# Introduction to Joint Latent Class Modelling: Definitions, Functions, and Applications

## Joint Latent Class Modelling: Basic definitions

Joint latent class modelling (JLCM) can be understood, conceptually, as a combination of group-based trajectory modelling and a time-to-event survival analysis. In JLCM, we assume that a latent group structure explains both the trajectory of a longitudinal variable and a survival outcome. For illustrative purposes, consider a study in which researchers repeatedly assess psychosocial risk factors associated with recidivism in a group of individuals recently released from incarceration, while simultaneously keeping track of recidivism outcomes. Suppose that in this study, risk factors are reassessed monthly until the individual either completes their required supervision, returns to incarceration, or until a year has passed. A JLCM approach would assume that the sample of individuals is heterogeneous, with individual differences in assessed risk factors (including both initial risk assessments and patterns of change in risk over time) and likelihood of recidivism, but consists of latent subgroups, within which individuals share both the same patterns of risk assessments and the same risk for recidivism.

Throughout this chapter, I will use several terms to discuss the various components of JLCM. Note that these terms also apply to other joint modelling approaches. I will return to the hypothetical study introduced above for reference as I define these. The longitudinal component of joint latent class models describes the longitudinal trajectory of a *marker variable*. In our example, this is the trajectory of assessed risk levels. The term “marker” comes from the method’s origin in biostatistics, where the longitudinal component often details repeated measures of a biomarker of a disease process (e.g., counts of helper t-cells in patients with HIV). The marker variable is sometimes referred to as an indicator variable or simply as a longitudinal variable in the literature. The survival component of the JLCM describes *time-to-event*. *Time-to-event* in the example study is the follow-up time, prior to study dropout or recidivism. Similarly to a traditional survival model, the JLCM’s survival component is concerned with the order in which participants experience the *survival event*. In the example study, recidivism is the survival event, or the explicit outcome of interest. JLCM also considers *censorship*, when follow-up data ceases to be collected for an individual despite their not experiencing the survival event. In the example study, censorship occurs when an individual completes their supervision period without recidivism, or when the study period ends. Censorship can also include loss to follow-up. Although some applications of JLCM focus on predicting the survival event, other applications use the survival event to control for non-randomly missing data, while describing the trajectory of the marker variable. I will discuss potential applications in more detail later.

## What kind of data is JLCM appropriate for?

JLCM is suited for longitudinal data with repeated measurements of marker variables and time-linked outcome data. One of the advantages of this approach, relative to traditional group trajectory modelling is that it readily accommodates uneven assessment schedules. This means that no imputation is required when scheduled assessments are missed, and all data may be used, even from participants who remain only briefly in the sample. Multiple mutually exclusive outcomes may be considered. For example, one may wish to consider competing risks of violent and non-violent recidivism, in addition to censorship.

## What can joint latent class modelling tell me?

Researchers can use JLCM to describe longitudinal data or to create individualized predictions of how likely participants are to experience a survival event. Depending on the application, researchers may emphasize different components of the latent class joint model.

### Describing Trajectories

Similar to traditional group-based trajectory models, joint latent class models describe points of support for unobserved heterogeneity within a sample. When used effectively, these points of support may reveal subgroups that differ from each other, not only in degree of change in a marker variable over time, but in the quality of that change. As with traditional group-based trajectory models, latent class joint models will describe latent classes, even when given an underlying continuous distribution. For this reason, it is important to have evidence that individual variation in slopes and intercepts is present and a theoretical basis to hypothesize the existence of groups with qualitatively distinct patterns of change (see Qureshi & Fang, 2011).

Given appropriate assumption testing and careful use, both JLCM and traditional group-based trajectory models allow researchers to describe subgroups with different trajectories of a marker variable. In addition to describing each group’s trajectory, latent class joint models will also describe that group’s time-to-event. Because the groups are defined based on both the trajectory of the marker variable and the time-to-event, the baseline hazard associated with a given group is descriptive rather than predictive. However, this becomes predictive when researchers use a participant’s assessment history to infer probable membership in a particular trajectory group.

A strong advantage of JLCM over traditional latent class modelling is the flexibility it lends for dealing with missing data and uneven assessment schedules. This allows researchers to consider all data points, without requiring imputation or removal of participants with incomplete data. In cases where data is systematically missing (for example, due to incarceration), JLCM allows researchers to correct trajectories for bias. This advantage may be especially attractive for research in populations relevant to correctional psychology, where participants are potentially more transient or likely to drop out of studies following relapse or recidivism, and the assumption that data is missing at random is often untenable.

**Potential applications:** Describing patterns of change in psychosocial risk factors following release from incarceration, while controlling for recidivism and reincarceration; describing life course trajectories of criminal activity while controlling for mortality and loss to follow-up.

### Creating Individual Dynamic Predictions

Although researchers have often used traditional group-based trajectory models to make inferences about the likelihood of associated outcomes, the analytic approaches are limited to either post-hoc analyses (for example, using group designations as independent variables in ANOVAs or comparing recidivism rates within groups) or split studies (for example, growth trajectory mixture survival analysis) where the first period of a study is used to define groups and the second is used for prediction. With JLCM, individual predictions can be extracted for prediction windows at any point during the study. Additionally, researchers may use models developed from one sample to estimate predictions for a new set of a participants, or even a single participant. The *prediction window* is a period of time, defined by the researcher. These predictions are dynamic because they use all of a participant’s information up until the beginning of the prediction window to predict the likelihood that a survival event will occur during that window. The researcher may then test the performance of the prediction, assessing the discriminative ability and acuity of their model with AUCs and Brier scores respectively.

**Potential applications:** comparing the predictive discrimination of different risk assessments; testing whether risk factors are more relevant to predicting recidivism early in re-entry as opposed to after more time spent in the community; comparing prediction of competing outcomes, such as non-violent and violent recidivism.

