ArchaeoGLOBE trend analysis

Nick Gauthier

Last knit on: 05 February, 2019

Sample analysis code for the ArchaeoGlobe database. Here we use 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 deviations 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 data. 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.

```r
library(tidyverse)
library(mgcv)
library(sf)
library(ggplot2)

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

Data import

Read in the latest version of the ArchaeoGLOBE database and the regions' shapefile from the Dataverse repository.

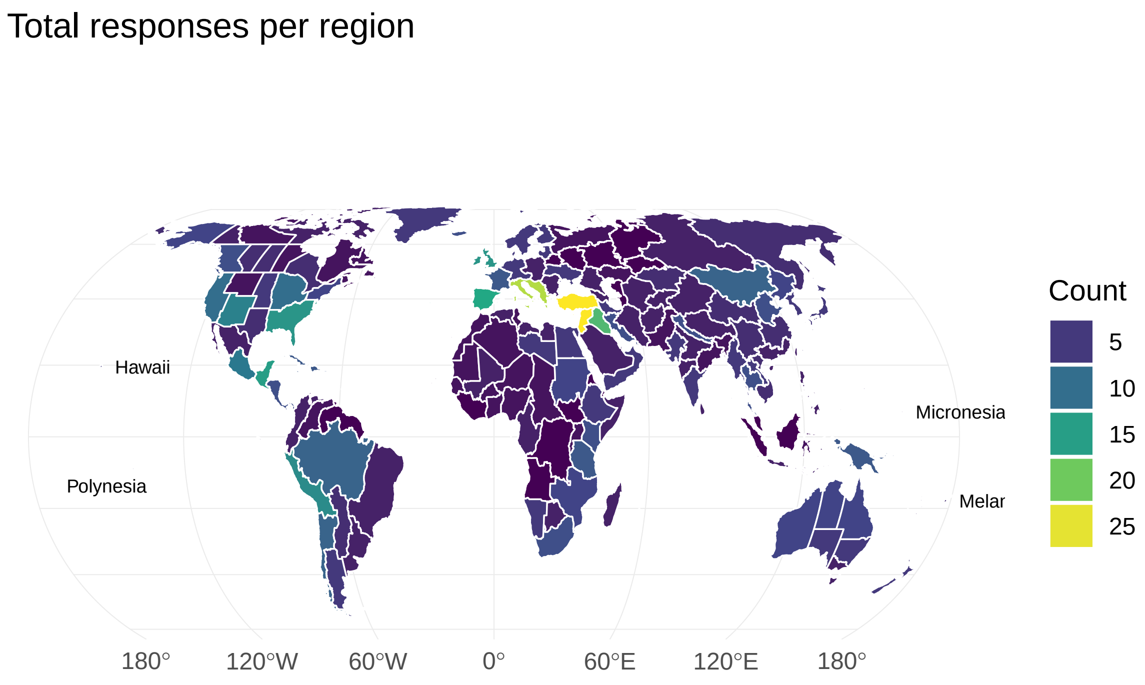
```r
library("dataverse")
Sys.setenv("DATAVERSE_SERVER" = "dataverse.harvard.edu")

# get data frame of files on dataverse
ArchaeoGLOBE_Public_Data_DOI <-
  "doi:10.7910/DVN/CNCANQ"
```

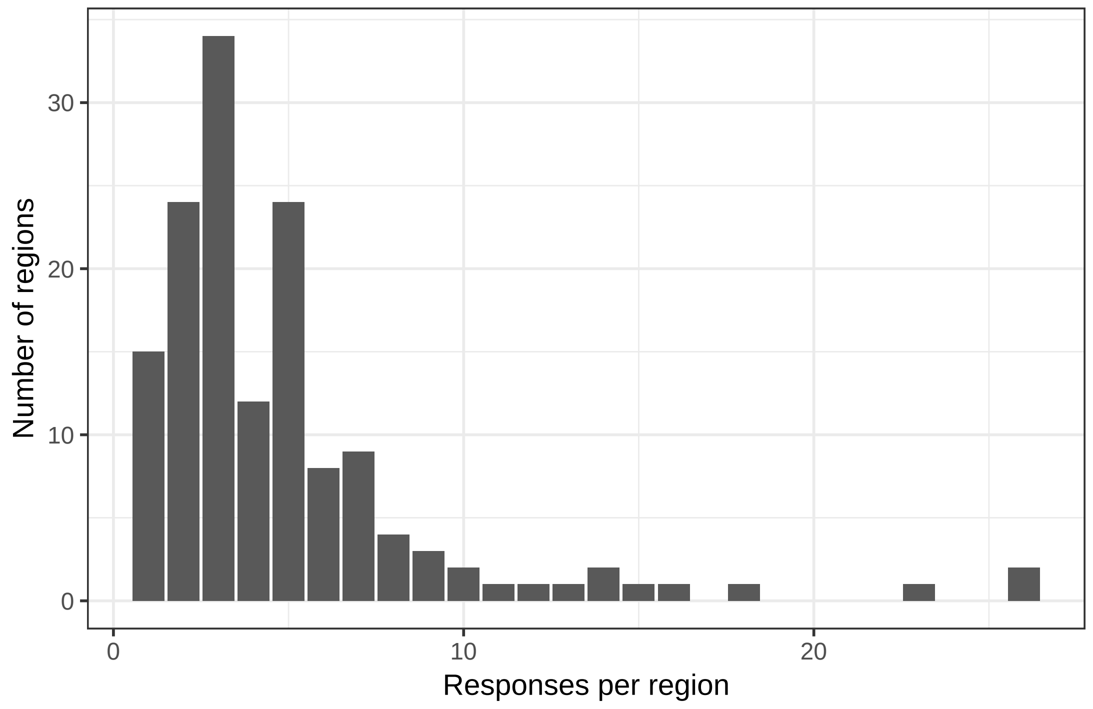
```r
ArchaeoGLOBE_Public_Data_df <-
  get_dataset(ArchaeoGLOBE_Public_Data_DOI)

# Only download the file we need here
ArchaeoGLOBE_Public_Data_df_files <-
  ArchaeoGLOBE_Public_Data_df$files[grepl("ARCHAEOGLOBE_PUBLIC_DATA|ARCHAEOGLOBE_CONSENSUS_ASSESSMENT",
                                          ArchaeoGLOBE_Public_Data_df$files$filename), ]

# read into local dir
walk(ArchaeoGLOBE_Public_Data_df_files$label,
     ~get_file(.x, ArchaeoGLOBE_Public_Data_DOI) %>%
       writeBin(paste0('data/raw-data/', .x)))

# read into the current environment
archaeoglobe <- read_csv('data/raw-data/ARCHAEOGLOBE_PUBLIC_DATA.tab')
consensus <- read_csv('data/raw-data/ARCHAEOGLOBE_CONSENSUS_ASSESSMENT.tab')


# repeat for shapefile
ArchaeoGLOBE_Regions_DOI <-
  "doi:10.7910/DVN/CQWUBI"

# get data frame of files on DV
ArchaeoGLOBE_Regions_df <-
  get_dataset(ArchaeoGLOBE_Regions_DOI)

# just download the shapefile we want
ArchaeoGLOBE_Regions_df_files <- ArchaeoGLOBE_Regions_df$files[ArchaeoGLOBE_Regions_df$files$filename =

# read into local dir
walk(ArchaeoGLOBE_Regions_df_files$label,
    ~get_file(.x, ArchaeoGLOBE_Regions_DOI) %>%
      writeBin(paste0('data/raw-data/', .x)))

unzip('data/raw-data/ArchaeGLOBE_Regions.zip',
      overwrite = TRUE,
      exdir = 'data/raw-data/ArchaeGLOBE_Regions')

