Data processing of cities.txt

Cassandra Gunasekaram & Onkar Sadekar

2026-02-05

This Rmarkdown file includes an example of how to sort your code and workflow. You can then knit this document into a readable pdf, html, word etc.

You can insert text, tables, figures and code in these files, whereby all can be edited.
For example, you can use **bold** with `**bold`, or *cursive* with `*cursive*`, highlide `code` with `code` etc.
To get text on a new line, use double space at the end of the line .

You can insert lists as in other text editors:
1. one
2. two

or

- one
- two

or

- ☐ one
- ☐ two

Now let's start with the actual document.

In the beginning you can put any prior information, e.g. description, notes, copyright note etc.

Workspace setup

First set up the workspace: Empty the environment, and set knitting options.

Next, install packages if necessary.

Load packages: set warning=FALSE to hide warnings.

```
library(here)               # manage directories
library(dplyr)              # data manipulation
library(ggplot2)            # plots
library(viridis)            # colour palettes
```

Set directories (pathways to folders). We will also create a results folder.

```r
# where are we:
getwd()
```

```
## [1] "C:/Users/cassi/OneDrive - Universität Zürich UZH/PhD_general/admin_teaching/labmeetings/heeg_lab
```

```r
# set directories
dir_raw <- here("data", "raw_data")
dir_processed <- here("data", "processed_data")
dir_derived <- here("data", "derived_data")
dir_code <- here("code")

# Check if results folder exists, if not, create one.
if(here("results")==FALSE){
  dir.create(here("results"))
} else{
  print("Results already exists")
}
```

```
## [1] "Results already exists"
```

```r
dir_results <- here("results")
```

Process raw file

Here we will use an example of city densities as anyslsis. We wil use an example data set shown in the image:

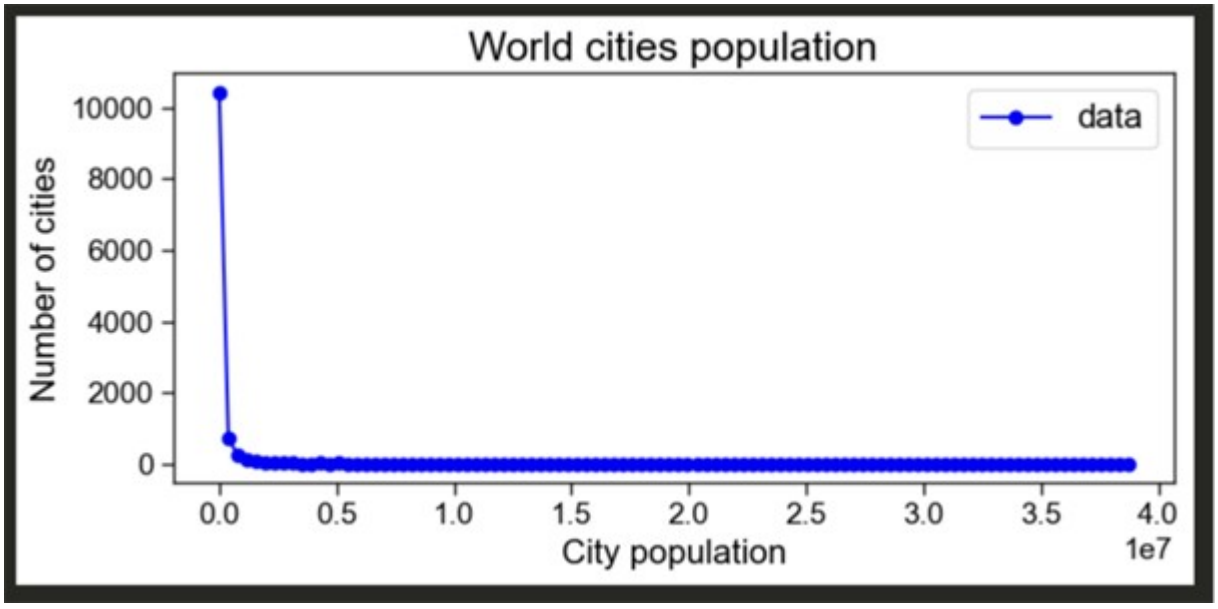


Figure 1: World cities populations

First, load the raw data:

```
cities <- read.delim(file.path(dir_raw, "cities.txt"))
cities_vec <- cities$X39105000

# Explore
summary(cities_vec)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     1012     8310    15920   106808    39836 35362000
```

Then process and save:

```
# Filter for >10000
cities_vec <- cities_vec[cities_vec > 10000]
write.csv(cities_vec, file.path(dir_processed, "cities_filtered.csv"),
          row.names = FALSE)
```

Analyse processed file

Get bins

In this example, we will bin the data and plot the histogram per bin.