# Implementation

## What goes into the model?

Recall that the JLCM combines group based trajectory modelling with survival analysis. It does this by combining three different submodels: a group membership submodel, a longitudinal submodel, and a survival/hazard submodel. I will describe each of these submodels in detail along with the decisions that researchers must make in the process. These are summarized in the table below for easy reference. For each of the following sections, more information can be found in Proust-Lima, Phillips, and Liquet (2017).

|  |  |
| --- | --- |
| Table 1. Summary of submodels within the latent class joint model. | |
| Group membership submodel | |
| Function | Describe the probability that an individual belongs to a group *g*, given a latent class structure |
| Equation |  |
| Researcher decisions | Whether to include covariates  Number of groups to consider |
| Longitudinal Submodel | |
| Function | Describe the class-specific trajectories of the marker variable |
| Equation | , |
| Researcher decisions | Choice of marker variable  Representation of time  Inclusion of class-general covariates  Inclusion of class-specific covariates  Inclusion of individual-level covariates  Trajectory shape (e.g., linear, polynomial, spline) |
| Hazard submodel | |
| Function | Describe the risk of the survival event occurring |
| Equation |  |
| Researcher decisions | Choice of survival event(s)  Inclusion of class-general covariates  Inclusion of class-specific covariates  Inclusion of cause-specific covariates  Family of hazard function (Weibull baseline risk function, piecewise constant risk function, or cubic M-splines baseline risk function) |

### Group membership submodel

The group membership submodel describes the likelihood that an individual belongs to a particular latent class. Researchers decide how many classes to include in the model. We assume that each individual *i* (*i* = 1,…, *N*) belongs to a single latent class characterised by marker trajectory and associated survival outcome. True latent class membership, , cannot be known. We therefore model the probability that an individual *i* belongs to each of the possible latent classes *g* (*g* = 1, …, *G*).

Researchers may choose to include covariates in their group membership submodel. This covariate should be time-independent. If a covariate is included, we would model probability that an individual *i* belongs to latent classes *g* (*g* = 1, …, *G*) according to covariates . In our example, I will use static risk as a group membership covariate.

Here, is the intercept for class *g* and is the vector of class-specific parameters associated with covariate . For identifiability, the scalar = 0 and the vector = 0. When no covariates are included in the group membership submodel, this equation reduced to a class-specific probability.

Researchers must specify the number of latent groups to be estimated. I recommend that researchers test a range of models with different numbers of groups. In our example, I will test models with up to 5 groups, but researchers often increase the number of groups until exclusion criteria are met (see model estimation, below, for a description of frequently-used exclusion criteria).

The latent class structure described by these equations is defined by both the longitudinal submodel and the survival submodel. For each of these submodels, researchers will have to decide on what variables to include.

### Longitudinal trajectory submodel

For the trajectory component of the model, researchers must decide on a markervariable. This is the variable that the trajectory will be describing. In our example, the marker variable is repeated assessments of dynamic risk. The trajectory of this marker variable is described using mixed modelling. Within the mixed model are fixed effects, mixture effects, and random effects. Intercepts are automatically included for each of these unless otherwise specified.

Fixed effects are common across individuals, and are assumed to have a uniform impact on the marker variable. The fixed effects include a *time* variable. Researchers may choose to enter time as a polynomial or as a linear term. In our example, the time variable is the week an assessment was completed after re-entry. One may also choose to include other covariates as fixed effects. Covariates could include demographic features, such as education level, or a measure of static risk – anything that needs to be accounted for when explaining how the marker variable changes over time. In our example, I include static risk as a fixed effect.

Mixture effects are only entered into the model when multiple latent groups are being estimated (i.e., not for the single group model). They describe class-specific fixed effects. The time variable must be included in the mixture effects. If additional covariates included in the fixed effects are expected to have class-specific regression parameters, they should be included as mixture effects as well.

Finally, there are random effects. These are the effects that are expected to vary across individuals. In the example study, participants vary in the number of reassessments recorded, so assessment times are included in the random effects.

In addition to deciding which covariates to include, researchers have the option to use a link function to define the trajectory shape (e.g., linear, polynomial) or approximate the shape using splines. I recommend the use of splines, unless there is reason to assume a specific shape.

Given a Gaussian outcome, the longitudinal trajectory of marker variable *y* for individual *i* at time *tij*(*j* = 1, …, *ni*) in class *g* is assumed to follow

,

where is the vector of covariates associated with class-common fixed effects, *,*; is the vector of those same covariates, associated with class-specific (mixture) effects, ; and is the vector of covariates associated with individual random effects, . describes a zero-mean stochastic process that is beyond the scope of what we need to understand and represents normally distributed standard error.

### Hazard submodel

The hazard submodel describes the class-specific likelihood that the survival event will occur prior to censorship. For this component of the latent class joint model, researchers must choose which survival event they wish to examine. In the example, the survival event is recidivism. Researchers may also choose to examine two competing events, if there are multiple outcomes of interest.

Within the hazard submodel, researchers may decide to include class-general covariates (assumed to have the same effect for all participants), class-specific covariates (effects vary depending on which latent group a participant probably belongs to), or, in the case of a competing risks model, cause-specific covariates (effects vary depending on which event is being predicted). These covariates should be time independent.

Researchers must choose what family of hazard function to use. Choices include Weibull baseline risk function, piecewise constant risk function, or cubic M-splines baseline risk function. Researchers should choose a baseline hazard function according to how survival events are distributed. There are several potential shapes of Weibull distribution. In re-entry studies, this is often a convex monotonic distribution (where plotted recidivism times make sort of a backwards J shape). Piecewise and spline functions are flexible and are generally able to accommodate many different hazard distributions. They estimate several different hazards for different portions of the follow-up period. I use a Weibull baseline risk function for our example.

In the class-specific survival model described below, denotes the time to the survival event, while denotes censoring time (i.e., time of exit from the dataset in the absence of the survival event). The time of exit from the dataset is thus and time to event is. Given the latent class *g* and survival up until time (*t*), the risk of an individual experiencing the survival event can be described using the following proportional hazard model:

,

Where indicates the class-specific baseline hazard, is the vector of covariates associated with parameters common over classes, *v*, and is the vector of covariates associated with class-specific parameters, .

# Procedure

## Data preparation

Data should be structured in person-period format: multiple rows of data for each participant, where each row indicates a reassessment. For each participant, some variables will be constant for each row. These variables include the participant identifier, the total survival time (i.e., either time until censorship or time until the survival event), the *status* of the participant, and any covariates that do not change over time (e.g., static risk assessments, time-independent demographic variables). The *status* variable refers to whether the participant experienced the survival event(s) or was otherwise censored. If a participant is censored, the value of their status variable would be 0; if they experienced the survival event it would be 1. Other variables will be time-linked, with different values for each row. These include the assessment time (this should be specified in the same unit as the variable denoting total survival time), the marker variable, and any covariates that are time-dependent. For illustration, this is what the dataset might look like for two individuals in the example study:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Participant Identifier | Total Survival Time (weeks) | Status | Static Risk Score (0-1.0) | Assessment time (weeks) | Dynamic Risk Score (0-15) |
| 1 | 20.00 | .00 | .21 | 0 | 7.00 |
| 1 | 20.00 | .00 | .21 | 1 | 7.00 |
| 1 | 20.00 | .00 | .21 | 2 | 7.00 |
| 1 | 20.00 | .00 | .21 | 3 | 7.00 |
| 1 | 20.00 | .00 | .21 | 4 | 6.00 |
| 1 | 20.00 | .00 | .21 | 5 | 6.00 |
| 2 | 9.00 | 1.00 | .52 | 0 | 8.00 |
| 2 | 9.00 | 1.00 | .52 | 1 | 8.00 |
| 2 | 9.00 | 1.00 | .52 | 2 | 9.00 |
| 2 | 9.00 | 1.00 | .52 | 3 | 9.00 |

Looking at the sample data, we can see that participant 1 had 6 separate assessments, and was censored after 20 weeks with no recidivism, whereas participant 2 was assessed on 4 separate occasions and recidivated after 9 weeks.