# read into the current environment, and simplify the polygons for faster plotting
regions_unsimplified <-
  st_read('data/raw-data/ArchaeGLOBE_Regions/ArchaeGLOBE_Regions.shp',
          quiet = TRUE)
regions <- rmapshaper::ms_simplify(regions_unsimplified)
```

Exploratory plots

Before running any analyses, let's look at the data. How many responses do we have per region?

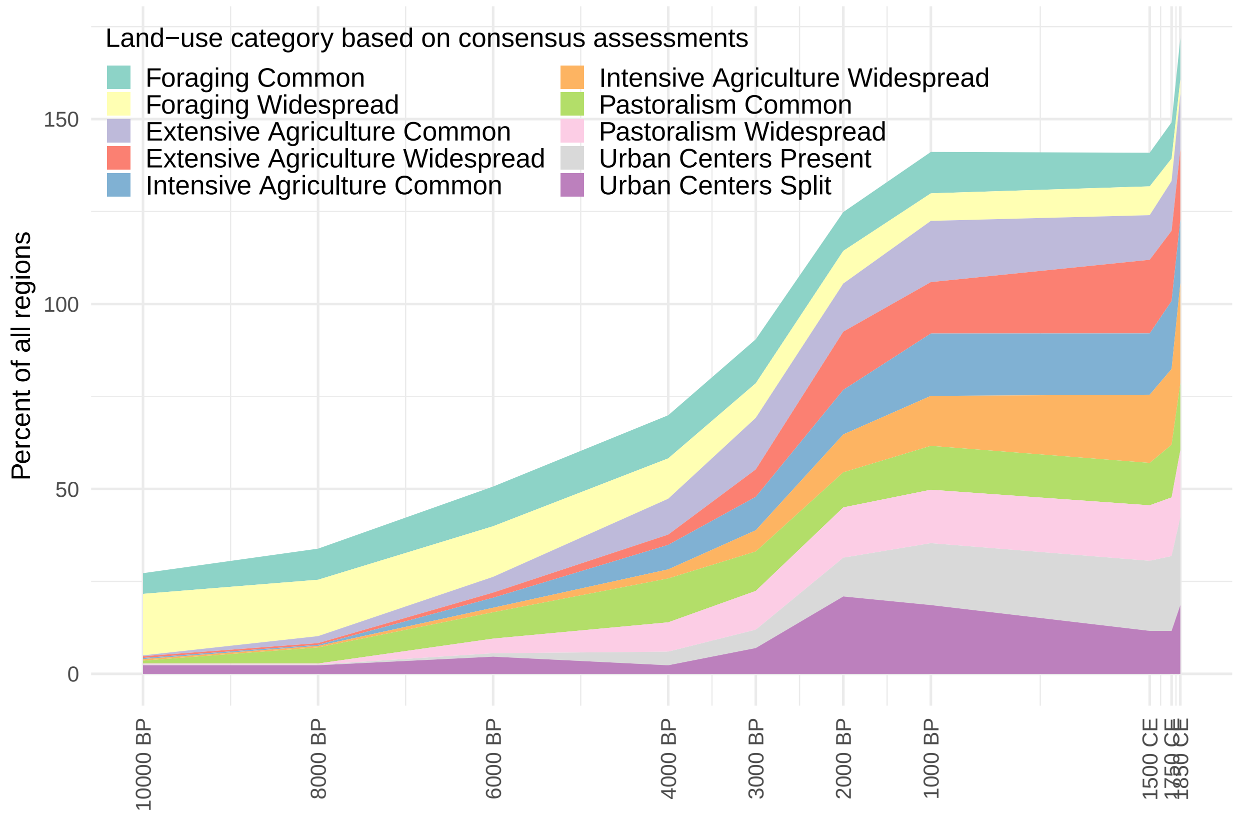
```r
response_counts <- archaeoglobe %>%
  group_by(REGION_ID) %>%
  count
```





Here is the cumulative summary of regions per land-use category based on consensus assessments (Common $>$ 1% to 20% regional land area; Widespread $>$ 20% regional land area).

```
cumsum_landuse_regions <-
consensus %>%
  select(Region, FHG_10KBP:URBAN_1850CE) %>%
  gather(variable, value, - Region) %>%
  filter(value %in% c("Widespread", "Common", "Split", "Present")) %>%
  separate(variable, into = c("land_use_category", "years_BP"), sep = "_") %>%
  mutate(land_use_category = ifelse(str_detect(land_use_category, "AGR"),
                                    str_replace_all(land_use_category ,
                                                    "AGR",
                                                    "AG"),
                                    land_use_category)) %>%
  mutate(land_use_category = case_when(
    land_use_category == "FHG"   ~  "Foraging",
    land_use_category == "EXAG"  ~  "Extensive Agriculture",
    land_use_category == "INAG"  ~  "Intensive Agriculture",
    land_use_category == "PAS"   ~  "Pastoralism",
    land_use_category == "URBAN" ~  "Urban Centers")
    ) %>%
  mutate(years_BP = ifelse(str_detect(tolower(years_BP), "kbp"),
                           -parse_number(years_BP) * 1000,
                           ifelse(str_detect(tolower(years_BP), "ce"),
                           parse_number(years_BP),
                           -parse_number(years_BP)))) %>%
  unite(land_use_category_consensus_assessments,
        c('land_use_category',
          'value'),
        sep = " ") %>%
  complete(land_use_category_consensus_assessments,
           nesting(years_BP)) %>%
  group_by(land_use_category_consensus_assessments,
           years_BP) %>%
  summarise(n = n()) %>%
  mutate(perc = n / sum(n) * 100)  %>%
  ungroup() %>%
  # make an ordered factor for nice plotting
  mutate(land_use_category_consensus_assessments =
           fct_relevel(land_use_category_consensus_assessments,
                       c(
                         "Foraging Common",
                         "Foraging Widespread",
                         "Extensive Agriculture Common",
                         "Extensive Agriculture Widespread",
                         "Intensive Agriculture Common",
                         "Intensive Agriculture Widespread",
                         "Pastoralism Common",
                         "Pastoralism Widespread",
                         "Urban Centers Present",
                         "Urban Centers Split"
                         )))


# make years label and breaks
years_label <- unique(ifelse(cumsum_landuse_regions$years_BP < 0,
                      str_glue('{-cumsum_landuse_regions$years_BP} BP'),
                      str_glue('{cumsum_landuse_regions$years_BP} CE')
                      ))

years_breaks <- unique(cumsum_landuse_regions$years_BP)

# draw the plot
ggplot(
  cumsum_landuse_regions,
  aes(years_BP,
      perc,
      fill = land_use_category_consensus_assessments)) +
  geom_area(position = 'stack') +
  scale_fill_brewer(palette = "Set3") +
  scale_x_continuous(labels = years_label,
                     breaks = years_breaks) +
  theme_minimal(base_size = 10) +
  theme(
    axis.text.x = element_text(
      angle = 90,
      hjust = 1,
      vjust = 0.5),
    #  x- and y- offsets from the bottom-left of the plot, ranging from 0 - 1.
    legend.position = c(0.4, 0.85),
    legend.text = element_text(size = 10),
    legend.key.size = unit(0.7, "line")) +
  guides(fill = guide_legend(ncol = 2,
                             title = "Land-use category based on consensus assessments")) +
  xlab("") +
  ylab("Percent of all regions")
```



```r
ggsave('figures/cumulative_sum_land_use.png', height = 5, width = 7)
```

Analysis functions

Define some analysis functions that we'll be using repeatedly in the analysis, so that we don't have to keep copying and pasting the same lines of code.

This function subsets the data to highlight a variable of interest, and converts it from a wide to a long "tidy" format to make analysis and plotting easier.

```r
preprocess <- function(prefix, categories){
  archaeoglobe %>% # start with the full ArcheoGlobe data
    # drop columns not related to the variable of interest
    select(c(CONTRIBUTR:LAND_AREA, starts_with(prefix))) %>%
    gather(time, value, starts_with(prefix)) %>% # one value per row
    mutate(time = parse_number(time) * -1, # convert time period labels to years
           value = ordered(value, levels = categories),
           cat_num = as.numeric(value)) %>%
    mutate_if(is.character, as.factor) # convert characters to factors
}
```

This function takes a data frame produced by the above function and fits GAM to the global trend and local deviations for each region, accounting for inter-observer variability. This function takes as arguments a preprocessed data frame containing time slices, regions, contributors, and the ordered categorical response variable transformed to a numeric vector.

```r
cores <- max(parallel::detectCores() / 2, 1) # physical cores for parallelization
cl <- parallel::makeCluster(cores)

fit_gam <- function(x, n_cats){
  bam(cat_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_LAB, bs = 'cs', m = 1) +
        # add back in region-specific intercepts
        REGION_LAB  +
        # model contributor as a random effect
        s(CONTRIBUTR, bs = 're'),
      data = x, # data frame to analyize
      family = ocat(R = n_cats), # ordered categorical with n levels
      # final 3 arguments just speed up the model fitting
      method = 'fREML',
      discrete = TRUE,
      cluster = cl)
}
```

This function extracts the estimated trends for each region, incorporating the global and regional splines as well as the region and contributor specific intercepts. Then it clusters these trends into 6 discrete clusters.

