Introduction to Quantitative Research Methods Seminar

Week 4

James Rice

James.rice@ucl.ac.uk

Thursday, February 5, 2026

Week 4 focus (recap: sourcing + preparing data)

- In real projects you usually need to **source** your own **data** (e.g. from national statistics agencies).
- This week's workflow:
  - **Download** an official dataset (ONS mid-year population estimates)
  - **Extract + select** the relevant tab/columns in Excel
  - **Save as CSV** and **read into R**
  - **Rename columns** into analysis-friendly names (e.g. `lad20_name`, `pop20`)
- Then we apply Week 1–3 skills:
  - **Subset/filter** (Base R `[rows, cols]` and `dplyr::filter()`)
  - **Describe + visualise** distributions (summary stats, boxplots, histograms)

Why Birmingham vs Manchester (in an ONS dataset)?

- The ONS mid-year estimates are reported for small areas (**MSOAs**), and each MSOA belongs to a **Local Authority District** (LAD).
- The dataset includes LAD identifiers for different boundary vintages (e.g. **2020 boundaries**), which is why we filter on `lad20_name`.
- Comparing Birmingham vs Manchester is a useful example because:
  - it demonstrates how to subset official data by an administrative geography
  - it highlights that we usually compare **distributions** (not just one number)
  - it motivates tidy, consistent variable names before analysis

Contents	Table SAPE23DT4: Mid-2020 Population Estimates for Middle Layer Super Output Areas in England and Wales by Single Year of Age and Sex - Supporting Information
Terms and conditions	Terms and conditions
Notes and definitions	Notes and definitions
Mid-2020 Persons	The number of persons by single year of age and sex for Middle Layer Super Output Areas in England and Wales, mid-2020.
Mid-2020 Males	The number of males by single year of age and sex for Middle Layer Super Output Areas in England and Wales, mid-2020.
Mid-2020 Females	The number of females by single year of age and sex for Middle Layer Super Output Areas in England and Wales, mid-2020.
Related publications	Provides links to further population statistics & related publications

Key variables (after preparing the ONS download)

The ONS download starts as an `.xlsx` file with multiple tabs and metadata; we manually select the **Mid-2020 Persons** data and save it as `midyear2020.csv`.

After importing into R, we use clear column names so the analysis is readable:

- `msoa_code`, `msoa_name`: small area identifiers
- `lad20_name`: Local Authority District name (2020 boundaries; e.g. Birmingham, Manchester)
- `pop20`: total population estimate ("All Ages") for each MSOA

```
> head(midyear)
# A tibble: 6 × 7
  msoa_code msoa_name      lad18_code lad18_name lad20_code lad20_name pop20
  <chr>     <chr>          <chr>      <chr>      <chr>      <chr>      <dbl>
1 E02002483 Hartlepool 001 E06000001  Hartlepool E06000001  Hartlepool 10332
2 E02002484 Hartlepool 002 E06000001  Hartlepool E06000001  Hartlepool 10440
3 E02002485 Hartlepool 003 E06000001  Hartlepool E06000001  Hartlepool  8165
4 E02002487 Hartlepool 005 E06000001  Hartlepool E06000001  Hartlepool  5174
5 E02002488 Hartlepool 006 E06000001  Hartlepool E06000001  Hartlepool  5894
6 E02002489 Hartlepool 007 E06000001  Hartlepool E06000001  Hartlepool  7638
>
```

Quick recap: sourcing + preparing data (before analysing)

- Official data often arrive in formats that are **not analysis-ready** (zipped files, multiple Excel tabs, metadata sheets).
- Before opening R, we need to:
  - extract the `.zip` (if needed)
  - copy only the relevant columns/rows from the correct tab (avoid empty rows/whitespace)
  - save as a clean `.csv` (comma-separated) and sanity-check in a text editor
- After importing, column names may differ from Excel (punctuation/spaces are normalised), so we often **rename** to tidy names.
- Once the data are tidy, we can compute summary statistics (`mean()`, `median()`, `sd()`) and make plots.

Example R code: Import + quick checks

(Recap from the sourcing/prep workflow: once you have saved a clean `.csv`, import it and do quick checks.)

```
library(tidyverse)

midyear <- read_csv("midyear2020.csv")

glimpse(midyear)
head(midyear)
summary(midyear$pop20)
```

- `glimpse()` tidyverse function that confirms rows/columns + data types.
- `summary()` helps show basic features of the data.

Base R subsetting: select columns with `[, c(1,5)]`

Square brackets subset a dataframe as `[rows, cols]`. To keep specific columns by position:

```
# keep all rows, but only columns 1 and 5
midyear_cols <- midyear[, c(1, 5)]

# or keep a range of columns
midyear_1to5 <- midyear[, 1:5]

# check what you kept
names(midyear_cols)
head(midyear_cols)


#Using Tidyverse
midyear_1to5 <- midyear %>% dplyr::select(1:5)
```

- If you want to keep columns by name instead, use `midyear[, c("lad20_name", "pop20")]` .

Example R code: Select + rename variables

Real downloads (raw data) often have extra columns; we keep what we need and rename for readability.

```
midyear_small <- midyear %>%
  select(msoa_code, msoa_name, lad20_name, pop20) %>%
  rename(pop20_numeric = pop20)
```

- If your file uses different column names, update `select(...)` / `rename(...)`.
- Always check the names with `names(midyear)` after import.