In some applications, researchers may wish to evaluate competing risks. The statistics package lcmm (Proust-Lima et al., 2017) allows for models with two competing survival events. In these applications, the status variable will differentiate between these; where a value of 0 indicates a participant being censored, 1 indicates survival event A, and 2 indicates survival event B. Although the example uses weeks as a measurement of time, other studies may use other units, such as participant age. Survival time in that case would be the age of the participant at the time of censorship or survival event. JLCM allows for left censoring, if there is variation in starting times. This would require an additional *entry* variable, constant for each participant, equal to the time of the first assessment. For example, in a study of life course crime rates, researchers could include an entry variable indicating the age at which participants began the study.

The software to conduct JLCM uses the R statistical program. An introduction to R is beyond the scope of this manual, but I have aimed to make the sample syntax as easy to adapt as possible. In the next sections, I will reference the example study when providing sample code. For this, I will name my variables, as one would for any data management software. Below are the variables from the example and their corresponding names for the sample code:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Variable | Participant identifier | Total Survival Time | Status | Static Risk Score | Assessment time | Dynamic Risk Score |
| Variable Name | ID | time | status | static | time\_assess | dynamic |

The code also requires a name for the data frame from which data is being drawn. The simulated dataset that I have created for this guide is named reentry\_data.

## Assumption checking

As mentioned previously, latent class joint modelling will estimate groups, even if the underlying distribution is continuous (Erosheva, Matsueda, & Telesca, 2014; Bauer & Curran, 2003). In large samples, indices of model fit may improve as groups are added despite a single underlying mean trajectory (Erosheva, Matsueda, & Telesca, 2014). For this reason, I recommend the framework prescribed by Qureshi and Fang (2011), in which the representation of time is determined and evidence of variation in individual slopes and intercepts is required prior to applying a latent class model.

### Specification of Time

Although Qureshi and Fang recommend latent curve modelling for this process, I found it easier to use mixed modelling with the lme4 package (Bates, Maechler, Bolker, and Walker 2014a) for R (R Core Team 2015). We can test how time ought to be represented by comparing indices of fit for models of our marker variable with different specifications of time (e.g., linear, quadratic, cubic). Using our sample data, the code to do this is:

|  |
| --- |
| Syntax for determining representation of time |
| Linear\_model<-lmer(dynamic~time\_assess+(1+time\_assess|ID),  data=reentry\_data)  summary(Linear\_model)  Quad\_model<-lmer(dynamic~poly(time\_assess,2)+(1+time\_assess|ID),  data=reentry\_data)  summary(Quad\_model)  Cubic\_model<-lmer(dynamic~poly(time\_assess,3)+(1+time\_assess|ID), data=reentry\_data)  summary(Cubic\_model)  anova(Linear\_model,Quad\_model,Cubic\_model) |

In this example, I have chosen to include random effects for intercepts and for the linear component of time. Quadratic and cubic components of time have been included as fixed effects only. The syntax includes a command to show summaries of the models before completing the ANOVA to compare fit. This is good practice in general, and allows researchers to check that they have entered everything correctly and that their model has converged appropriately. Sometimes, models may have difficulty converging as they become more complex. Adding an optimizer to the syntax can help with this. As an example, this is what the quadratic model would look like with the “bobyqa” optimizer:

|  |
| --- |
| Syntax for adding an optimizer |
| Quad\_model<-lmer(dynamic~poly(time\_assess,2)+(1+time\_assess|ID),  data=reentry\_data,  control = lmerControl(optimizer = "bobyqa",  optCtrl = list(maxfun = 1e5)))  summary(Quad\_model) |
|  |

Output from the ANOVA using our sample data looks like this:

|  |
| --- |
| Sample ANOVA output for time specification |
| Data: reentry\_data  Models:  linear\_model: dynamic ~ time\_assess + (1 + time\_assess | ID)  quad\_model: dynamic ~ poly(time\_assess, 2) + (1 + time\_assess | ID)  cubic\_model: dynamic ~ poly(time\_assess, 3) + (1 + time\_assess | ID)  npar AIC BIC logLik deviance Chisq Df Pr(>Chisq)  linear\_model 6 940.45 966.17 -464.22 928.45  quad\_model 7 942.29 972.30 -464.14 928.29 0.1592 1 0.6899  cubic\_model 8 943.35 977.65 -463.67 927.35 0.9385 1 0.3327 |
|  |

From this output we can see that AICs and BICs are both lowest for the linear model. We therefore opt to use a linear representation of time.

**Variability in slopes and intercepts.** We can test for variability in slopes and intercepts using the same program. To do this, we will consider fit indices of three different models of our marker variable. The first model, m1, will contain only the marker variable regressed onto random intercepts. The second model, m2, will regress the marker variable onto the time variable, with random intercepts and fixed slopes. In the third model, m3, slopes and intercepts are both entered as random effects. We use ANOVA to test if allowing slopes and intercepts to vary improved model fit. The sample syntax for this is:

|  |
| --- |
| Syntax for establishing variability of slopes and intercepts |
| m1<-lmer(dynamic~(1|ID),data= reentry\_data)  summary(m1)  m2<-lmer(dynamic~time\_assess+(1|ID),data= reentry\_data)  summary(m2)  m3<-lmer(dynamic~time\_assess+(1+time\_assess|ID),data= reentry\_data)  summary(m3)  anova(m1,m2,m3) |

Output from the ANOVA (below) indicates that model fit is improved by allowing slopes and intercepts to vary. This suggests that, given a theoretical reason to support a group-trajectory approach, it is appropriate to estimate trajectory groups.