```r
extract_trends <-function(mod, n_clusters = 6){
  set.seed(1000) # set seed for reproducability of clusters
  archaeoglobe %>% # create dummy data for prediction in the following lines
    select(REGION_LAB) %>%
    group_by(REGION_LAB) %>%
    slice(1) %>%
    slice(rep(1:n(), each = 198)) %>%
    ungroup %>%
    mutate(time = rep_len(seq(-10000, -150, 50), n()),
           CONTRIBUTR = 'CYRBU') %>% # select an arbitrary contributor
    mutate(preds = predict(mod, .)) %>% # estimate trend lines
    mutate(preds = plogis(preds)) %>% # transform responses to [0,1] scale
    spread(time, preds) %>%
    # next is the actual kmeans clustering code
    mutate(cluster = kmeans(.[,-c(1,2)], n_clusters, iter.max = 100, nstart = 100)$cluster)
}
```

Analysis

Now we use the functions defined above on the ArchaeoGlobe data. For convenience, first define a data frame that lists the prefixes of the variables we are interested in (e.g. “EXP” for expertise) and the levels of the ordered factors associated with each variable. This will make it easier to quickly focus on a specific variable. The `tribble` command is simply a way to make a data frame by row rather than column, which makes the code easier to read.

```r
response_levels <- tribble(
  ~prefix, ~categories,
  'EXP', c('None', 'Low', 'High'),                  # Expertise
  'DQ', c('Unknown', 'Low', 'Moderate', 'Good'),    # Data Quality
  'HUNT', c('none', 'minimal (<1%)', 'common (1-20%)', 'widespread (>20%)'),
  'EXAG', c('none', 'minimal (<1%)', 'common (1-20%)', 'widespread (>20%)'),
  'INAG', c('none', 'minimal (<1%)', 'common (1-20%)', 'widespread (>20%)'),
  'PAST', c('none', 'minimal (<1%)', 'common (1-20%)', 'widespread (>20%)'),
  'URBN', c('Absent', 'Present')
)
```

Now map each of the above functions to each variable. This allows us to run the analysis for all variables of interest in a single step, and save all the outputs in a tibble format for easy plotting. If you're running this for the first time, it should take about 40 minutes to run on a Intel NUC with a 5th-gen Intel Core i7-5557U processor and 16gb of RAM running Linux.

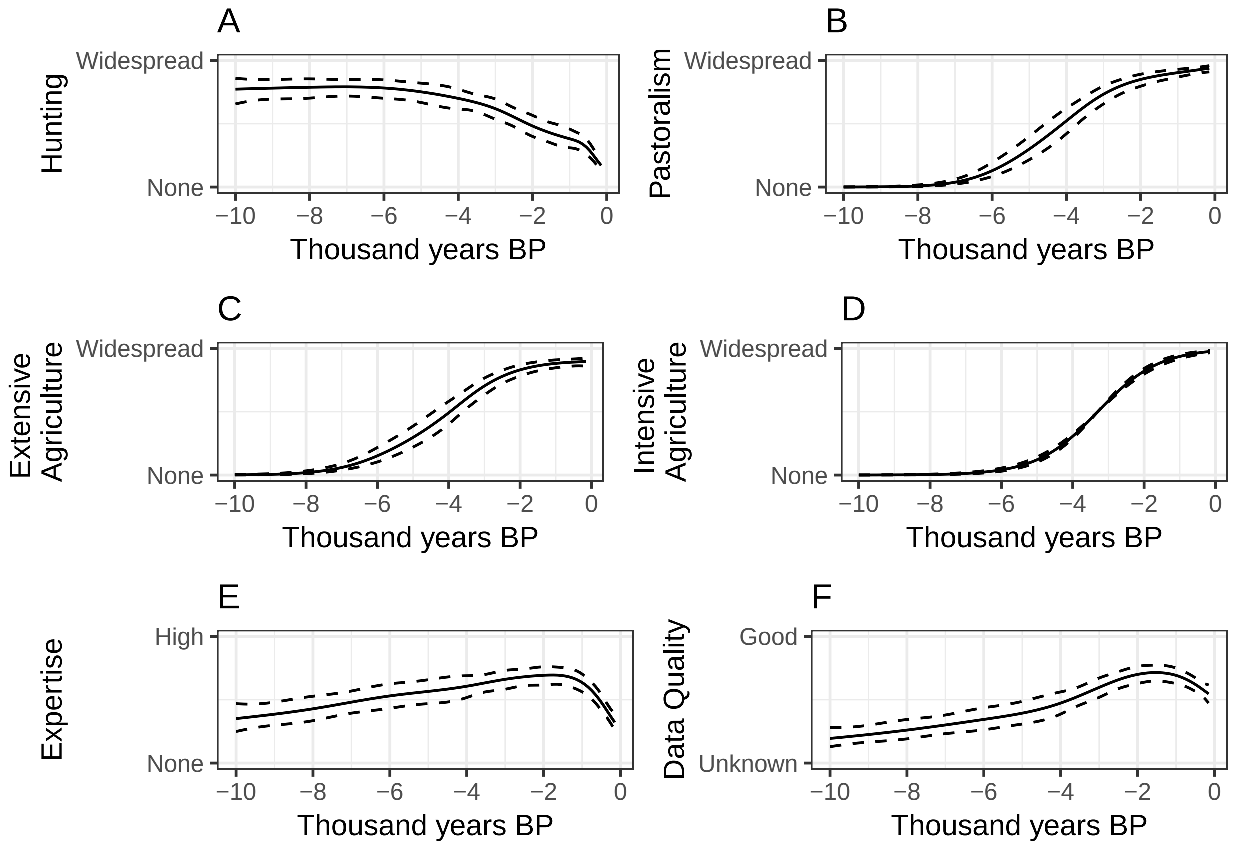
```
# A simple type of caching...
# Do we want to run the modelling code, or
# load a previously saved result from disk, or
# download a previously saved result from a repository?
# default is not run, then check if there is a saved file, and use that, or download

# devtools::install_github('centerforopenscience/osfr')
library(osfr)

rerun_time_consuming_analysis <- FALSE # FALSE means do not run the modelling code when knitting

if(rerun_time_consuming_analysis) {
  message("running the modelling code, this may take 30-50 min...")
  # go to the next chunk of code
} else {
  # check if there is a local file and if so, load it
  if(file.exists('data/derived-data/trend_dat.rda')) {
    # the file exists on the local disk, so just read it in
    message("Loading previously saved model results from disk...")
    trend_dat <- readRDS('data/derived-data/trend_dat.rda')
  } else {
  # we don't want to run the modelling code, and the result don't exist locally,
  # so download
  message("Downloading previously saved model results, takes 2-3 min...")
  trend_dat <- osf_retrieve_file("kcr2e") %>% osf_download('data/derived-data/trend_dat.rda')
  message("Loading the data downloaded from osf.io...")
  trend_dat <- readRDS('data/derived-data/trend_dat.rda')
  writeLines(paste0('trend_dat.rda downloded from https://osf.io/kcr2e/ on ', Sys.Date()), con = 'data/
  message("Done.")
  }
}

trend_dat <- response_levels %>%
  mutate(data = map2(prefix, categories, ~preprocess(.x,.y)),
         n_cats = map_dbl(categories, length),
         mod = map2(data, n_cats, fit_gam),
         trends = map(mod, extract_trends))

# save to disk
```

```
saveRDS(trend_dat,
        file = "data/derived-data/trend_dat.rda")

