# ArchaeoGLOBE trend analysis

*Nick Gauthier*

*August 23, 2018*

Sample analysis code for the ArchaeoGlobe database. Here we fit Generalized Additive Models (GAMs), a flexible form of nonlinear regression model capable of fitting smooth, time-varying trends to the ordered categorical ArchaeoGLOBE response data.

We model ordered categorical data using a latent variable following a logistic distribution. The model identifies a series of cut points, which correspond the the probabilities of the latent variable falling within each of our categories.

We fit two sets of trends. One trend is fitted to all the data simultaneously, representing the global trend across all archaeological regions. Then we fit region-level trends, which represent the deviation of each region from the global trend. By penalizing the "wiggliness" of the trend lines, we allow regional trends that don't significantly deviate from the global trend to be penalized to 0, effectively reducing that particular region to the global trend. This is a form of partial pooling, allowing the model to share information between groups and in so doing make the results less sensitive to regions with exceptionally low response rates.

After fitting the model, we can extract the region-specific diviations from the global trend, use a k-means clustering alogirithm to group together regions with similar trends, and map the results. We repeat this analysis for both self-reported expertise and perceived data quality.

## Setup

Import packages needed for analysis. We'll use packages from the `tidyverse`, such as `readr`, `dplyr`, and `ggplot2` for data import, processing, and plotting. We'll also use `mgcv` for fitting nonlinear trends to the expertise and quality datasets. We'll use the `sf` package to help us plot shapefiles in a tidy context. Finally, we'll use `patchwork` to combine multiple ggplots in the same image.

```
library(tidyverse)
library(mgcv)
library(sf)

#install patchwork from github
#devtools::install_github('thomasp85/patchwork')
library(patchwork)
```

## Data import

Read in the latest version of the ArchaeoGLOBE database.

```
dat <- read_csv('data/Survey_scrubbed_Aug20_IDs.csv') %>%
  select(contributor = CONTRIBUTR,
         macroregion = WORLD_LAB,
         region = REGION_LAB,
         region_id = REGION_ID,
         EXP_10000:DQ_00150) %>%
  mutate_all(as.factor)
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   WORLD_ID = col_integer(),
##   REGION_ID = col_integer(),
##   TOT_AREA = col_double(),
##   LAND_AREA = col_double()
## )

## See spec(...) for full column specifications.
```

Separate the ArchaeoGLOBE data into expertise and quality datasets, each in long "tidy" format to make analysis and plotting easier.

```
exp_dat <- dat %>%
  select(-c(DQ_10000:DQ_00150)) %>% # drop the DQ columns
  gather(time, expertise, EXP_10000:EXP_00150) %>% # format so one expertise value per row
  mutate(time = parse_number(time) * -1, # convert time period labels to years
         expertise = ordered(expertise, levels = c('None', 'Low', 'High')),
         exp_num = as.numeric(expertise))

qual_dat <- dat %>%
  select(-c(EXP_10000:EXP_00150)) %>% # drop the EXP columns columns
  mutate_at(vars(DQ_10000:DQ_00150), funs(ordered(., levels = c('Unknown', 'Low', 'Moderate', 'Good'))))
  gather(time, quality, DQ_10000:DQ_00150) %>% # format so one expertise value per row
  mutate(time = parse_number(time) * -1, # convert time period labels to years
         quality = ordered(quality, levels = c('Unknown', 'Low', 'Moderate', 'Good')),
         qual_num = as.numeric(quality))
```

Finally import the regions shapefile.

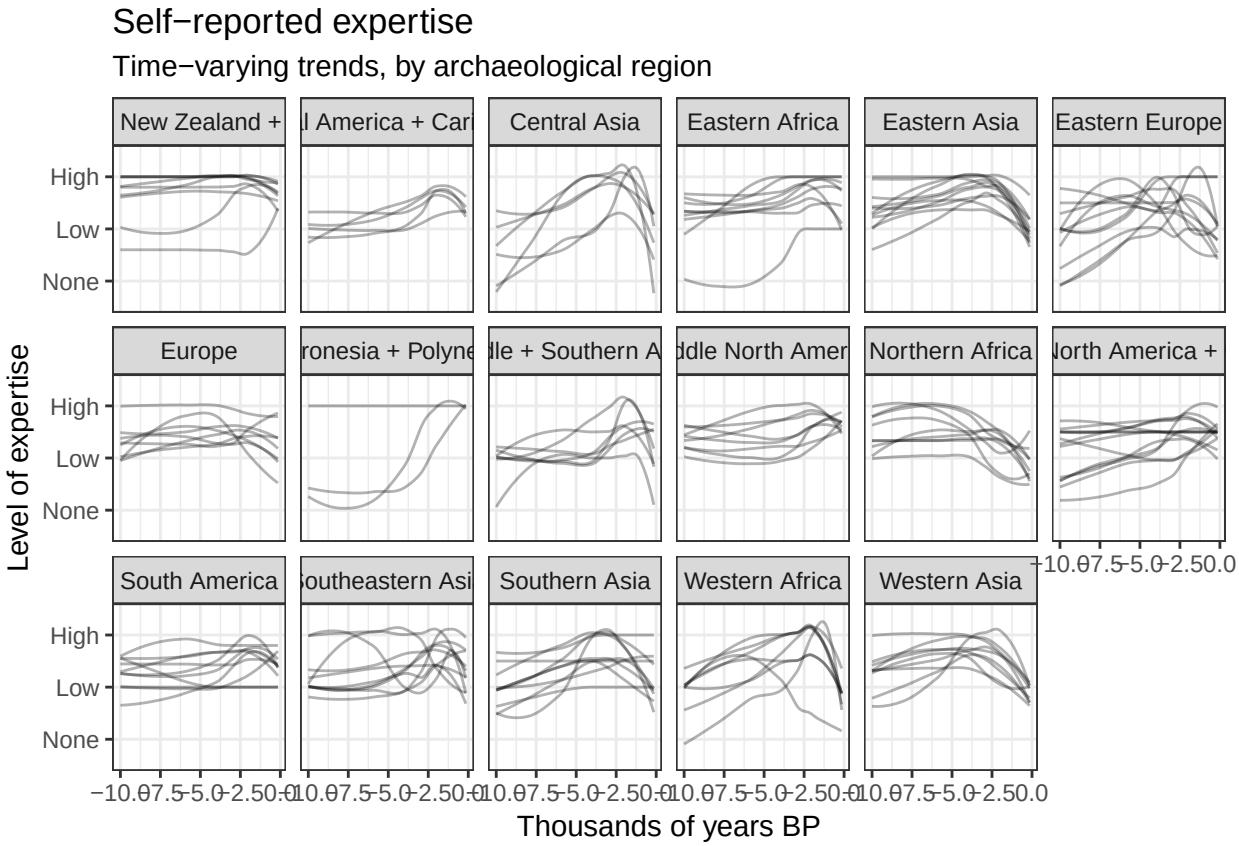
```
regions <- st_read('data/ArchaeGLOBE_Regions.shp')

## Reading layer `ArchaeGLOBE_Regions' from data source `/home/nick/gdrive/Projects/ArchaeoGLOBE/data/A
## Simple feature collection with 146 features and 6 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -16653410 ymin: -6637605 xmax: 16821760 ymax: 8375497
## epsg (SRID):    NA
## proj4string:    +proj=eck4 +lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs
```

