Analysis Codes for ESS

read me

Below are the R scripts I used to analyze the European Social Survey (Wave 3) for Study 2 of my writing sample.

In total, I wrote five R scripts:

1. `clean_ESS.R`
2. `explore_ESS.R`
3. `efa_ESS.R`
4. `analysis_ESS.R`
5. `util_ESS.R`

Here, I include only scripts **3** (`efa_ESS.R`) and **4** (`analysis_ESS.R`), as they are probably more interesting to look at than the rest.

efa_ESS.R

```
################################################
## Setup
################################################
pkgs <- c("tidyverse",
          "psych", # fa
          "GPArotation", # oblimin / promax
          "nFactors", # parallel analysis
          "factoextra", # PCA
          "EGAnet", # Exploratory Graph Analysis
          "blavaan", # Bayesian EFA
          "dplyr", "purrr", "kableExtra")
to_get <- setdiff(pkgs, rownames(installed.packages()))
if(length(to_get)) install.packages(to_get)
lapply(pkgs, require, character.only = TRUE)

source("R/util_ESS.R")
source("data/dv_ESS.R")

data <- read_csv("data/ESS3_clean_with_indices.csv", show_col_types = FALSE)
```

```r
################################################
## Choose DVs
################################################
dv_choice <- choose_dvs("fa", "domain") # custom function
                                        # returns list(vars = ..., labels = ...)
                                        # based on the dictionary below

# dv_desc_short_fa <- list(
#   richness = list(
#     impdiff = "Seek new things",
#     ipadvnt = "Seek exciting life",
#     ipcrtiv = "Seek new ideas",
#     ipudrst = "Seek different perspectives",
#     lrnnew  = "Love learning new things"
#   ),
#   positive_affect = list(
#     wrhpp = "Happy last week",
#     enjlf = "Enjoyed life last week"
#   ),
#   negative_affect = list(
#     fltsd  = "Sad last week",
#     fltanx = "Anxious last week"
#   ),
#   life_satisfaction = list(
#     happy   = "Happy (11-point)",
#     stflife = "Satisfied nowadays",
#     stflfsf = "Satisfied so far",
#     lfcllk  = "Life close to ideal"
#   ),
#   meaning = list(
#     dngval = "Worthwhile",
#     accdng = "Accomplished"
#   )
# )

outcomes <- dv_choice$vars
labels  <- clean_labels(dv_choice$labels)

data_complete <- data %>%
  dplyr::select(all_of(outcomes)) %>%
  tidyr::drop_na()

################################################
## Factors
################################################
########################
## Parallel & Factor analysis
########################
run_efa <- function(df, vars, names, nfactors = NULL, n.iter = 100){
  tmp <- df %>% select(all_of(vars)) %>% drop_na()
  colnames(tmp) <- names
  if(is.null(nfactors)){
    nfactors <- fa.parallel(tmp, fa = "fa", n.iter = n.iter,
```

```
                              show.legend = FALSE)$nfact
  }
  fa(tmp, nfactors = nfactors, fm = "pa", rotate = "oblimin")
}

efa <- run_efa(data_complete, outcomes, labels)
print(efa, cut = .30)

########################
## nfactors
########################
ev <- eigen(cor(data_complete))$values
ap <- nFactors::parallel(
  subject = nrow(data_complete),
  var     = ncol(data_complete),
  rep     = 100,
  cent    = .05
)
ns <- nScree(x = ev, aparallel = ap$eigen$qevpea)
plotnScree(ns)

########################
## PCA loading plot - each item as an arrow (Components 1-2)
########################
pca_data <- data_complete
colnames(pca_data) <- labels

res.pca <- prcomp(pca_data, scale. = TRUE)

fviz_eig(res.pca, addlabels = TRUE) +
  ggtitle("Scree plot")

fviz_pca_var(
  res.pca, axes = c(1,2),
  col.var = "contrib",
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
  repel = TRUE) +
  ggtitle("PCA loadings (PC1-PC2)")

########################
## EGA
########################
ega_data <- data_complete
colnames(ega_data) <- labels

ega_out <- EGAnet::EGA(
  ega_data,
  type  = "ordinal", # DVs are not normally distributed
  model = "glasso",
  plot.EGA = TRUE
)
print(ega_out$wc)
```

```
########################
## Cronbach Alpha
########################
alpha_results <- imap_dfr(dv_desc_short_fa, ~{
  domain     <- .y # e.g., "seek_richness"
  short_list <- .x # e.g., c(impdiff="Seek new things", ...)

