ., scale = "free_y")+
  guides(col = FALSE)+
  ggtitle(ca)

model_B_SSR = opt_par_df %>%
  filter(country_area == ca, model == "B", BC == "all", sex_type == "unprot_sex", Gsd == 10) %>%  select(SSR) %>% unlist()
g_bar =  ggplot(SSR_df %>% filter(country_area == ca, Gsd == 10), aes(x = 1, y = SSR))+
  coord_flip()+
  geom_bar(stat = "identity")+
  geom_hline(yintercept = model_B_SSR, col = "cadetblue1")+
  facet_grid(G ~ ., scale = "free_y")+
  xlab("")+scale_x_continuous(breaks = NULL)+
  ggtitle(ca)

g_combined = plot_grid(g,g_bar, ncol = 2, nrow = 1, rel_widths = c(3,1), align = "v" )
print(g_combined)

}
```
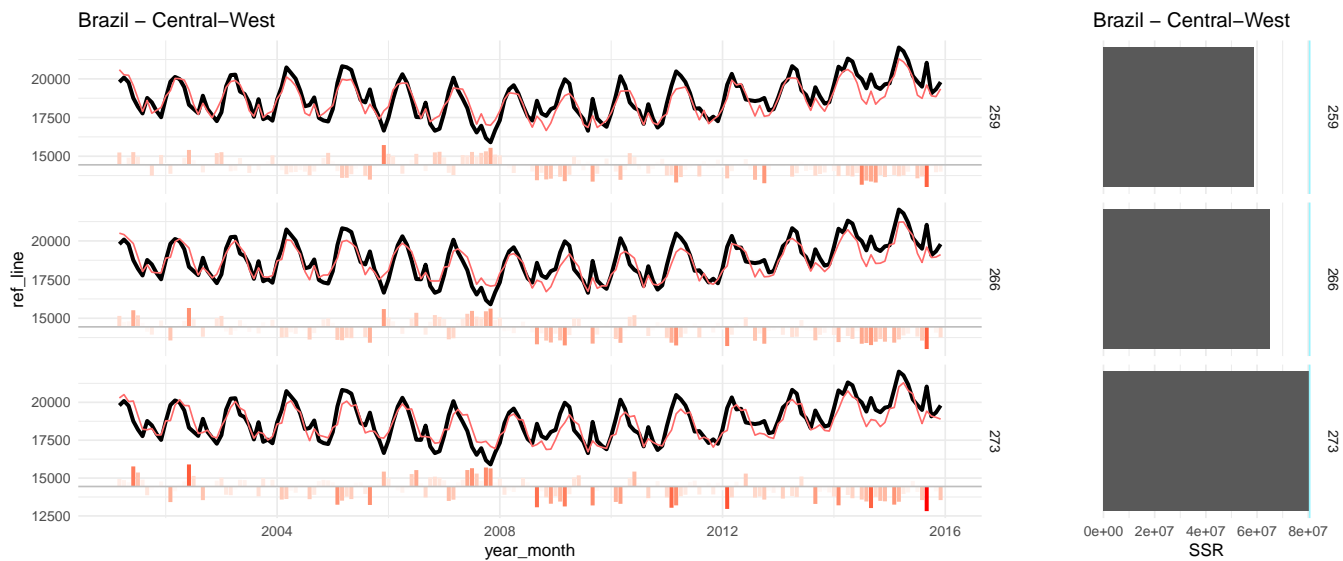

Figure 5.53: [main panel - left] Actual (black) and simulated (with model B, red) births with different average gestation durations. [side panel - right] SSR (sum of square of the residuals) for each different average gestation duration.
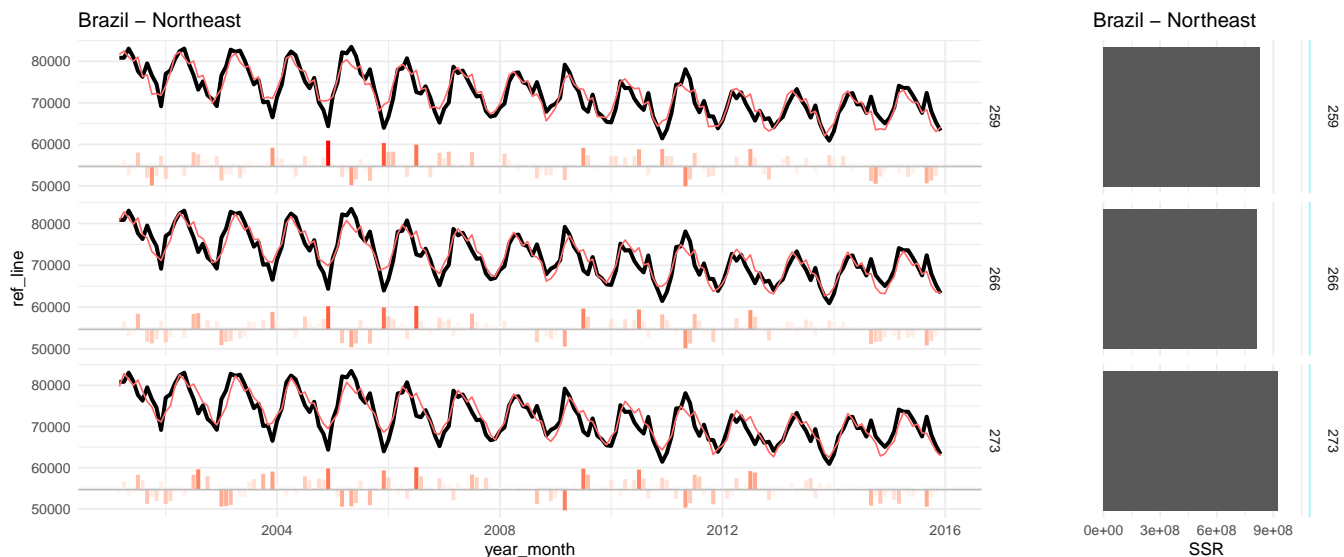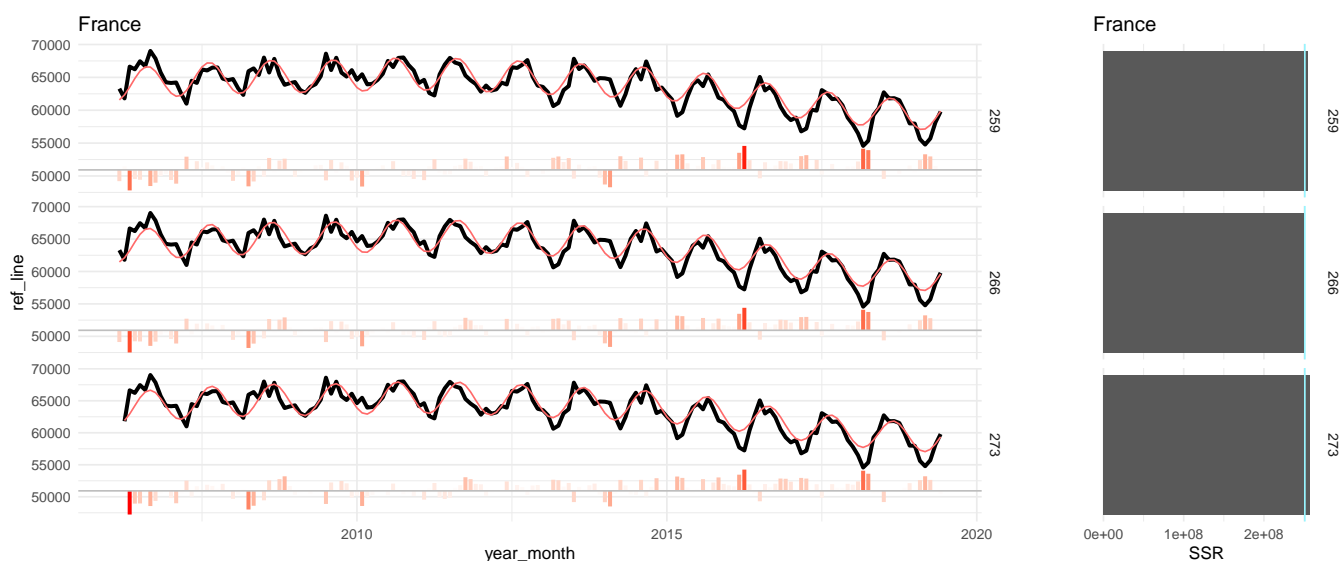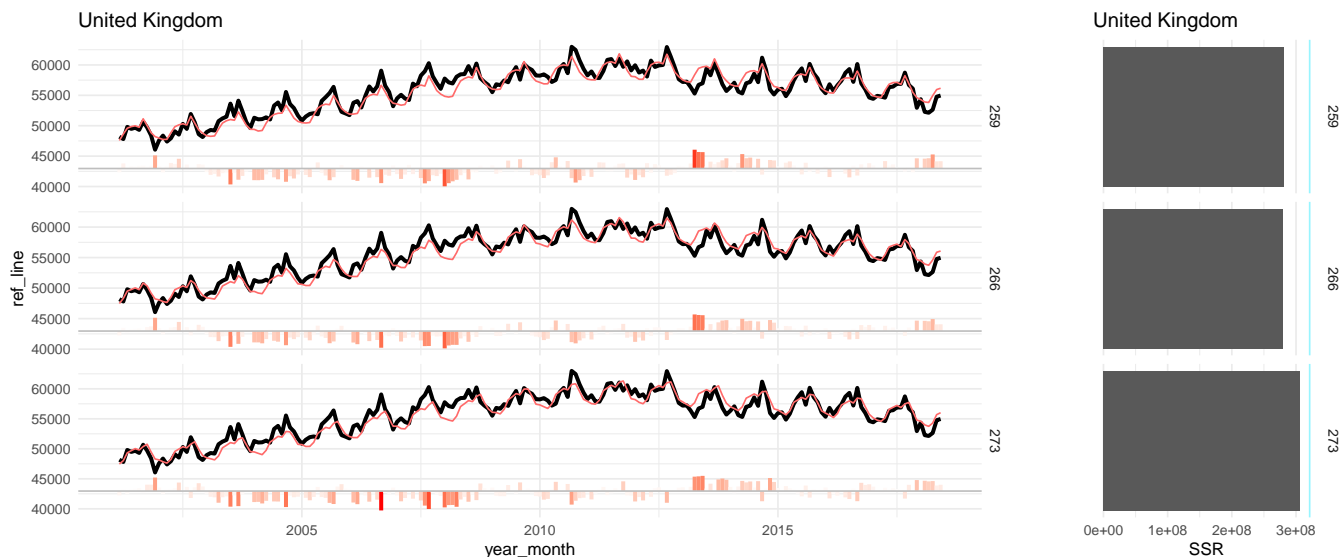
Figure 5.54: [main panel - left] Actual (black) and simulated (with model B, red) births with different average gestation durations. [side panel - right] SSR (sum of square of the residuals) for each different average gestation duration.
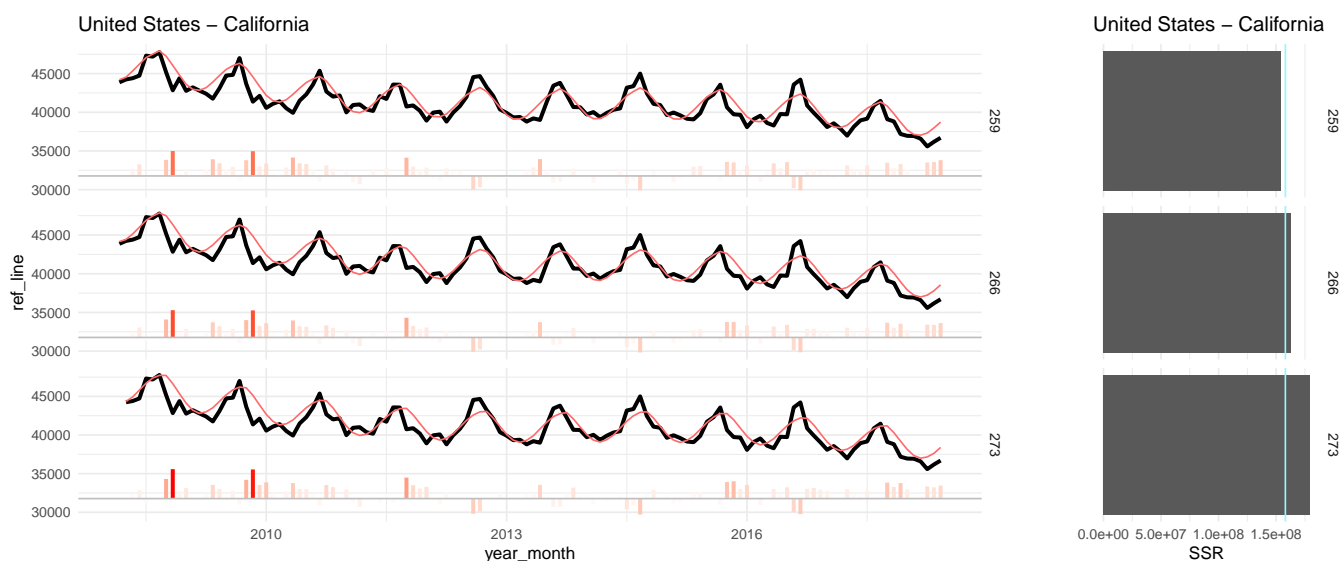

Figure 5.55: [main panel - left] Actual (black) and simulated (with model B, red) births with different average gestation durations. [side panel - right] SSR (sum of square of the residuals) for each different average gestation duration.
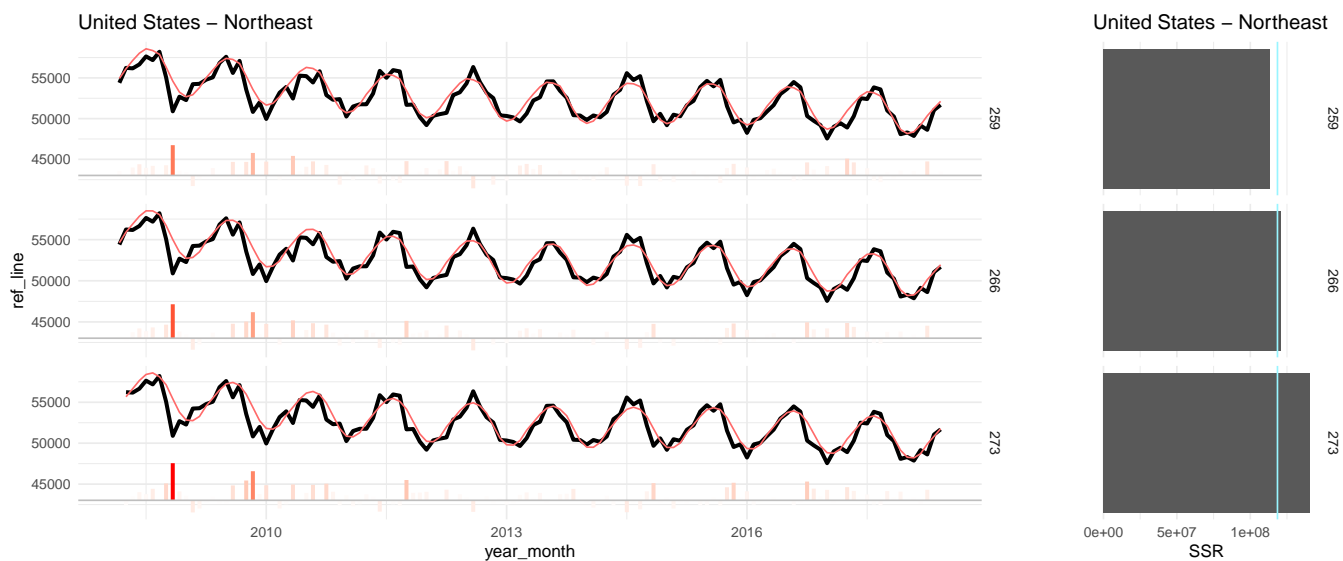
Figure 5.56: [main panel - left] Actual (black) and simulated (with model B, red) births with different average gestation durations. [side panel - right] SSR (sum of square of the residuals) for each different average gestation duration.

Figure 5.57: [main panel - left] Actual (black) and simulated (with model B, red) births with different average gestation durations. [side panel - right] SSR (sum of square of the residuals) for each different average gestation duration.

Figure 5.58: [main panel - left] Actual (black) and simulated (with model B, red) births with different average gestation durations. [side panel - right] SSR (sum of square of the residuals) for each different average gestation duration.