## Analysis of expertise

How does self-professed level of expertise vary in each region over time?

Before doing anything else, let's plot out the data.

```
ggplot(exp_dat, aes(time / 1000, expertise, group = region)) +
  stat_smooth(geom='line', alpha=0.3, se=FALSE, method = 'loess') +
  facet_wrap(~macroregion, nrow = 3) +
  labs(title = 'Self-reported expertise',
       subtitle = 'Time-varying trends, by archaeological region',
       x = 'Thousands of years BP', y = 'Level of expertise') +
  theme_bw()
```



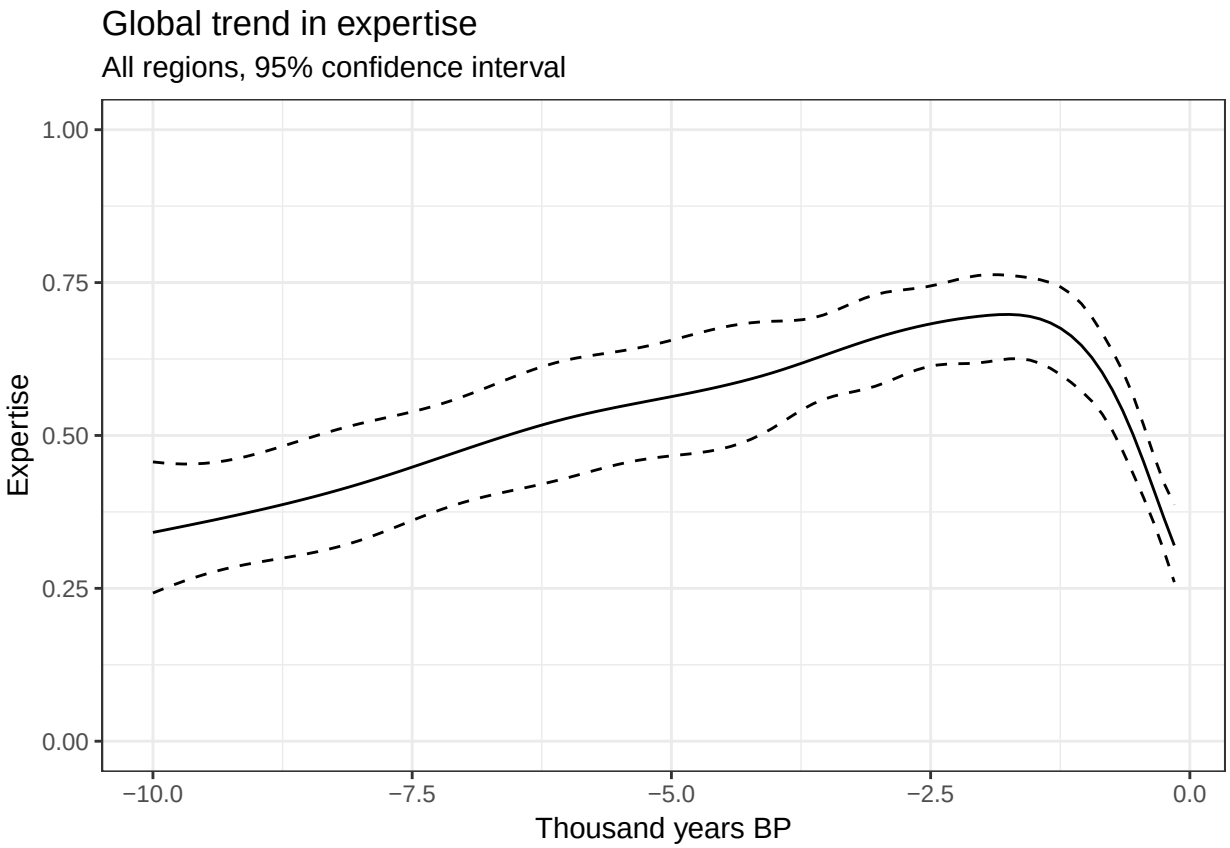

This plot will be somewhat misleading, because we aren't accounting for individual observer effects, such as observers who are more likely to enter high expertise regardless of region. Let's try fitting a multilevel GAM.

```
exp_mod <- bam(exp_num ~
               # this spline is for the global trend
               s(time, bs = 'cr', m = 2) +
               # region-specific trends. bs = 'ts' and m = 1
               # help penalize deviation from the global model
               s(time, by = region, bs = 'cs', m = 1) +
               # add back in region-specific intercepts
               region +
               # model contributor as a random effect
               s(contributor, bs = 're', k = 252),
             method = 'fREML',
             discrete = TRUE,
             nthreads = 2,
             family = ocat(R = 3), # ordered categorical with 3 levels
             data = exp_dat)
```

