Term Project-Data Cleaning

Brynn Woolley

1. Setup and Packages

```
library(tidyverse)
library(stats19)
library(lubridate)
library(here)
library(janitor)
library(sf)

set.seed(506)
```

2. Pull Stats19 Data

```
## ---- stats19-pull-functions ----

pull_stats19_collisions <- function(years) {
  purrr::map_dfr(
    years,
    ~ get_stats19(
        year = .x,
        type = "collision",
        ask  = FALSE
      ) %>%
        mutate(year = .x)
  )
}

## Years of interest
years_multi <- 2020:2024
```

```r
## Pull data
collisions_raw_multi <- pull_stats19_collisions(years_multi)

## Pull single-year
collisions_raw_2023 <- collisions_raw_multi %>%
  filter(year == 2023)
```

3. Filter to York

```r
## ---- york-filter-function ----

filter_to_york <- function(collisions_df) {
  collisions_df %>%
    filter(local_authority_ons_district == "York")
}

collisions_york_multi <- filter_to_york(collisions_raw_multi)
collisions_york_2023  <- filter_to_york(collisions_raw_2023)

## Sanity checks
collisions_york_multi %>%
  count(year)
```

```
# A tibble: 5 x 2
   year     n
  <int> <int>
1  2020   267
2  2021   299
3  2022   297
4  2023   314
5  2024   304
```

```r
collisions_york_multi %>%
  summarise(
    min_date = min(date),
    max_date = max(date),
    n_collisions = n()
  )
```

```
# A tibble: 1 x 3
  min_date   max_date   n_collisions
  <date>     <date>            <int>
1 2020-01-04 2024-12-30         1481
```

```r
collisions_york_2023 %>%
  summarise(
    min_date = min(date),
    max_date = max(date),
    n_collisions = n()
  )
```

```
# A tibble: 1 x 3
  min_date   max_date   n_collisions
  <date>     <date>            <int>
1 2023-01-01 2023-12-28          314
```

4. Initial Data Integrity Checks

- Row counts
- Missing coordinates
- CRS validity

```r
## Crashes by month (multi-year)
collisions_york_multi %>%
  mutate(month = floor_date(date, unit = "month")) %>%
  count(year, month)
```

```
# A tibble: 60 x 3
    year month          n
   <int> <date>     <int>
 1  2020 2020-01-01    34
 2  2020 2020-02-01    24
 3  2020 2020-03-01    23
 4  2020 2020-04-01     6
 5  2020 2020-05-01    10
 6  2020 2020-06-01    16
 7  2020 2020-07-01    22
 8  2020 2020-08-01    26
 9  2020 2020-09-01    26
10  2020 2020-10-01    24
# i 50 more rows
```

```r
## Crashes by month (2023 only)
collisions_york_2023 %>%
  mutate(month = floor_date(date, unit = "month")) %>%
  count(month)
```

```
# A tibble: 12 x 2
   month          n
   <date>     <int>
 1 2023-01-01    30
 2 2023-02-01    27
 3 2023-03-01    29
 4 2023-04-01    22
 5 2023-05-01    26
 6 2023-06-01    32
 7 2023-07-01    33
 8 2023-08-01    25
 9 2023-09-01    19
10 2023-10-01    26
11 2023-11-01    28
12 2023-12-01    17
```

```r
## Check Missingness
check_missingness <- function(df) {
  df %>%
    select(
      road_type,
      speed_limit,
      junction_detail,
      light_conditions,
      weather_conditions,
      road_surface_conditions
    ) %>%
    summarise(across(everything(), ~ mean(is.na(.))))
}

## Apply checks
check_missingness(collisions_york_multi)
```

```
# A tibble: 1 x 6
  road_type speed_limit junction_detail light_conditions weather_conditions
      <dbl>       <dbl>           <dbl>            <dbl>              <dbl>
```

```
1         0           0               0               0                   0
# i 1 more variable: road_surface_conditions <dbl>
```

```
check_missingness(collisions_york_2023)
```

```
# A tibble: 1 x 6
  road_type speed_limit junction_detail light_conditions weather_conditions
      <dbl>       <dbl>           <dbl>            <dbl>               <dbl>
1         0           0               0                0                   0
# i 1 more variable: road_surface_conditions <dbl>
```

5. Feasibility EDA

- Crashes per month
- Overall crash volume
- Candidate covariate availability

```
## Monthly crash counts by year (multi-year)
collisions_york_multi %>%
  mutate(month = floor_date(date, "month")) %>%
  count(year, month)
```

```
# A tibble: 60 x 3
    year month          n
   <int> <date>     <int>
 1  2020 2020-01-01    34
 2  2020 2020-02-01    24
 3  2020 2020-03-01    23
 4  2020 2020-04-01     6
 5  2020 2020-05-01    10
 6  2020 2020-06-01    16
 7  2020 2020-07-01    22
 8  2020 2020-08-01    26
 9  2020 2020-09-01    26
10  2020 2020-10-01    24
# i 50 more rows
```

```
## Overall crash volume
collisions_york_multi %>%
  count(year)
```

```
# A tibble: 5 x 2
   year     n
  <int> <int>
1  2020   267
2  2021   299
3  2022   297
4  2023   314
5  2024   304
```

```r
collisions_york_2023 %>%
  summarise(n_collisions = n())
```

```
# A tibble: 1 x 1
  n_collisions
         <int>
1          314
```

```r
## Candidate covariate availability
check_missingness(collisions_york_multi)
```

```
# A tibble: 1 x 6
  road_type speed_limit junction_detail light_conditions weather_conditions
      <dbl>       <dbl>           <dbl>            <dbl>               <dbl>
1         0           0               0                0                   0
# i 1 more variable: road_surface_conditions <dbl>
```

