Continuous-Time Dynamic Models

Table of contents

To illustrate continuous-time dynamic modelling, we specified a Bayesian hierarchical continuous time dynamic model, with a measurement model with the five observed life satisfaction items loading onto a latent life satisfaction factor. The general change in this latent life satisfaction factor was modelled with an initial level and an auto effect. To estimate the impact of the widowhood transition, we created a transitionTime variable indicating the timing of the widowhood transition occurrence, and estimated a transition input effect and a transition auto effect. Note that the terms *drift* and *auto effect* refer to the same concept.

To reproduce the results, it is necessary to prepare the data set, plot base, and training and test data sets, as outlined in the "Data Preparation" section.

Preparation

Loading Required Packages and Data

Load the necessary packages, data sets, and other supporting files. Each element serves a specific purpose:

- **`tidyverse`**: For data manipulation and visualisation.
- **`ctsem`**: To fit the continuous time dynamic model.
- **`rstan`**: Required for Bayesian continuous time dynamic model.
- **`caret`**: To compute model performance indices.
- **`plot_base`**: A pre-configured ggplot object for visualisation.
- **Training and Test Data sets**: Required for cross-validation.

```
# Load necessary packages
library(tidyverse)
library(ctsem)
library(rstan)
library(caret)

# Load the data set
load("data/wido.rdata")

# Load the pre-configured plot base
plot_base <- readRDS("objects/plot_base.rds")

# Load training and test datasets for cross-validation
training_datasets <- readRDS("objects/training_datasets.rds")
test_datasets <- readRDS("objects/test_datasets.rds")
```

Preparing the Data

To specify the ctsem model, we need to create a variable called transitionTime, which should be set to 1 at the time point when the transition (widowhood) occurs, and 0 at all other time points. Since the transition occurred at mnths = 0 for all individuals, this variable must have a value of 1 at mnths = 0 for each person.

However, our current dataset only includes rows where life satisfaction data is available. As a result, not all individuals have an observation/row where mnths = 0.

To ensure the transitionTime variable correctly reflects the timing of the transition for every person, we need to add a row with mnths = 0 for each individual who does not already have one. This will allow us to assign the transition indicator consistently across all participants.

```
# Check for which individuals mnths = 0 is missing
ids_to_add <- wido %>%
  group_by(id) %>%
  filter(all(mnths != 0)) %>%
  distinct(id)

# Create new rows for these individuals, with transitionTime = 1, and mnths =
 ↪  0
new_rows <- ids_to_add %>%
  mutate(transitionTime = 1, mnths = 0)

# Combine these new rows with the original data
wido_extrarows <- wido %>%
  bind_rows(new_rows)

# Create transitionTime variable for all individuals, which is always 0,
 ↪  except when mnths = 0, for those rows transitionTime = 1
wido_extrarows <- wido_extrarows %>%
  mutate(transitionTime = if_else(mnths == 0, 1, 0))
```

We use the default priors, which are set up to be weakly informative for typical applications in the social sciences, on data that is centered and scaled (https://cran.r-project.org/web/packages/ctsem/vignettes/hierarchicalmanual.pdf, p.4). It is recommended to grand mean center and scale the outcome variable, but not the transitionTime variable.

```
# Grand mean scale life satisfaction
wido_extrarows$lifesatisfaction_scaled <-
 ↪  scale(wido_extrarows$lifesatisfaction)

# Grand mean scale life satisfaction items, for the measurement model
wido_extrarows$cp014_s <- scale(wido_extrarows$cp014)
wido_extrarows$cp015_s <- scale(wido_extrarows$cp015)
wido_extrarows$cp016_s <- scale(wido_extrarows$cp016)
wido_extrarows$cp017_s <- scale(wido_extrarows$cp017)
wido_extrarows$cp018_s <- scale(wido_extrarows$cp018)
```

Further, to align with the default priors, a time interval of 1 should reflect some 'moderate change' (https://cran.r-project.org/web/packages/ctsem/vignettes/hierarchicalmanual.pdf,

p.4). We create a variable "fiveyrs" which is a time variable coded in five year intervals, which we expect to reflect moderate change in life satisfaction surrounding widowhood.

```
# Create a time variable indicating the timing of life satisfaction
 ↪  measurements on a five year scale
wido_extrarows$fiveyrs <- round(wido_extrarows$mnths / 60, digits = 2)
```