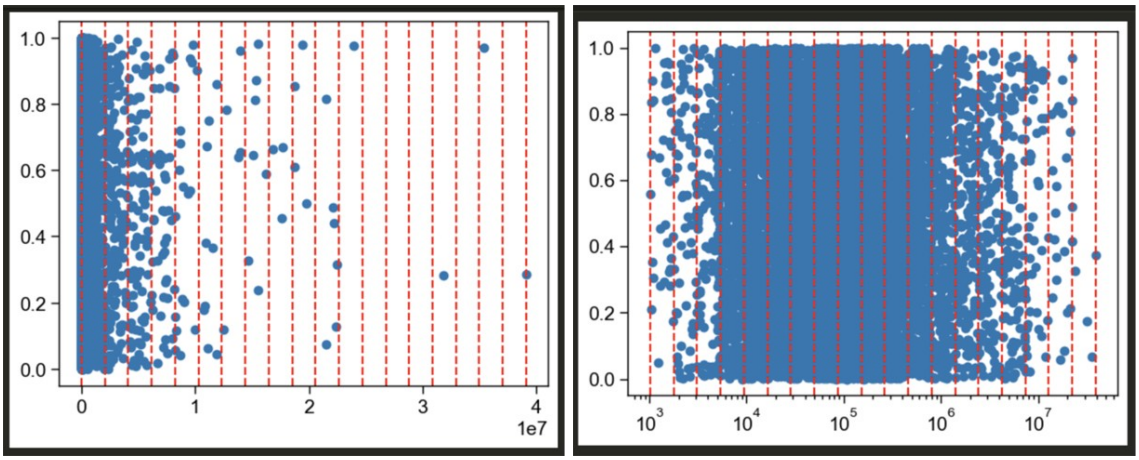


Figure 2: Spaced binning: linear (left) vs log (right).

Load the processed file:

```
# First read file
cities_filtered <- read.csv(file.path(dir_processed, "cities_filtered.csv"))
```

Then get the bins by sourcing the function. Save the Binned data set in the derived folder.

```
# Get function to logspace
source(file.path(dir_code, "get_logspaced_bins.R"))

bins <- get_log_bins(min(cities_filtered$x), max(cities_filtered$x), n_bins = 27)
```

```r
counts <- table(cut(cities_filtered$x, breaks = bins, include.lowest = TRUE, right = FALSE))

# View
as.data.frame(counts)
```

```
##                   Var1 Freq
## 1     [1e+04,1.35e+04) 4985
## 2  [1.35e+04,1.83e+04) 4225
## 3  [1.83e+04,2.48e+04) 3807
## 4  [2.48e+04,3.36e+04) 3162
## 5  [3.36e+04,4.54e+04) 2639
## 6  [4.54e+04,6.15e+04) 2066
## 7  [6.15e+04,8.32e+04) 1680
## 8  [8.32e+04,1.13e+05) 1247
## 9  [1.13e+05,1.52e+05) 1004
## 10 [1.52e+05,2.06e+05)  775
## 11 [2.06e+05,2.79e+05)  581
## 12 [2.79e+05,3.78e+05)  468
## 13 [3.78e+05,5.11e+05)  357
## 14 [5.11e+05,6.92e+05)  314
## 15 [6.92e+05,9.36e+05)  211
## 16 [9.36e+05,1.27e+06)  193
## 17 [1.27e+06,1.72e+06)  121
## 18 [1.72e+06,2.32e+06)   89
## 19 [2.32e+06,3.14e+06)   95
## 20 [3.14e+06,4.25e+06)   55
## 21 [4.25e+06,5.75e+06)   78
## 22 [5.75e+06,7.79e+06)   46
## 23 [7.79e+06,1.05e+07)   24
## 24 [1.05e+07,1.43e+07)   14
## 25 [1.43e+07,1.93e+07)   12
## 26 [1.93e+07,2.61e+07)    9
## 27 [2.61e+07,3.54e+07]    2
```

```r
# Save
write.csv(counts, file.path(dir_derived, "logspaced_density.csv"),
          row.names = FALSE)
```

