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.

## 1 Preparation

### 1.1 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")

### 1.2 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)

## 2 Analysis

### 2.1 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  
  
 T0MEANS = 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  
)

### 2.2 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 T0MEANS 1 1 0.0929 0.0196 0.0539 0.0929 0.1309  
2 T0MEANS 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 0.6637 0.0420 0.5866 0.6626 0.7437  
AutoEffectLS 0.5816 0.1365 0.3543 0.5683 0.8837  
AutoEffectTransition 13.7779 5.5800 3.1013 13.6462 25.0453  
transitionEffect 17.9433 7.8872 4.9129 17.2545 35.0901  
  
$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 / unconstrained correlation form -- see $parmatrices for simpler interpretations!"  
  
$loglik  
[1] -11950.85  
  
$npars  
[1] 31  
  
$aic  
[1] 23963.69  
  
$logposterior  
[1] -11950.85

# 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 (T0MEANS) to the raw scale  
pmat$Mean[pmat$matrix == "T0MEANS" & pmat$row == 1] \* LS\_sd + LS\_mean

[1] 5.068943

## 3 Visualisation

### 3.1 Predicting Average Trajectory

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

# 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 (T0MEANS) matrix   
T0MEANS\_MATRIX <- matrix(c(  
 pmat$Mean[pmat$matrix == "T0MEANS" & 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]  
  
# 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 <- T0MEANS\_MATRIX # starting values  
xtraj[1, ] <- c(T0MEANS\_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()  
  
# Merge predicted values into main dataset  
wido <- wido %>%  
 left\_join(df, by = "mnths")

### 3.2 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.

# 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$T0MEANS[j], 0), nrow = 2)  
 TDPREDEFFECT\_MATRIX <- matrix(c(0, df\_posteriorsamples$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 <- T0MEANS\_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)  
  
# 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")

### 3.3 Predicting Individual Trajectories

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

# 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  
 T0MEANS\_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 <- T0MEANS\_MATRIX  
 xtraj[1, ] <- c(T0MEANS\_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"))

### 3.4 Selecting a Random Sample for Plotting

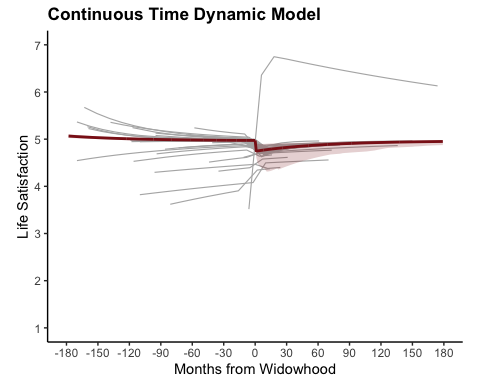
For better visualisation, select a random sample of individuals to display their individual trajectories.

# 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)

### 3.5 Creating the Plot

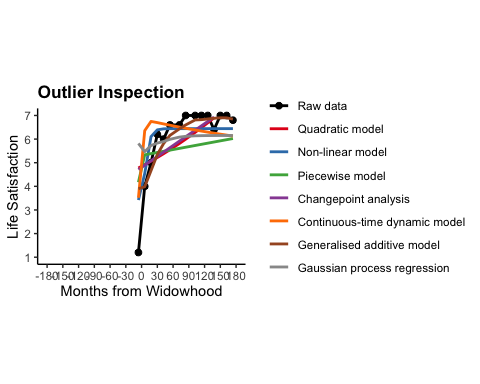
Combine all elements to create the plot, which includes individual trajectories, the population trajectory, and the confidence interval of the population trajectory.

# 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"))



### 3.6 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.



## 4 Model Performance

### 4.1 Evaluating the Model

Assess the model’s performance using the Bayesian Information Criterion (BIC), R-squared (R²), 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

# 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

### 4.2 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 R², MAE, and RMSE.

# --- 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)  
   
 # 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))  
   
 # 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