Analysis

Specifying the model

```
# Specify the model
model <- ctModel(
  manifestNames = c("cp014_s", "cp015_s", "cp016_s", "cp017_s", "cp018_s"),
  # Names of the observed (manifest) variables in the dataset (questionnaire
   ↪  items)

  latentNames = c("lifesatisfaction", "transitionresponse"),
  # Names of the unobserved (latent) variables: one for the personality trait
   ↪  (life satisfaction)
  # and one for the transition (transitionresponse)

  TDpredNames = c("transitionTime"),
  # Name of the "transitionTime" variable, that indicates when the transition
   ↪  occurs

  time = 'fiveyrs',
  # Name of the time variable indicating the timing of life satisfaction
   ↪  measurements on a five year scale

  id = 'id',
  # Name of the subject ID column in the dataset

  type = 'stanct',
  # Specifies the use of a continuous-time model implemented in STAN

  LAMBDA = matrix(c(1,1,1,1,1,0,0,0,0,0), nrow = 5, ncol = 2),
  # Factor loading matrix: maps latent variables to manifest variables
```

```
# The first latent variable (lifesatisfaction) loads on all 5 observed
 ↪  items
# The second latent variable (transitionresponse) does not directly load on
 ↪  any observed variable

DRIFT = matrix(c('AutoEffectLS||TRUE', 0, 1, 'AutoEffectTransition||TRUE'),
 ↪  nrow = 2, ncol = 2),
# DRIFT matrix defines how latent variables evolve over time
# [1,1]: self-regulation of lifesatisfaction (auto-effect)
# [2,1]: how transitionresponse influences change in lifesatisfaction
 ↪  (cross-lagged effect)
# [2,2]: self-regulation of transitionresponse (auto-effect)
# "||TRUE" indicates random effects are estimated for this parameter

DIFFUSION = matrix(c('systemNoise', 0, 0, 0), nrow = 2, ncol = 2),
# System noise (stochastic variability) for the latent processes
# Only lifesatisfaction has system noise; transitionresponse is
 ↪  deterministic

MANIFESTVAR = matrix(c(
  'residualSD1', 0,           0,           0,           0,
  'residCov21', 'residualSD2', 0,           0,           0,
  'residCov31', 'residCov32', 'residualSD3', 0,           0,
  'residCov41', 'residCov42', 'residCov43', 'residualSD4', 0,
  'residCov51', 'residCov52', 'residCov53', 'residCov54', 'residualSD5'
), nrow = 5, byrow = TRUE),
# Residual variance-covariance matrix for the manifest variables
 ↪  (measurement error)
# Diagonal = residual variances; lower triangle = covariances between item
 ↪  residuals

CINT = matrix(c(0, 0), nrow = 2, ncol = 1),
# Continuous intercepts (fixed baseline drift) for latent variables set to
 ↪  0

MANIFESTMEANS = matrix(c(0), nrow = 5, ncol = 1),
# Mean of the manifest variables fixed to 0

TOMEANS = matrix(c("initialLS||TRUE", 0), nrow = 2, ncol = 1),
# Initial mean of life satisfaction estimated freely; transitionresponse
 ↪  initial value fixed to 0
# Again: "||TRUE" indicates random effects are estimated for this parameter
```

```
  TDPREDEFFECT = matrix(c(0, "transitionEffect||TRUE"), nrow = 2, ncol = 1)
  # Effect of "transitionTime" on the latent variables:
  # No direct effect on lifesatisfaction; freely estimated effect on
   ↪  transitionresponse
)
```