  sub_df <- data[, names(short_list), drop = FALSE]
  alpha_out <- psych::alpha(sub_df, check.keys = TRUE)

  tibble(
    Domain    = domain,
    n_items   = length(short_list),
    Alpha     = round(alpha_out$total$raw_alpha, 3)
  )
})

alpha_results %>%
  kbl(
    caption   = "Cronbach's Alpha per Domain (ESS)",
    col.names = c("Domain", "N (items)", "Alpha"),
    booktabs  = TRUE
  ) %>%
  kable_styling(latex_options = c("hold_position"), full_width = FALSE)
```

analysis_ESS.R

```
################################################
## Setup
################################################
pkgs <- c("tidyverse","lmerTest","modelsummary","performance", "lme4", "jtools", "randomForest")
lapply(pkgs, require, character.only = TRUE)

source("R/util_ESS.R")
source("data/dv_ESS.R")
data <- read_csv("data/ESS3_clean_with_indices.csv",
                 show_col_types = FALSE)

data <- data %>%
  mutate(
    gndr      = factor(gndr,      levels = c("Male", "Female")),
    maritala  = factor(maritala,  levels = c(
      "Married",
      "Civil-partner",
      "Separated m",
      "Separated cp",
      "Divorced",
      "Widowed",
      "Ex-CP dissolved",
      "Ex-CP died",
```

```
      "Never married"
  )) %>%
    forcats::fct_relevel("Married"),  # Married as the reference level
  chldhhe   = factor(chldhhe,   levels = c("Yes", "No"))
)

################################################################
## HLM
################################################################
dvs <- c("fltbrd_z",
         "seek_richness_z", "positive_affect_z", "negative_affect_z", "life_satisfaction_z", "meaning_z
         "jbintr_z", "jbstrs_z", "stfjb_z")

########################
## Fixed effect blocks
########################
blocks <- list(
  education   = "edulvla",
  demo        = c("gndr","age","health", "hincfel"),
  labor       = c("pdwrk","uempla","uempli","dsbld","rtrd","cmsrv","hswrk","dngoth"),
  family      = c("maritala","chldhhe"),
  parentsEdu  = c("edulvlfa","edulvlma"),
  religiosity = "ever_religious",
  personality = "seek_richness_z"
)

########################
## Build models
########################
model_seq <- list(
  m1 = c("education"), # simple
  m2 = c("education","demo", "labor", "family"), # full
  m3 = c("education","demo", "labor", "family", "parentsEdu", "religiosity"),  # robust 1
  m4 = c("education","demo", "labor", "family", "personality") # robust 2
)

formula_list <- fit_models(
  data      = dat,
  blocks    = blocks,
  seq_list  = model_seq,
  random    = "(1 | cntry)"
)

########################
## Fit models
########################
all_fits <- purrr::imap(dvs, \(dv, idx){
  apply_dv(dv, formula_list, data)
}) |> setNames(dvs)

summary(all_fits$fltbrd_z$m2)
summary(all_fits$stfjb_z$m3)
```

```r
########################
## Summarize coefficients (for robust 1 / m3)
########################
get_ed_coef <- function(fit){
  # get the education row
  ed <- broom::tidy(fit, conf.int = TRUE) |>
    filter(term == "edulvla")

  # compute R2
  r2_vals <- suppressWarnings(performance::r2(fit))

  # add N, R2m, R2c
  ed |>
    mutate(
      N   = nobs(fit),
      R2m = round(r2_vals$R2_marginal,    3),
      R2c = round(r2_vals$R2_conditional, 3)
    ) |>
    select(estimate, std.error, p.value, N, R2m, R2c)
}

results_tbl <- purrr::imap_dfr(
  all_fits,
  \(fit_list, dv) {
    out <- get_ed_coef(fit_list$m3)
    out$DV <- dv
    out
  }
) |>
  relocate(DV)

print(results_tbl, n = Inf)

# write.csv(results_tbl, "output_ESS/educ_effects_m3.csv", row.names = FALSE)

########################
## Latex table
########################
library(kableExtra)

############
## Helper objects
############
dv_labels <- c(