|  |
| --- |
| Sample ANOVA output for establishing variability of slopes and intercepts |
| Data: reentry\_data  Models:  m1: dynamic ~ (1 | ID)  m2: dynamic ~ time\_assess + (1 | ID)  m3: dynamic ~ time\_assess + (1 + time\_assess | ID)  npar AIC BIC logLik deviance Chisq Df Pr(>Chisq)  m1 3 1048.66 1061.52 -521.33 1042.66  m2 4 1024.91 1042.06 -508.46 1016.91 25.747 1 3.892e-07 \*\*\*  m3 6 940.45 966.17 -464.22 928.45 88.466 2 < 2.2e-16 \*\*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1 |

## Model selection

After we have decided which variables to include in our model, demonstrated variability in slopes and intercepts, and have theoretical reason to believe that latent groups may exist, it is time to begin with model selection. As with traditional group-based trajectory modelling, this process requires caution on the part of the researcher. The number of groups will ultimately be chosen according to a variety of indicators. These indicators may differ significantly with small changes in model specification. It is especially prudent to have logical or theoretical justification for the covariates included.

One way that we can increase our confidence that our results reflect real trends in the data (in contrast to artefacts of group specification) is to use a split sample. This is also helpful, as it reduce computation times for more complex models. As our sample data includes on 100 participants, I will not be splitting the sample. If I were to do so, I would apply the model selection steps to 60% of the participants, then test the selected model using the remaining 40% of the participants. If the trajectories are similar across both groups, it supports that the estimated trajectories are truly representative of the data.

To select a model, we must compare models with different numbers of latent classes specified. This will be done using the R program, lcmm (Proust-Lima, Philipps, Diakite, & Liquet, 2019). It is generally recommended to test models with up to 6 latent classes, but it is first necessary to fit a model with the same structure but with only one latent class. I will call this model mod\_init, and use the following syntax:

|  |
| --- |
| Syntax for initial model |
| mod\_init<-Jointlcmm(dynamic~time\_assess+static,random=~time\_assess,  survival=Surv(time,status)~static,  hazard="Weibull",subject="ID",data=reentry\_data,  ng=1,link="splines")  summary(mod\_init) |

As this model only uses one group, I have not included any covariates in the group membership submodel. I have chosen to include static as a covariate in the longitudinal submodel and as a class-general effect in the survival submodel. Notice that there are no mixture effects when only one group is estimated. We have used the default hazard, which is a Weibull baseline risk function. Additionally, we have selected splines as our link function. This allows for flexibility in trajectory shapes. Some post-fit plotting functions in lcmm only work with linear link functions (e.g., expected prognostic observed cross-entropy, see Proust-Lima, 2017). Researchers wishing to use these function may specify time as a polynomial directly, rather than using splines.

The single-group model will be used to generate initial values for the next models. Initial values are important for making sure that maximum likelihood estimates are based on global rather than local maxima. They also reduce computation time. Initial values are discussed more thoroughly in the appendix. We will be using a gridsearch function of the initial model to generate initial values for subsequent models. Sample syntax for a two-group model (to increase the number of groups, change ng = to 3, 4, etc.) is:

|  |
| --- |
| Syntax for joint latent class models |
| mod\_2g<-gridsearch(rep=3,maxiter=50,minit=mod\_init,  Jointlcmm(dynamic~time\_assess+static,  mixture=~time\_assess,  random=~time\_assess,  survival=Surv(time,status)~static,  classmb=~static,  hazard="Weibull",subject="ID",  data=reentry\_data,  ng=2,link="splines",verbose=FALSE)) |

In the two-group model we are able to introduce mixture effects (mixture=~time\_assess) and a group membership covariate (classmb=~static). As the models become more complex, their computation times will increase. Although specifying appropriate initial values and using a split sample helps reduce computation time, a single model may still take several hours to process.

To select a model, we need to consider several selection criteria. One of these is the Bayesian information criterion (BIC). This an indicator of model fit, where smaller values indicate better fit. BICs often continue to drop as groups are added, especially in large samples. For this reason, it is good to consider other selection criteria. The conditional independence test statistic should be used to give preference to models where the assumption of conditional independence is upheld. Both of these values are easily obtained with the summary command.

|  |
| --- |
| Syntax for model summary |
| summary(mod\_2g) |

This yields output:

|  |
| --- |
| Sample output for model summary |
| > summary(mod\_2g)  Joint latent class model for quantitative outcome and competing risks  fitted by maximum likelihood method    Jointlcmm(fixed = dynamic ~ time\_assess + static, mixture = ~time\_assess,  random = ~time\_assess, subject = "ID", classmb = ~static,  ng = 2, survival = Surv(time, status) ~ static, hazard = "Weibull",  link = "splines", data = reentry\_data, verbose = FALSE)    Statistical Model:  Dataset: reentry\_data  Number of subjects: 100  Number of observations: 538  Number of latent classes: 2  Number of parameters: 21  Event 1:  Number of events: 27  Class-specific hazards and  Weibull baseline risk function  Link function for dynamic: Quadratic I-splines with nodes 7 8 9 10 13    Iteration process:  Convergence criteria satisfied  Number of iterations: 1  Convergence criteria: parameters= 2e-16  : likelihood= 1.1e-13  : second derivatives= 3.5e-17    Goodness-of-fit statistics:  maximum log-likelihood: -561.9  AIC: 1165.8  BIC: 1220.51  Score test statistic for CI assumption: 0.102 (p-value=0.9504)    Maximum Likelihood Estimates:    Fixed effects in the class-membership model:  (the class of reference is the last class)  coef Se Wald p-value  intercept class1 8.12774 4.29987 1.890 0.05873  static class1 -11.59128 5.72182 -2.026 0.04279  Parameters in the proportional hazard model:  coef Se Wald p-value  event1 +/-sqrt(Weibull1) class 1 0.04035 0.01730 2.332 0.01972  event1 +/-sqrt(Weibull2) class 1 1.18183 0.12791 9.240 0.00000  event1 +/-sqrt(Weibull1) class 2 0.06180 0.05028 1.229 0.21907  event1 +/-sqrt(Weibull2) class 2 1.82574 0.53591 3.407 0.00066  static 9.28773 1.67990 5.529 0.00000  Fixed effects in the longitudinal model:  coef Se Wald p-value  intercept class1 (not estimated) 0  intercept class2 -0.86267 1.22181 -0.706 0.48015  time\_assess class1 -0.19204 0.06245 -3.075 0.00210  time\_assess class2 -0.10179 0.12890 -0.790 0.42972  static 4.66054 1.81611 2.566 0.01028  Variance-covariance matrix of the random-effects:  intercept time\_assess  intercept 9.06030  time\_assess -0.29338 0.18235  Residual standard error (not estimated) = 1  Parameters of the link function:  coef Se Wald p-value  I-splines1 -4.60523 0.78084 -5.898 0.00000  I-splines2 1.24974 0.08252 15.146 0.00000  I-splines3 1.19259 0.16832 7.085 0.00000  I-splines4 1.92987 0.11528 16.741 0.00000  I-splines5 1.74417 0.15809 11.033 0.00000  I-splines6 2.31915 0.20321 11.413 0.00000  I-splines7 1.04728 0.41861 2.502 0.01236 |

