Reject inference with nested conditional models based on joint risk and fraud scores

Ross Gayler

28 August, 2013

veda
applied intelligence

Problem setting

- Application scoring
- “Competing” outcomes: Risk Bad (B), Fraud-like Bad (N)
  - Not convinced “competing” is the right conception because there are not necessarily two independent processes
  - Fraud-like outcome may have been the intent at time of application (but maybe not)
- Two application scorecards developed
- Scores to be used jointly in decision-making
  - Matrix approach
- Needed to populate joint score matrix with historical outcomes to guide strategy setting
- Some historical outcomes are necessarily censored (e.g. B or N can only be observed if the application was accepted by lender and taken up by applicant)

Why use a joint score matrix?

- Outcome probabilities may be interactive functions of the two scores
  - That seemed very likely given the outcomes are mutually exclusive
  - Exploiting that interaction yields a more predictive system
  - If no interaction, nothing lost by considering interaction
  - Can always drop back to using the scores separately

- Decision allocation is naturally a function of both scores
  - Joint consideration is the most flexible
  - Contains separate consideration strategies as a special case

Application scorecard design

- Design needed to recognise joint use in matrix
- Known that B and N look relatively similar
  - G vs B and G vs N scores would have been highly correlated
  - Joint score matrix would be populated on the diagonal only
- Want the two scores to be relatively independent
  - Joint score matrix reasonably well filled

Rscore = G vs B

- Conceptually simple
- We would have one useable model if Nscore didn't pan out

Nscore = B vs N

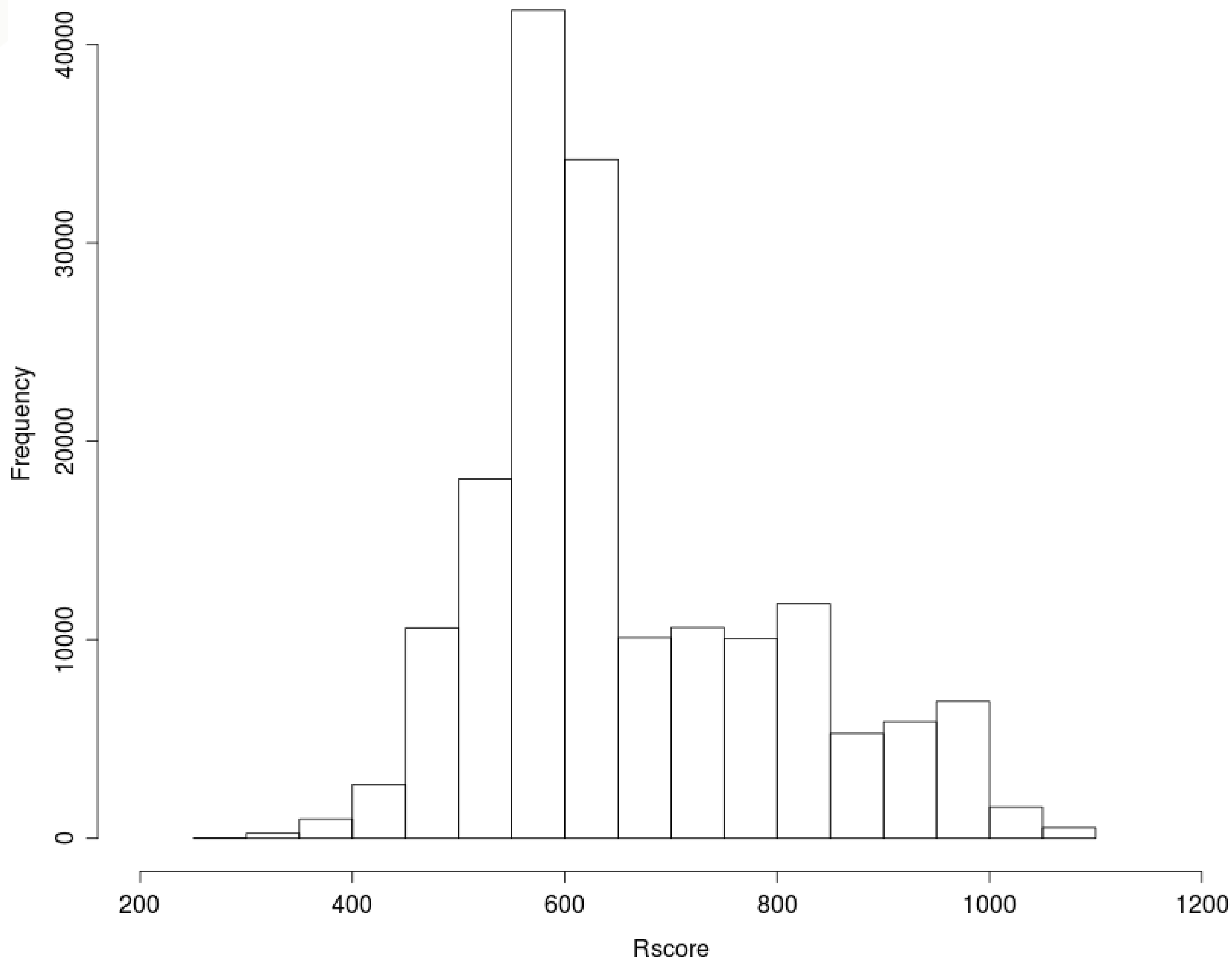
- Concentrate on discriminating between the two (similar) outcomes
- Cases concentrated at the B+N end of the space
- Might not be so good at the G end (but that is less important)