  fltbrd_z            = "Felt bored (r)",
  jbintr_z            = "Job interesting",
  seek_richness_z     = "Seek richness",
  life_satisfaction_z = "Life satisfaction",
  stfjb_z             = "Job satisfaction",
  positive_affect_z   = "Positive affect",
  negative_affect_z   = "Negative affect (r)",
  jbstrs_z            = "Job stressful (r)",
  meaning_z           = "Meaning"
```

```
)

domain_lkp <- list(
  `Psychological richness` = c("Felt bored (r)", "Job interesting", "Seek richness"),
  `Subjective well-being`  = c("Life satisfaction", "Job satisfaction",
                               "Positive affect", "Negative affect (r)", "Job stressful (r)"),
  `Meaning in life`        = "Meaning"
)

format_pval <- function(p){
  dplyr::case_when(
    p < .001 ~ "$< .001$",
    p < .01  ~ "$< .01$",
    p < .05  ~ "$< .05$",
    TRUE     ~ sprintf("$%.2f$", p)
  )
}

############
## Build table for one model
############

get_table <- function(all_fits, model_name){

  purrr::imap_dfr(all_fits, \(fit_list, dv){

    mdl <- fit_list[[model_name]]
    r2  <- suppressWarnings(performance::r2(mdl))

    dom <- names(domain_lkp)[vapply(domain_lkp,
                                    \(v) dv_labels[dv] %in% v,
                                    logical(1))]
    if (length(dom) == 0) dom <- ""

    broom::tidy(mdl, conf.int = TRUE) |>
      dplyr::filter(term == "edulvla") |>
      dplyr::mutate(
        Domain   = dom,
        DV = dv_labels[dv],
        Estimate_raw = estimate,
        Estimate = sprintf("%.3f", estimate),
        SE       = sprintf("%.3f", std.error),
        p        = p.value,
        p_fmt    = format_pval(p.value),
        N        = nobs(mdl),
        R2m      = sprintf("%.3f", r2$R2_marginal),
        R2c      = sprintf("%.3f", r2$R2_conditional),
        stars    = dplyr::case_when(
          p < .001 ~ "***",
          p < .01  ~ "**",
          p < .05  ~ "*",
          TRUE     ~ ""
        )
```

```
      )
  }) |>
    dplyr::arrange(factor(DV, levels = unlist(domain_lkp))) |>
    dplyr::mutate(
      Estimate = kableExtra::cell_spec(
        paste0(Estimate, stars),
        format = "latex",
        bold   = p < .05)
    )
}


############
## Kables for all four models
############
model_names <- c(m1 = "Simple Model", m2 = "Full Model",
                 m3 = "Robust Model 1", m4 = "Robust Model 2")

for (m in names(model_names)) {

  tab <- get_table(all_fits, m)

  tab <- tab |>
    dplyr::mutate(
      DV = ifelse(DV %in% c("Job interesting", "Job satisfaction", "Job stressful (r)"),
                  kableExtra::cell_spec(DV, format = "latex", italic = TRUE),
                  DV)
    )

  if (m == "m4") {
    tab <- tab |> dplyr::filter(DV != "Seek richness")
  }

  kbl(tab |> dplyr::select(Domain, DV, Estimate, SE, p_fmt, N, R2m, R2c),
      format    = "latex",
      booktabs  = TRUE,
      escape    = FALSE,
      align     = c("l","l","r","r","c","r","r","r"),
      caption   = paste("Relationship between education and well-being -",
                        model_names[[m]]),
      col.names = c("", "DV", "Estimate", "S.E.", "$p$", "N",
                    "$R^{2}_{m}$", "$R^{2}_{c}$")
  ) |>
    kable_styling(latex_options = c("hold_position"),
                  font_size = 9) |>
    column_spec(1, width = "2.5cm") |>
    collapse_rows(columns = 1, valign = "top") |>
    footnote(
      general = c("All variables are standardized. Felt bored, negative affect and
                  job stressful are reversed to align with their respective domains.",
                  "Signif. codes: ***$p < .001$, **$p < .01$, *$p < .05$"),
      threeparttable = TRUE, escape = FALSE, footnote_as_chunk = FALSE