# write a note to indicate the provenance of this file
this_commit <-  git2r::revparse_single(git2r::repository('.'), "HEAD")
writeLines(paste0('trend_dat generated on ',
                   Sys.Date() , ' from archaeoglobe.Rmd at git commit ',
                    this_commit$sha, ' made by ',
                    this_commit$author$name, ' on ',
                    this_commit$author$when, " with the message '",
                    this_commit$summary, "'"),
           con = 'data/derived-data/README.md',
           sep = '')
message("Done.")
```

Results

Global Trends

First we plot out the global trends for each land use type. Please refer to the source .rmd file for the plotting code.



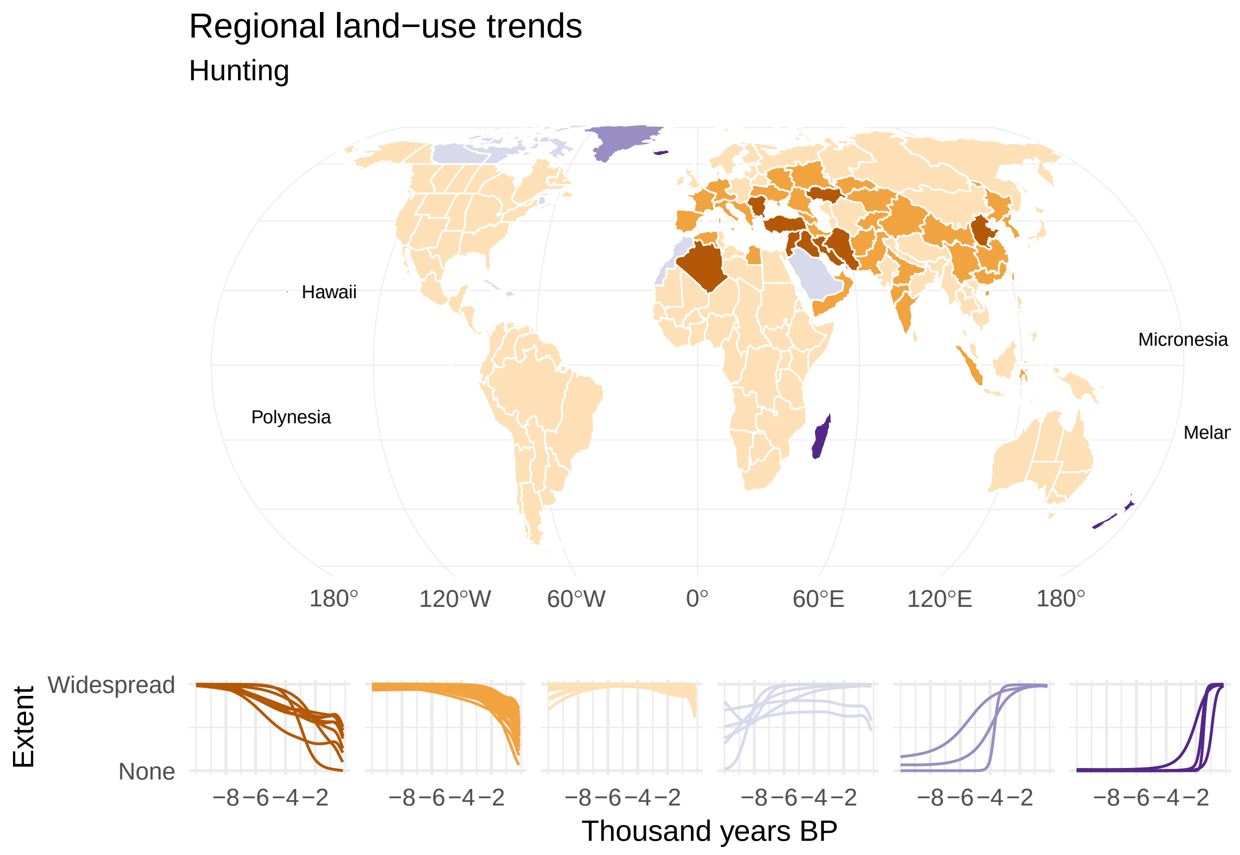


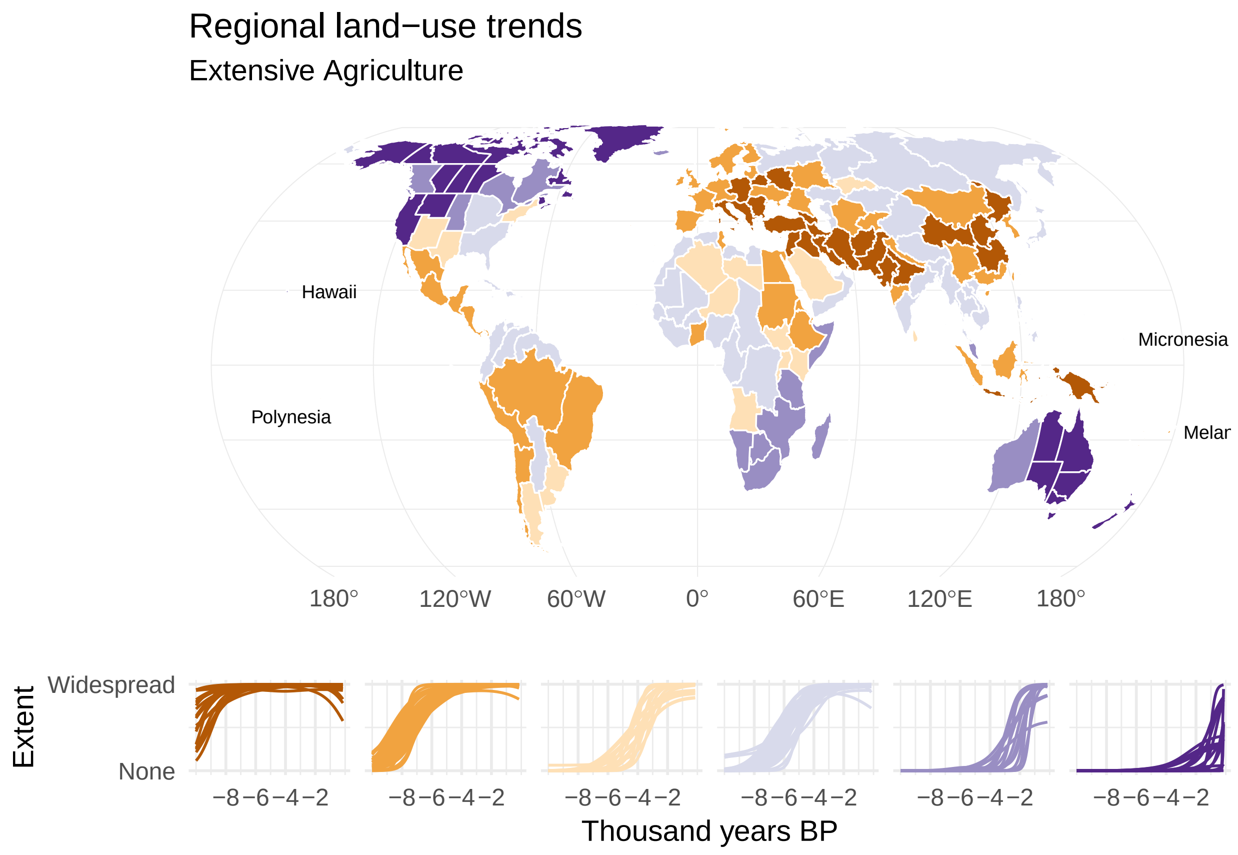
Figure 1: Regional trends in the areal extent of hunting. (A) Global trend (all regions) with 95% confidence interval. (B) Regional deviations from global trend, clustered via k-means. (C) Map of the local deviations from the global trend, same clusters as in B.

Hunting

The global trend in hunting shows constant high prevalence until around 6,000 years ago, after which there is a smooth decline until the present day when it is very rare. Mapping out the clusters reveals a clear east-west divide, which regions in Afro-eurasia seeing hunting earlier then the global mean, and regions in the Americas and Oceania seeing later peaks in hunting.

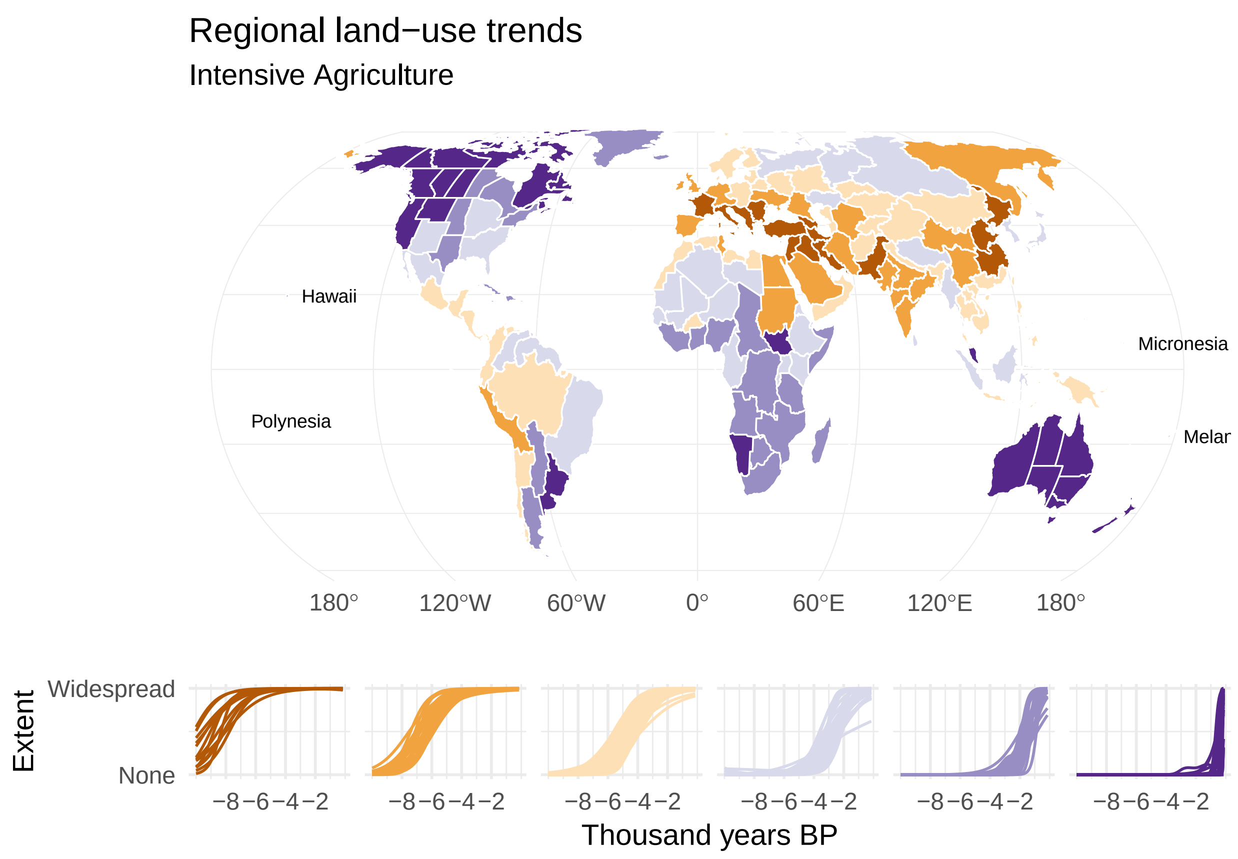
Extensive Agriculture

The global trends in the prevalence of pastoralism, extensive and intensive agriculture, and urbanism all follow a sigmoidal curve, which means the trend is linear on the scale of the linear predictor (the ordered categorical GAM uses a logit transform as a latent link function). This means that there is a simple increase in the probability of each land use type being prevalent over time.