Distributions of scores

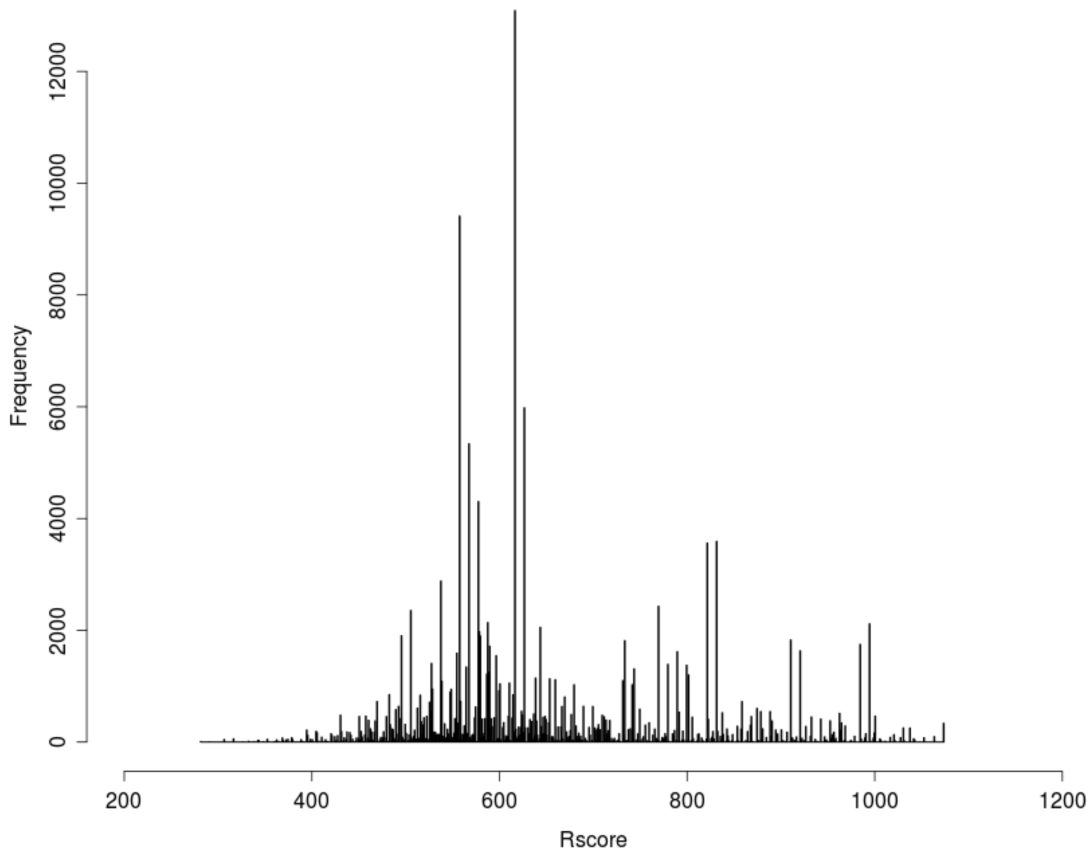
- Scorecards built using typical “classed” logistic regression methodology
- Each scorecard reasonable as a scorecard to use separately
- Rscore more predictive of G vs B than Nscore of B vs N
  - That’s more a statement about the relative homgeneity of the populations than about the scorecards