First we plot the global trend. We see a linear increase in self-reported expertise from 10ka BP up to 2ka BP, then a falloff continuing to the present day. This makes sense, as it points to both the increased frequency of preserved archaeological materials with time as well as the reduction in archaeological attention in periods with extensive historical records.

```
plot(exp_mod, select = 0) %>%
  .[1] %>%
```

```
map(~tibble(time = .$x, fit = c(.$fit), se = .$se)) %>%
.[[1]] %>%
ggplot(aes(time / 1000, plogis(fit)))+
geom_line() +
geom_line(aes(y = plogis(fit + 2 * se)), linetype = 2) +
geom_line(aes(y = plogis(fit - 2 * se)), linetype = 2) +
scale_y_continuous(limits = c(0,1)) +
labs(title = 'Global trend in expertise',
     subtitle = 'All regions, 95% confidence interval',
     x = 'Thousand years BP', y = 'Expertise') +
theme_bw()
```



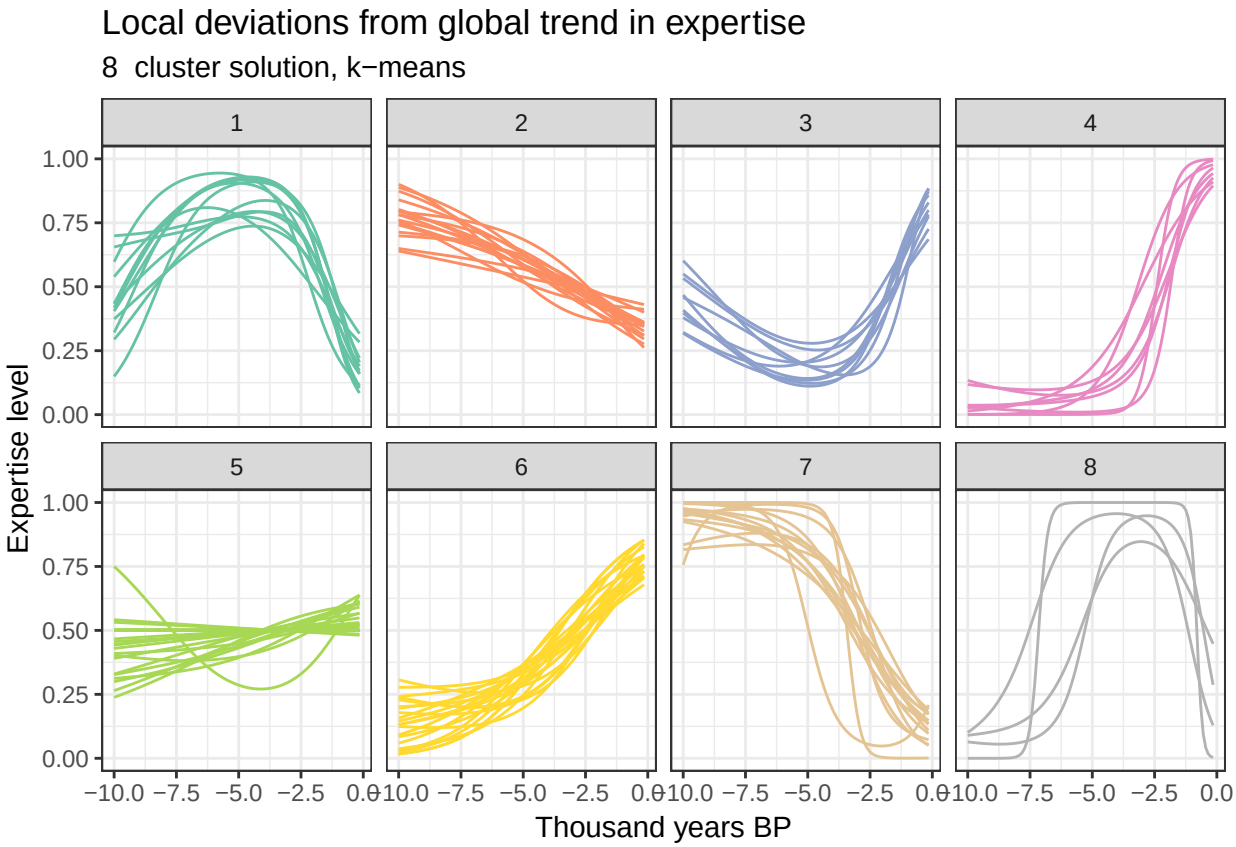

Now let's investigate the local deviations from this global trend. Start by extracting the fitted splines for each region, ignoring factors such as the global trend and region and contributor specific intercepts so that the focus is on the shape of the local trends.

```
exp_local_trends <- exp_mod %>%
  plot(select = 0) %>% # plot for the side effect of printing smoothed fits
  .[2:147] %>% # extract the local trends
  map(~tibble(region =.$ylab, time = .$x, fit = c(.$fit))) %>%
  bind_rows %>%
  mutate(fit = plogis(fit)) %>%
  spread(time, fit)
```

Now we cluster together the local deviations from the global trend using a k-means algorithm. We need to set the seed first to ensure reproduceability of the cluster solutions and ordering. The selection of 8 clusters

is somewhat arbitrary, and is made simply based on visual comparisons of different cluster solutions with the goal of retaining as few clusters as possible while keeping their interpretations distinct.

```
n_clusters <- 8

set.seed(1000) # set seed for reproduceability
exp_local_trends %>%
  mutate(cluster = as.factor(kmeans(.[,-1], n_clusters, iter.max = 100, nstart = 100)$cluster)) %>%
  gather(time, expertise, 2:101) %>%
  ggplot(aes(as.numeric(time) / 1000, expertise,
             group = region, color = cluster)) +
  geom_line() +
  scale_color_brewer(palette = 'Set2', guide = 'none') +
  facet_wrap(~cluster, nrow = 2) +
  labs(title = 'Local deviations from global trend in expertise',
       subtitle = paste(n_clusters, ' cluster solution, k-means'),
       x = 'Thousand years BP', y = 'Expertise level') +
  theme_bw()
```