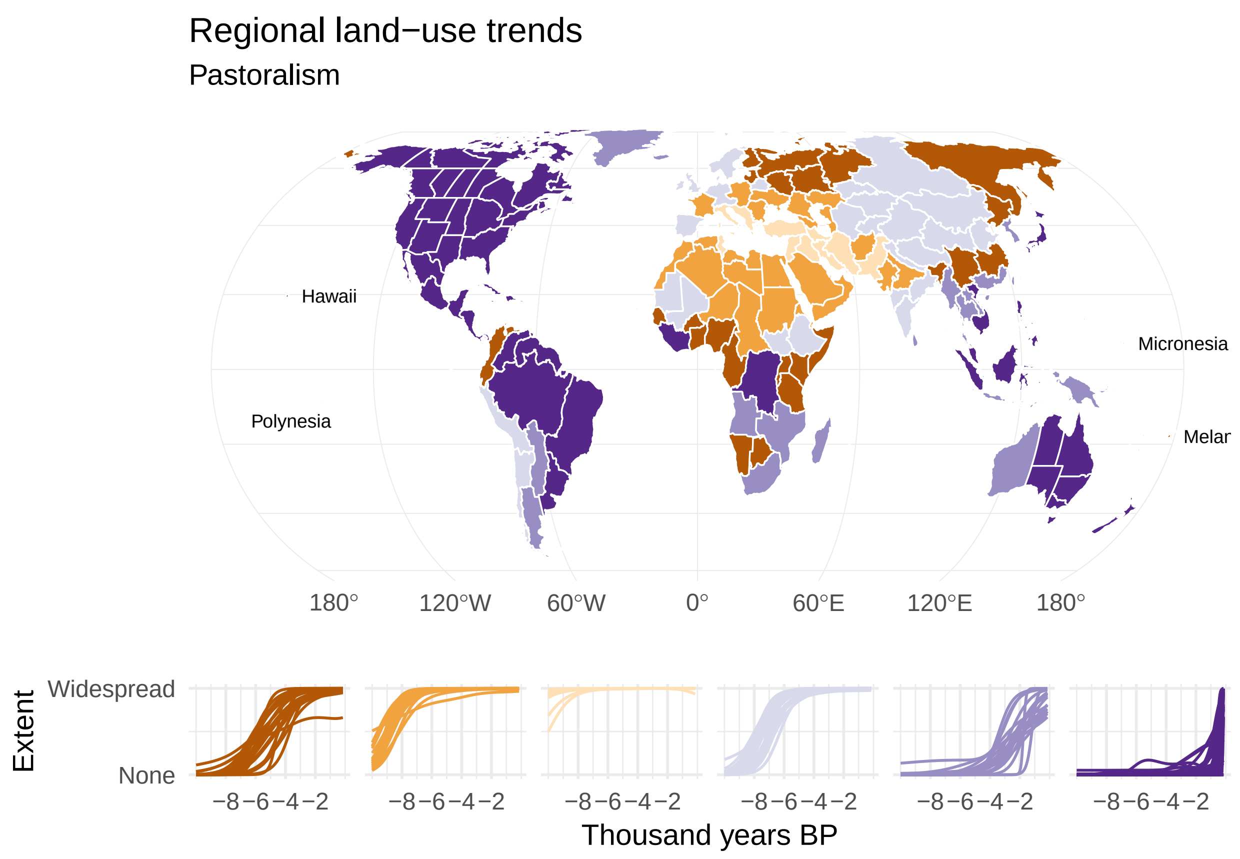
Intensive Agriculture

See above.

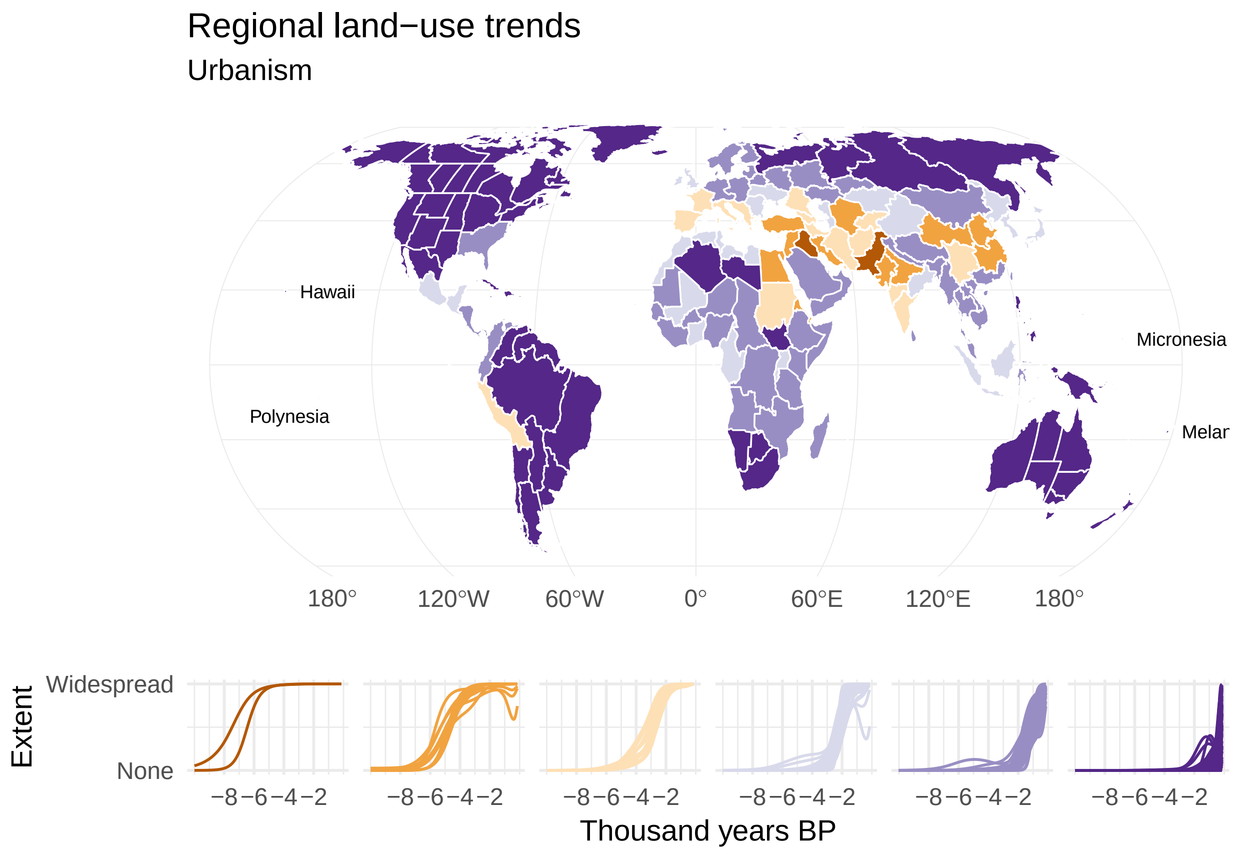


Pastoralism

See above.


Regional land−use trends
Pastoralism
Hawaii
Polynesia
Micronesia
Melar
180°
120°W
60°W
0°
60°E
120°E
180°
Extent
Widespread
None
−8 −6 −4 −2
Thousand years BP

Urbanism



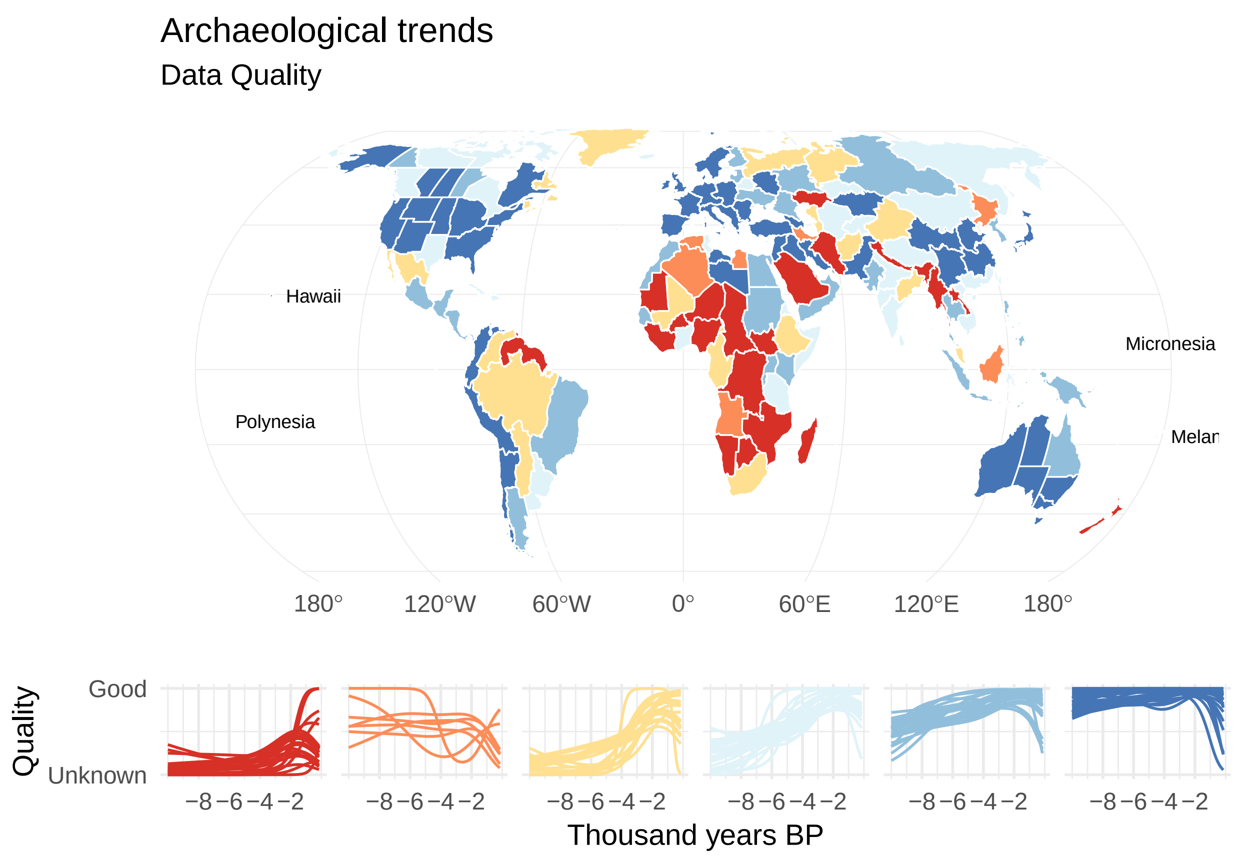
See above.

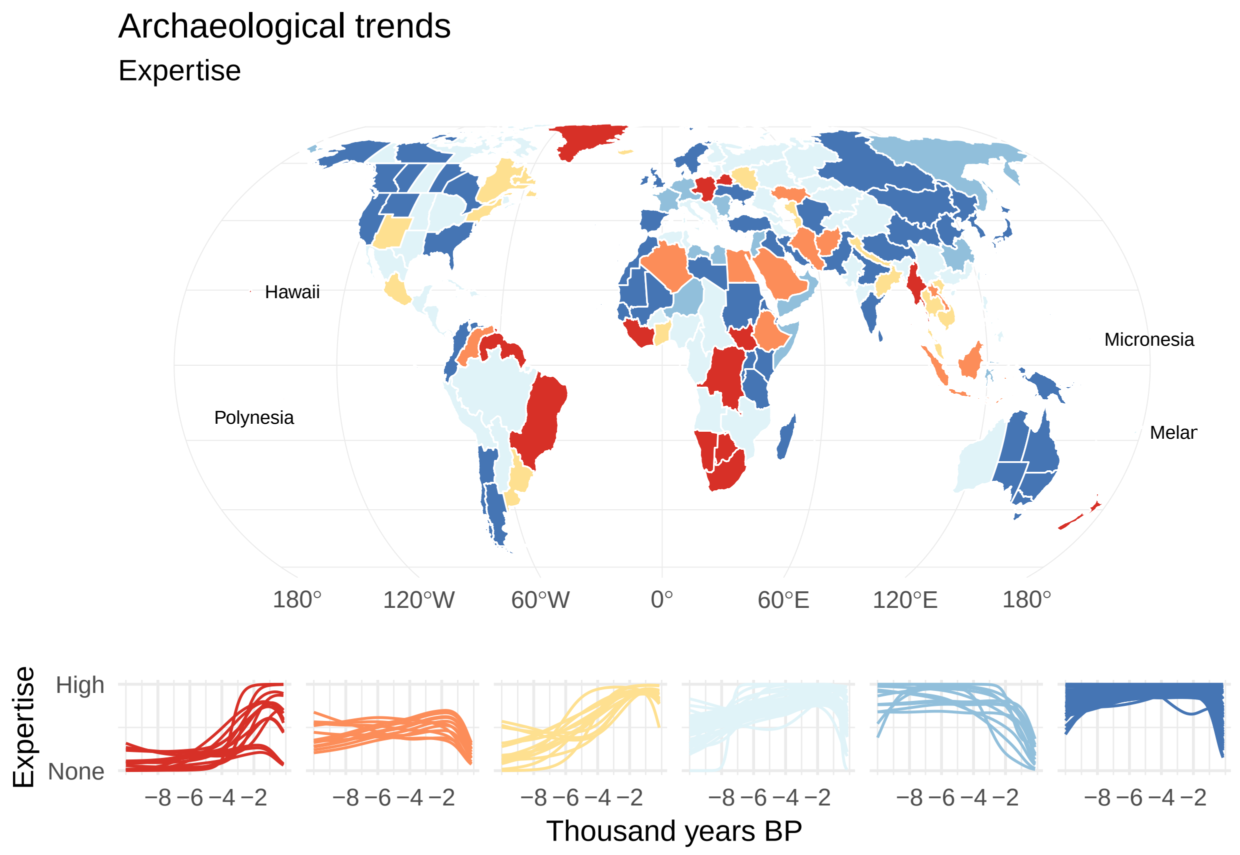
Expertise and Data Quality

How does self-professed level of expertise vary in each region over time? The global trend is a roughly linear increase in self-reported expertise from 10ka BP up to 2ka BP, then a falloff continuing to the present day. The present day expertise values are approximately the same as at 10ka BP. 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.

Now we cluster together the local deviations from the global trend using a k-means algorithm. The selection of 6 clusters is somewhat arbitrary, and is made simply based on visual comparisons of different cluster solutions with the goal making the results visually interpretable. The trajectories in these clusters are deviations from the global trend, so a horizontal line would indicate no deviation from the global trend.

The global trend in data quality 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. Unlike expertise, which reaches the same values at 10ky BP and present, data quality in the present day remains high in spite of the falloff in the last 2 millennia. Also note the confidence interval for the global trend is generally wider than for the expertise responses.


Archaeological trends
Data Quality
Hawaii
Polynesia
Micronesia
Melan
180°
120°W
60°W
0°
60°E
120°E
180°
Quality
Good
Unknown
−8 −6 −4 −2
Thousand years BP



Was the abandonment of widespread foraging more correlated closely with the spread of pastoralism than crop agriculture?