Fitting the Model

```
# Options to speed up model fitting
options(mc.cores = 4)
rstan_options(threads_per_chain = 40)
```

```
# For reproducibility
set.seed(123)

# Fit the CTDM
fit <- ctStanFit(datalong = wido_extrarows, ctstanmodel = model, iter = 2000,
 ↪  chains = 4)

summary(fit)
```

```
$residCovStd
        cp014_s cp015_s cp016_s cp017_s cp018_s
cp014_s   0.661   0.499   0.435   0.213   0.079
cp015_s   0.499   0.719   0.498   0.244   0.049
cp016_s   0.435   0.498   0.672   0.257   0.073
cp017_s   0.213   0.244   0.257   0.624   0.152
cp018_s   0.079   0.049   0.073   0.152   0.644

$resiCovStdNote
[1] "Standardised covariance of residuals"

$rawpopcorr
                                          mean     sd    2.5%     50%  97.5%
AutoEffectLS__initialLS                -0.3408 0.3320 -0.7526 -0.4505 0.4461
AutoEffectTransition__initialLS        -0.2380 0.2452 -0.6544 -0.2519 0.2797
transitionEffect__initialLS            -0.2359 0.2597 -0.6107 -0.2822 0.3982
AutoEffectTransition__AutoEffectLS      0.1163 0.4009 -0.6419  0.1427 0.7787
```

```
transitionEffect__AutoEffectLS          0.2554 0.8355 -0.9590  0.7988 0.9949
transitionEffect__AutoEffectTransition  0.3376 0.3139 -0.3626  0.3793 0.8289
                                             z
AutoEffectLS__initialLS                -1.0265
AutoEffectTransition__initialLS        -0.9704
transitionEffect__initialLS            -0.9081
AutoEffectTransition__AutoEffectLS      0.2900
transitionEffect__AutoEffectLS          0.3057
transitionEffect__AutoEffectTransition  1.0754

$parmatrices
             matrix row col     Mean      sd     2.5%      50%   97.5%
1           TOMEANS   1   1   0.0929  0.0196   0.0539   0.0929  0.1309
2           TOMEANS   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
3            LAMBDA   1   1   1.0000  0.0000   1.0000   1.0000  1.0000
4            LAMBDA   1   2   0.0000  0.0000   0.0000   0.0000  0.0000
5            LAMBDA   2   1   1.0000  0.0000   1.0000   1.0000  1.0000
6            LAMBDA   2   2   0.0000  0.0000   0.0000   0.0000  0.0000
7            LAMBDA   3   1   1.0000  0.0000   1.0000   1.0000  1.0000
8            LAMBDA   3   2   0.0000  0.0000   0.0000   0.0000  0.0000
9            LAMBDA   4   1   1.0000  0.0000   1.0000   1.0000  1.0000
10           LAMBDA   4   2   0.0000  0.0000   0.0000   0.0000  0.0000
11           LAMBDA   5   1   1.0000  0.0000   1.0000   1.0000  1.0000
12           LAMBDA   5   2   0.0000  0.0000   0.0000   0.0000  0.0000
13            DRIFT   1   1  -0.1814  0.0610  -0.3240  -0.1723 -0.0848
14            DRIFT   1   2   1.0000  0.0000   1.0000   1.0000  1.0000
15            DRIFT   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
16            DRIFT   2   2 -45.7765 21.8041 -88.8045 -45.6937 -3.0859
21    MANIFESTMEANS   1   1   0.0000  0.0000   0.0000   0.0000  0.0000
22    MANIFESTMEANS   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
23    MANIFESTMEANS   3   1   0.0000  0.0000   0.0000   0.0000  0.0000
24    MANIFESTMEANS   4   1   0.0000  0.0000   0.0000   0.0000  0.0000
25    MANIFESTMEANS   5   1   0.0000  0.0000   0.0000   0.0000  0.0000
26             CINT   1   1   0.0000  0.0000   0.0000   0.0000  0.0000
27             CINT   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
32     TDPREDEFFECT   1   1   0.0000  0.0000   0.0000   0.0000  0.0000
33     TDPREDEFFECT   2   1  -9.4293  4.3073 -18.0734  -9.3916 -1.0166
34         asymCINT   1   1   0.0000  0.0000   0.0000   0.0000  0.0000
35         asymCINT   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
36 asymDIFFUSIONcov   1   1   0.3926  0.1005   0.2271   0.3825  0.6037
37 asymDIFFUSIONcov   1   2   0.0000  0.0000   0.0000   0.0000  0.0000
38 asymDIFFUSIONcov   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
39 asymDIFFUSIONcov   2   2   0.0000  0.0000   0.0000   0.0000  0.0000
```

```
40     DIFFUSIONcov   1   1   0.1335  0.0288   0.0865   0.1304  0.1948
41     DIFFUSIONcov   1   2   0.0000  0.0000   0.0000   0.0000  0.0000
42     DIFFUSIONcov   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
43     DIFFUSIONcov   2   2   0.0000  0.0000   0.0000   0.0000  0.0000
44      MANIFESTcov   1   1   0.5183  0.0150   0.4890   0.5178  0.5479
45      MANIFESTcov   1   2   0.3554  0.0180   0.3203   0.3547  0.3904
46      MANIFESTcov   1   3   0.2937  0.0172   0.2607   0.2940  0.3276
47      MANIFESTcov   1   4   0.0745  0.0133   0.0499   0.0741  0.0998
48      MANIFESTcov   1   5  -0.0593  0.0126  -0.0843  -0.0589 -0.0337