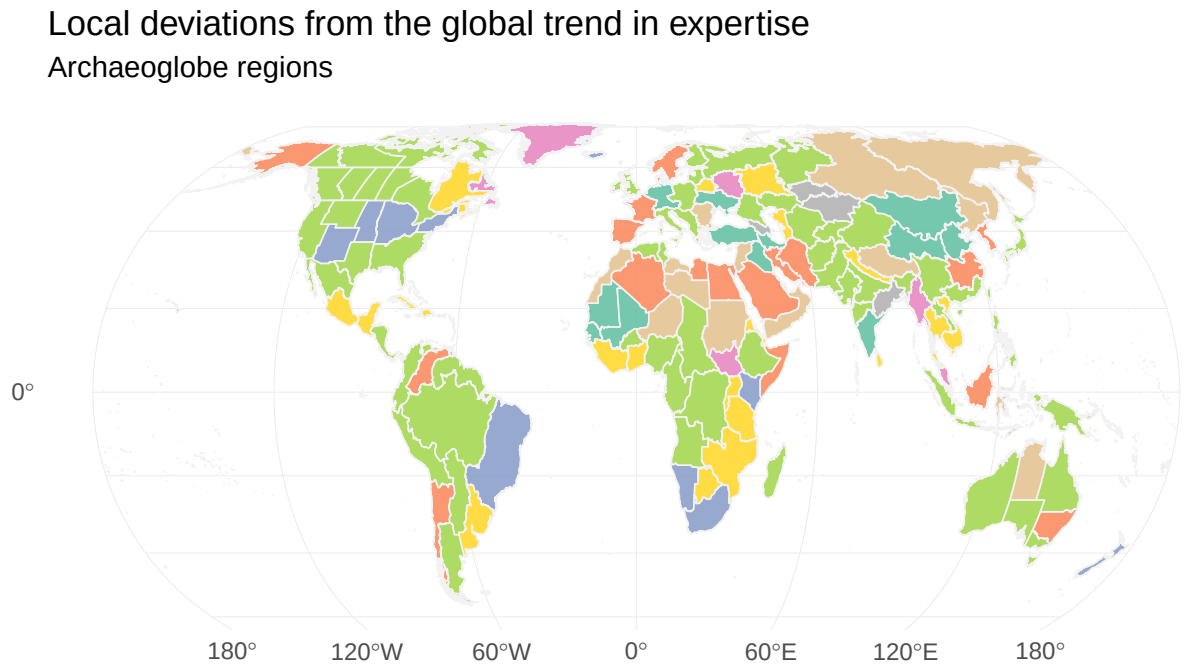



Next we map out the archaeological regions, showing which region belongs to each cluster.

```
set.seed(1000) # use the same seed as before so clusters match
exp_local_trends %>%
  mutate(cluster = as.factor(kmeans(.[,-1], n_clusters, iter.max = 100, nstart = 100)$cluster)) %>%
  select(region, cluster) %>% # just select the columns of interest
  separate(region, c('extra', 'region'), sep =  'region') %>%
```

```
  left_join(select(exp_dat, region:region_id)) %>%
  mutate(region_id = as.numeric(region_id)) %>%
  group_by(region) %>%
  filter(row_number() == 1) %>%
  left_join(regions, ., by = c('Archaeo_ID' = 'region_id')) %>% # join to the region shapes
  ggplot() + # plot
  geom_sf(aes(fill = cluster), size=.3, color = 'grey95', alpha=0.9) +
  scale_fill_brewer(palette = 'Set2', guide = 'none') +
  labs(subtitle = 'Archaeoglobe regions',
       title = 'Local deviations from the global trend in expertise') +
  theme_minimal()
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

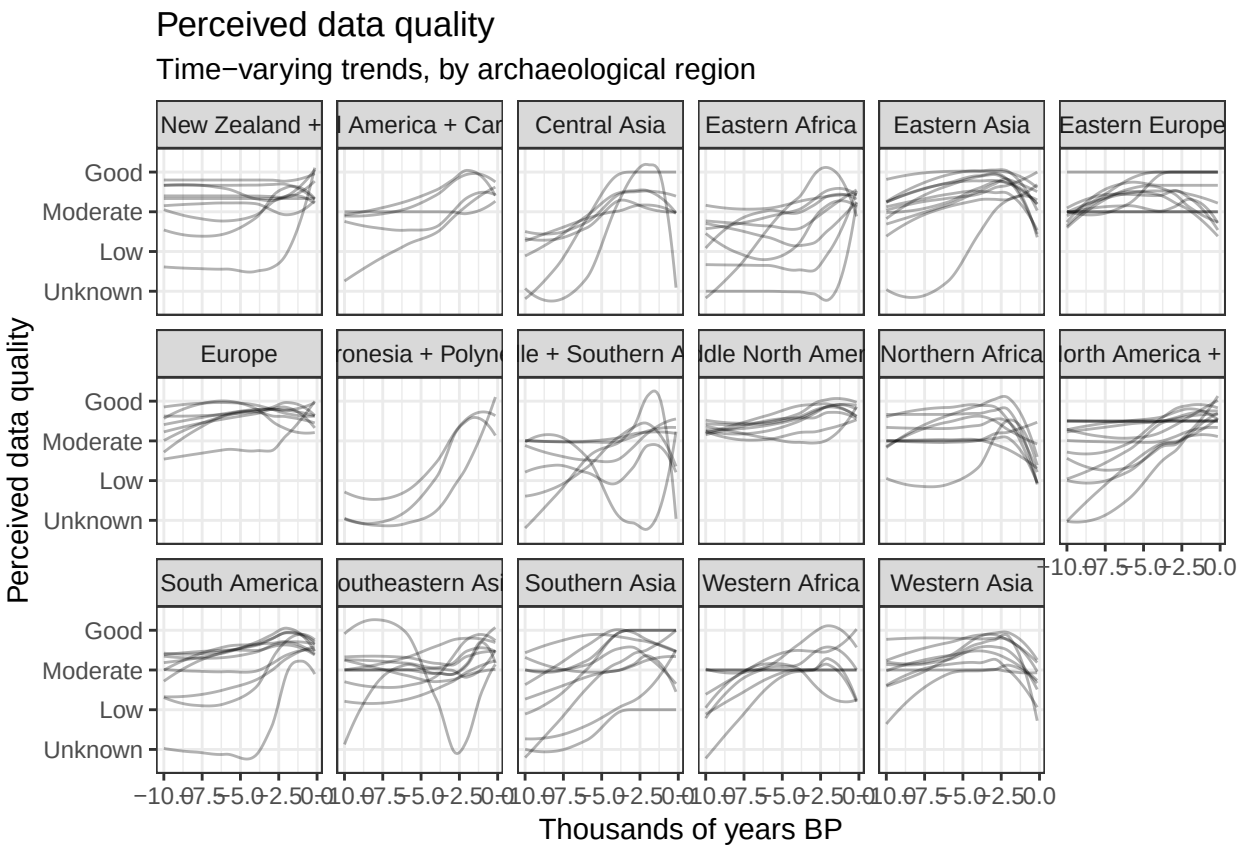



## Analysis of data quality

Repeat the above analysis, now looking at how perceptions of data quality vary in each region over time. Start with just a naive plot of the data.