We generated a structural equation model to estimate parameters for a path schematic to test hypotheses about causal relationships between latent variables in the consensus land use data. Structural equation modeling is a multivariate statistical method for analyzing structural relationships that include latent variables (Bollen 1989, Beaujean 2014). The consensus data were recoded to ordinal factors and then input to a diagonally weighted least squares procedure to estimate the structural equation model parameters. To investigate weather the abandonment of widespread foraging was more correlated closely with the spread of pastoralism than crop agriculture, we modeled the consensus responses for foraging, pastoralism and crop agriculture for all regions during the middle and late Holocene. We used the lavaan R package (Rosseel 2012) to fit a model with a Model Fit Test Statistic of 75.199, 18 degrees of freedom. The model fit is good, as indicated by a Comparative Fit Index (CFI) 0.997 and a Root Mean Square Error of Approximation (RMSEA) of 0.148. The model output shows a regression estimate of -0.674 for foraging and pastoralism, compared to -0.574 for foraging and crop agriculture. The regression coefficient of foraging predicting pastoralism is more negative than foraging predicting crop agriculture, indicating that as foraging was abandoned it was more often replaced by pastoralism than crop agriculture.

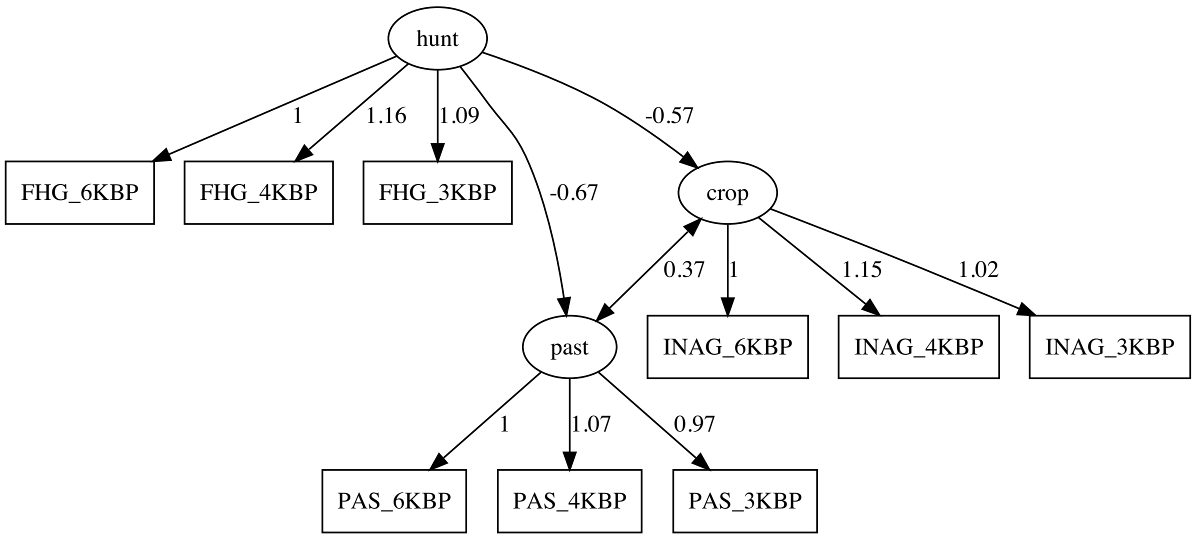
```
consensus_cat <-
  consensus %>%
  # convert consensus variables to ordinal factors
  mutate_at(.vars = vars(FHG_10KBP:URBAN_1850CE),
            .funs = funs(case_when(. == "Widespread" ~ 3,
                                   . == "Common" ~ 2,
                                   . == "Minimal" ~  1,
                                  . == "None" ~ 0))) %>%
  mutate_at(.vars = vars(FHG_10KBP:URBAN_1850CE),
            .funs = funs(factor(., ordered = TRUE)))