49      MANIFESTcov   2   1   0.3554  0.0180   0.3203   0.3547  0.3904
50      MANIFESTcov   2   2   0.5736  0.0177   0.5391   0.5729  0.6087
51      MANIFESTcov   2   3   0.3555  0.0184   0.3196   0.3552  0.3916
52      MANIFESTcov   2   4   0.1037  0.0144   0.0770   0.1029  0.1336
53      MANIFESTcov   2   5  -0.0907  0.0133  -0.1168  -0.0907 -0.0644
54      MANIFESTcov   3   1   0.2937  0.0172   0.2607   0.2940  0.3276
55      MANIFESTcov   3   2   0.3555  0.0184   0.3196   0.3552  0.3916
56      MANIFESTcov   3   3   0.5326  0.0160   0.5016   0.5321  0.5632
57      MANIFESTcov   3   4   0.1199  0.0143   0.0920   0.1199  0.1466
58      MANIFESTcov   3   5  -0.0640  0.0131  -0.0885  -0.0635 -0.0379
59      MANIFESTcov   4   1   0.0745  0.0133   0.0499   0.0741  0.0998
60      MANIFESTcov   4   2   0.1037  0.0144   0.0770   0.1029  0.1336
61      MANIFESTcov   4   3   0.1199  0.0143   0.0920   0.1199  0.1466
62      MANIFESTcov   4   4   0.4889  0.0097   0.4701   0.4895  0.5073
63      MANIFESTcov   4   5   0.0166  0.0128  -0.0080   0.0164  0.0415
64      MANIFESTcov   5   1  -0.0593  0.0126  -0.0843  -0.0589 -0.0337
65      MANIFESTcov   5   2  -0.0907  0.0133  -0.1168  -0.0907 -0.0644
66      MANIFESTcov   5   3  -0.0640  0.0131  -0.0885  -0.0635 -0.0379
67      MANIFESTcov   5   4   0.0166  0.0128  -0.0080   0.0164  0.0415
68      MANIFESTcov   5   5   0.5079  0.0080   0.4931   0.5077  0.5244
69            T0cov   1   1   0.4423  0.0524   0.3530   0.4396  0.5491
70            T0cov   1   2   0.0000  0.0000   0.0000   0.0000  0.0000
71            T0cov   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
72            T0cov   2   2   0.0000  0.0000   0.0000   0.0000  0.0000
73          dtDRIFT   1   1   0.8356  0.0499   0.7232   0.8417  0.9187
74          dtDRIFT   1   2   0.0443  0.1234   0.0089   0.0185  0.2675
75          dtDRIFT   2   1   0.0000  0.0000   0.0000   0.0000  0.0000
76          dtDRIFT   2   2   0.0182  0.1242   0.0000   0.0000  0.0457

$popsd
                        mean     sd   2.5%     50%   97.5%
initialLS             0.6645 0.0419 0.5873  0.6630  0.7523
AutoEffectLS          0.5804 0.1350 0.3521  0.5693  0.8699
AutoEffectTransition 13.7902 5.5916 3.0284 13.6892 24.9229
```

```
transitionEffect     17.9639 7.8939 4.8419 17.2697 34.7972

$popmeans
                             mean      sd     2.5%      50%   97.5%
initialLS                  0.0929  0.0196   0.0539   0.0929  0.1309
systemNoise                0.3633  0.0387   0.2942   0.3611  0.4414
residualSD1                0.7198  0.0104   0.6993   0.7196  0.7402
residCov21                 0.3577  0.0111   0.3369   0.3577  0.3801
residualSD2                0.7573  0.0117   0.7343   0.7569  0.7802
residCov31                 0.2796  0.0119   0.2556   0.2801  0.3032
residCov32                 0.3492  0.0113   0.3265   0.3496  0.3703
residualSD3                0.7297  0.0109   0.7083   0.7295  0.7504
residCov41                 0.0543  0.0118   0.0307   0.0542  0.0764
residCov42                 0.0858  0.0120   0.0626   0.0857  0.1101
residCov43                 0.1122  0.0122   0.0882   0.1126  0.1349
residualSD4                0.6992  0.0070   0.6856   0.6996  0.7122
residCov51                -0.0452  0.0127  -0.0701  -0.0448 -0.0206
residCov52                -0.0819  0.0127  -0.1072  -0.0818 -0.0579
residCov53                -0.0515  0.0129  -0.0764  -0.0509 -0.0254
residCov54                 0.0246  0.0130  -0.0008   0.0249  0.0494
residualSD5                0.7126  0.0056   0.7022   0.7125  0.7241
T0var_transitionresponse   0.0000  0.0007   0.0000   0.0000  0.0000
AutoEffectLS              -0.1814  0.0610  -0.3240  -0.1723 -0.0848
AutoEffectTransition     -45.7765 21.8041 -88.8045 -45.6937 -3.0859
transitionEffect          -9.4293  4.3073 -18.0734  -9.3916 -1.0166

$popNote
[1] "popmeans are reported as specified in ctModel -- covariance related matrices are in sd ,

$loglik
[1] -11950.85

$npars
[1] 31

$aic
[1] 23963.69

$logposterior
[1] -11950.85
```

```r
# Get the original mean and SD of life satisfaction, to interpret the initial
 ↪  life satisfaction level on the raw scale
LS_mean <- mean(wido_extrarows$lifesatisfaction, na.rm = TRUE)
LS_sd <- sd(wido_extrarows$lifesatisfaction, na.rm = TRUE)

# Extract parameter matrix from model fit
pmat <- summary(fit)$parmatrices

# Convert the initial life satisfaction level (TOMEANS) to the raw scale
pmat$Mean[pmat$matrix == "TOMEANS" & pmat$row == 1] * LS_sd + LS_mean
```

```
[1] 5.068943
```