```
save.image(file = str_c(IO$tmp_clue,"workspace.Rdata"))
# load(file = str_c(IO$tmp_clue,"workspace.Rdata"))
```

6 Checking for reporting biases from the app users

6.1 Loading data and detrending curves

Loading the aggregated logs

```
control_features_agg = read_feather(path =str_c(IO$out_Rdata,"aggregated_control_features_counts_clue_July2017-June2019_incl.feather"))
str(control_features_agg)
```

```
## tibble [157,680 x 7] (S3: tbl_df/tbl/data.frame)
## $ country_area  : chr [1:157680] "Brazil - Central-West" "Brazil - Central-West" "Brazil - Central-West" "Brazil - Central-West" ...
## $ BC            : chr [1:157680] "F" "F" "F" "F" ...
## $ age_cat       : chr [1:157680] "<= 23" "<= 23" "<= 23" "<= 23" ...
## $ date          : Date[1:157680], format: "2017-07-01" "2017-07-01" ...
## $ n_users       : num [1:157680] 1781 1781 1781 1781 954 ...
## $ control_features: chr [1:157680] "exercise" "long_sleep" "medium_bleeding" "breast_pain" ...
## $ n             : num [1:157680] 0 0 76 63 0 0 48 20 0 0 ...
```

```
clue_sex_agg = read_feather(path =str_c(IO$out_Rdata,"aggregated_sex_counts_clue_July2017-June2019_incl.feather"))
str(clue_sex_agg)
```

```
## tibble [118,260 x 7] (S3: tbl_df/tbl/data.frame)
## $ country_area: chr [1:118260] "Brazil - Central-West" "Brazil - Central-West" "Brazil - Central-West" "Brazil - Central-West" ...
## $ BC          : chr [1:118260] "F" "F" "F" "F" ...
## $ age_cat     : chr [1:118260] "<= 23" "<= 23" "<= 23" "> 23" ...
## $ date        : Date[1:118260], format: "2017-07-01" "2017-07-01" ...
## $ n_users     : num [1:118260] 1781 1781 1781 954 954 ...
## $ sex_type    : chr [1:118260] "all_sex" "prot_sex" "unprot_sex" "all_sex" ...
## $ n           : num [1:118260] 114 47 37 89 23 43 135 53 40 92 ...
```

We do this analysis for BC = "all" and age_cat = "all" and the last year of data (we only take the last year because some features are less reported and there were not enough logs to build meaningful time-series).

```
control_features_agg = control_features_agg %>%
  filter(BC == "all", age_cat == "all", date >= as.Date("2018-07-01"))

clue_sex_agg = clue_sex_agg %>%
  filter(BC == "all", age_cat == "all", date >= as.Date("2018-07-01"), sex_type == "all_sex")
```

We compute the relatives changes and detrend the curves.

```
# relative change

control_features_agg = control_features_agg %>%
  mutate(r = n/n_users)

clue_sex_agg = clue_sex_agg %>%
  mutate(r = n/n_users)

# trend

control_features_agg = control_features_agg %>%
  arrange(country_area, BC, age_cat, control_features, date) %>%
  group_by(country_area, BC, age_cat, control_features) %>%
  mutate(t = row_number(),
         trend = predict(loess(r ~ t))) %>%
  ungroup() %>% select(-t)


clue_sex_agg = clue_sex_agg %>%
  arrange(country_area, BC, age_cat, sex_type, date) %>%
```

```
  group_by(country_area, BC, age_cat, sex_type) %>%
  mutate(t = row_number(),
      trend = predict(loess(r ~ t))) %>%
  ungroup() %>% select(-t)


# relative change

control_features_agg = control_features_agg %>% dplyr::mutate(x = r/trend)
clue_sex_agg = clue_sex_agg %>% dplyr::mutate(x = r/trend)
```

6.2   Comparison of the control feature logs with the sex logs

```
for(ca in unique(control_features_agg$country_area)){

 CF = control_features_agg %>% filter(country_area == ca)
 S = clue_sex_agg %>% filter(country_area == ca)

 g = ggplot(CF, aes(x = date, y = x, col = control_features))+
   geom_hline(yintercept = 1)+
   geom_line(data = S, aes(x = date, y = x), col = "gray50")+
   geom_line()+
   guides(col = FALSE)+
   facet_grid(control_features ~ ., scale = "free")+
   ggtitle(ca)


 J = full_join(CF, S, by = c("country_area", "BC", "age_cat", "date"), suffix = c(".CF",".S"))

 g2 = ggplot(J, aes(x = x.CF, y = x.S, col = control_features))+
   coord_fixed()+
   geom_abline(intercept = 0, slope = 1)+
   geom_point()+
   xlab("Control feature")+ylab("Sex")+
   guides(col = FALSE)+
   facet_wrap(control_features ~ .)+
   theme(strip.background = element_rect(fill = "gray80", color = "transparent"))+
   ggtitle(ca)


 g_combined = cowplot::plot_grid(g, g2, ncol = 1)
 print(g_combined)

