Data Preparation

Table of contents

This script prepares the data for analysis and creates objects for the visualisation and cross-validation.

Load Required Packages

```
# Load necessary R packages
library(foreign)   # For reading SPSS files
library(tidyverse) # For data manipulation and cleaning
```

Download Data

Create folders to organise the data files.

```
dir.create("data")
dir.create("data/back") #For the monthly background variables
dir.create("data/pers") #For the personality questionnaires
```

Download the monthly background variables files ('avars_(...).sav') from 2007 to 2023 from the LISS website and save them in the folder `data/back`. Download the personality questionnaires ('cp_(...).sav') from 2008 to 2023 from LISS and save them in the folder `data/pers`.

Load and Clean Data

Load these files in R.

```
# List all .sav files in the 'data/back' directory
files <- list.files(path = "data/back/", pattern = "*.sav", full.names =
 ↪  TRUE)

# Combine all the files into one dataset 'back.all'
back.all <- do.call('bind_rows', lapply(files, function(x) read.spss(x,
 ↪  use.value.labels = FALSE, to.data.frame = TRUE)))
```

```
# Load personality data for each wave (2008 to 2023)
pe08  <- read.spss('data/pers/cp08a_1p_EN.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe09  <- read.spss('data/pers/cp09b_1.0p_EN.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe10  <- read.spss('data/pers/cp10c_1.0p_EN.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe11  <- read.spss('data/pers/cp11d_1.0p_EN.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe12  <- read.spss('data/pers/cp12e_1.0p_EN.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe13  <- read.spss('data/pers/cp13f_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe14  <- read.spss('data/pers/cp14g_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe15  <- read.spss('data/pers/cp15h_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe17  <- read.spss('data/pers/cp17i_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe18  <- read.spss('data/pers/cp18j_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
```

```
pe19  <- read.spss('data/pers/cp19k_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe20  <- read.spss('data/pers/cp20l_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe21  <- read.spss('data/pers/cp21m_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe22  <- read.spss('data/pers/cp22n_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
pe23  <- read.spss('data/pers/cp23o_EN_1.0p.sav', use.value.labels=F,
 ↪  to.data.frame=T)
```

Prepare the personality questionnaires for merging by renaming them consistently and removing unnecessary columns.

```
# Make the names uniform by removing the wave qualifier (e.g., "08a")
names(pe08) <- gsub("08a", "", names(pe08))
names(pe09) <- gsub("09b", "", names(pe09))
names(pe10) <- gsub("10c", "", names(pe10))
names(pe11) <- gsub("11d", "", names(pe11))
names(pe12) <- gsub("12e", "", names(pe12))
names(pe13) <- gsub("13f", "", names(pe13))
names(pe14) <- gsub("14g", "", names(pe14))
names(pe15) <- gsub("15h", "", names(pe15))
names(pe17) <- gsub("17i", "", names(pe17))
names(pe18) <- gsub("18j", "", names(pe18))
names(pe19) <- gsub("19k", "", names(pe19))
names(pe20) <- gsub("20l", "", names(pe20))
names(pe21) <- gsub("21m", "", names(pe21))
names(pe22) <- gsub("22n", "", names(pe22))
names(pe23) <- gsub("23o", "", names(pe23))

# Remove variables indicating date and duration of the questionnaire,
# Because we don't need them and otherwise they cause trouble with merging as
 ↪  their variable type differs per wave
pe08 <- pe08 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe09 <- pe09 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe10 <- pe10 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe11 <- pe11 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe12 <- pe12 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe13 <- pe13 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe14 <- pe14 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
```

```r
pe15 <- pe15 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe17 <- pe17 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe18 <- pe18 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe19 <- pe19 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe20 <- pe20 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe21 <- pe21 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe22 <- pe22 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))
pe23 <- pe23 %>% dplyr::select(-c(cp189, cp190, cp191, cp192, cp193))

# Merge them all together
# To receive one file including all waves for the personality data
pe.all <- bind_rows(pe08, pe09, pe10, pe11, pe12, pe13, pe14, pe15, pe17,
 ↪  pe18, pe19, pe20, pe21, pe22, pe23)

# Rename "cp_m" into "wave" to be consistent with "back.all"
names(pe.all)[3] <- c("wave")

# Remove everything besides of participant ID ("nomem_encr"), Wave and Life
 ↪  Satisfaction items from data set
pe.all <- pe.all %>% dplyr::select(c(nomem_encr, wave, cp014, cp015, cp016,
 ↪  cp017, cp018))
```