```
ggplot(qual_dat, aes(time / 1000, quality, group = region)) +
  stat_smooth(geom = 'line', alpha = 0.3, se = FALSE, method = 'loess') +
  facet_wrap(~macroregion, nrow = 3) +
  labs(title = 'Perceived data quality',
```

```
      subtitle = 'Time-varying trends, by archaeological region',
      x = 'Thousands of years BP', y = 'Perceived data quality') +
  theme_bw()
```

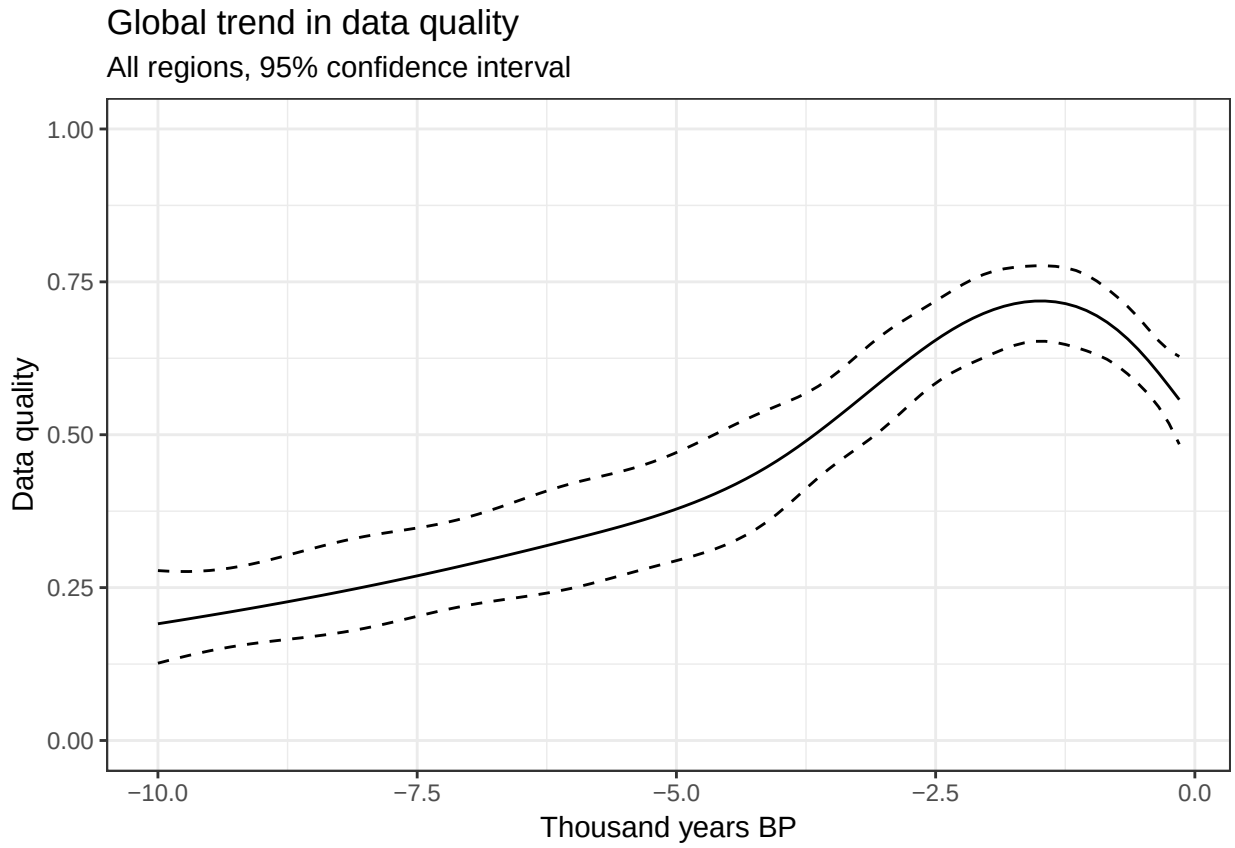



Next fit the GAM.