From this long list of output, we would focus on the goodness of fit statistics. For our tables, we’ll be using the BIC and the score for the test statistic of the conditional independence (CI) assumption. Note that the output indicates a non-significant CI test statistic (p-value = 0.95).

Other selection criteria are based on post-fit indices. These are more important for ensuring that the groups identified are meaningful. Generally, groups identified should contain at least 1% of the sample to be considered. Mean posterior probabilities should be greater than 0.7 for each group. These are used as *exclusion criteria,* meaning that we exclude solutions that yield groups that do not meet these criteria. Group sizes and posterior probabilities can be retrieved with the postprob command.

|  |
| --- |
| Syntax for post-fit computations |
| postprob(mod\_2g) |

The output from this command looks like this:

|  |
| --- |
| Sample output for post-fit computations |
| > postprob(mod\_2g)    Posterior classification based on longitudinal and time-to-event data:  class1 class2  N 79 21  % 79 21    Posterior classification table:  --> mean of posterior probabilities in each class  prob1 prob2  class1 0.9634 0.0366  class2 0.0855 0.9145    Posterior probabilities above a threshold (%):  class1 class2  prob>0.7 96.20 90.48  prob>0.8 94.94 90.48  prob>0.9 91.14 66.67      Posterior classification based only on longitudinal data:  class1 class2  N 81 19  % 81 19 |

From these output, we can see that 79% of participants likely belong to group 1, and that the mean posterior probability for this group is 0.96. We can see that 21% of participants likely belong to group 2, and that the mean posterior probability for this group is 0.91. As no exclusion criteria are met, we can go on to test the three-group model and so on. Syntax for these models (as well as all presented syntax) are available on my GitHub (<https://github.com/agstone90/JLCM>)

Relative entropy is also useful to consider in selecting a model. I use an equation provided by Brilleman (2019). A separate equation is needed for calculating relative entropy. Here is the syntax to calculate relative entropy for our two-group model:

|  |
| --- |
| Syntax for computing relative entropy |
| entropy <- function(mod\_2g) {  pp <- mod\_2g$pprob[3:ncol(mod\_2g$pprob)]  N <- nrow(pp) # number of individuals  J <- mod\_2g$ng # number of latent classes  ent <- -sum(pp \* log(pp)) # entropy  res <- 1 - (ent / (N \* log(J))) # relative entropy  res  }  entropy(mod\_2g) |

The entropy for the 2 group model is 0.82. Higher entropy indicates a better fitting model.

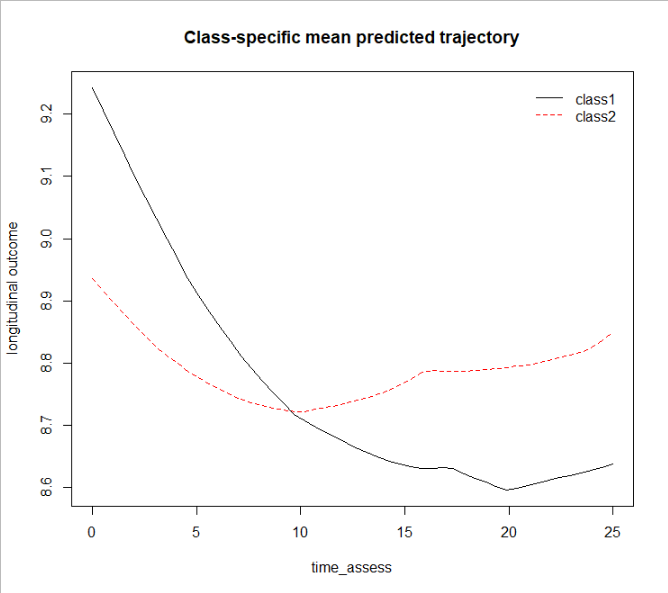
It can also be helpful at this point to plot predicted mean trajectories for your models as a visual aid. To do this in lcmm we need to create a new data frame, with only our time variable and covariates. The syntax below specifies that the new data frame will include 50 time points, between assessment times 0 and 25, as well as static assessments set to 0.46, repeated 50 times (0.46 is the mean static score; the number of repeats doesn’t matter, but it should be the same for both the time and the covariate).

|  |
| --- |
| Syntax for creating a new data frame |
| newdata<-data.frame(time\_assess=seq(0,25,length=50),  static=rep(0,50)) |
|  |

Creating this data frame with covariate levels fixed is necessary, because the mean predicted trajectories are dependent on the levels of the covariate. This new data frame is used to create predicted values of the marker variable for plotting. The resulting plots show what the mean trajectories of the marker variable would be if participants remained in the sample for 25 weeks. The syntax below would be used to calculate and plot predicted trajectories for the two-group model:

|  |
| --- |
| Syntax for predicting and plotting trajectories |
| mod2\_predtraj<-predictY(mod\_2g,newdata, var.time="time\_assess")  plot(mod2\_predtraj) |
|  |

The resulting plot should look something like this. Note that as we haven’t specified limits for the y axis the amount of change appears greater than it actually is.



Once we have run all of our models for different group specifications and compiled the corresponding selection criteria, it is time to select a model. We have organized the output from our sample data into a handy table to help with this process. As our sample was small, only 2 and 3 group solutions converged, but researchers often test as many as 8 groups.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | BIC | CI Test Statistic | Relative Entropy | Mean Posterior Probability within Class (Percentage) | | | | | | | |
|  |  | 1 | 2 | 3 | 4 | 5 | | 6 | |
| 2 classes | 1220.51 | 0.10 | 0.82 | 0.96 (79) | 0.91 (21) |  |  | |  | |  |
| 3 classes | 1234.38 | 0.57 | 0.86 | 0.92 (22) | 0.95 (75) | 0.98 (3) |  | |  | |  |

Sometimes, all of the selection criteria will converge on the same solution. More frequently, there will be conflicting indications. Different researchers choose to deal with this in different ways, and this is a major criticism of latent class modelling. It is common to look for inflection point in BICs (a point at which model fit no longer improves as classes are added). However, in large samples BICs will often simply continue to drop. When this happens, it can be useful to use a process of elimination. In joint latent class modelling, a good first step is to eliminate from consideration any models in which the conditional independence assumption is violated. We can also eliminate models in which any of the groups has less than 1% of the sample or where mean posterior probabilities within class is lower than 0.7. Using this process with the results from our sample data, we would select the 2 group model, as BIC is lower.