Merge Background and Personality Data

Merge the background data (back.all) with the personality data (pe.all) on the nomem_encr (participant ID) and wave variables.

```r
# Merge background data and personality data
data <- list(back.all, pe.all) %>% reduce(full_join, by=c('nomem_encr',
 ↪  'wave'))
```

Compute Life Satisfaction Scores

Compute life satisfaction scores by averaging responses from its five items (cp014 to cp018). Higher scores indicate more life satisfaction.

```r
# Compute mean life satisfaction for each row
data <- data %>%
  rowwise() %>%
  mutate(lifesatisfaction = mean(c(cp014, cp015, cp016, cp017, cp018)))
```

Create Timeline Variable

Create variables for year and month of measurement, and a continuous timeline of measurements.

```
# Create year and month variables
data$year <- as.numeric(substr(data$wave, 1,4))
data$month <- as.numeric(substr(data$wave, 5,6))

# Create a continuous time axis (timeline in months)
# NOTE: as the survey started in november 2007, this timeline is not
 ↪  completely correct yet
data$timeline <- (data$year-2008)*12 + data$month

# Adjust timeline to start from the correct point in time (November 2007)
data$timeline <- data$timeline + 2
data <- data[order(data$nomem_encr, data$wave),]

# Remove objects from environment we do not need anymore
rm(list = setdiff(ls(), "data"))
```

Create Widowhood Transition Variable

Create a transition variable that indicates when a participant experienced widowhood, based on a marital status change ("burgstat"). To do so, first create a variable that indicates a person's previous marital status (from the previous wave).

```
# Create a parallel dataset that consists of past data

# Order data: rearrange data ascending based on nomem_encr (=participant id)
data <- data[order(data$nomem_encr, data$wave),]

# Create a variable that indicates the number of waves that a person has
 ↪  contributed (including current wave)
data$k <- ave(data$wave, data$nomem_encr, FUN = seq_along)

# Mirror the dataset
pastdata <- data

# Select the relevant variables
pastdata <- pastdata[,c("nomem_encr", "k", "burgstat")]
```

```
# Add one to the wave number
pastdata$k.pa <- pastdata$k

# The following trick (+1) ensures that row k = 10 of "data" is combined with
 ↪  row 9 (k = 9 + 1 = 10) of "pastdata", which is therefore the "past"
pastdata$k <- pastdata$k + 1

# Rename the variables to indicate the past
names(pastdata) <- c("nomem_encr", "k", "burgstat.pa", "k.pa")

# Create an additional "past id variable", as a check (to check whether there
 ↪  are no "nomadic" lags)
pastdata$nomem_encr.pa <- pastdata$nomem_encr

# Merge files, exclude the highest lag of the "pastdata" file via "all.x = T"
 ↪
# (because the last wave cannot be a lag)
data <- merge(data, pastdata, by=c("nomem_encr", "k"), all.x=T)

# Remove pastdata, because we do not need it anymore
rm(pastdata)
```

Create the widowhood transition variable.

```
# Create a variable indicating when a participant became widowed
data <- data %>%
  mutate(wido.ev = ifelse(
    nomem_encr == nomem_encr.pa & burgstat == 4 & burgstat.pa == 1,
    1,
    0
  ))
```