}
```
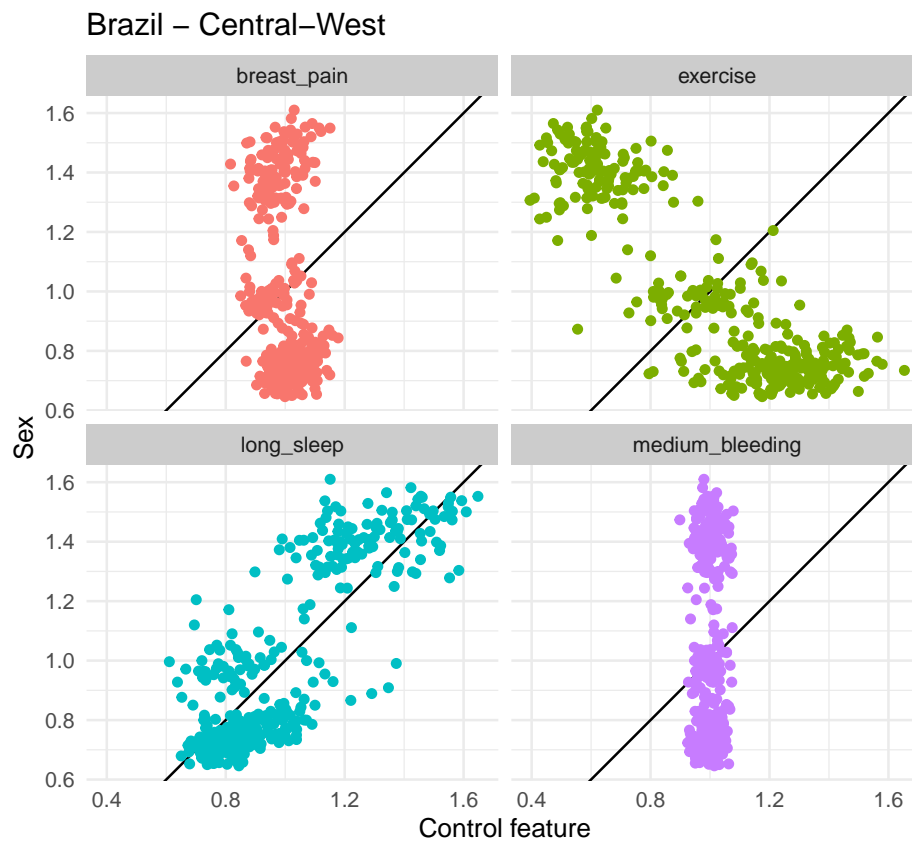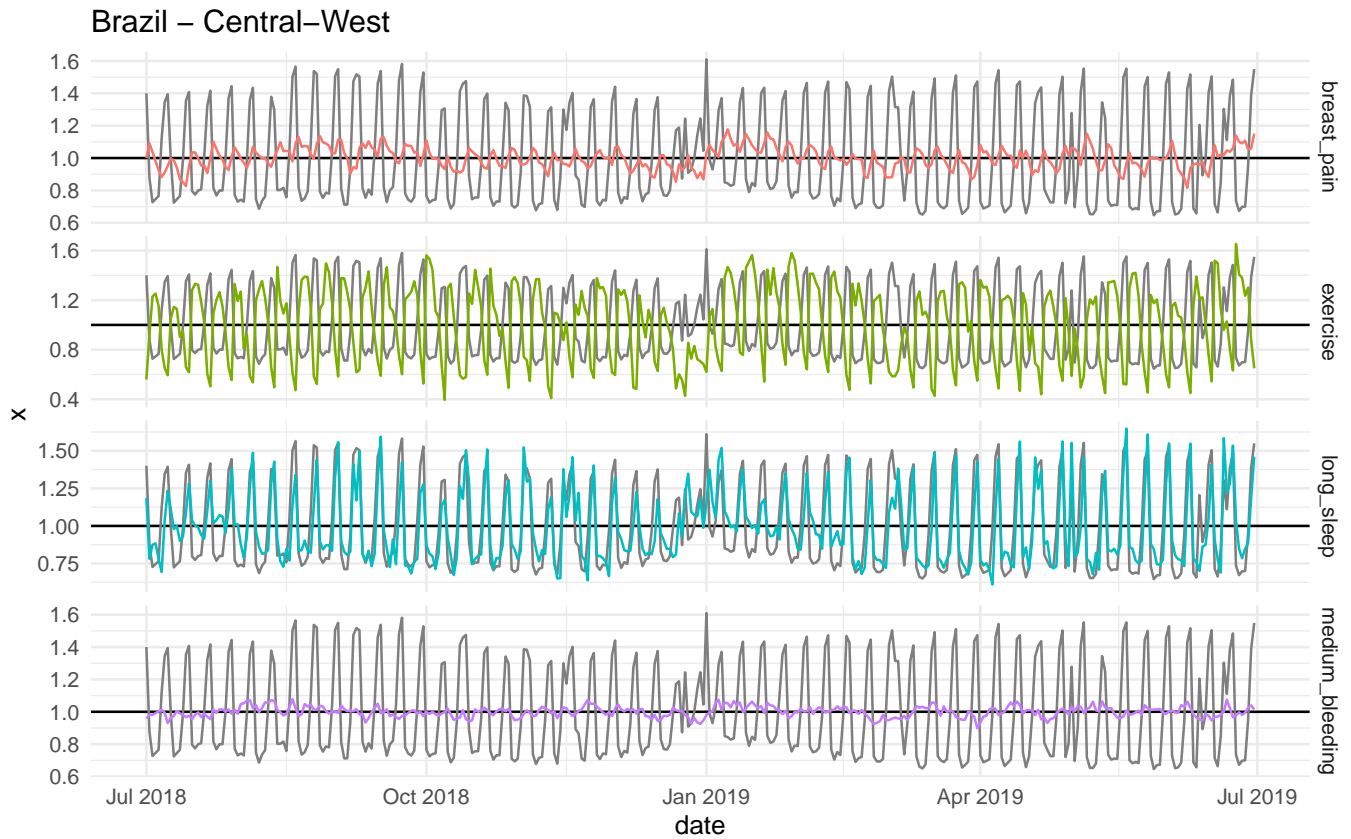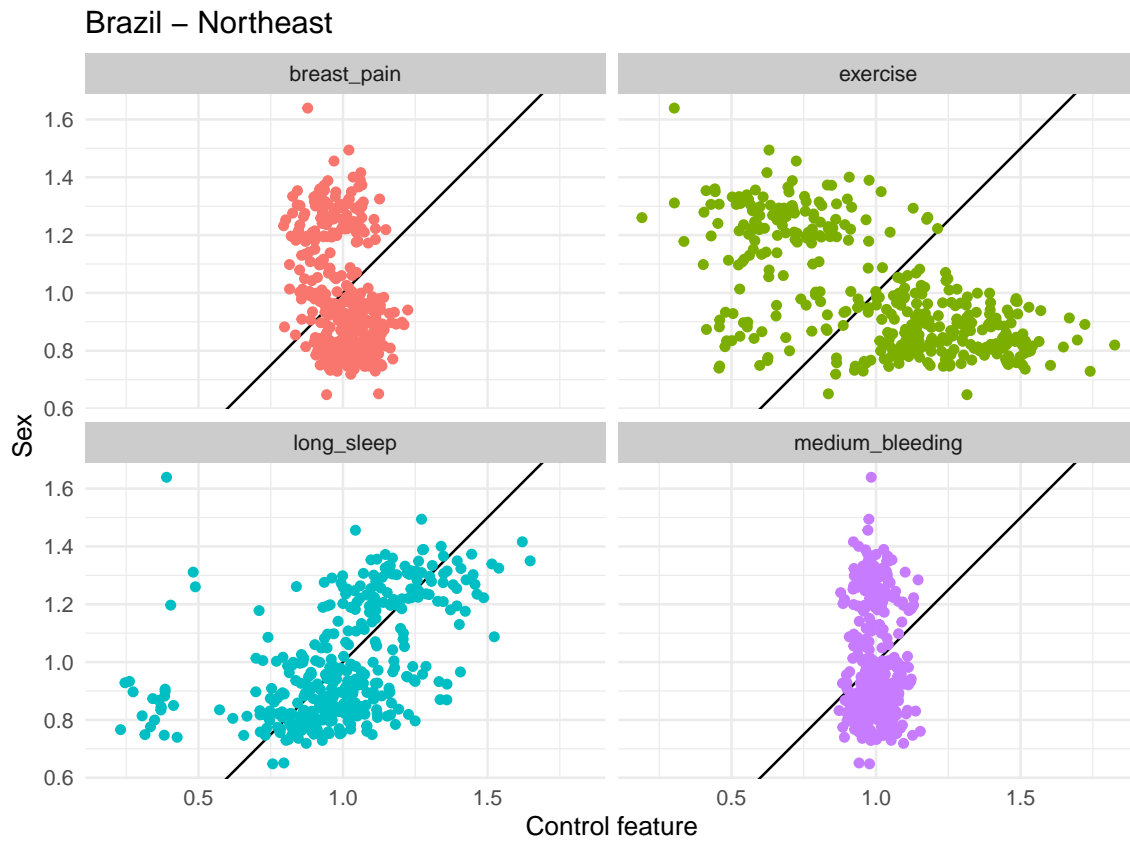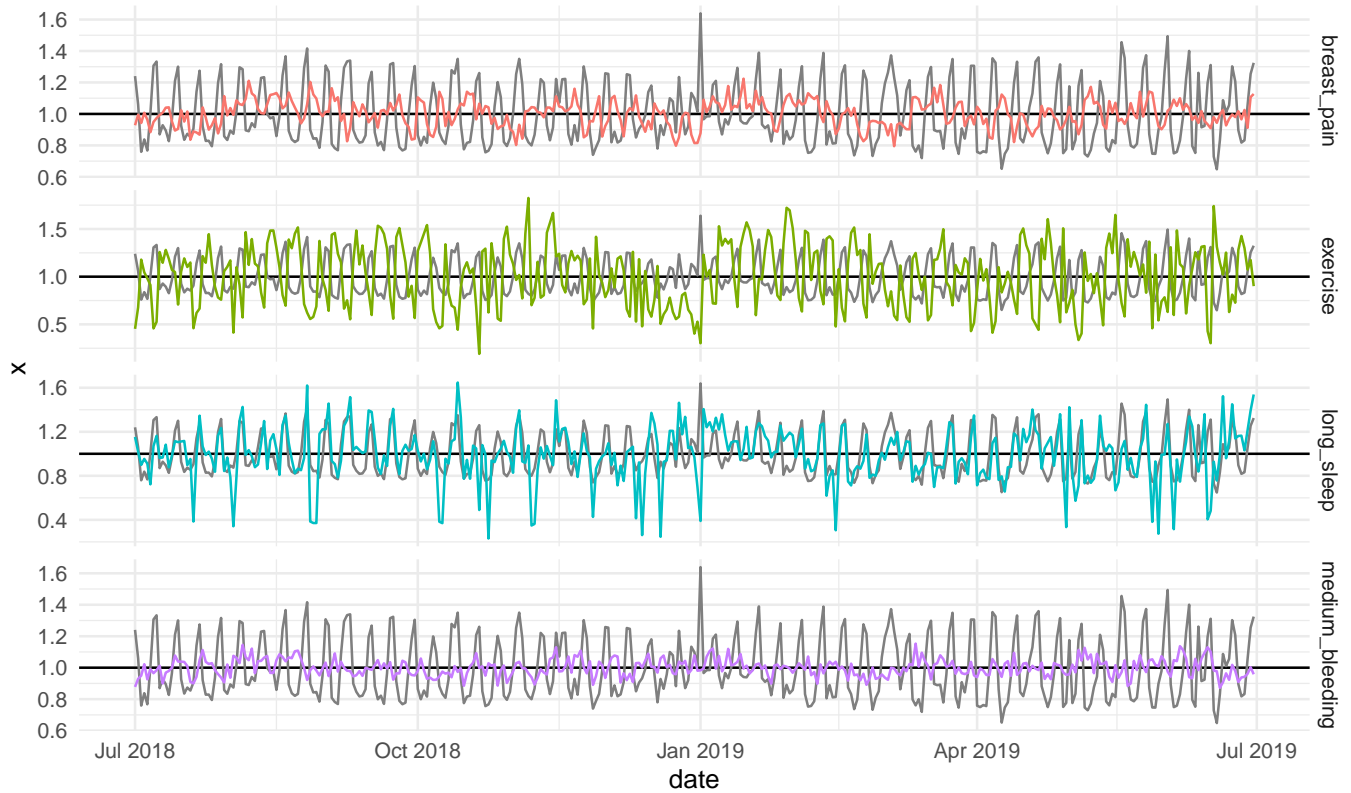
Figure 6.1: (Top) Detrended time-series for the control features (colored lines) and for sex (any sex type, gray line). (Bottom) Relative changes in the control features (x axis) vs the relative changes in sexual activity (y axis).

Figure 6.2: (Top) Detrended time-series for the control features (colored lines) and for sex (any sex type, gray line). (Bottom) Relative changes in the control features (x axis) vs the relative changes in sexual activity (y axis).
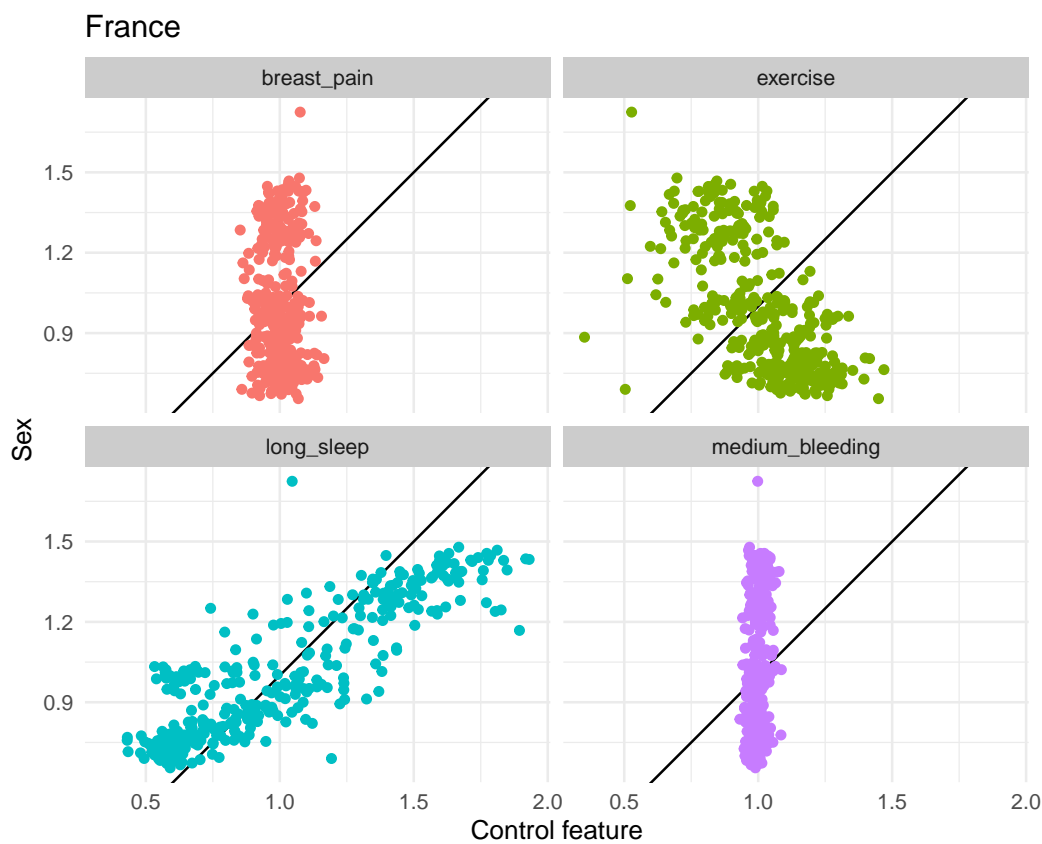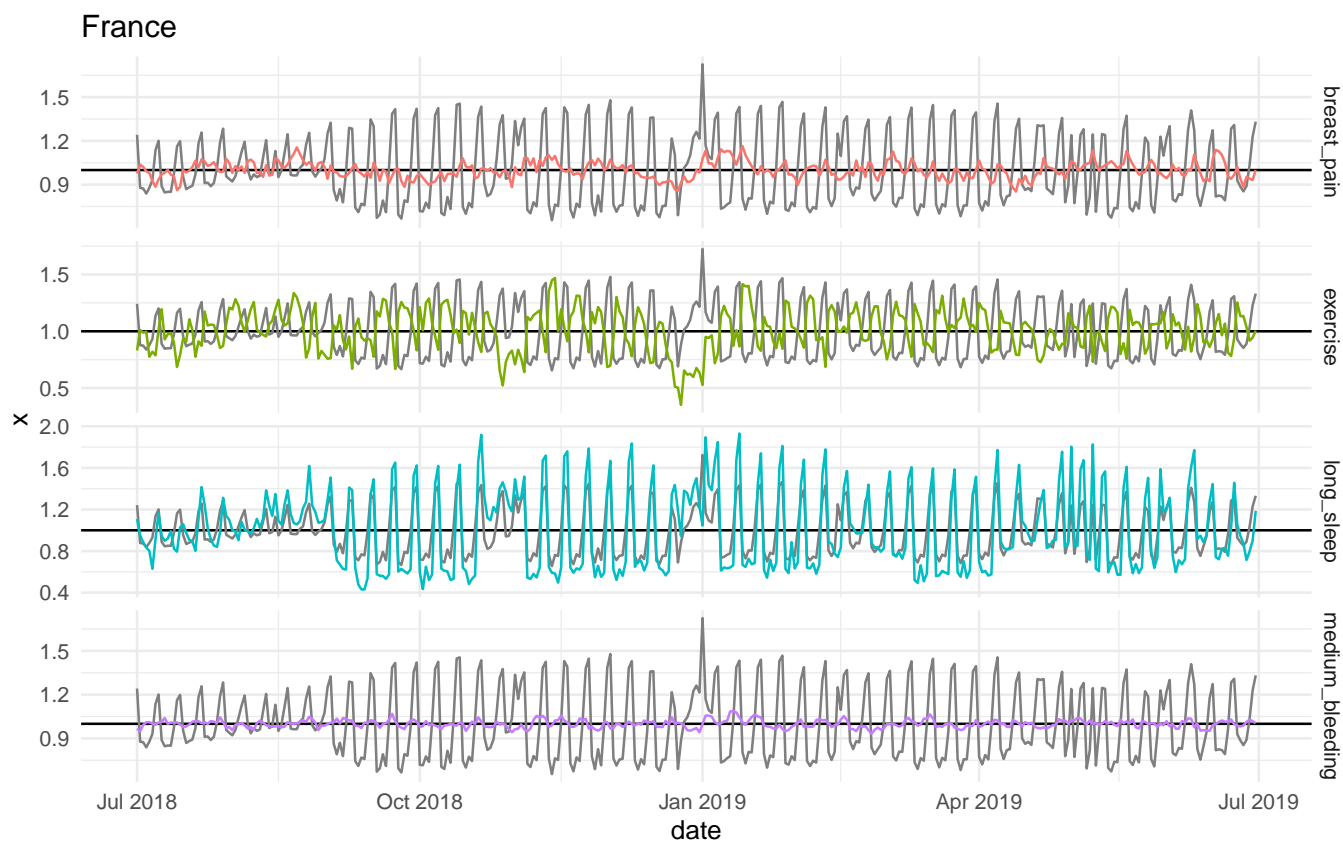
Figure 6.3: (Top) Detrended time-series for the control features (colored lines) and for sex (any sex type, gray line). (Bottom) Relative changes in the control features (x axis) vs the relative changes in sexual activity (y axis).
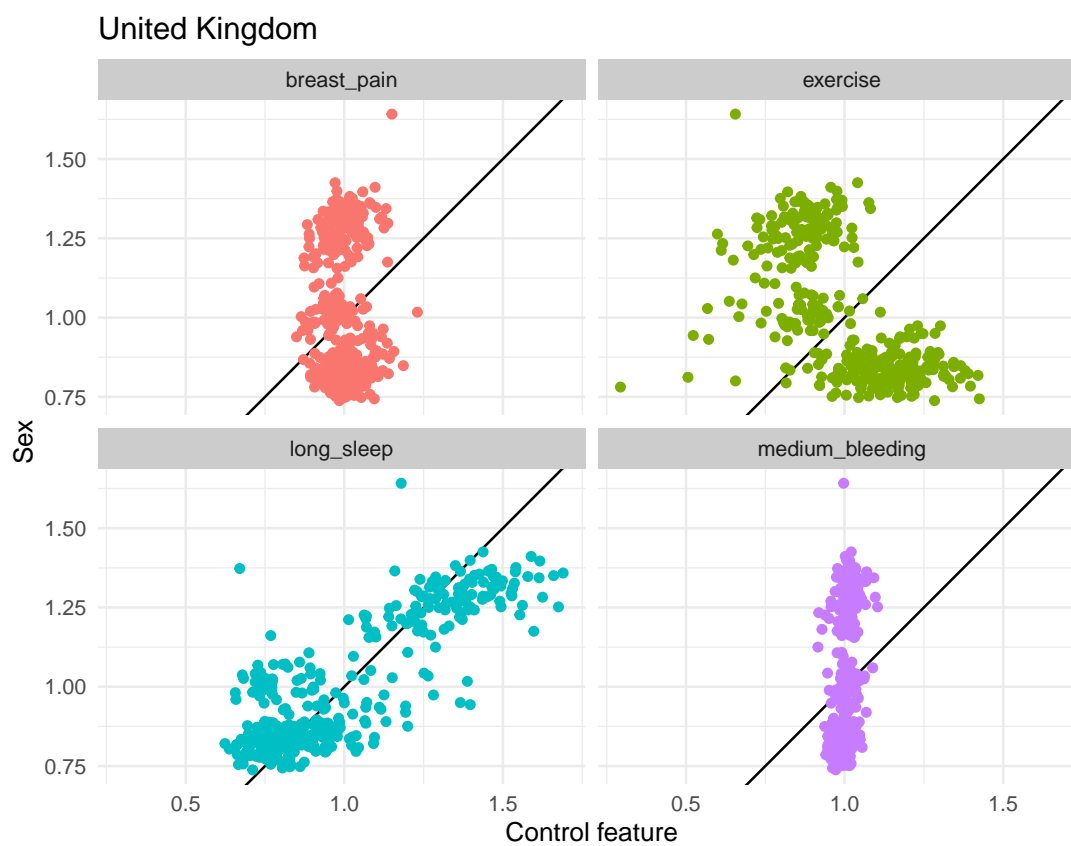