This process can seem unnervingly arbitrary at times. Indeed, small changes to model specification may result in different balances of selection criteria, despite trajectory shapes remaining relatively stable. This should serve as a reminder to researchers: the groups being identified are not objectively real entities; they merely describe patterns of unobserved heterogeneity. An individual participant’s designation in a given group is probabilistic and not definite. Groups are defined by the trajectories and survival times of individuals within them and not vice versa. With this in mind, we may move on to checking the model selected from the calibration sample against the remaining test sample.

## Presenting the final model

We can use the selected model to make a variety of plots. Those presented here are not exhaustive, see Proust-Lima et al (2017, 2019) for additional available plotting functions.

### Predicted mean trajectories, hazard rates, and survival curves

Plotting predicted mean trajectories is useful to illustrate differences in trajectory shapes. This can be done using the same syntax we used during model selection. We may also wish to include confidence intervals. The syntax below calculates values for predicted trajectories.

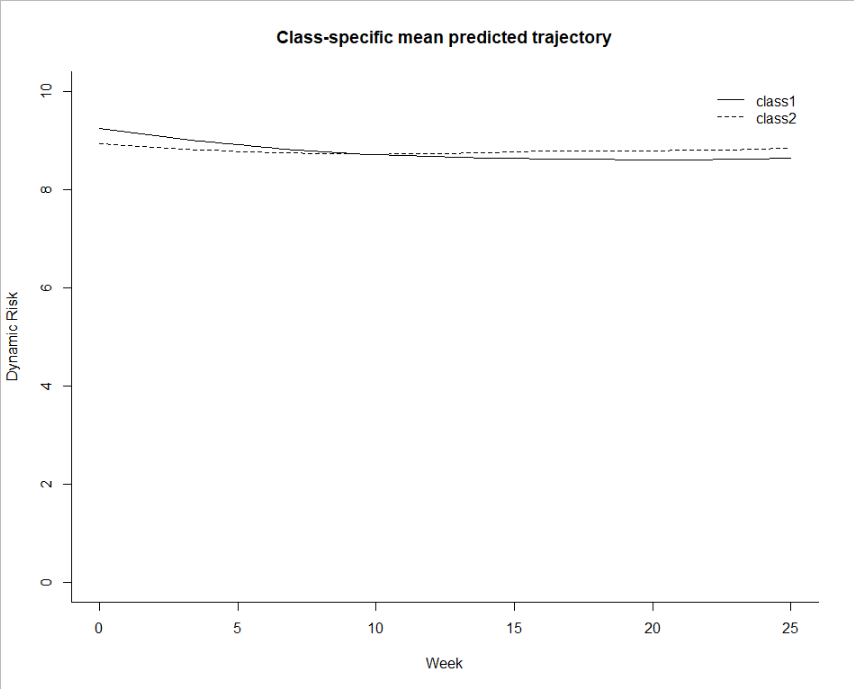
|  |
| --- |
| Syntax for creating a new data frame and predicting trajectories |
| newdata<-data.frame(time\_assess=seq(0,25,length=50),  static=rep(0.46,50))  predtraj<-predictY(mod\_2g,newdata, var.time="time\_assess")  plot(predtraj) |
|  |

We can calculate Monte Carlo confidence intervals by adapting the syntax slightly:

|  |
| --- |
| Syntax for predicting trajectories with confidence intervals |
| predtraj\_MC<-predictY(mod\_2g,newdata,draws=TRUE,var.time="time\_assess") |
|  |

Because this is our final model, we may want to add specifications that will make the plot look nice:

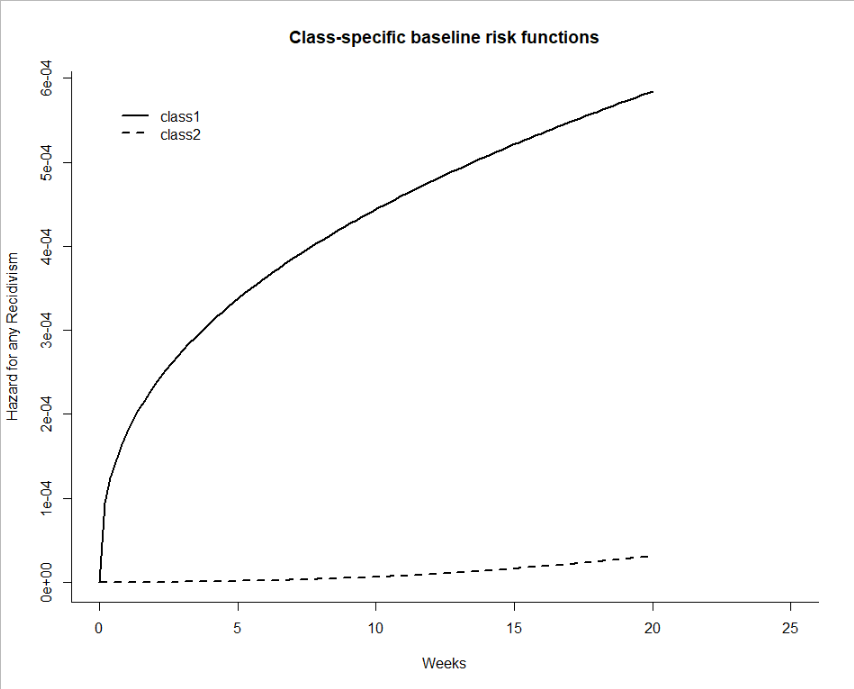
|  |
| --- |
| Syntax for plotting predicted trajectories but fancy |
| plot(predtraj, bty = "l",  ylim = c(0, 15), xlim=c(0,25),  ylab = "Dynamic Risk",  xlab = "Week", lwd = 1,col=1) |
|  |

