## 2. Modeling

The next code block executes the cNORM modeling process, in which age-related development of a latent ability is modeled using raw scores from a normative sample. This model of development is eventually operationalized as a set of raw-to-norm score lookup tables.

As noted previously, cNORM processes one raw score at a time. Here, read\_csv() is used to read in one of the single-score data frames written out by upstream code. The resulting object is named input. The file path is concatenated from previously initialized tokens. In the present example, iws\_sum is the raw score to be normed.

cnorm() is the modeling function, and its arguments are as follows:

* **raw**: designates the raw score column in the input data file (here input$raw).
* **group**: designates the age group column in the input data file (here input$group).
* **k**: a power constant that sets the general limit on the expansion of the Taylor polynomial series, which controls the precision of estimation of the normative model (i.e., the limit for location and age). The default value is 4, and values of 3 and 5 can be used in searching for an acceptable model. k = 5 is computationally intensive and entails processing times of several minutes or more on many CPUs.
* **t**: a power constant that also sets the limit on the expansion of the Taylor polynomial series, but for the explanatory variable (e.g. age or grade) only. The optimal t depends on the relation between the dependent variable and the explanatory variable. For example, if the dependent variable increases linearly with the explanatory variable, t should be set to 1. But most cases, the combination of k = 5 and t = 3 will provide good results with limited computational effort.
* **terms**: sets the number of terms in the regression equation that expresses the normative model. cnorm() finds the best-fitting model with this number of terms. By convention, we use a starting value of 4 for terms.
* **scale**: sets the metric of the norm score. Values can be "IQ","T","z", "percentile", or a vector that provides the mean and standard deviation of the preferred metric (e.g., c(10, 3)).

The initial output of cnorm() is a model summary printed in the console, and a plot of the observed and predicted percentile curves associated with the norming model. The plot shown below is for the iws raw score in the input sample.

At this point, the norming workflow extends outside the mere sequential execution of this script. The initial model is evaluated, using diagnostic tools available within cNORM. If this initial model is judged acceptable, the output phase of the norms process can proceed. However, if the initial model is problematic, we re-run the cnorm() function with different values of k and terms, and examine these subsequent models with the diagnostic aids, repeating the process until we settle on a model that is acceptable (or, at least, the model with the fewest observable flaws).

*Monotonicity* is the primary criterion for evaluating the normative model. When we test a developing cognitive ability, we expect raw scores to increase monotonically (that is, increase without ever decreasing) with increasing age. Consequently, we expect that a specific raw score will be associated with progressively lower norm scores as age increases. This occurs because the mean raw score increases from one age group to the next, and, as a result, an identical raw score moves to a lower rank in the next oldest age group, and so on.

The best way to check monotonicity after running cnorm() is to examine the plot of percentile curves associated with the model. In this plot, the *x*-axis variable is the explanatory variable, and the *y*-axis variable is raw score. The plot illustrates seven percentile ranks: 2.5, 10, 25, 50, 75, 90, 97.5. The colored circles represent the actual raw scores associated with these percentiles, per age group, in the input data. The solid lines represent the outcome of the modeling process, whereby the selected regression equation is used to smooth the age-related progression of each percentile rank.

If those curves *do not* intersect (as in the example below), the model exhibits adequate monotonicity, and we can proceed to the output phase. If, on the other hand, the percentile curves *do* intersect, it indicates an unexpected change in the raw to norm-score relationship from one age group to the next (i.e., an absence of monotonicity). For example, the plot may reveal that certain raw scores are associated with *higher* norm scores in the next oldest age group. This counter-intuitive outcome is referred to as a norm-score *reversal*.

The plot also shows the proportion of variance in raw scores that is explained by the regression model (here, R2 = 0.9677). This value is often quite large, even in models that lack monotonicity. This is because lacks of monotonicity almost always occur at the tails of the distributions, but the contribution of the tails to the total variance is small. For our purposes, R2 is less important than monotonicity in terms of evaluating the adequacy of the model. Nevertheless, R2 can also be regarded as a general indicator of the quality of the test scale. For example, poor test scales with low reliability generally yield lower R2.

When the plot shows intersecting percentile curves, we re-run the cnorm() modeling function with different parameters (i.e., different values of k and terms), in order to find a model that yields non-intersecting curves (if such a model exists). To examine alternative models, we use another diagnostic tool: plot(model, "series", end = 8). This call of plot() returns a series of percentile graphs for terms = 1 through terms = 8. Often, varying the number of regression terms will result in a plot with non-intersecting curves. Please not that after visually selecting an adequate model from the series of plots, the cnorm() modeling function has to be rerun with the according number of terms.

In general, parsimony applies: it's best to select a model with fewer terms, all else being equal. Increasing the number of terms increases the risk of *overfitting* the model (that is, modeling error in addition to the developmental gradient of the latent ability). In the percentile plot, an overfitted model is visualized in curves that twist and turn sharply around the raw score data points (colored circles). In contrast, a properly fitted model will yield smoothly increasing parallel colored lines.

Another diagnostic aid is checkConsistency(model), which offers a programmatic check on monotonicity. When checkConsistency() returns FALSE, it indicates that the norming model is adequately specified, with no violations of monotonicity. If checkConsistency() finds problems with the model, it returns the ages at which violations of monotonicity are identified. These findings can be checked against the model plot. In addition, we can generate raw-to-norm score lookup tables based on a problematic model, to see where score reversals occur.

A final note about model specification is that with some data sets, it may not be possible to find a norming model that is completely free of violations of monotonicity. Finalizing the norms then becomes a process of selecting the least-flawed model, and then manually correcting score reversals and other anomalies in the resulting raw-to-norm-score lookup tables. Moreover, it can also be necessary to limit the range of the raw-to-norm-score lookup tables to the areas with reasonable outcome.