- Look at the separate and joint distributions of the scores

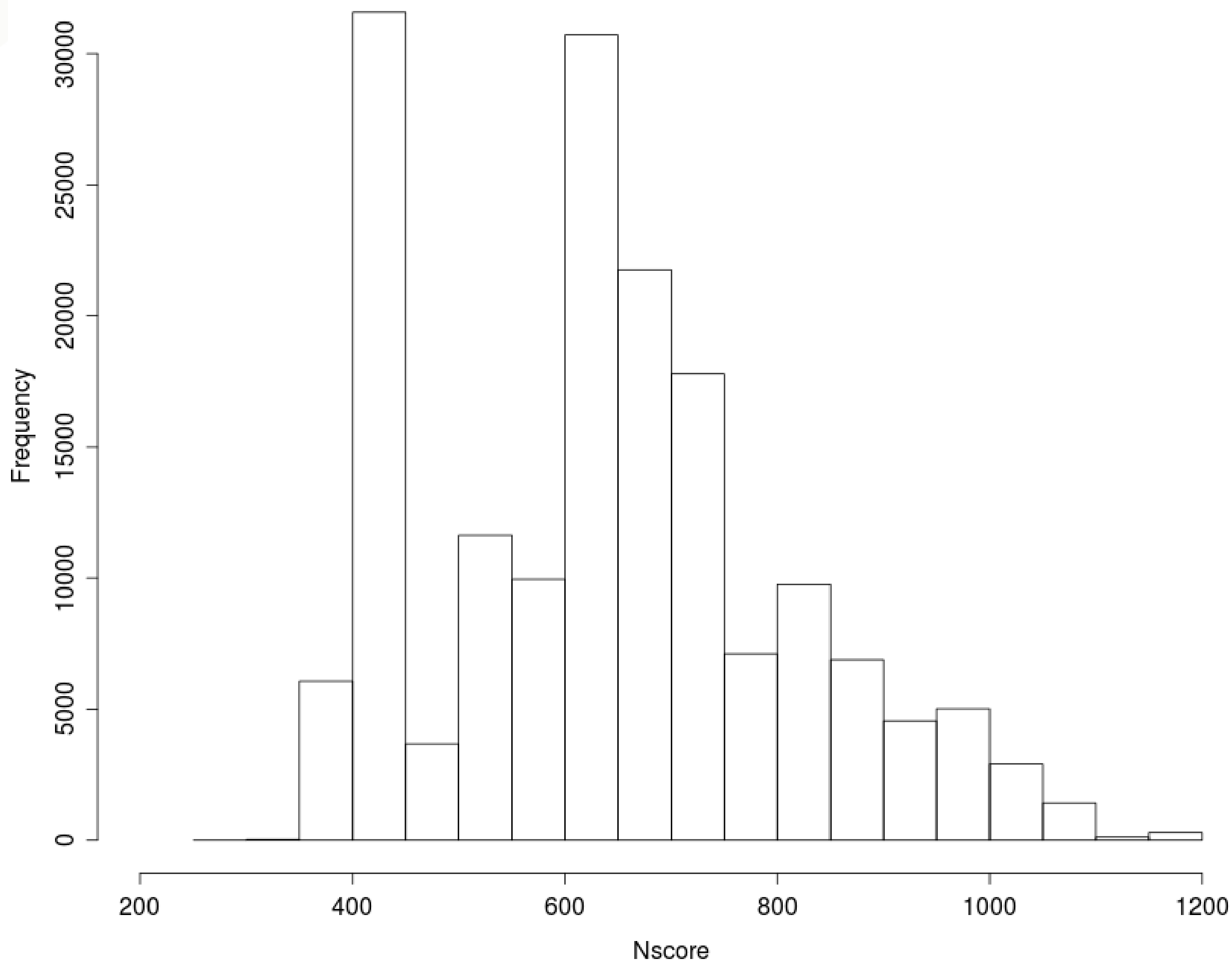
Distribution of Rscore (low res.)