Figure 6.4: (Top) Detrended time-series for the control features (colored lines) and for sex (any sex type, gray line). (Bottom) Relative changes in the control features (x axis) vs the relative changes in sexual activity (y axis).
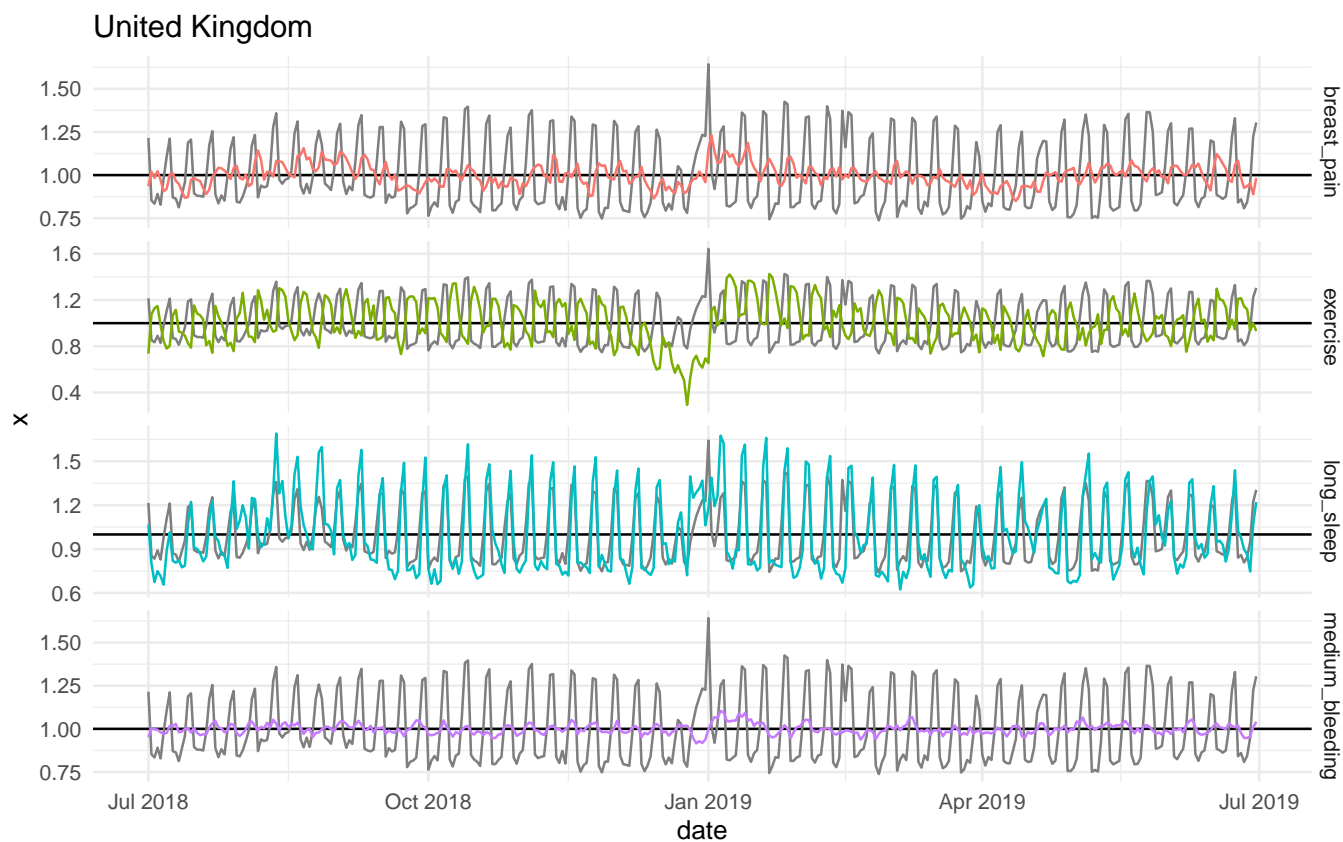
Figure 6.5: (Top) Detrended time-series for the control features (colored lines) and for sex (any sex type, gray line). (Bottom) Relative changes in the control features (x axis) vs the relative changes in sexual activity (y axis).
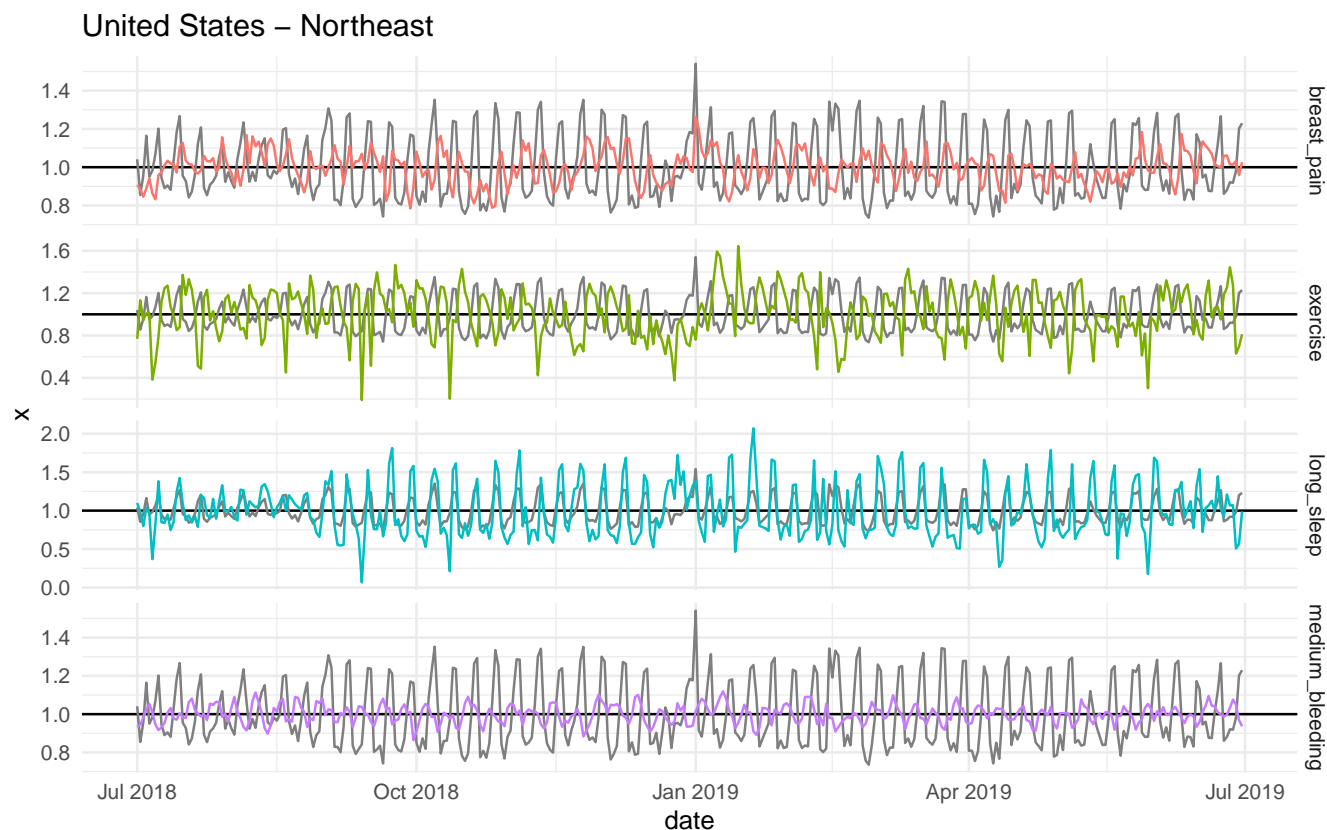
Figure 6.6: (Top) Detrended time-series for the control features (colored lines) and for sex (any sex type, gray line). (Bottom) Relative changes in the control features (x axis) vs the relative changes in sexual activity (y axis).
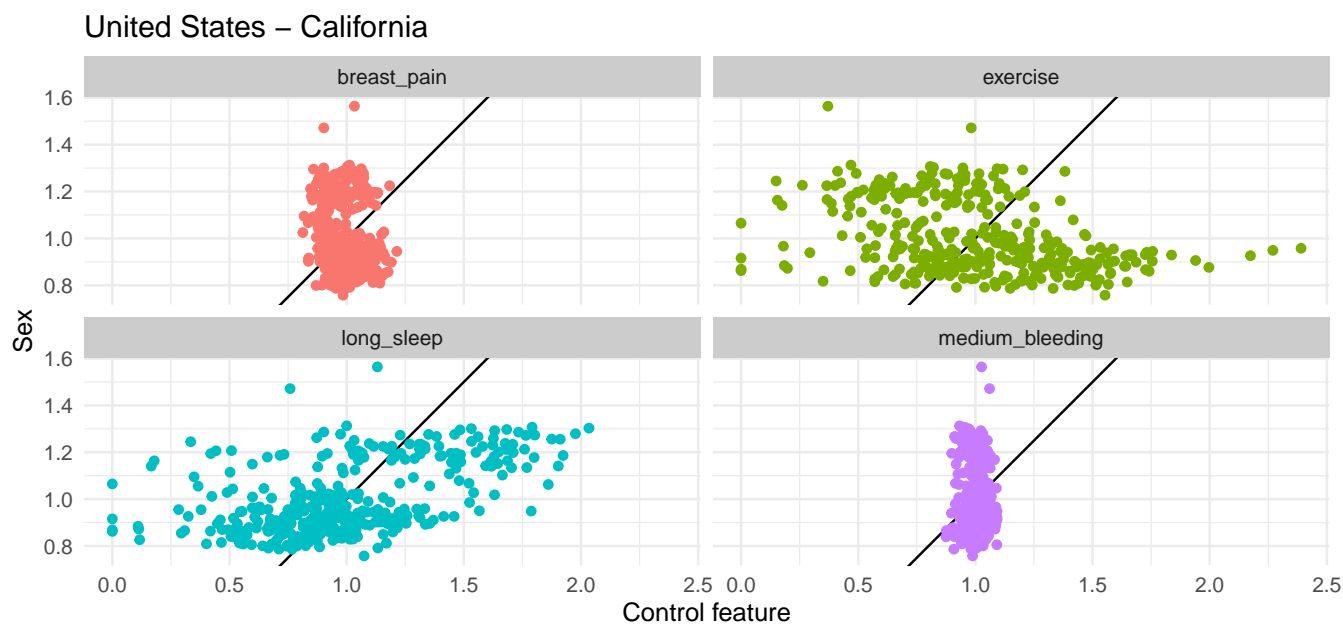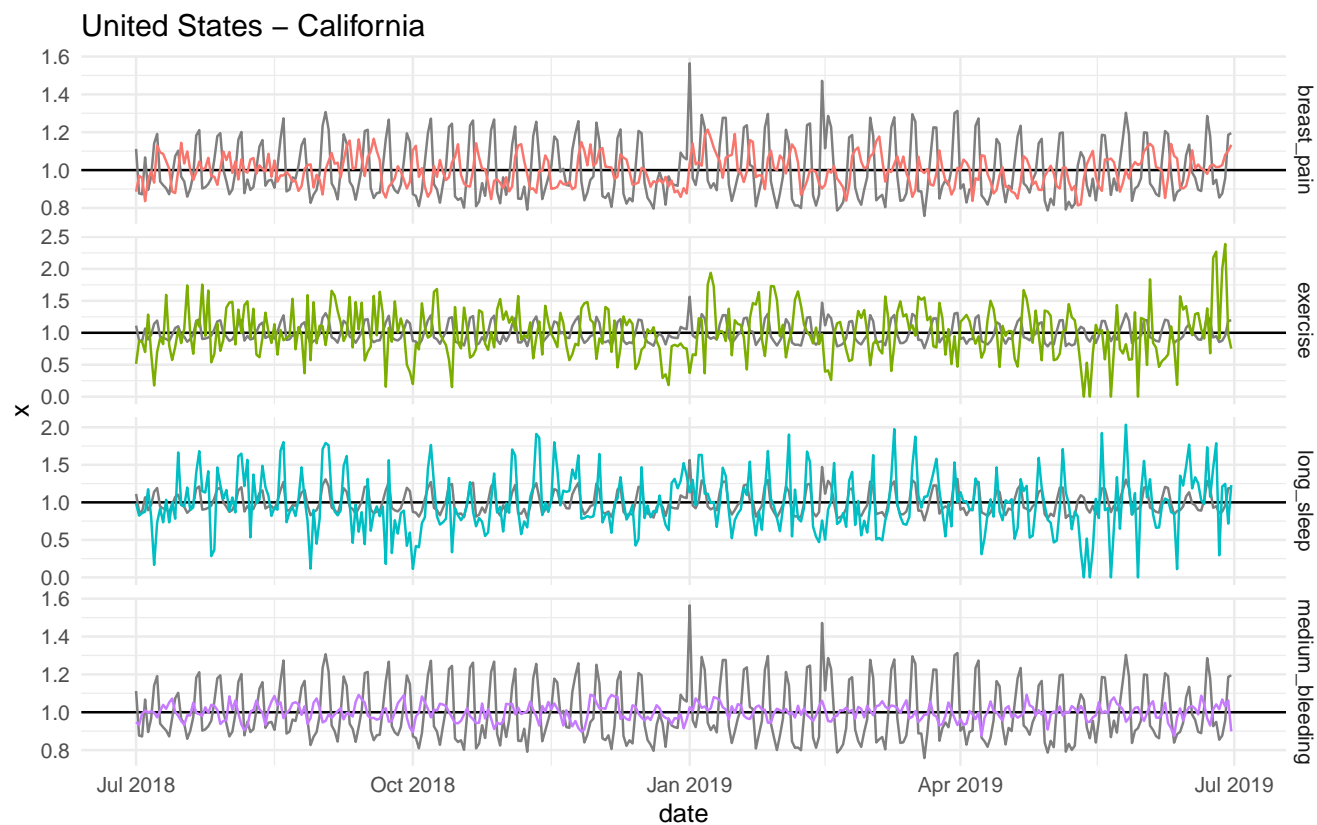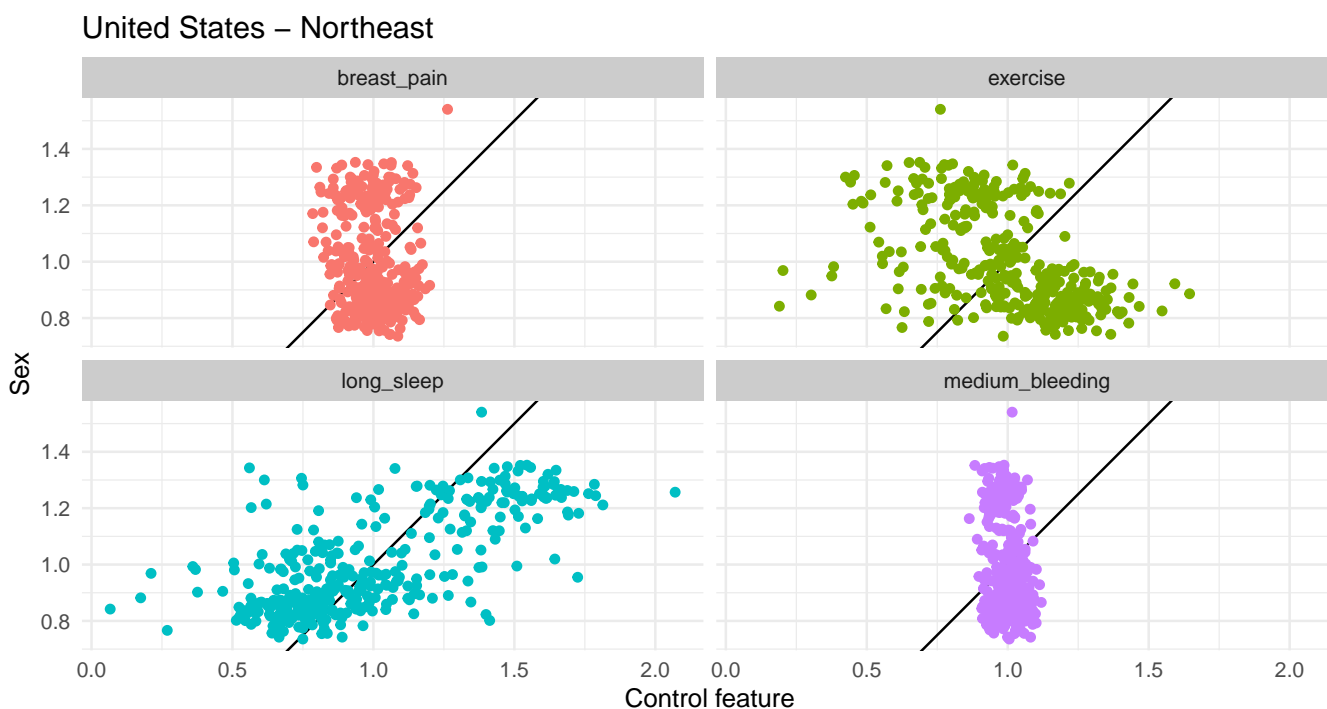