```
qual_mod <- bam(qual_num ~
              # this spline is for the global trend
              s(time, bs = 'cr', m = 2) +
              # region-specific trends. bs = 'ts' and m = 1
              # help penalize deviation from the global model
              s(time, by = region, bs = 'cs', m = 1) +
              # we need to add back in region-specific intercepts
              region +
              # model contributor as a random effect
              s(contributor, bs = 're', k = 252),
            method = 'fREML',
            discrete = TRUE,
            nthreads = 2,
            family = ocat(R = 4), # ordered categorical with 4 levels
            data = qual_dat)
```

The global trend is more or less the same as the expertise data, with the peak in data quality occurring more recently than for expertise and with a less dramatic falloff leading to the present day. Also note the confidence interval for the global trend is generally wider than for the expertise responses.

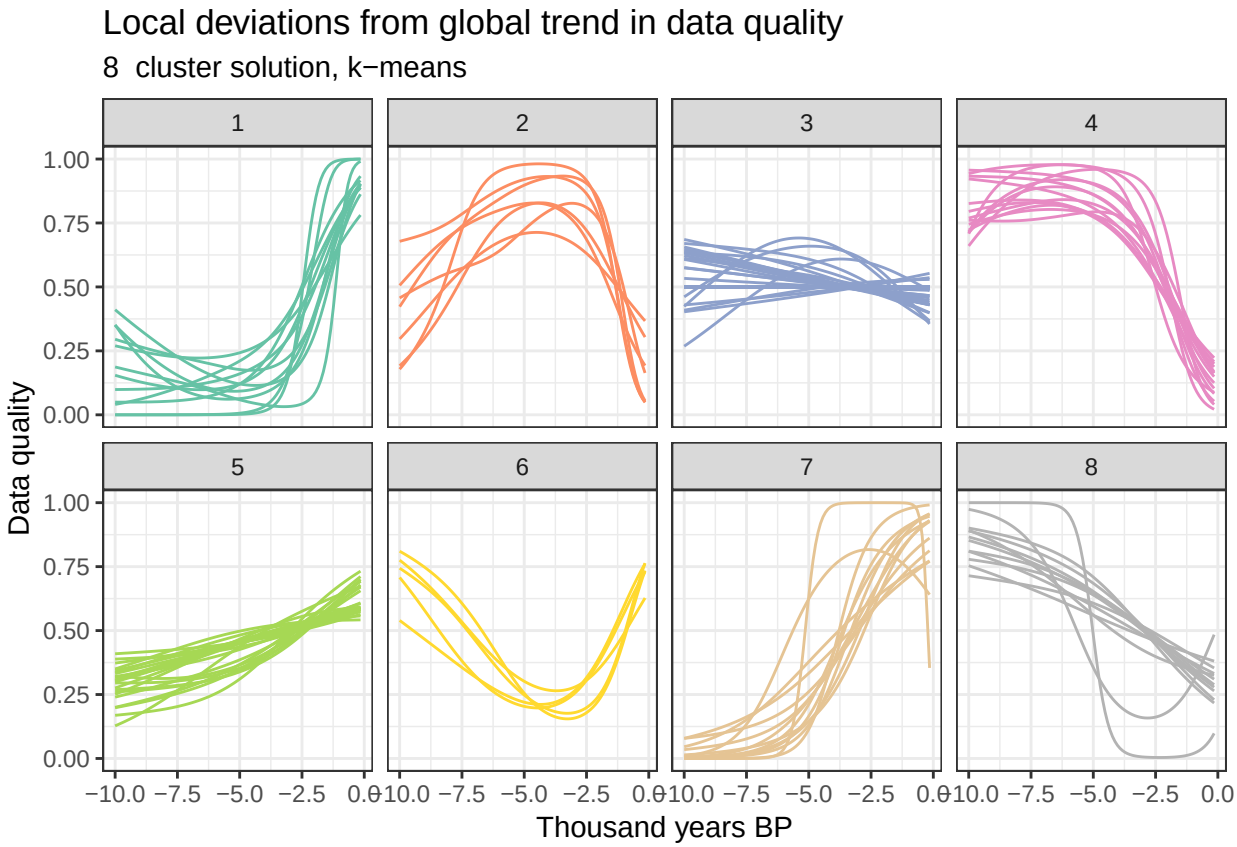
```
plot(qual_mod, select = 0) %>%
  .[1] %>%
  map(~tibble(time = .$x, fit = c(.$fit), se = .$se)) %>%
  .[[1]] %>%
  ggplot(aes(time / 1000, plogis(fit)))+
  geom_line() +
  geom_line(aes(y = plogis(fit + 2 * se)), linetype = 2) +
  geom_line(aes(y = plogis(fit - 2 * se)), linetype = 2) +
  scale_y_continuous(limits = c(0,1)) +
  labs(title = 'Global trend in data quality',
       subtitle = 'All regions, 95% confidence interval',
       x = 'Thousand years BP', y = 'Data quality') +
  theme_bw()
```




```
qual_local_trends <- qual_mod %>%
  plot(select = 0) %>% # plot for the side effect of printing smoothed fits
  .[2:147] %>% # extract the local trends
  map(~tibble(region =.$ylab, time = .$x, fit = c(.$fit))) %>%
  bind_rows %>%
  mutate(fit = plogis(fit)) %>%
  spread(time, fit)
```

We select a 8 cluster solution here, although as above the cluster selection is done primarily to aide visual interpretation and should not be taken as the only possible solution.