mod.sem1 <- '
# latent variable definitions
hunt  =~ FHG_6KBP + FHG_4KBP + FHG_3KBP
past  =~ PAS_6KBP + PAS_4KBP + PAS_3KBP
crop  =~ INAG_6KBP + INAG_4KBP + INAG_3KBP
# regressions
past ~ hunt
crop ~ hunt
# residual correlations
FHG_6KBP ~~ PAS_6KBP + INAG_6KBP
FHG_4KBP ~~ PAS_4KBP + INAG_4KBP
FHG_3KBP ~~ PAS_3KBP + INAG_3KBP
'

library(lavaan)
sem.fit <- sem(mod.sem1,
               data = consensus_cat[,-c(1:2)])
# inspect the summary
summary(sem.fit,
        fit.measures=TRUE,
        standardized = TRUE)

## lavaan 0.6-3 ended normally after 36 iterations
##
##   Optimization method                           NLMINB
##   Number of free parameters                         45
##
##   Number of observations                           146
##
##   Estimator                                       DWLS      Robust
##   Model Fit Test Statistic                      75.199     133.621
##   Degrees of freedom                                18          18
##   P-value (Chi-square)                           0.000       0.000
##   Scaling correction factor                                  0.589
##   Shift parameter                                            6.040
##     for simple second-order correction (Mplus variant)
##
## Model test baseline model:
##
##   Minimum Function Test Statistic            21562.343    9594.431
##   Degrees of freedom                                36          36
##   P-value                                        0.000       0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.997       0.988
##   Tucker-Lewis Index (TLI)                       0.995       0.976
##
##   Robust Comparative Fit Index (CFI)                            NA
##   Robust Tucker-Lewis Index (TLI)                               NA
```

```
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.148       0.210
##   90 Percent Confidence Interval          0.114  0.183       0.178  0.245
##   P-value RMSEA <= 0.05                          0.000       0.000
##
##   Robust RMSEA                                                  NA
##   90 Percent Confidence Interval                                NA     NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.084       0.084
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model         Unstructured
##   Standard Errors                            Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   hunt =~
##     FHG_6KBP          1.000                                0.860    0.860
##     FHG_4KBP          1.156    0.064   18.158    0.000    0.995    0.995
##     FHG_3KBP          1.094    0.044   24.624    0.000    0.941    0.941
##   past =~
##     PAS_6KBP          1.000                                0.947    0.947
##     PAS_4KBP          1.074    0.024   43.895    0.000    1.018    1.018
##     PAS_3KBP          0.973    0.015   64.871    0.000    0.922    0.922
##   crop =~
##     INAG_6KBP         1.000                                0.901    0.901
##     INAG_4KBP         1.153    0.050   23.244    0.000    1.038    1.038
##     INAG_3KBP         1.021    0.040   25.725    0.000    0.920    0.920
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   past ~
##     hunt             -0.674    0.062  -10.795    0.000   -0.612   -0.612
##   crop ~
##     hunt             -0.574    0.071   -8.134    0.000   -0.548   -0.548
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .FHG_6KBP ~~
##    .PAS_6KBP          0.145    0.060    2.406    0.016    0.145    0.888
##    .INAG_6KBP        -0.113    0.077   -1.461    0.144   -0.113   -0.511
##  .FHG_4KBP ~~
##    .PAS_4KBP          0.084    0.026    3.168    0.002    0.084    4.273
##    .INAG_4KBP         0.059    0.034    1.733    0.083    0.059    2.024
##  .FHG_3KBP ~~
##    .PAS_3KBP         -0.070    0.022   -3.237    0.001   -0.070   -0.532
##    .INAG_3KBP        -0.059    0.033   -1.768    0.077   -0.059   -0.442
##  .past ~~
##    .crop              0.373    0.054    6.881    0.000    0.661    0.661
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .FHG_6KBP          0.000                               0.000    0.000
##    .FHG_4KBP          0.000                               0.000    0.000
##    .FHG_3KBP          0.000                               0.000    0.000
##    .PAS_6KBP          0.000                               0.000    0.000
##    .PAS_4KBP          0.000                               0.000    0.000
##    .PAS_3KBP          0.000                               0.000    0.000
##    .INAG_6KBP         0.000                               0.000    0.000
##    .INAG_4KBP         0.000                               0.000    0.000
##    .INAG_3KBP         0.000                               0.000    0.000
##     hunt              0.000                               0.000    0.000
##    .past              0.000                               0.000    0.000
##    .crop              0.000                               0.000    0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     FHG_6KBP|t1      -1.541    0.164   -9.388    0.000   -1.541   -1.541
##     FHG_6KBP|t2      -1.063    0.129   -8.271    0.000   -1.063   -1.063
##     FHG_6KBP|t3      -0.103    0.104   -0.990    0.322   -0.103   -0.103
##     FHG_4KBP|t1      -1.738    0.187   -9.287    0.000   -1.738   -1.738
##     FHG_4KBP|t2      -0.752    0.116   -6.510    0.000   -0.752   -0.752
##     FHG_4KBP|t3       0.173    0.105    1.649    0.099    0.173    0.173
##     FHG_3KBP|t1      -1.822    0.199   -9.155    0.000   -1.822   -1.822
##     FHG_3KBP|t2      -0.580    0.111   -5.243    0.000   -0.580   -0.580
##     FHG_3KBP|t3       0.332    0.106    3.128    0.002    0.332    0.332
##     PAS_6KBP|t1       0.369    0.107    3.456    0.001    0.369    0.369
##     PAS_6KBP|t2       0.775    0.116    6.665    0.000    0.775    0.775
##     PAS_6KBP|t3       1.305    0.144    9.085    0.000    1.305    1.305
##     PAS_4KBP|t1       0.000    0.104    0.000    1.000    0.000    0.000
##     PAS_4KBP|t2       0.260    0.105    2.472    0.013    0.260    0.260
##     PAS_4KBP|t3       0.871    0.120    7.274    0.000    0.871    0.871
##     PAS_3KBP|t1      -0.086    0.104   -0.825    0.410   -0.086   -0.086
##     PAS_3KBP|t2       0.155    0.105    1.484    0.138    0.155    0.155
##     PAS_3KBP|t3       0.664    0.113    5.881    0.000    0.664    0.664
##     INAG_6KBP|t1      1.005    0.126    7.998    0.000    1.005    1.005
##     INAG_6KBP|t2      1.390    0.150    9.250    0.000    1.390    1.390
##     INAG_6KBP|t3      2.043    0.238    8.589    0.000    2.043    2.043
##     INAG_4KBP|t1      0.406    0.107    3.783    0.000    0.406    0.406
##     INAG_4KBP|t2      0.871    0.120    7.274    0.000    0.871    0.871
##     INAG_4KBP|t3      1.738    0.187    9.287    0.000    1.738    1.738
##     INAG_3KBP|t1      0.120    0.104    1.154    0.248    0.120    0.120
##     INAG_3KBP|t2      0.520    0.109    4.759    0.000    0.520    0.520
##     INAG_3KBP|t3      1.305    0.144    9.085    0.000    1.305    1.305
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .FHG_6KBP          0.260                               0.260    0.260
##    .FHG_4KBP          0.011                               0.011    0.011
##    .FHG_3KBP          0.114                               0.114    0.114
##    .PAS_6KBP          0.103                               0.103    0.103
##    .PAS_4KBP         -0.035                              -0.035   -0.035
##    .PAS_3KBP          0.151                                   0.151    0.151
##    .INAG_6KBP         0.188                                   0.188    0.188
##    .INAG_4KBP        -0.078                                  -0.078   -0.078
##    .INAG_3KBP         0.154                                   0.154    0.154
##     hunt              0.740    0.058   12.744    0.000    1.000    1.000
##    .past              0.561    0.048   11.709    0.000    0.626    0.626
##    .crop              0.568    0.060    9.466    0.000    0.700    0.700
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     FHG_6KBP          1.000                               1.000    1.000
##     FHG_4KBP          1.000                               1.000    1.000
##     FHG_3KBP          1.000                               1.000    1.000
##     PAS_6KBP          1.000                               1.000    1.000
##     PAS_4KBP          1.000                               1.000    1.000
##     PAS_3KBP          1.000                               1.000    1.000
##     INAG_6KBP         1.000                               1.000    1.000
##     INAG_4KBP         1.000                               1.000    1.000
##     INAG_3KBP         1.000                               1.000    1.000
```