Given our sample data, the resulting plot looks like 

Plotting predicted survival curves and hazard rates is also helpful to illustrate group differences. These plots are simple to make and do not require the creation of a new data frame. The syntax for creating a plot of hazard rates is:

|  |
| --- |
| Syntax for plotting predicted hazard rates |
| plot(mod\_2g, which = "hazard",  lwd = 2,bty = "l",  xlab = "Weeks",  ylab = "Hazard for any Recidivism",  xlim=c(0,25),col=1) |
|  |

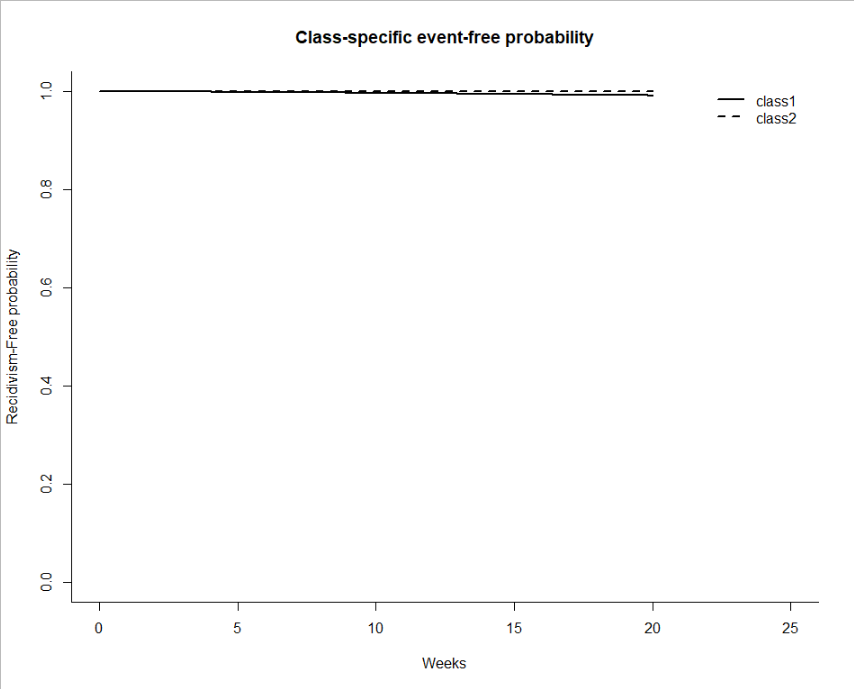
The resulting plot looks like



The syntax for creating a plot of predicted survival curves is similar. We will need to change the prediction function from hazard to survival and change the axis labels accordingly:

|  |
| --- |
| Syntax for plotting predicted hazard rates |
| plot(mod\_2g, which = "survival",  lwd = 2,bty = "l",  xlab = "Weeks",  ylab = "Recidivism-Free probability",  xlim=c(0,25),col=1) |
|  |

Which, using our data, produces the corresponding plot



### Illustrate within-group variability with sample trajectories

Although the mean predicted trajectory plots in the above section are good summaries of the trajectory shapes, they do little to illustrate either trends in trajectory length or variability within latent groups. Plotting sample trajectories is an effective way to illustrate both noise within groups and how time-to-event varies between groups. In this section, I describe the steps and provide sample syntax for creating these plots.

The first step for doing this is to extract group membership and posterior probabilities for group membership from the selected model. The syntax below creates a new data frame, m2\_pprob and joins that with the main data frame, reentry\_data. This requires the dplyr (Wickham et al., 2021)package.

|  |
| --- |
| Syntax for extracting groups and joining with main data frame |
| m2\_pprob<-as.data.frame(mod\_2g$pprob[,1:4])  reentry\_data<-left\_join(reentry\_data, m2\_pprob) |
|  |

Once the variable class is joined with the main data frame, we can create separate data frames for participants designated to each latent group. To create a data frame containing only information for participants designated to the first latent group, we would use the syntax:

|  |
| --- |
| Syntax for creating specific group data frame |
| m2\_group1<-reentry\_data[which(reentry\_data$class==1),] |
|  |

From this data frame we can then select 50 sample participants into a data frame named group1\_sample:

|  |
| --- |
| Syntax for selecting sample of 50 |
| ID<-as.character(m2\_group1$ID)  ID<-as.data.frame(ID,ID=ID)  ID<-ID[!duplicated(ID$ID),]  ID<-as.character(ID)  ID<-as.data.frame(ID, ID=ID)  ID$ID<-as.character(ID$ID)  m2\_group1\_sample<-sample(x=ID$ID, size = 50, replace = F,  set.seed(1))  m2\_group1\_sample<-data.frame(m2\_group1\_sample, ID=m2\_group1\_sample)  m2\_group1\_sample<-m2\_group1[m2\_group1$ID%in%m2\_group1\_sample$ID,] |

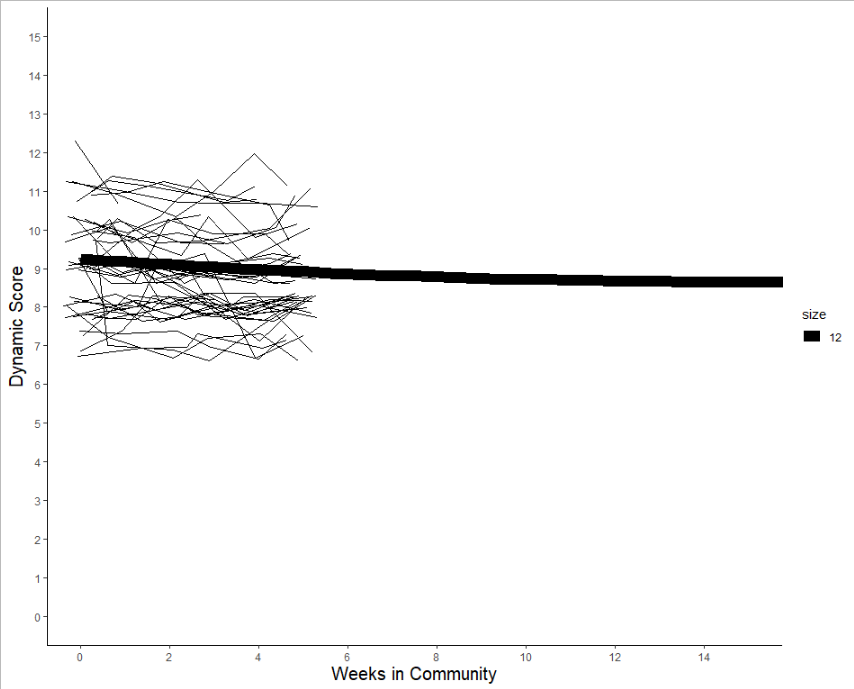
We can then plot these 50 sample trajectories. For illustrative purposes, it may be useful to include the mean predicted trajectory for the group in this plot. Previously, I provided syntax to calculate predicted trajectories for plotting (object named predtraj). To include them in the sample trajectory plots, we will need to set these predicted trajectories into their own data frame, predtraj\_df:

|  |
| --- |
| Syntax for creating a data frame of predicted mean trajectories |
| predtraj\_df<-as.data.frame(predtraj$pred[,1:2],predtraj$times[,1]) |
|  |

