R Markdown (UPDATED: World Mapping)

Emma Chapman-Banks

2025-01-26

Part 1: Load packages/set up world map

Load necessary packages

```
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library("sf")
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

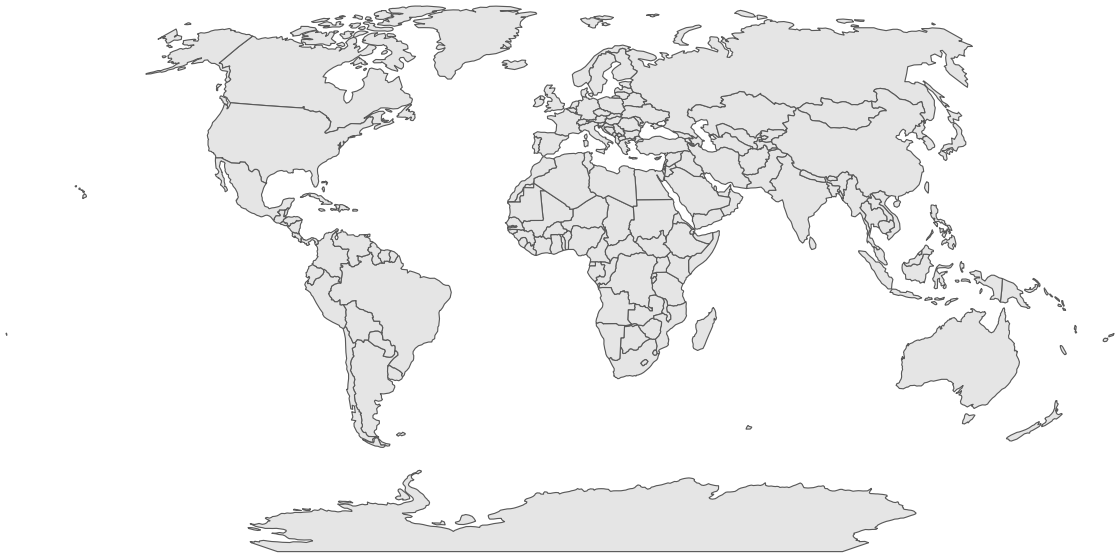
```
library("rnaturalearth")
library("countrycode")
library("ggrepel")
library("stringr")
```

World Data

```
# Get world data
world <- ne_countries(scale = "small", returnclass = "sf")
world$iso_a3 <- ifelse(world$iso_a3 == "-99",
                       world$iso_a3_eh,
                       world$iso_a3)

world$iso_a3 <- ifelse(world$iso_a3 == "-99",
                       world$sov_a3,
                       world$iso_a3)
```

```r
# Change map projection
world %>%
  st_transform(crs = "+proj=wintri") %>%
  ggplot() +
  geom_sf() +
  coord_sf(datum = NA) + # no graticule
  theme_minimal()
```



Part 2: Identify Unmapped Locations

```r
# Load new dataset with geolocations.
abstracts_data_clean <- read.csv("data/Combined_Location_With_Metadata_All.csv", stringsAsFactors = FALS

# Process `location` column
abstracts_data_clean <- abstracts_data_clean %>%
  filter(!is.na(location) & location != "") %>%
  separate_rows(location, sep = ";\\s*") %>%  # Split multiple locations into rows
  mutate(
    Iso3 = countrycode::countrycode(
      sourcevar = location,
      origin = "country.name",
      destination = "iso3c",
      warn = FALSE  # Suppress warnings for unmatched entries
    ),
    Iso3 = ifelse(is.na(Iso3), "unmapped", Iso3)  # Mark unmatched values as "unmapped"
  ) %>%
```

```r
  mutate(Status = ifelse(Iso3 == "unmapped", "Unmapped", "Mapped"))

# Look at unmatched values
unmatched <- abstracts_data_clean %>%
  filter(Status == "Unmapped") %>%
  distinct(location) %>%
  arrange(location)

# Print unmatched values for verification
#print(unmatched)

# Look at matched values
matched <- abstracts_data_clean %>%
  filter(Status == "Mapped") %>%
  distinct(location, Iso3) %>%
  arrange(Iso3)

# Print matched values
#print(matched)
```

Part 3: Clean and re-map unmapped entries

After identifying the list of unmapped entries, I did a quick scan through to understand how I could process the values. I decided to group all the values by country as some of the locations cited were cities/regions/places/hospitals that the Iso3 code would not pick up.

View unmatched entries in the US