```
n_clusters <- 8

set.seed(1000)
qual_local_trends %>%
  mutate(cluster = as.factor(kmeans(.[,-1], n_clusters, iter.max = 100, nstart = 100)$cluster)) %>%
  gather(time, quality, 2:101) %>%
  ggplot(aes(as.numeric(time) / 1000, quality, group = region, color = cluster)) +
  geom_line() +
  scale_color_brewer(palette = 'Set2', guide = 'none') +
  facet_wrap(~cluster, nrow = 2) +
  labs(title = 'Local deviations from global trend in data quality',
       subtitle = paste(n_clusters, ' cluster solution, k-means'),
       x = 'Thousand years BP', y = 'Data quality') +
  theme_bw()
```



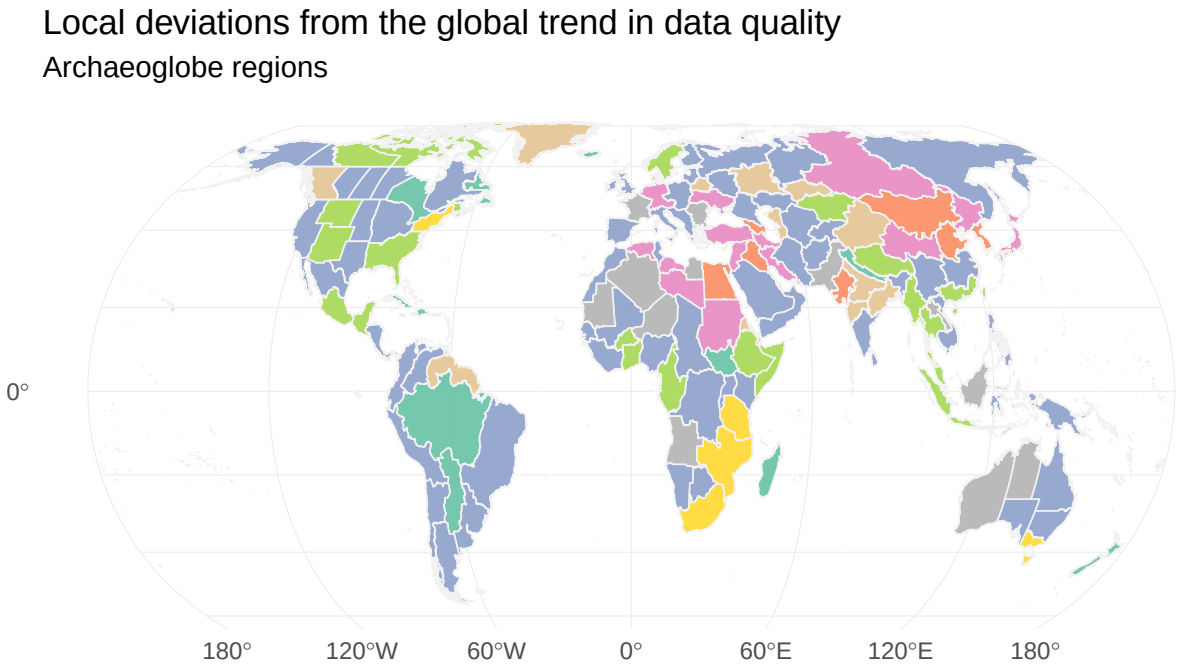

Map the clusters once more.

```
set.seed(1000)
qual_local_trends %>%
  mutate(cluster = as.factor(
    kmeans(.[,-1], n_clusters, iter.max = 100, nstart = 100)$cluster)) %>%
  select(region, cluster) %>% # just select the columns of interest
  separate(region, c('extra', 'region'), sep =  'region') %>%
  left_join(select(qual_dat, region:region_id)) %>%
  mutate(region_id = as.numeric(region_id)) %>%
  group_by(region) %>%
```

```
  filter(row_number() == 1) %>%
  left_join(regions, ., by = c('Archaeo_ID' = 'region_id')) %>% # join to the region shapes
  ggplot() + # plot
    geom_sf(aes(fill = cluster), size=.3, color = 'grey95', alpha=0.9) +
    scale_fill_brewer(palette = 'Set2', guide = 'none') +
    labs(subtitle = 'Archaeoglobe regions',
         title = 'Local deviations from the global trend in data quality') +
    theme_minimal()
```

```
## Joining, by = "region"
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```




## Plots

Our final step is to combine the results from all the analyses above into two nice figures. We recreate all the plots above (not shown) saving the call to an R object. Then we plot them all together using `patchwork`.

```
((p1 | p2 ) + plot_layout(widths = c(1,2))) /
  p3 +
  plot_layout(heights = c(1,2))
```