Plot results

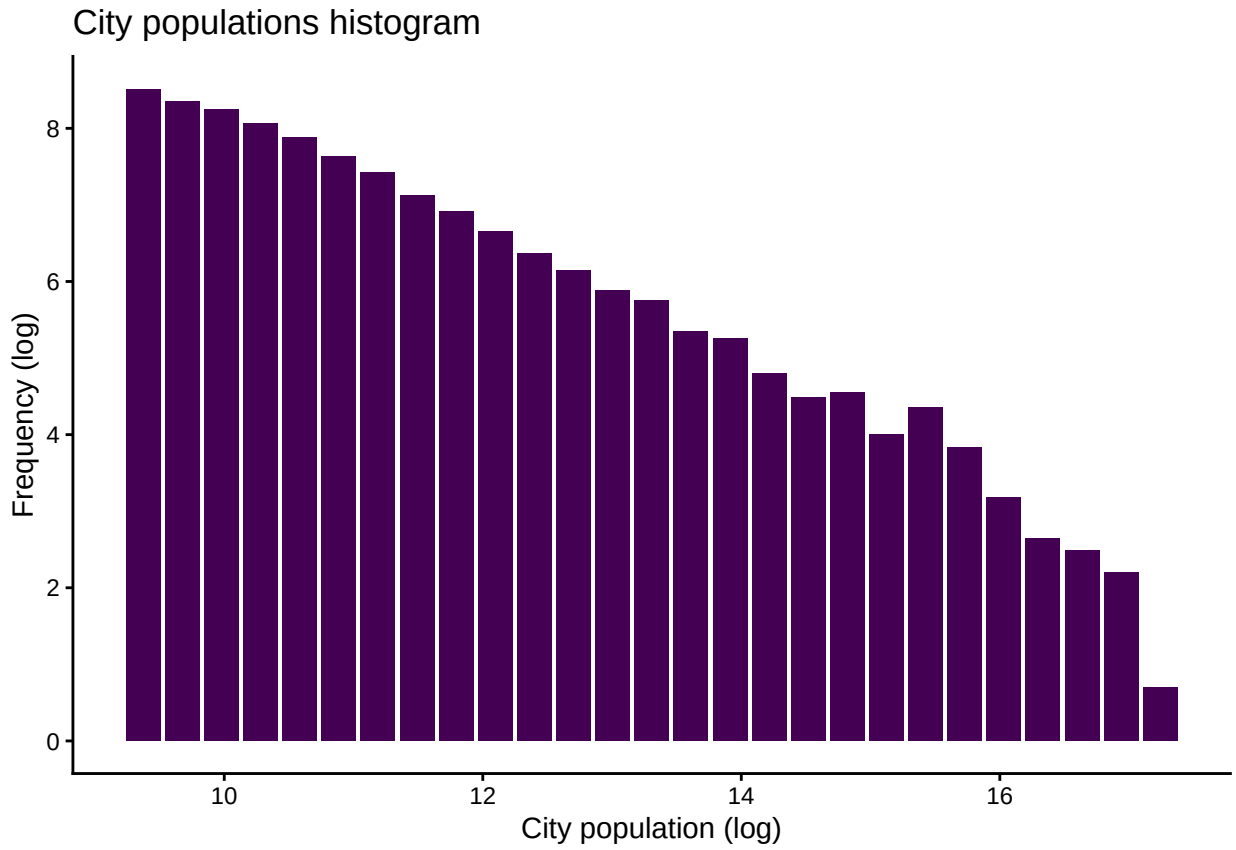
Finally, we can analyse the results by plotting. First load the necessary data sets and get the average per bin.

```r
densities <- read.csv(file.path(dir_derived, "logspaced_density.csv"))
str(densities)
```

```
## 'data.frame':    27 obs. of  2 variables:
##  $ Var1: chr  "[1e+04,1.35e+04)" "[1.35e+04,1.83e+04)" "[1.83e+04,2.48e+04)" "[2.48e+04,3.36e+04)" .
##  $ Freq: int  4985 4225 3807 3162 2639 2066 1680 1247 1004 775 ...
```

```r
# use custom function to get mean of each interval
densities$avg_bins <- get_interval_mean(densities$Var1)
```

Plot using the viridis palette: (when you plot make sure to put it on a new line with double spacing).



Save the output into results.