```
modificationindices(sem.fit, sort = TRUE)
```

```
##           lhs op       rhs     mi    epc sepc.lv sepc.all sepc.nox
## 87       past =~  FHG_3KBP 53.302 -0.636  -0.602   -0.602   -0.602
## 97   FHG_6KBP ~~  FHG_4KBP 51.422  0.513   0.513    1.006    1.006
## 86       past =~  FHG_4KBP 41.727  0.724   0.686    0.686    0.686
## 85       past =~  FHG_6KBP 39.788  0.594   0.563    0.563    0.563
## 93       crop =~  FHG_3KBP 39.644 -0.574  -0.518   -0.518   -0.518
## 103  FHG_4KBP ~~  FHG_3KBP 38.895 -0.568  -0.568   -1.682   -1.682
## 91       crop =~  FHG_6KBP 31.944  0.529   0.477    0.477    0.477
## 92       crop =~  FHG_4KBP 26.354  0.590   0.531    0.531    0.531
## 81       hunt =~  PAS_3KBP 18.874  0.513   0.441    0.441    0.441
## 100  FHG_6KBP ~~  PAS_3KBP 13.299  0.312   0.312    1.577    1.577
## 109  FHG_3KBP ~~  PAS_4KBP 13.250 -0.208  -0.208   -0.615   -0.615
## 99   FHG_6KBP ~~  PAS_4KBP 11.469  0.274   0.274    0.537    0.537
## 95       crop =~  PAS_4KBP 11.076 -0.518  -0.467   -0.467   -0.467
## 82       hunt =~ INAG_6KBP 10.157 -0.376  -0.323   -0.323   -0.323
## 108  FHG_3KBP ~~  PAS_6KBP 10.041 -0.188  -0.188   -1.735   -1.735
## 126 INAG_4KBP ~~ INAG_3KBP  8.703  0.208   0.208    0.528    0.528
## 105  FHG_4KBP ~~  PAS_3KBP  7.269  0.203   0.203    0.524    0.524
## 88       past =~ INAG_6KBP  6.982  0.375   0.355    0.355    0.355
## 84       hunt =~ INAG_3KBP  6.818  0.303   0.261    0.261    0.261
## 124 INAG_6KBP ~~ INAG_4KBP  6.305 -0.180  -0.180   -0.415   -0.415
## 79       hunt =~  PAS_6KBP  5.740 -0.241  -0.208   -0.208   -0.208
## 90       past =~ INAG_3KBP  4.959 -0.335  -0.318   -0.318   -0.318
## 98   FHG_6KBP ~~  FHG_3KBP  4.860 -0.182  -0.182   -1.053   -1.053
## 101  FHG_6KBP ~~ INAG_4KBP  4.358  0.187   0.187    0.367    0.367
## 110  FHG_3KBP ~~ INAG_6KBP  3.881 -0.176  -0.176   -1.198   -1.198
## 83       hunt =~ INAG_4KBP  3.732  0.252   0.217    0.217    0.217
## 89       past =~ INAG_4KBP  3.558 -0.334  -0.317   -0.317   -0.317
## 102  FHG_6KBP ~~ INAG_3KBP  3.416  0.155   0.155    0.771    0.771
## 114  PAS_6KBP ~~ INAG_6KBP  2.252  0.113   0.113    0.810    0.810
## 106  FHG_4KBP ~~ INAG_6KBP  2.189 -0.140  -0.140   -0.323   -0.323
## 120  PAS_4KBP ~~ INAG_3KBP  1.959 -0.107  -0.107   -0.272   -0.272
## 117  PAS_4KBP ~~  PAS_3KBP  1.956  0.077   0.077    0.197    0.197
## 80       hunt =~  PAS_4KBP  1.876  0.155   0.133    0.133    0.133
## 116  PAS_6KBP ~~ INAG_3KBP  1.778 -0.112  -0.112   -0.887   -0.887
## 111  FHG_3KBP ~~ INAG_4KBP  1.463 -0.087  -0.087   -0.259   -0.259
## 119  PAS_4KBP ~~ INAG_4KBP  1.172 -0.074  -0.074   -0.074   -0.074
## 112  PAS_6KBP ~~  PAS_4KBP  1.136 -0.061  -0.061   -0.191   -0.191
## 122  PAS_3KBP ~~ INAG_4KBP  1.084  0.064   0.064    0.165    0.165
## 107  FHG_4KBP ~~ INAG_3KBP  0.893  0.075   0.075    0.192    0.192
## 96       crop =~  PAS_3KBP  0.780  0.116   0.104    0.104    0.104
## 123  PAS_3KBP ~~ INAG_3KBP  0.504  0.049   0.049    0.322    0.322
## 125 INAG_6KBP ~~ INAG_3KBP  0.346 -0.039  -0.039   -0.232   -0.232
## 118  PAS_4KBP ~~ INAG_6KBP  0.326  0.043   0.043    0.099    0.099
## 121  PAS_3KBP ~~ INAG_6KBP  0.229  0.041   0.041    0.246    0.246
## 104  FHG_4KBP ~~  PAS_6KBP  0.188  0.032   0.032    0.101    0.101
## 115  PAS_6KBP ~~ INAG_4KBP  0.149 -0.026  -0.026   -0.081   -0.081
## 113  PAS_6KBP ~~  PAS_3KBP  0.147 -0.021  -0.021   -0.172   -0.172
## 94       crop =~  PAS_6KBP  0.084  0.040   0.036    0.036    0.036
```

```r
# look at the diagram:
library(lavaanPlot)
png("figures/lavaanPlot.pdf")
lavaanPlot(name="",
           model=sem.fit,
           coefs=TRUE,
           covs=TRUE,
           sig=1.00)
```



```r
dev.off()
```

```
## pdf
## 2
```

Structural equation modelling shows that a causal inference of abandonment of widespread foraging correlating more closely with the spread of pastoralism than crop agriculture is consistent with the data.

Bollen, K. A. (1989). Structural Equations with Latent Variables (Wiley Series in Probability and Statistics, Canada

Beaujean, A. A. (2014). Latent variable modeling using R: A step-by-step guide. Routledge.

Rosseel, Y. (2012). Lavaan: An R package for structural equation modeling and more. Version 0.5–12 (BETA). Journal of statistical software, 48(2), 1-36.