Visualisation

Predicting Average Trajectory

Predict the population-level (fixed effects) trajectory of life satisfaction.

```r
# Predict average (population-level) trajectory of life satisfaction using a
 ↪  custom function
# (We will use this function again later on)
predict_average_trajectory <- function() {
# Construct initial state (TOMEANS) matrix
TOMEANS_MATRIX <- matrix(c(
  pmat$Mean[pmat$matrix == "TOMEANS" & pmat$row == 1],
  0), nrow = 2, byrow = FALSE)

# Construct matrix of transition effect
TDPREDEFFECT_MATRIX <- matrix(c(
  0,
  pmat$Mean[pmat$matrix == "TDPREDEFFECT" & pmat$row == 2]),
  nrow = 2, byrow = FALSE)

# Extract drift parameters (auto effects)
DRIFT_LS <- pmat$Mean[pmat$matrix == "DRIFT" & pmat$row == 1 & pmat$col == 1]
DRIFT_transition <- pmat$Mean[pmat$matrix == "DRIFT" & pmat$row == 2 &
 ↪  pmat$col == 2]
```

```r
# Combine drift/auto effect values into a matrix
DRIFT_MATRIX <- matrix(c(DRIFT_LS, 1, 0, DRIFT_transition), nrow = 2, byrow =
 ↪  TRUE)

# Define time points (in years)
times <- seq(-180, 180, by = 1) / 12
dt <- diff(times)[1] # Define the time step to take to compute the change
 ↪  over time

# Compute matrix exponential for the change over time
DRIFT_MATRIX_STAR <- Matrix::expm(DRIFT_MATRIX * dt)

# Initialise matrix to store the predicted values
xtraj <- matrix(0, nrow = length(times), ncol = 2)
x <- TOMEANS_MATRIX  # starting values
xtraj[1, ] <- c(TOMEANS_MATRIX)

# Predict the values over time
for (i in seq_along(times)[-1]) {
    t <- times[i]
    x <- DRIFT_MATRIX_STAR %*% x + TDPREDEFFECT_MATRIX * (abs(t) ==
 ↪  min(abs(times)))
    xtraj[i, ] <- c(as.matrix(c(x)[[1]]))
}

# Label the values of life satisfaction and the transition response
colnames(xtraj) <- c("LS", "TR")

# Rescale life satisfaction values to original scale
LS_raw <- xtraj[, 1] * LS_sd + LS_mean

# Create data frame for plotting
df <- data.frame(
  mnths = seq(-180, 180, by = 1),
  ctdm_pred_f = LS_raw
)
  return(df)
}

df <- predict_average_trajectory()
```

```r
# Merge predicted values into main dataset
wido <- wido %>%
  left_join(df, by = "mnths")
```

Computing Confidence Intervals

Compute the 95% confidence intervals for the average trajectory using 1,000 posterior sample draws. A posterior sample draw is a set of parameter values drawn from the posterior distribution — which reflects what we believe about the parameters after seeing the data, given our model and prior assumptions.

```r
# Extract posterior samples of population-level parameters from fit
posteriorsamples <- ctExtract(fit)

# Organise parameter draws into separate vectors
posteriorsamples_t0means <- matrix(posteriorsamples[["pop_T0MEANS"]], ncol =
 ↪  5)[, 1]
posteriorsamples_autoeffect_LS <- matrix(posteriorsamples[["pop_DRIFT"]],
 ↪  ncol = 4)[, 1]
posteriorsamples_autoeffect_transition <-
 ↪  matrix(posteriorsamples[["pop_DRIFT"]], ncol = 4)[, 4]
posteriorsamples_tdpredeffect <-
 ↪  matrix(posteriorsamples[["pop_TDPREDEFFECT"]], ncol = 2)[, 2]

# Combine into a tidy data frame
df_posteriorsamples <- data_frame(
  T0MEANS = posteriorsamples_t0means,
  TDPREDEFFECT = posteriorsamples_tdpredeffect,
  DRIFT_LS = posteriorsamples_autoeffect_LS,
  DRIFT_transition = posteriorsamples_autoeffect_transition
)

# Initialise storage for predicted trajectories based on posterior draws
n <- nrow(df_posteriorsamples)
results_list <- vector("list", n)

# Predict one trajectory per posterior draw
for (j in 1:n) {

  # Define parameter matrices for draw j
  T0MEANS_MATRIX <- matrix(c(df_posteriorsamples$T0MEANS[j], 0), nrow = 2)
```

```r
  TDPREDEFFECT_MATRIX <- matrix(c(0, df_posteriorsamples$TDPREDEFFECT[j]),
↪  nrow = 2)

  DRIFT_MATRIX <- matrix(
    c(df_posteriorsamples$DRIFT_LS[j], 1,
      0, df_posteriorsamples$DRIFT_transition[j]),
    nrow = 2, byrow = TRUE
  )

  # Set up time vector
  times <- seq(-180, 180, by = 1) / 12  # months to years
  dt <- diff(times)[1]
  DRIFT_MATRIX_STAR <- Matrix::expm(DRIFT_MATRIX * dt)

  # Predict values
  xtraj <- matrix(0, nrow = length(times), ncol = 2)
  x <- TOMEANS_MATRIX
  xtraj[1, ] <- c(x)

  for (i in seq_along(times)[-1]) {
    t <- times[i]
    x <- DRIFT_MATRIX_STAR %*% x + TDPREDEFFECT_MATRIX * (abs(t) ==
↪  min(abs(times)))
    xtraj[i, ] <- c(as.matrix(c(x)[[1]]))
  }

  # Extract and rescale life satisfaction predictions to raw scale
  LS_raw <- xtraj[, 1] * LS_sd + LS_mean

  # Store in list
  results_list[[j]] <- LS_raw
}

# Combine results into data frame with one column per posterior draw
posterior_ci <- data.frame(mnths = seq(-180, 180, by = 1))

for (j in 1:n) {
  posterior_ci[[paste0("ctdm_pred_f_", j)]] <- results_list[[j]]
}

# Identify prediction columns
pred_cols <- grep("^ctdm_pred_f_", names(posterior_ci), value = TRUE)
```

```r
# Compute 95% credible intervals (CI) across draws for each time point
ci_stats <- t(apply(posterior_ci[, pred_cols], 1, function(x) {
  quants <- quantile(x, probs = c(0.025, 0.975), names = FALSE)
  c(lower_CI = quants[1], upper_CI = quants[2])
}))

# Add CI to data frame
posterior_ci$lower_CI <- ci_stats[, "lower_CI"]
posterior_ci$upper_CI <- ci_stats[, "upper_CI"]

# Keep only relevant columns
posterior_ci <- posterior_ci %>% dplyr::select(mnths, lower_CI, upper_CI)

# Join CI estimates to main dataset
wido <- wido %>%
  left_join(posterior_ci, by = "mnths")
```