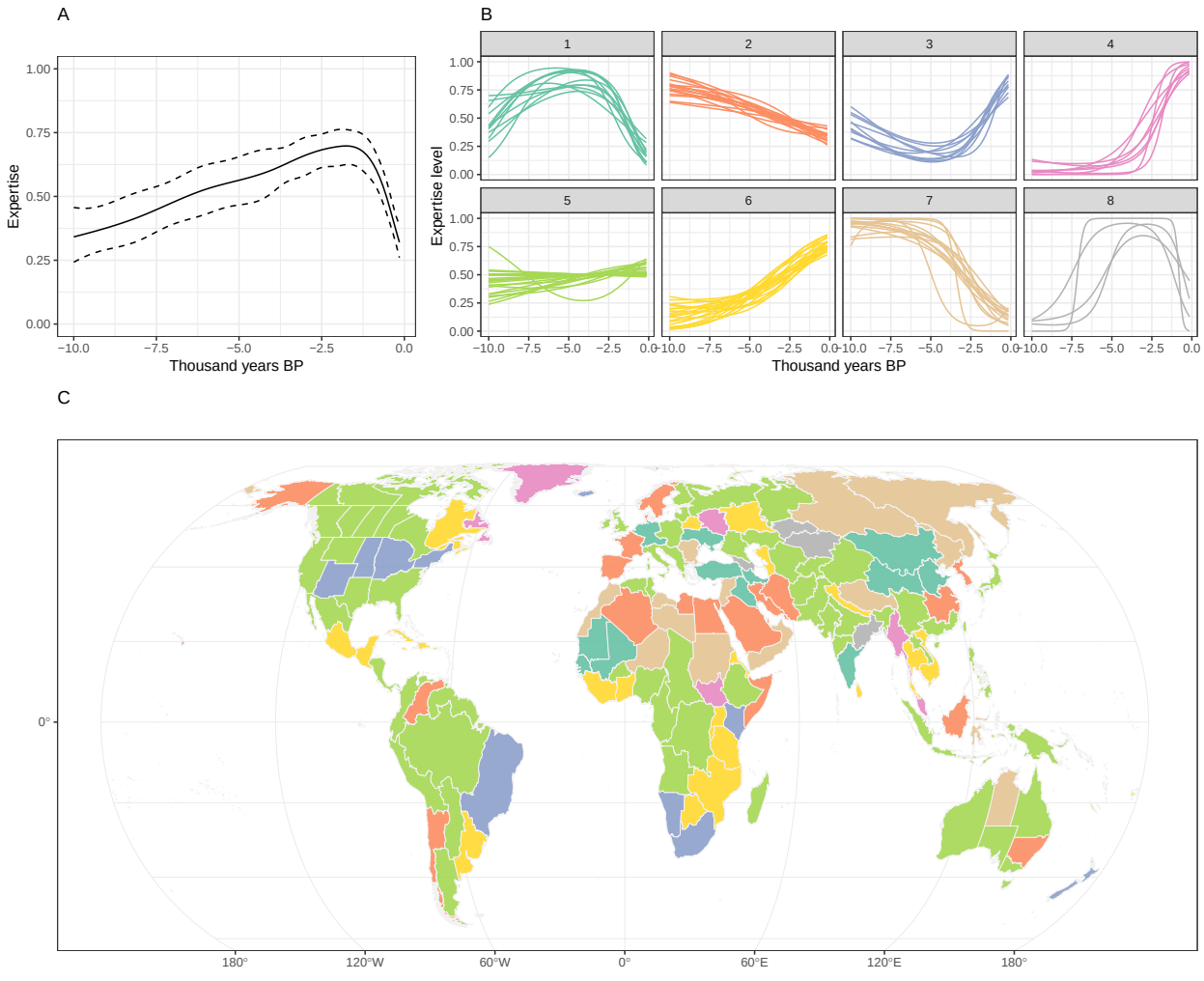



Figure 1: Global and regional trends in self-reported expertise. (A) Global trend (all regions) with 95% confidence interval. (B) Regional deviations from global trend, clustered via k-means. (C) Map of the local deviations from the global trend, same clusters as in B.

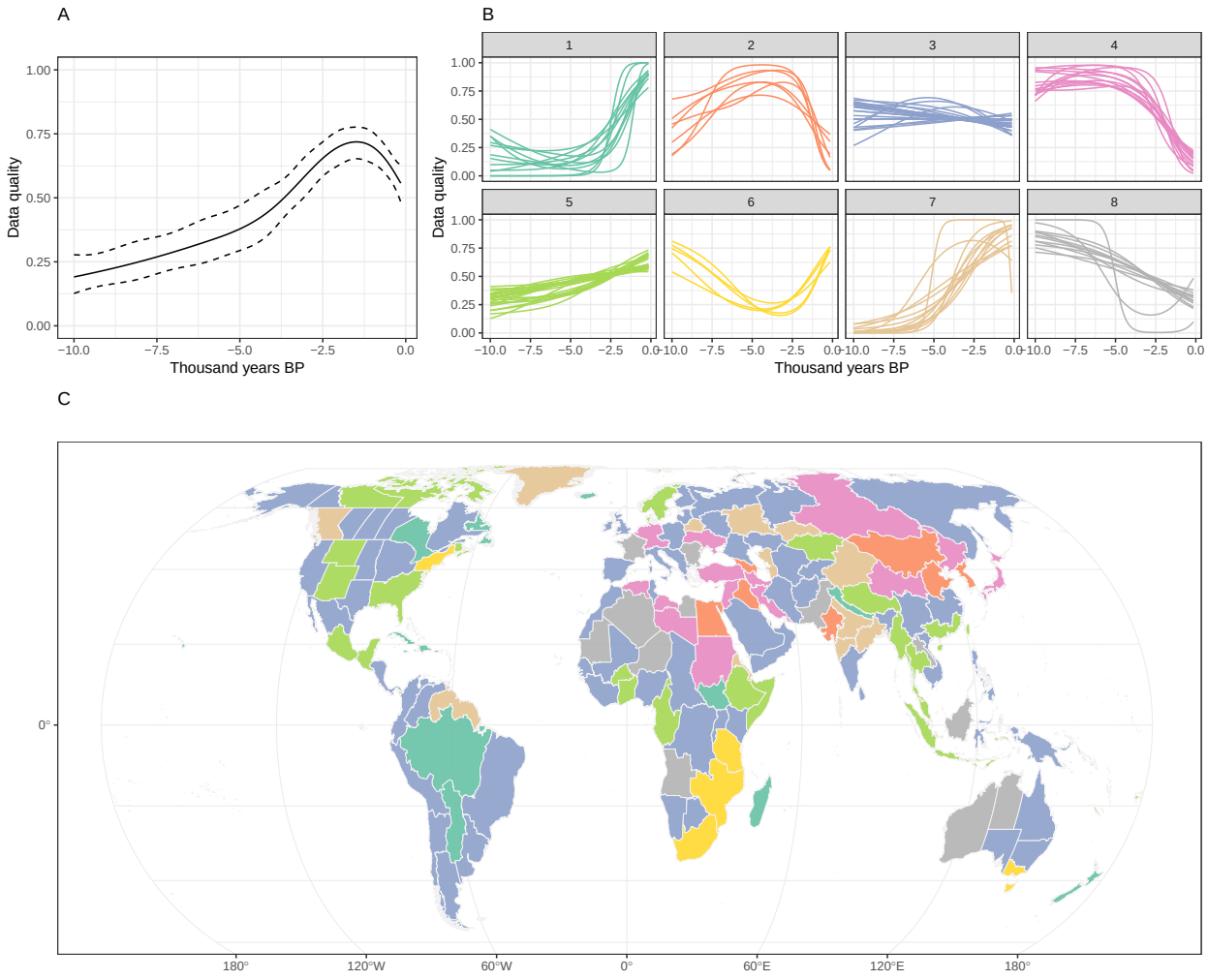



Figure 2: Global and regional trends in perceived data quality. (A) Global trend (all regions) with 95% confidence interval. (B) Regional deviations from global trend, clustered via k-means. (C) Map of the local deviations from the global trend, same clusters as in B.

```
((p4 | p5 ) + plot_layout(widths = c(1,2))) /
  p6 +
  plot_layout(heights = c(1,2))
```