```r
# Filter for US-related unmatched entries
unmatched_us <- unmatched %>%
  filter(str_detect(location, regex("U\\.S\\.|US|USA|United States|North America", ignore_case = TRUE)))

# Preview the unmatched US entries
#print(unmatched_us)
```

View unmatched entries in the UK

```r
# Filter for UK-related unmatched entries
unmatched_uk <- unmatched %>%
  filter(str_detect(location, regex("U\\.K\\.|UK|United Kingdom|England|Scotland|Ireland|Wales|Welsh|Br

# Preview the unmatched UK entries
#print(unmatched_uk)
```

Identify any rows that mention US states and other cities/counties (as some don't explictly say US/USA/U.S)

```r
# List and define US States
us_states <- paste(
  c(
    "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut", "Delaware", "Fl
  collapse = "|"
```

```
)

# Filter for rows mentioning any US state
unmatched_usstates <- unmatched %>%
  filter(str_detect(location, regex(us_states, ignore_case = TRUE)))

# Preview the unmatched state-related entries
#print(unmatched_usstates)
```

Now, let's do China (and chinese cities + beijing)

```
# Filter for China unmatched entries
unmatched_china <- unmatched %>%
  filter(str_detect(location, regex("China|Chinese cities|Chinese Cities|Beijing|Chinese provinces|Guang

# Preview the unmatched China entries
#print(unmatched_china)
```

View Candian cities/provinces

```
# Filter for Canadian unmatched entries
unmatched_canada <- unmatched %>%
  filter(str_detect(location, regex("Canadian|Quebec|Ontario|Alberta|British Columbia|Winnipeg, Manitoba

# Preview the unmatched Canada entries
#print(unmatched_canada)
```

Group Japan-associated areas

```
# Filter for Japanese unmatched entries
unmatched_japan <- unmatched %>%
  filter(str_detect(location, regex("Kanagawa|Osaka|Tokyo", ignore_case = TRUE)))

# Preview the unmatched Japanese entries
#print(unmatched_japan)
```

Group Australia-associated areas

```
# Filter for Australia unmatched entries
unmatched_australia <- unmatched %>%
  filter(str_detect(location, regex("Melbourne, Victoria|Queensland|Tasmania|Victoria" , ignore_case = T

# Preview the unmatched Australian entries
#print(unmatched_australia)
```

Identiy Yemen-associated areas

```
# Filter for Yemen unmatched entries
unmatched_yemen <- unmatched %>%
  filter(str_detect(location, regex("Yemen|Sana'a, Yemen|Yemen Arab Republic" , ignore_case = TRUE)))

# Preview the unmatched Yemen entries
#print(unmatched_yemen)
```

Identiy Brazil-associated areas

```
# Filter for Brazil unmatched entries
unmatched_brazil <- unmatched %>%
  filter(str_detect(location, regex("Amazon|Amazon Basin|Amazonian", ignore_case = TRUE)))

# Preview the unmatched Brazil entries
#print(unmatched_brazil)
```

View Indian-associated areas

```
# Filter for Indian unmatched entries
unmatched_india <- unmatched %>%
  filter(str_detect(location, regex("Andhra Pradesh|Benga|Bengal Delta|Delhi|Gujarat|Karnataka|Kerala|Ma

# Preview the unmatched Indian entries
#print(unmatched_india)
```

View Italian-associated areas

```
# Filter for Italian unmatched entries
unmatched_italy <- unmatched %>%
  filter(str_detect(location, regex("Apulia|Basilicata|Calabria|Campania|Emilia Romagna|Lazio|Lombardia

# Preview the unmatched Italian entries
#print(unmatched_italy)
```

View Thai-associated areas

```
# Filter for Thai unmatched entries
unmatched_thai <- unmatched %>%
  filter(str_detect(location, regex("Bangkok|Phramongkutklao Hospital|rural Thai settings", ignore_case

# Preview the unmatched Thai entries
#print(unmatched_thai)
```

View Indonesia-associated areas

```
# Filter for Indonesia unmatched entries
unmatched_indonesia <- unmatched %>%
  filter(str_detect(location, regex("Banjarmasin, South Kalimantan|Papua, Indonesia|West Papua, Indones

# Preview the unmatched Indonesia entries
#print(unmatched_indonesia)
```

View Spanish-associated areas

```
# Filter for Spanish unmatched entries
unmatched_spain <- unmatched %>%
  filter(str_detect(location, regex("Basque Country|Catalonia|Lleida, Catalonia", ignore_case = TRUE)))