From these figures, we observe that there is a positive correlation between the sexual activity and sleeping over 9h; both of these self-reported variables have their peaks during week-ends and holidays. Reported breast pain and medium bleeding do not appear to have any temporal structure and are thus not correlated with sexual activity. Exercise is negatively correlated with sexual activity in most locations (Clue users report exercising

more during week-days than during week-ends). In California, the negative correlation is not as strong as in other locations such as the Central-West region in Brazil. Altogether, these observations are matching expected patterns and likely indicate that the temporal patterns in sexual activity are not driven by reporting biases but rather reflect actual variations in sexual activity of the app users.

7 Reproducibility receipt

Execution datetime:

[1] "2020-10-02 15:29:02 PDT"

----

sessionInfo :

R version 4.0.2 (2020-06-22)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Catalina 10.15.7
##
Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
attached base packages:
[1] parallel stats   graphics grDevices utils   datasets methods
[8] base
##
other attached packages:
[1] magrittr_1.5   forcats_0.5.0   stringr_1.4.0   purrr_0.3.4
[5] tidyr_1.1.2   tibble_3.0.3   tidyverse_1.3.0 forecast_8.13
[9] biwavelet_0.20.19 bookdown_0.20   styler_1.3.2   tictoc_1.0
[13] doParallel_1.0.15 iterators_1.0.12 foreach_1.5.0   dplyr_1.0.2
[17] plyr_1.8.6   timeDate_3043.102 lubridate_1.7.9 chron_2.3-56
[21] reshape_0.8.8   scales_1.1.1   ggmap_3.0.0   rworldmap_1.3-6
[25] sp_1.4-2   magick_2.4.0   ggpubr_0.4.0   cowplot_1.1.0
[29] kableExtra_1.2.1 hexbin_1.28.1   quantreg_5.67   SparseM_1.78
[33] gridExtra_2.3   mapdata_2.3.0   maps_3.3.0   ggthemes_4.2.0
[37] ggplot2_3.3.2   zoo_1.8-8   MASS_7.3-53   feather_0.3.5
[41] readr_1.3.1   knitr_1.29
##
loaded via a namespace (and not attached):
[1] colorspace_1.4-1 ggsignif_0.6.0   rjson_0.2.20