The program ggplot2 allows us to plot both of these together with the following syntax:

|  |
| --- |
| Syntax for plotting sample trajectories with mean trajectories |
| p1<-ggplot()+geom\_line(data=m2\_group1\_sample,  aes(x=time\_assess,y=dynamic,group=ID),position="jitter")+ geom\_line(data=predtraj\_df,aes(x=as.numeric(row.names(predtraj\_df)), y=Ypred\_class1, size=12))+  ylab("Dynamic Score")+  xlab("Weeks in Community")+  coord\_cartesian(xlim=c(0,15),ylim=c(0,15))+  scale\_x\_continuous(breaks=seq(0,25,2))+  scale\_y\_continuous(breaks=seq(0,15,1))+  theme\_classic()+  theme(axis.title.x = element\_text(size=15),  axis.title.y = element\_text(size=15))  p1 |
|  |

This produces a plot that looks like



Setting the position to jitter prevents trajectories from overlapping. This makes the individual trajectories more apparent, especially when the marker variable is ordinal rather than continuous, but also causes trajectories to appear as though they are oscillating when they may not be. The choice to use jittering is one of personal preference, and is easily reversed by removing the specification.

We may repeat this process for the remaining latent groups. Note that if a latent group contains fewer than 50 participants, it will be necessary to reduce the number of participants sampled for plotting.

### Dynamic Predictions

JLCM allows the exciting prospect of individualized predictions. This function also allows us to test the predictive discrimination and acuity of our selected model. The procedure to do this is described in detail by Blanche et al. (2015), and the syntax required are included in the supplemental materials of their [manuscript](https://onlinelibrary.wiley.com/doi/full/10.1111/biom.12232) on the *Biometrics* website on Wiley Online Library. Here, I will describe the preliminary steps necessary to use those syntax to compute dynamic AUCs and Brier Scores from the model.

The function to create dynamic predictions, dynpred requires users to input data including all the variables included in their joint model. Because we are using dynamic prediction to test prediction within our own sample, this data frame will be identical to our original dataframe (reentry\_data). Recall that researchers define the prediction window for dynamic predictions. The dynpred function requires us to select “landmarks” to indicate the beginning of our desired prediction windows, and a single “horizon” to indicate how long we want the prediction window to be. Researchers should describe these in the same unit as their time variables. The syntax below instructs the program to return predictions for three prediction windows, from weeks 4 to 14, 12 to 22, and 24 to 34:

|  |
| --- |
| Syntax for creating dynamic predictions |
| dpred <-dynpred(mod\_2g, landmark = c(4,12,24),  newdata = reentry\_data ,horizon=c(10),  var.time = "time\_assess", event = 1) |

Running this function should return a list of survival probabilities for each participant during the prediction windows. Note that the function will automatically only use data from participants who have not yet been censored at the beginning of the prediction window.

|  |
| --- |
| Sample output of dynamic predictions |
| ID landmark horizon pred  [1,] 1 4 10 0.065219781  [2,] 1 12 10 0.083891551  [3,] 1 24 10 0.102837210  [4,] 2 4 10 0.055400684  [5,] 2 12 10 0.071376484  [6,] 2 24 10 0.087652399  [7,] 3 4 10 0.020972528  [8,] 3 12 10 0.027171332  [9,] 3 24 10 0.033555905 |

To use these dynamic predictions to calculate AUCs, we need to create a data frame where each participant’s data is in a single row. The easiest way to do this is to first extract IDs, landmarks, and predictions from the dynpred output into a new data frame and convert the format using the tidyr program’s pivot\_wider function.

|  |
| --- |
| Syntax for creating data frame of predictions |
| pred\_df<-dpred$pred%>%dplyr::select(ID, landmark, pred)  pred\_df<-pivot\_wider(pred\_df, id\_cols = ID,  names\_from = landmark, values\_from = pred) |

We then need create a data frame containing those participants’ time and status variables in single rows (along with ID for matching) and then connect the two data frames:

|  |
| --- |
| Syntax for adding participants’ survival time and status to prediction data frame |
| timestatus<-reentry\_data[c("ID","time","status")]  timestatus<-timestatus[!duplicated(timestatus),]  pred\_df<-left\_join(pred\_df,timestatus ) |

Entering view(pred\_df) allows us to see the new data frame, consisting of six variables: ID, 4, 12, 24, time, and status. We want to re-name variables 4, 12, and 24, because having them as just numbers is confusing. We will also save the data frame as a CSV file, as this is the format that the syntax provided by Blanche et al. (2015) uses.

|  |
| --- |
| Syntax for changing variable names and creating CSV |
| names(pred\_df)<-c("ID","pred.dynamic.4","pred.dynamic.12","pred.dynamic.24","time","status")  write.csv(pred\_df,"pred\_df.csv") |

The rest of the procedure to calculate AUCs and Brier scores requires the syntax from Blanche and colleagues’ [supplemental files](https://onlinelibrary.wiley.com/doi/full/10.1111/biom.12232). I will not reproduce their syntax in full here, but I will provide instructions for adapting their syntax to work for our example data. We only need to alter the syntax in the “Main.R” file. The syntax in the “Rfunctions” folder are source code (basically instructions for functions in the Main.R file) and should be run in full prior to running the adapted Main.R syntax.

After changing the “read data” section to read our CSV instead of the sample data from Blanche et al, we will need to change the parameter definitions for the prediction windows. The adapted syntax will look like:

|  |
| --- |
| Adapted syntax defining parameters |
| ComputeTheLoop <- TRUE)  landmark.times <- c(4,12,24)  windowt <- 8 |

After that, we just need to change object names to match our variables. The easiest way to do this is to just control+F and replace their variable “IST” with our variable “dynamic”. We can delete (or # out) the syntax for MMSE (lines 50-56, 67, 91-111, 119, and 122) because we only have one predictor variable.