Seminar task: Local Authority populations

Using the **cleaned midyear population dataset**, complete the following tasks:

- Create a new object that only contains data for the Local Authority District (2020 boundaries) of **Manchester**.
- Create a new object that only contains data for the Local Authority District (2020 boundaries) of **Birmingham**.

For *both* new datasets:

- Calculate the **mean**, **median**, and **standard deviation** of `pop20_numeric`.
- Create a **boxplot** of `pop20_numeric` using `ggplot2`.
- Create a **histogram** of `pop20_numeric` using `ggplot2`. Select a bin width that you think is appropriate.

Seminar questions: interpretation

1. Compare the descriptive statistics (mean, median, standard deviation) for Birmingham and Manchester.

   What do these statistics tell you about:
   - typical population size?
   - variability in population across MSOAs?
   - skewness or the presence of extreme values?
2. Why did you select the bin width you used for your histograms?
3. Compare the histograms for Birmingham and Manchester. What do they suggest about differences in the **population distribution** of the two Local Authority Districts?

Step 0: Packages + load data (script template)

```
library(tidyverse)
library(ggplot2)
library(gridExtra)

rm(list = ls())
gc()
dev.off()

setwd("~")

# load the super cleaned data
midyear <- read_csv('midyear2020.csv')
```

- If your `.csv` is in a different folder, adjust `setwd(...)` (or provide a full file path to `read_csv`).
- First check column names with `names(midyear)` / `glimpse(midyear)`.

Step 1: Subset by Local Authority (Base R vs tidyverse)

```
# Option 1: Base R
midyr_birmingham_baser <- midyear[midyear$lad20_name=="Birmingham",]

# Option 2: Tidyverse
midyr_birmingham_tidyv <- filter(midyear, lad20_name=="Birmingham")

# Option 1: Base R
midyr_manchester_baser <- midyear[midyear$lad20_name=="Manchester",]

# Option 2: Tidyverse
midyr_manchester_tidyv <- filter(midyear, lad20_name=="Manchester")

filter_bir_manc_baser <- midyear[
  midyear$lad20_name=="Birmingham" | midyear$lad20_name=="Manchester",
]
```

Step 2: Descriptive statistics (report these)

```
birmingham_result <- midyr_birmingham_baser %>%
  summarize(mean = mean(pop20), sd = sd(pop20), median = median(pop20))
View(birmingham_result)

manchester_result <- midyr_manchester_baser %>%
  summarize(mean = mean(pop20), sd = sd(pop20), median = median(pop20))
View(manchester_result)
```

- If your variable is named `pop20_numeric` in your cleaned file, swap `pop20` for `pop20_numeric`.

Birmingham

	mean	sd	median
1	8640.341	2118.727	8267.5

Manchester

	mean	sd	median
1	9749.842	2463.864	9283

Step 3: Dual boxplot (same y-scale, fair comparison)

```
p.boxplot.all <- ggplot(filter_bir_manc_baser, aes(x=lad20_name, y=pop20)) +
  geom_boxplot() +
  ggtitle("Birmingham & Manchester") +
  xlab("Area Name") +
  ylab("Population") +
  theme_classic()
```

- For other comparisons: change the filter object and update the plot title.

Step 4: Histograms (choose binwidth deliberately)

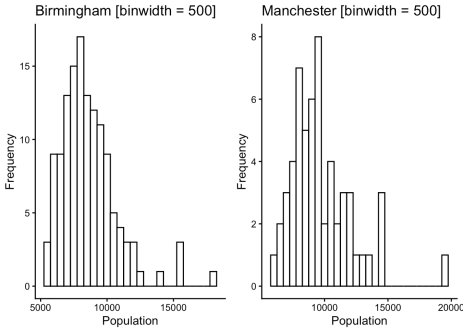
```
hist1 <- ggplot(midyr_birmingham_baser, aes(x=pop20)) +
  geom_histogram(binwidth = 500, fill="white", colour="black") +
  ggtitle("Birmingham [binwidth = 500]") +
  xlab("Population") +
  ylab("Frequency") +
  theme_classic()

hist2 <- ggplot(midyr_manchester_baser, aes(x=pop20)) +
  geom_histogram(binwidth = 500, fill="white", colour="black") +
  ggtitle("Manchester [binwidth = 500]") +
  xlab("Population") +
  ylab("Frequency") +
  theme_classic()

plot.hist_1_2 <- grid.arrange(hist1, hist2, ncol=2, nrow=1)
```

- To answer Q2 (bin width), re-run with a different `binwidth` and justify the choice.

Arrange Histograms in a Single Panel

The script uses `grid.arrange()` to place plots side-by-side.

```
plot.hist_1_2 <- grid.arrange(hist1, hist2, ncol=2, nrow=1)
```



What you should end up with

- Filtered datasets: `midyr_birmingham_baser`, `midyr_manchester_baser`, `filter_bir_manc_baser`.
- Descriptive statistics: a screenshot of `birmingham_result` and `manchester_result`.
- Visuals: dual boxplot + two-panel histogram (same binwidth).