    ) |>
```

```
    save_kable(sprintf("output_ESS/ESS_edu_%s.tex", m))
}

############################################################
## Forest partial plots
############################################################
########################
## Run and plot forest
########################
pacman::p_load(randomForest, pdp)

# helper
ess_plot_partial_forest <- function(data,
                                    dv,
                                    x.var       = "age",
                                    controls    = NULL,
                                    ntree       = 500,
                                    seed        = 23,
                                    model_tag   = "simple",
                                    base_dir    = "output_ESS",
                                    save_png    = TRUE,
                                    save_objects = TRUE) {

  ## clean predictor name
  x_chr <- trimws(if (is.character(x.var)) x.var[1]
                  else deparse(substitute(x.var))[1])

  ## folders
  plot_dir <- file.path(base_dir, paste0("partial_plots_", model_tag))
  obj_dir  <- file.path(plot_dir, "objects")
  dir.create(plot_dir, recursive = TRUE, showWarnings = FALSE)
  if (save_objects) dir.create(obj_dir, recursive = TRUE, showWarnings = FALSE)

  ## modelling data
  controls <- unique(c(x_chr, controls)); controls <- controls[!is.na(controls)]
  all_vars <- c(dv, controls)
  if (length(miss <- setdiff(all_vars, names(data))))
    stop("Missing in data: ", paste(miss, collapse = ", "))

  df    <- na.omit(data[, all_vars, drop = FALSE])
  n_obs <- nrow(df)

  ## fit random-forest
  set.seed(seed)
  rf <- randomForest(as.formula(paste(dv, "~ .")),
                     data = df, ntree = ntree, keep.forest = TRUE)

  ## partial dependence via pdp
  pdp_df <- pdp::partial(
    object   = rf,
    pred.var = x_chr,
    train    = df,
    plot     = FALSE,
```

```
    grid.resolution = 100,
    center = TRUE
  )

  ## PNG
  if (save_png) {
    png_path <- file.path(plot_dir,
                          sprintf("%s_%s_%s.png", x_chr, dv, model_tag))
    png(png_path, 800, 600)
    plot(pdp_df[[x_chr]], pdp_df$yhat, type = "l",
         main = paste("Partial Effect of", x_chr, "on", dv),
         xlab = x_chr, ylab = "Partial Dependence")
    dev.off()
  }

  ## save objects
  if (save_objects) {
    tag <- sprintf("%s_%s_%s", x_chr, dv, model_tag)
    saveRDS(rf,             file.path(obj_dir, paste0(tag, "_model.rds")))
    saveRDS(importance(rf), file.path(obj_dir, paste0(tag, "_importance.rds")))
    saveRDS(pdp_df,         file.path(obj_dir, paste0(tag, "_pdp.rds")))
  }

  message("ESS PDP saved for ", dv, " vs ", x_chr,
          "  (", n_obs, " rows; ntree=", ntree, ")")
  invisible(pdp_df)
}



### Fixed-effect blocks
blocks <- list(
  education   = "edulvla",
  demo        = c("gndr","age","health","hincfel"),
  labor       = c("pdwrk","uempla","uempli","dsbld","rtrd",
                  "cmsrv","hswrk","dngoth"),
  family      = c("maritala","chldhhe"),
  parentsEdu  = c("edulvlfa","edulvlma"),
  religiosity = "ever_religious",
  personality = "seek_richness_z",
  country     = "cntry"
)


simple_controls <- character(0)
full_controls   <- unlist(blocks[c("demo","labor","family","country")])
robust2_controls<- unlist(blocks[c("demo","labor","family","country",
                                   "personality")])

control_map <- list(simple = simple_controls,
                    full   = full_controls,
                    robust2 = robust2_controls)
```

```
########################
## Spec grid (for overlaying Forest graphs)
########################
library(tibble)

ess_specs <- tribble(