Predicting Individual Trajectories

Predict the individual-level (random effects) trajectories of life satisfaction.

```r
# Extract individual-level posterior means (random effects) for all
 ↪  participants
subjectpars <- ctStanSubjectPars(fit, pointest = TRUE, cores = 4, nsamples =
 ↪  'all')
subjectpars <- as.data.frame(subjectpars[1, , ])
subjectpars <- subjectpars %>%
  mutate(id = row_number())  # Add subject ID

# Create function to predict an individual's trajectory over time
predict_individual_trajectory <- function(subject_id, drift_ls,
 ↪  drift_transition, transition_effect, initial_ls) {

  # Define time vector (in years)
  times <- seq(-180, 180, by = 1) / 12
  dt <- diff(times)[1]

  # Construct required matrices based on subject-specific parameters
  TOMEANS_MATRIX <- matrix(c(initial_ls, 0), nrow = 2)
  TDPREDEFFECT_MATRIX <- matrix(c(0, transition_effect), nrow = 2)
```

```
  DRIFT_MATRIX <- matrix(c(drift_ls, 1, 0, drift_transition), nrow = 2, byrow
↪  = TRUE)
  DRIFT_MATRIX_STAR <- Matrix::expm(DRIFT_MATRIX * dt)

  # Initialise storage
  xtraj <- matrix(0, nrow = length(times), ncol = 2)
  x <- TOMEANS_MATRIX
  xtraj[1, ] <- c(TOMEANS_MATRIX)

  # Predict values
  for (i in seq_along(times)[-1]) {
    t <- times[i]
    x <- DRIFT_MATRIX_STAR %*% x + TDPREDEFFECT_MATRIX * (abs(t) ==
↪  min(abs(times)))  # Apply transition effect only at t = 0
    xtraj[i, ] <- c(as.matrix(c(x)[[1]]))
  }

  # Transform life satisfaction to raw scale
  LS_raw <- xtraj[, 1] * LS_sd + LS_mean

  # Return a tidy data frame
  data.frame(
    id = subject_id,
    mnths = seq(-180, 180, by = 1),
    ctdm_pred_r = LS_raw
  )
}

# Predict trajectories for all individuals using their subject-specific
↪  parameters
predict_list <- pmap(
  list(
    subjectpars$id,
    subjectpars$AutoEffectLS,
    subjectpars$AutoEffectTransition,
    subjectpars$transitionEffect,
    subjectpars$initialLS
  ),
  ~ predict_individual_trajectory(
      subject_id = ..1,
      drift_ls = ..2,
      drift_transition = ..3,
```

```
      transition_effect = ..4,
      initial_ls = ..5
  )
)

# Combine individual data frames into one long-format data frame
individualtrajectories <- bind_rows(predict_list)

# Prepare 'id' variable for merging (ensure consistent type and labels)
wido <- wido %>%
  mutate(id2 = as.factor(as.numeric(factor(id))))

individualtrajectories$id2 <- as.factor(individualtrajectories$id)
individualtrajectories <- individualtrajectories %>% dplyr::select(-id)

# Merge individual-level predicted trajectories into main data frame
wido <- wido %>%
  left_join(individualtrajectories, by = c("id2", "mnths"))
```