After exploratory analysis of road collisions in York from 2020–2024, I found consistent monthly crash volumes across all years, with counts generally ranging between approximately 10 and 35 crashes per month. This supports aggregation at the monthly level while preserving temporal variation. Key roadway and environmental variables—including road type, speed limit, junction detail, lighting, weather, and road surface—exhibit complete or near-complete coverage across years. Spatial coordinates are available in OSGB format for all York collisions, enabling consistent spatial aggregation using hexagonal grid cells.

For reference, the 2023 subset contains 314 collisions distributed across all 12 months.

```r
## ---- york-sf-conversion ----

make_york_sf <- function(collisions_df) {
  collisions_df %>%
    filter(
```

```
      !is.na(location_easting_osgr),
      !is.na(location_northing_osgr)
    ) %>%
    st_as_sf(
      coords = c("location_easting_osgr", "location_northing_osgr"),
      crs = 27700,
      remove = FALSE
    )
}

collisions_york_multi_sf <- make_york_sf(collisions_york_multi)
collisions_york_2023_sf  <- make_york_sf(collisions_york_2023)
```

6. Cleaning Decisions

6.1 Administrative Filtering

- The analysis is restricted to collisions occurring within the City of York. Collisions were identified using the `local_authority_ons_district` field, which contains authority names in the 2023 STATS19 release.
- The `local_authority_district` variable was not used, as it is deprecated and uninformative in the current dataset.

6.2 Spatial Coordinates

- Spatial locations were constructed using OSGB eastings and northings (`location_easting_osgr`, `location_northing_osgr`), which are populated for all York collisions.
- Longitude and latitude fields were not used, as they are unpopulated in the 2023 STATS19 data.
- All spatial data were assigned the British National Grid coordinate reference system (EPSG:27700).

6.3 Environmental Covariates

- All candidate road and environmental covariates exhibited complete coverage in the York subset and were retained.
- To ensure adequate cell counts after spatial and temporal aggregation, environmental variables with many low-frequency categories were collapsed into interpretable binary indicators.

```r
## ---- environmental-covariates ----

add_environmental_bins <- function(sf_df) {
  sf_df %>%
    mutate(
      light_bin = if_else(
        light_conditions == "Daylight",
        "Daylight",
        "Dark"
      ),
      weather_bin = case_when(
        weather_conditions == "Fine no high winds" ~ "Fine",
        weather_conditions == "Data missing or out of range" ~ NA_character_,
        TRUE ~ "Other"
      )
    )
}

collisions_york_multi_sf <- collisions_york_multi_sf %>%
  add_environmental_bins()

collisions_york_2023_sf <- collisions_york_2023_sf %>%
  add_environmental_bins()
```

6.4 Roadway Covariates

- The `road_type` variable in the STATS19 data describes roadway configuration (e.g., single carriageway, roundabout) rather than hierarchical road class.
- In the York subset, `road_type` was highly skewed, with the majority of collisions occurring on single carriageways and sparse representation in other categories.
- To retain roadway context while avoiding sparse categories and collinearity after spatial–temporal aggregation, road types were collapsed into a small number of functional categories.
- The resulting categorical variable reflects roadway structure rather than importance and was used in place of finer-grained road classifications in the aggregated models.

```r
## ---- roadway-covariates ----

add_road_categories <- function(sf_df) {
  sf_df %>%
    mutate(
```

```
      road_cat = case_when(
        road_type == "Single carriageway" ~ "standard",
        road_type == "Dual carriageway"   ~ "divided",
        road_type %in% c("Roundabout", "Slip road") ~ "complex",
        road_type %in% c("One way street", "Unknown") ~ "other"
      ),
      road_cat = factor(
        road_cat,
        levels = c("standard", "divided", "complex", "other")
      )
    )
}

collisions_york_multi_sf <- collisions_york_multi_sf %>%
  add_road_categories()

collisions_york_2023_sf <- collisions_york_2023_sf %>%
  add_road_categories()
```

7. Save Cleaned Dataset

```
## ---- save-cleaned-data ----

saveRDS(
  collisions_york_2023_sf,
  here::here("Term Project", "Data", "collisions_york_2023_sf.rds")
)

saveRDS(
  collisions_york_multi_sf,
  here::here("Term Project", "Data", "collisions_york_2020_2024_sf.rds")
)
```

GitHub Link

- Repo: https://github.com/brynnwoolley/STATS-506#