Create Time Relative To Transition Variable

Create a variable indicating the timing of measurements relative to the transition.

```
# Create variables for the exact time of widowhood event (year and month)
data <- data %>%
  group_by(nomem_encr) %>%
```

```
  mutate(wido_time = ifelse(wido.ev == 1, paste0(year, month), NA)) %>%
  mutate(wido_time =  ifelse(any(!is.na(wido_time)), max(wido_time, na.rm =
   ↪  TRUE), NA))

# Split 'wido_time' into year and month components
data$wido.year <- as.numeric(substr(data$wido_time, 1,4))
data$wido.month <- as.numeric(substr(data$wido_time, 5,6))

# Create variable indicating the timing of measurements relative to the
 ↪  widowhood transition
# ("mnths" because it is the timing of measurements relative to widowhood in
 ↪  months)
data$mnths <- (data$year - data$wido.year)*12 + (data$month -
 ↪  data$wido.month)
```

Filter Participants for Analysis

Filter participants who have experienced widowhood, and for whom at least one observation before and after widowhood is available.

```
# Rename ID variable and recode into a factor
data$id <- factor(data$nomem_encr)

# Select only participants who have experienced widowhood
wido <- data %>%
  group_by(id) %>%
  filter(any(wido.ev == 1)) %>%
  filter(!is.na(lifesatisfaction))

# Select participants with at least one observation before and one after
 ↪  widowhood
wido <- wido %>%
  group_by(id) %>%
  filter(any(mnths < 0) & any(mnths > 0))
```

Variable for Model Performance Indices

Create a variable for the model performance indices: the mean of life satisfaction per month, to compare the fixed effects trajectory as estimated by the models, with the observed mean life satisfaction trajectory.

```
# Create a variable for the model performance indices: the mean of life
 ↪  satisfaction per month
wido <- wido %>%
  group_by(mnths) %>%
  mutate(m_lifesat_per_mnth = mean(lifesatisfaction, na.rm = TRUE)) %>%
  ungroup()
```

Save Data

Select relevant variables, and save the data.

```
# Select only the variables we need
wido <- wido %>% dplyr::select(c(id, lifesatisfaction, mnths,
 ↪  m_lifesat_per_mnth))

# Save data
save(wido, file = "data/wido.rdata")
```

Create Objects for Visualisation and Cross-Validation

Create objects to use for visualisation and cross-validation. First create a folder to save these.

```
dir.create("objects")
```

Create a basic plot to visualise the model predictions, and save it in the folder.

```
# Create plot
plot_base <-
  ggplot() + theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_line(colour = "black")
  ) +
  labs(x = "Months from Widowhood", y = "Life Satisfaction") +
  scale_y_continuous(breaks = seq(1, 7, by = 1), limits = c(1, 7)) +
  scale_x_continuous(breaks = seq(-180, 180, by = 30),
  limits = c(-180, 180)) +
  theme(legend.position = "none") +
```

```
  coord_fixed(ratio = 45)

# Save it
saveRDS(plot_base, file = "objects/plot_base.rds")
```

Randomly distribute the participants in 5 groups, and create training and test data sets for cross-validation, by sampling from these groups. Save the lists of training and test data sets in the folder.

```
# Set seed for reproducibility
set.seed(123)

# Create a unique list of participants and assign them randomly to 5 groups
group <- wido %>%
  distinct(id) %>%
  mutate(group = sample(1:5, size = n(), replace = TRUE)) %>%
  ungroup()

# Join this group information back to the original data
wido <- wido %>% left_join(group, by = "id")

# Create training and test data sets
training1 <- wido %>% filter(group != 1)
test1 <- wido %>% filter(group == 1)

training2 <- wido %>% filter(group != 2)
test2 <- wido %>% filter(group == 2)

training3 <- wido %>% filter(group != 3)
test3 <- wido %>% filter(group == 3)

training4 <- wido %>% filter(group != 4)
test4 <- wido %>% filter(group == 4)

training5 <- wido %>% filter(group != 5)
test5 <- wido %>% filter(group == 5)

# Create lists of these data sets
training_datasets <- list(training1, training2, training3, training4,
 ↪  training5)
test_datasets <- list(test1, test2, test3, test4, test5)
```

```r
# Save these lists
saveRDS(training_datasets, file = "objects/training_datasets.rds")
saveRDS(test_datasets, file = "objects/test_datasets.rds")
```

Done!