Selecting a Random Sample for Plotting

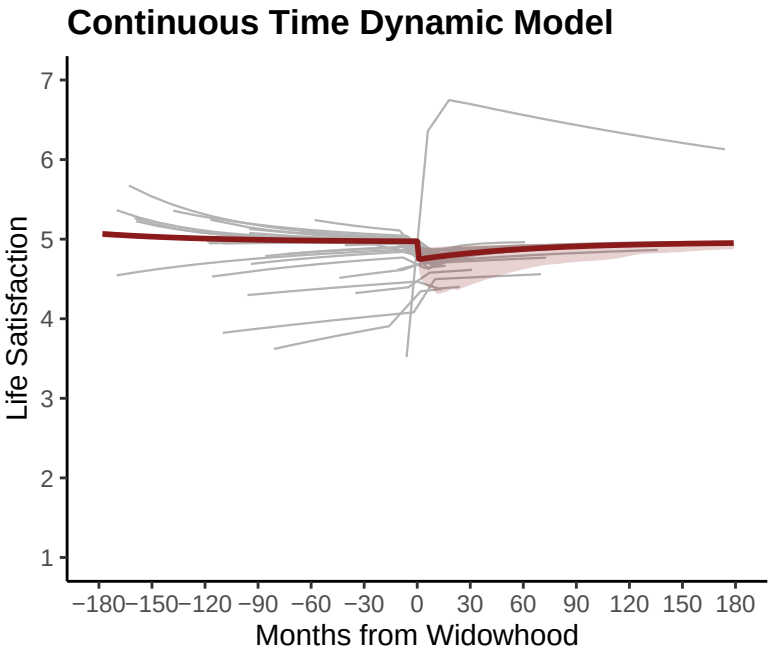
For better visualisation, select a random sample of individuals to display their individual trajectories.

```
# For reproducibility
set.seed(123)

# Randomly sample 50 participants
rsample_ids <- sample(unique(wido$id), 50)

# Filter the data to include only the randomly selected participants
wido_rsample <- wido %>%
  filter(id %in% rsample_ids)
```

Creating the Plot

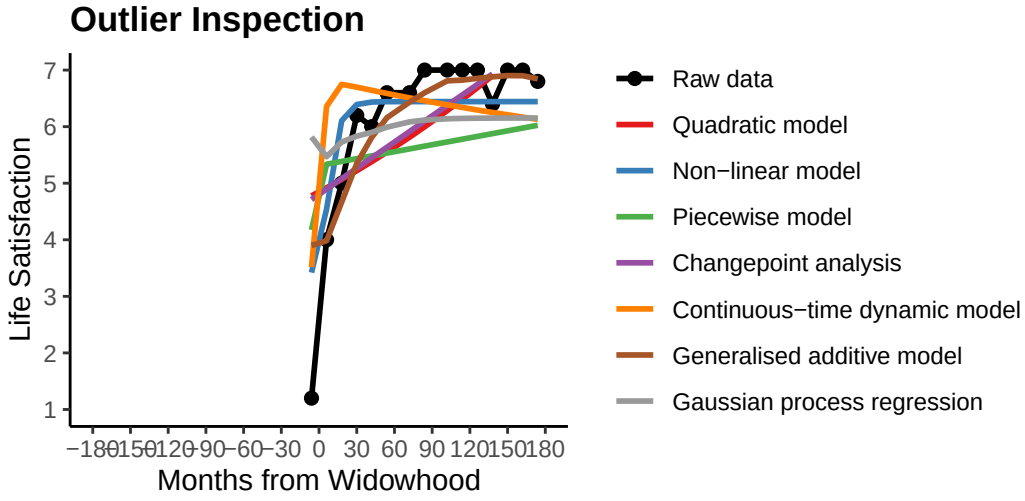
Combine all elements to create the plot, which includes individual trajectories, the population trajectory, and the confidence interval of the population trajectory.

```r
# Create the plot using the pre-configured plot base
plot_base +
  geom_line(
    data = wido_rsample,
    aes(x = mnths, y = ctdm_pred_r, group = id),
    color = "grey70",
    linewidth = 0.4
  ) +
  geom_ribbon(
    data = wido,
    aes(x = mnths, ymin = lower_CI, ymax = upper_CI),
    fill = "firebrick4",
    alpha = 0.2
  ) +
  geom_line(
    data = wido,
    aes(x = mnths, y = ctdm_pred_f),
    color = "firebrick4",
    linewidth = 1
  ) +
  ggtitle("Continuous Time Dynamic Model") +
  theme(plot.title = element_text(size = 13, face = "bold"))
```



Outlier Inspection

The plot above suggests that, for one individual, the continuous-time model produces an unusual prediction. To explore this further, the figure below displays this person's raw data (black connected points) alongside their predicted trajectory from each of the other models (coloured lines). Notably, this individual had only a single observation before widowhood, close to the time of widowhood, which was a very low life satisfaction score. Subsequently, their life satisfaction increased rapidly. While the continuous-time model's prediction is actually quite similar to those of the other models, it appears more prominent because this model tends to predict less variation across the other individuals' trajectories than the other models do. As a result, this particular case stands out more clearly in the continuous-time model than in the others.