  ~dv,                    ~x.var,      ~model_tag,
  #  AGE overlays
  # "happy_z",            "age",       "simple",
  # "happy_z",            "age",       "full",
  # "fltbrd_z",           "age",       "simple",
  # "fltbrd_z",           "age",       "full",

  # Full
  "fltbrd_z",          "edulvla",       "full",
  "meaning_z",             "edulvla",  "full",
  "life_satisfaction_z", "edulvla",  "full",
  # Robust-2
  "fltbrd_z",            "edulvla",   "robust2",
  "meaning_z",           "edulvla",   "robust2",
  "life_satisfaction_z", "edulvla",   "robust2"
)


library(purrr)

pmap(
  ess_specs,
  \(dv, x.var, model_tag)
  ess_plot_partial_forest(
    data        = data,
    dv          = dv,
    x.var       = x.var,
    controls    = control_map[[model_tag]],
    ntree       = 500,
    seed        = 23,
    model_tag   = model_tag,
    base_dir    = "output_ESS"
  )
)


########################
## Overlay graphs
########################
library(tidyverse)

## Curves to overlay
overlay_specs <- tribble(
  ~dv,                    ~label,        ~colour,
  "fltbrd_z",            "Richness",    "#D55E00",
  "life_satisfaction_z", "Happiness",   "#0072B2",
  "meaning_z",           "Meaning",     "#009E73"
```

```
)

## Parameters
base_dir  <- "output_ESS"
model_tag <- "robust2" # Model selection, e.g., "full", "robust2"
x_chr     <- "edulvla"
span_val  <- 0.45
plot_title <- "Partial Dependence of Education on Well-being (ESS)"

## Helper - read, smooth & centre a PDP
read_pdp <- function(dv, label) {
  fp <- file.path(
    base_dir, paste0("partial_plots_", model_tag), "objects",
    sprintf("%s_%s_%s_pdp.rds", x_chr, dv, model_tag)
  )
  readRDS(fp) %>%
    rename(x = !!sym(x_chr), y = yhat) %>%
    {
      lo <- loess(y ~ x, data = ., span = span_val)
      mutate(., y = predict(lo) - mean(predict(lo)),
             DV = label)
    }
}

## Stack all curves
pdp_overlay <- overlay_specs |>
  mutate(data = map2(dv, label, read_pdp)) |>
  unnest(data)

## Build the plot
overlay_plot <- ggplot(pdp_overlay, aes(x, y, colour = DV)) +
  geom_hline(yintercept = 0, colour = "grey60", linewidth = .3) +
  geom_line(linewidth = 1) +
  scale_color_manual(values = deframe(overlay_specs[, c("label","colour")])) +
  scale_x_continuous(
    breaks = 1:5,
    labels = c("Primary", "Middle", "High", "Some college", "College+")
  ) +
  labs(
    title = plot_title,
    x = "Education level (ISCED)",
    y = "Partial Dependence"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_line(colour = "grey90"),
    panel.grid.minor   = element_blank(),
    panel.background   = element_rect(fill = "white", colour = NA),
    axis.line          = element_line(colour = "black"),
    legend.position    = "bottom",
    legend.title       = element_blank(),
    legend.text        = element_text(size = 12)
```

```
    # plot.title         = element_text(
    #   size   = 18, face = "bold",
    #   margin = ggplot2::margin(t = 12, b = 8)
    # ),
    # plot.margin = ggplot2::margin(t = 20, r = 28, b = 12, l = 12)
  )


out_dir <- file.path(base_dir, paste0("partial_plots_", model_tag))
dir.create(out_dir, recursive = TRUE, showWarnings = FALSE)

ggsave(
  filename = file.path(out_dir, sprintf("overlay_%s.png", x_chr)),
  plot     = overlay_plot +
    theme(
      plot.title.position = "plot",  # title inside plot
      plot.title         = element_text(
        face   = "bold",
        size   = 14,
        margin = ggplot2::margin(t = 12, b = 8)
      ),
      plot.margin        = ggplot2::margin(15, 28, 15, 15)
    ),
  width  = 160/25.4,
  height = 120/25.4,
  units  = "in",
  dpi    = 300,
  bg     = "white"
)

cat("PNG written to:", out_dir, "\n")
```