# Preview the unmatched Spanish entries
#print(unmatched_spain)
```

View French-associated areas

```
# Filter for French unmatched entries
unmatched_france <- unmatched %>%
  filter(str_detect(location, regex("French West Indies|Institut du Cancer de Montpellier|Reunion Islan

# Preview the unmatched French entries
#print(unmatched_france)
```

View Korea-associated areas

```
# Filter for Korea unmatched entries
unmatched_korea <- unmatched %>%
  filter(str_detect(location, regex("Daegu, Republic of Korea|Daegu-Kyungpook Province, South Korea", ig

# Preview the unmatched Korea entries
#print(unmatched_korea)
```

View Tanzania-associated areas

```
# Filter for Tanzania unmatched entries
unmatched_tanzania <- unmatched %>%
  filter(str_detect(location, regex("Rufiji, Tanzania|Pemba, Zanzibar|Pemba Island, Zanzibar|Zanzibar",

# Preview the unmatched Tanzania entries
#print(unmatched_tanzania)
```

View Switzerland-associated areas

```
# Filter for Switzerland unmatched entries
unmatched_switzerland <- unmatched %>%
  filter(str_detect(location, regex("Lake Geneva|University Hospital of Zurich", ignore_case = TRUE)))

# Preview the unmatched Switzerland entries
#print(unmatched_switzerland)
```

View Micronesia-associated areas

```
# Filter for Micronesia unmatched entries
unmatched_micronesia <- unmatched %>%
  filter(str_detect(location, regex("Fais, Micronesia|Yap, Micronesia|Yap Island|Yap Island, Oceania|Ya

# Preview the unmatched Micronesian entries
#print(unmatched_micronesia)
```

View Saudi Arabia-associated areas

```
# Filter for Saudi unmatched entries
unmatched_saudi <- unmatched %>%
  filter(str_detect(location, regex("Gazan Province, Saudi Arabia|KSA", ignore_case = TRUE)))

# Preview the unmatched Saudi entries
#print(unmatched_saudi)
```

View Greek-associated areas

```
# Filter for Greece unmatched entries
unmatched_greece <- unmatched %>%
  filter(str_detect(location, regex("Crete|Region of Central Macedonia, Greece", ignore_case = TRUE)))

# Preview the unmatched Greece entries
#print(unmatched_greece)
```

Sudan

```
# Filter for Sudan unmatched entries
unmatched_sudan <- unmatched %>%
  filter(str_detect(location, regex("Gedaref State, Sudan|Gezira state, Sudan", ignore_case = TRUE)))

# Preview the unmatched Sudan entries
#print(unmatched_sudan)
```

Nigeria

```
# Filter for Nigeria unmatched entries
unmatched_nigeria <- unmatched %>%
  filter(str_detect(location, regex("Benin City, Nigeria|Benin, Nigeria", ignore_case = TRUE)))

# Preview the unmatched Nigeria entries
#print(unmatched_nigeria)
```

Sao Tome and Principe

```
# Filter for STP unmatched entries
unmatched_stp <- unmatched %>%
  filter(str_detect(location, regex("Principe|Principe Island, West Africa", ignore_case = TRUE)))

# Preview the unmatched STP entries
#print(unmatched_stp)
```

UPDATED FINAL CODE

```
# Process abstracts with new matched columns
abstracts_data_new <- abstracts_data_clean %>%
  filter(!is.na(location) & location != "") %>%
  separate_rows(location, sep = ";\\s*") %>%  # Split multiple locations into rows
  mutate(
    Iso3 = case_when(
      location %in% unmatched_us$location ~ "USA",  # US column
      location %in% unmatched_uk$location ~ "GBR",
      location %in% unmatched_usstates$location ~ "USA", # US states and other places
      location %in% unmatched_china$location ~ "CHN",
      location %in% unmatched_canada$location ~ "CAN",
      location %in% unmatched_japan$location ~ "JPN",
      location %in% unmatched_australia$location  ~ "AUS",
      location %in% unmatched_greece$location ~ "GRC",
```