Model Performance

Evaluating the Model

Assess the model's performance using the Bayesian Information Criterion (BIC), R-squared (R^2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

```
# Calculate R², MAE, and RMSE for the fixed effects predictions
data.frame(
  R2_FE = round(R2(wido$ctdm_pred_f, wido$m_lifesat_per_mnth), 2),
  MAE_FE = round(MAE(wido$ctdm_pred_f, wido$m_lifesat_per_mnth), 2),
  RMSE_FE = round(RMSE(wido$ctdm_pred_f, wido$m_lifesat_per_mnth), 2)
)
```

```
  R2_FE MAE_FE RMSE_FE
1  0.24   0.33    0.43
```

```r
# Calculate R², MAE, and RMSE for the random effects predictions
data.frame(
  R2_RE = round(R2(wido$ctdm_pred_r, wido$lifesatisfaction), 2),
  MAE_RE = round(MAE(wido$ctdm_pred_r, wido$lifesatisfaction), 2),
  RSME_RE = round(RMSE(wido$ctdm_pred_r, wido$lifesatisfaction), 2)
)
```

```
  R2_RE MAE_RE RSME_RE
1  0.27   0.81       1
```

Cross-Validation

To assess the replicability of the model, perform cross-validation using the training and test datasets. For each training dataset, fit the model and compute performance metrics for the associated test dataset R^2, MAE, and RMSE.

```r
# --- Preprocessing: Ensure a transition row (mnths == 0) for each individual
 ↪  ---
for (i in 1:length(training_datasets)) {

  training_data <- training_datasets[[i]]

  # Identify individuals missing a row at the transition point (mnths == 0)
  ids_to_add <- training_data %>%
    group_by(id) %>%
    filter(all(mnths != 0)) %>%
    distinct(id)

  # Create rows with mnths == 0 and transitionTime == 1 for those individuals
  new_rows <- ids_to_add %>%
    mutate(transitionTime = 1, mnths = 0)

  # Combine with original dataset
  training_data <- bind_rows(training_data, new_rows)
```

```r
  # Create or update the transitionTime variable (1 if mnths == 0, otherwise
  ↪  0)
  training_data <- training_data %>%
    mutate(transitionTime = if_else(mnths == 0, 1, 0))

  # Recode time into 5-year units for moderate time-scale dynamics
  training_data$fiveyrs <- round(training_data$mnths / 60, digits = 2)

  # Grand-mean scaling of life satisfaction items for the measurement model
  training_data$cp014_s <- scale(training_data$cp014)
  training_data$cp015_s <- scale(training_data$cp015)
  training_data$cp016_s <- scale(training_data$cp016)
  training_data$cp017_s <- scale(training_data$cp017)
  training_data$cp018_s <- scale(training_data$cp018)

  # Save the updated dataset back to the list
  training_datasets[[i]] <- training_data
}

# --- Initialise vectors for storing performance metrics ---
R2_values_ctdm <- c()
RMSE_values_ctdm <- c()
MAE_values_ctdm <- c()

# --- Fit the model and evaluate performance on each train/test split ---
for (i in 1:length(training_datasets)) {

  training_data <- training_datasets[[i]]
  test_data <- test_datasets[[i]]

  # Fit the ctsem model
  fit <- ctStanFit(
    datalong = training_data,
    ctstanmodel = model,
    iter = 2000,
    chains = 4
  )

  # Compute average life satisfaction per month from the test set
  test_data <- test_data %>%
    group_by(mnths) %>%
    mutate(m_lifesat_per_mnth = mean(lifesatisfaction, na.rm = TRUE))
```

```r
  # Predict population-level trajectory from model
  pred <- predict_average_trajectory()

  # Merge predictions with test data by time
  pred_ctdm_f <- merge(test_data, pred, by = "mnths")

  # Evaluate model performance
  R2_value <- R2(pred_ctdm_f$ctdm_pred_f, pred_ctdm_f$m_lifesat_per_mnth)
  RMSE_value <- RMSE(pred_ctdm_f$ctdm_pred_f, pred_ctdm_f$m_lifesat_per_mnth)
  MAE_value <- MAE(pred_ctdm_f$ctdm_pred_f, pred_ctdm_f$m_lifesat_per_mnth)

  # Store metrics
  R2_values_ctdm <- c(R2_values_ctdm, R2_value)
  RMSE_values_ctdm <- c(RMSE_values_ctdm, RMSE_value)
  MAE_values_ctdm <- c(MAE_values_ctdm, MAE_value)
}

# --- Aggregate the metrics across all folds ---
combined_metrics_ctdm <- data.frame(
  Metric = c("R²", "MAE", "RMSE"),
  Mean = round(c(mean(R2_values_ctdm), mean(MAE_values_ctdm),
   ↪  mean(RMSE_values_ctdm)), 2),
  SD = round(c(sd(R2_values_ctdm), sd(MAE_values_ctdm),
   ↪  sd(RMSE_values_ctdm)), 2)
)

# Display the results
print(combined_metrics_ctdm)
```

```
  Metric Mean   SD
1     R² 0.09 0.03
2    MAE 0.60 0.07
3   RMSE 0.78 0.10
```