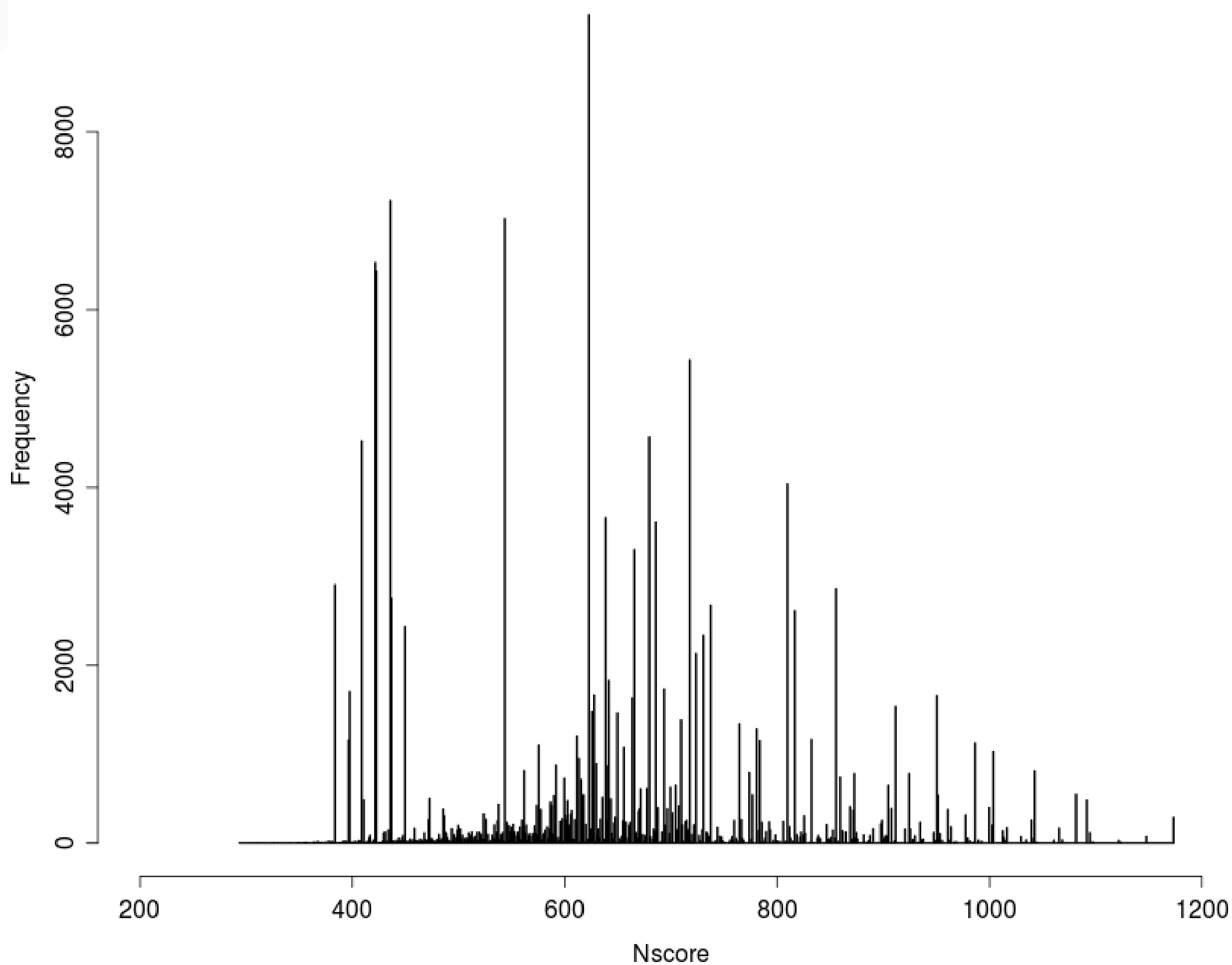
Distribution of Rscore (high res.)