[4] ellipsis_0.3.1   rio_0.5.16   fs_1.5.0
[7] rstudioapi_0.11   farver_2.0.3   MatrixModels_0.4-1
[10] fansi_0.4.1   xml2_1.3.2   codetools_0.2-16
[13] jsonlite_1.7.1   spam_2.5-1   broom_0.7.0
[16] dbplyr_1.4.4   png_0.1-7   compiler_4.0.2
[19] httr_1.4.2   backports_1.1.9   assertthat_0.2.1
[22] Matrix_1.2-18   cli_2.0.2   htmltools_0.5.0
[25] tools_4.0.2   dotCall64_1.0-0   gtable_0.3.0
[28] glue_1.4.2   Rcpp_1.0.5   carData_3.0-4
[31] cellranger_1.1.0 fracdiff_1.5-1   vctrs_0.3.4
[34] nlme_3.1-149   urca_1.3-0   conquer_1.0.2
[37] lmtest_0.9-38   xfun_0.17   openxlsx_4.1.5
[40] rvest_0.3.6   lifecycle_0.2.0   rstatix_0.6.0
[43] hms_0.5.3   fields_11.4   yaml_2.2.1
[46] quantmod_0.4.17   curl_4.3   stringi_1.5.3
[49] maptools_1.0-2   tseries_0.10-47   TTR_0.24.2
[52] zip_2.1.1   RgoogleMaps_1.4.5.3 rlang_0.4.7
[55] pkgconfig_2.0.3   matrixStats_0.56.0 bitops_1.0-6
[58] evaluate_0.14   lattice_0.20-41   tidyselect_1.1.0
[61] R6_2.4.1   generics_0.0.2   DBI_1.1.0