```
    location %in% unmatched_saudi$location ~ "SAU",
    location %in% unmatched_micronesia$location ~ "FSM",
    location %in% unmatched_switzerland$location ~ "CHE",
    location %in% unmatched_tanzania$location ~ "TZA",
    location %in% unmatched_korea$location ~ "KOR",
    location %in% unmatched_france$location ~ "FRA",
    location %in% unmatched_spain$location ~ "ESP",
    location %in% unmatched_indonesia$location ~ "IDN",
    location %in% unmatched_thai$location ~ "THA",
    location %in% unmatched_italy$location ~ "ITA",
    location %in% unmatched_india$location ~ "IND",
    location %in% unmatched_yemen$location ~ "YEM",
    location %in% unmatched_brazil$location ~ "BRA",
    location %in% unmatched_sudan$location ~ "SDN",
    location %in% unmatched_nigeria$location ~ "NGA",
    location %in% unmatched_stp$location ~ "STP",
    location == "Chang Gung Memorial Hospital" ~ "TWN",
    location == "Czechoslovakia" ~ "NA",
    location == "former Czechoslovakia" ~ "NA",
    location == "Congo RDC" ~ "COD",
    location == "Durban" ~ "ZAF",
    location == "Ferlo" ~ "SEN",
    location == "Yugoslavia" ~ "SRB",
    location == "San Juan, PR" ~ "PRI",
    location == "Omsk, Western Siberia" ~ "RUS",
    location == "Nile Delta" ~ "EGY",
    location == "Southwestern Vietnam" ~ "VNM",
    location == "Southern Vietnam" ~ "VNM",
    location == "Eire" ~ "IRL",
    location == "Pomerania" ~ "POL",
    location == "Dutch administrative areas" ~ "NLD",
    location == "Bioko Island" ~ "GNQ",
    location == "Bavaria" ~ "DEU",
    location == "German Ruhr Area" ~ "DEU",
    location == "Khuzestan Province" ~ "IRN",
    location == "Madeira" ~ "PRT",
    location == "Malindi District" ~ "KEN",
    location == "Oslo University Hospital" ~ "NOR",
    location == "Auckland" ~ "NZL",
    location == "Saxony" ~ "DEU",
    location == "Al-Bahrani" ~ "ARE",
    location == "D.Z.)" ~ "DZA",
    location == "I.S." ~ "ISL",
    location == "MS" ~ "MSR",
    location == "Haseki Research and Training" ~ "TUR",
    location == "Kruger National Park" ~ "ZAF",
    location == "Rakai" ~ "UGA",
    location == "Clinic for Nephrology and Clinical Immunology of the Clinical Centre of Vojvodina" ~
    TRUE ~ countrycode::countrycode(
      sourcevar = location,
      origin = "country.name",
      destination = "iso3c",
      warn = FALSE  # Suppress warnings for unmatched entries
```

```
      )
    ),
    Iso3 = ifelse(is.na(Iso3), "unmapped", Iso3)  # Mark unmatched values as "unmapped"
  )

# Identify unmatched values for review
unmatched_new <- abstracts_data_new %>%
  filter(Iso3 == "unmapped") %>%
  distinct(location) %>%
  arrange(location)

# Print unmatched values for verification
#print(unmatched_new)
```

```
# Identify newly matched values
newly_matched <- abstracts_data_new %>%
  filter(location %in% unmatched$location, Iso3 != "unmapped") %>%
  distinct(location, Iso3)

# Preview the newly matched values
#print(newly_matched)
```

Now we have as many matched values as we could. We will now combine matched (from part 2) and newly_matched so that we can map everything onto the plot.

```
# Combine newly_matched with matched
all_matched <- bind_rows(matched, newly_matched)

# Summarize by Iso3 (count the number of abstracts per country)
all_matched_summary <- all_matched %>%
  group_by(Iso3) %>%
  summarize(count = n(), .groups = "drop") %>%
  arrange(desc(count))

# Fix iso_a3 for France
world <- world %>%
  mutate(iso_a3 = ifelse(name == "France", "FRA", iso_a3))

# Join the summary data with the world map
world_with_data <- world %>%
  mutate(iso_a3 = ifelse(iso_a3 == "-99", sov_a3, iso_a3)) %>%
  left_join(all_matched_summary, by = c("iso_a3" = "Iso3"))

# Plot the world map
ggplot(world_with_data) +
  geom_sf(aes(fill = count)) +
  scale_fill_viridis_c(option = "plasma", na.value = "grey90", name = "Abstract Count") +
  theme_minimal() +
  labs(
    title = "Global Distribution of Abstract Locations",
    subtitle = "Number of abstracts mapped by country"
  )
```

Global Distribution of Abstract Locations

Number of abstracts mapped by country