```
## [64] pillar_1.4.6      haven_2.3.1      foreign_0.8-80
## [67] withr_2.2.0       xts_0.12.1       abind_1.4-5
## [70] nnet_7.3-14       modelr_0.1.8     crayon_1.3.4
## [73] car_3.0-9         rmarkdown_2.3    jpeg_0.1-8.1
## [76] grid_4.0.2        readxl_1.3.1     data.table_1.13.0
## [79] blob_1.2.1        reprex_0.3.0     digest_0.6.25
## [82] webshot_0.5.2     munsell_0.5.0    viridisLite_0.3.0
## [85] quadprog_1.5-8
```

References

Cavanaugh, Joseph E. 1997. "Unifying the derivations for the Akaike and corrected Akaike information criteria." *Statistics and Probability Letters* 33 (2): 201–8. https://doi.org/10.1016/s0167-7152(96)00128-9.

Delnord, Marie, Laust Mortensen, Ashna D. Hindori-Mohangoo, Béatrice Blondel, Mika Gissler, Michael R. Kramer, Jennifer L. Richards, et al. 2018. "International variations in the gestational age distribution of births: An ecological study in 34 high-income countries." *European Journal of Public Health* 28 (2): 303–9. https://doi.org/10.1093/eurpub/ckx131.

Delnord, Marie, and Jennifer Zeitlin. 2019. "Epidemiology of late preterm and early term births – An international perspective." *Seminars in Fetal and Neonatal Medicine* 24 (1): 3–10. https://doi.org/10.1016/j.siny.2018.09.001.

Jukic, A. M., D. D. Baird, C. R. Weinberg, D. R. Mcconnaughey, and A. J. Wilcox. 2013. "Length of human pregnancy and contributors to its natural variation." *Human Reproduction* 28 (10): 2848–55. https://doi.org/10.1093/humrep/det297.

MacDorman, Marian F., T. J. Matthews, Ashna D. Mohangoo, and Jennifer Zeitlin. 2014. "International comparisons of infant mortality and related factors: United States and Europe, 2010." *National Vital Statistics Reports : From the Centers for Disease Control and Prevention, National Center for Health Statistics, National Vital Statistics System* 63 (5): 1–6.