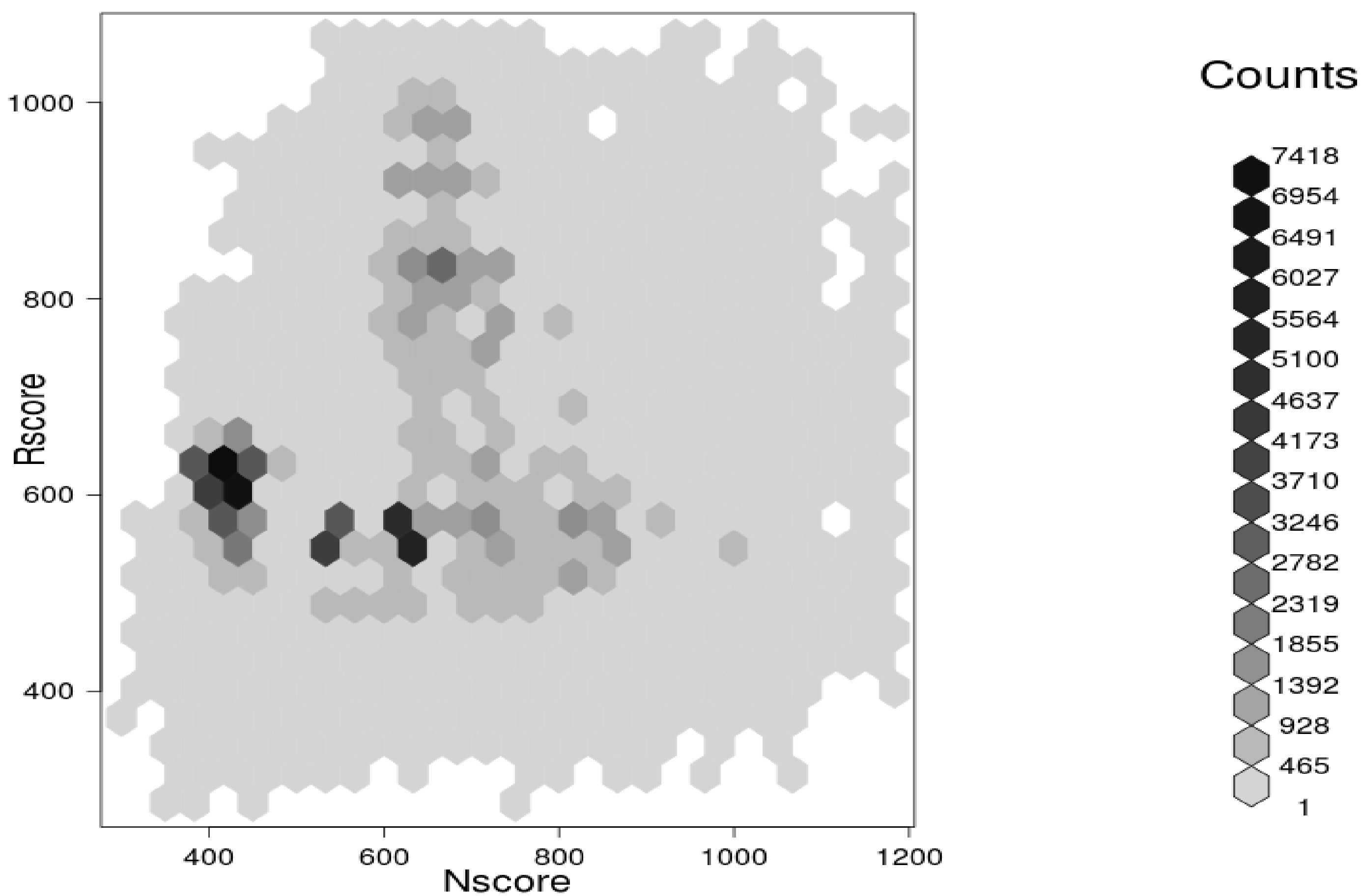
Distribution of Nscore (low res.)



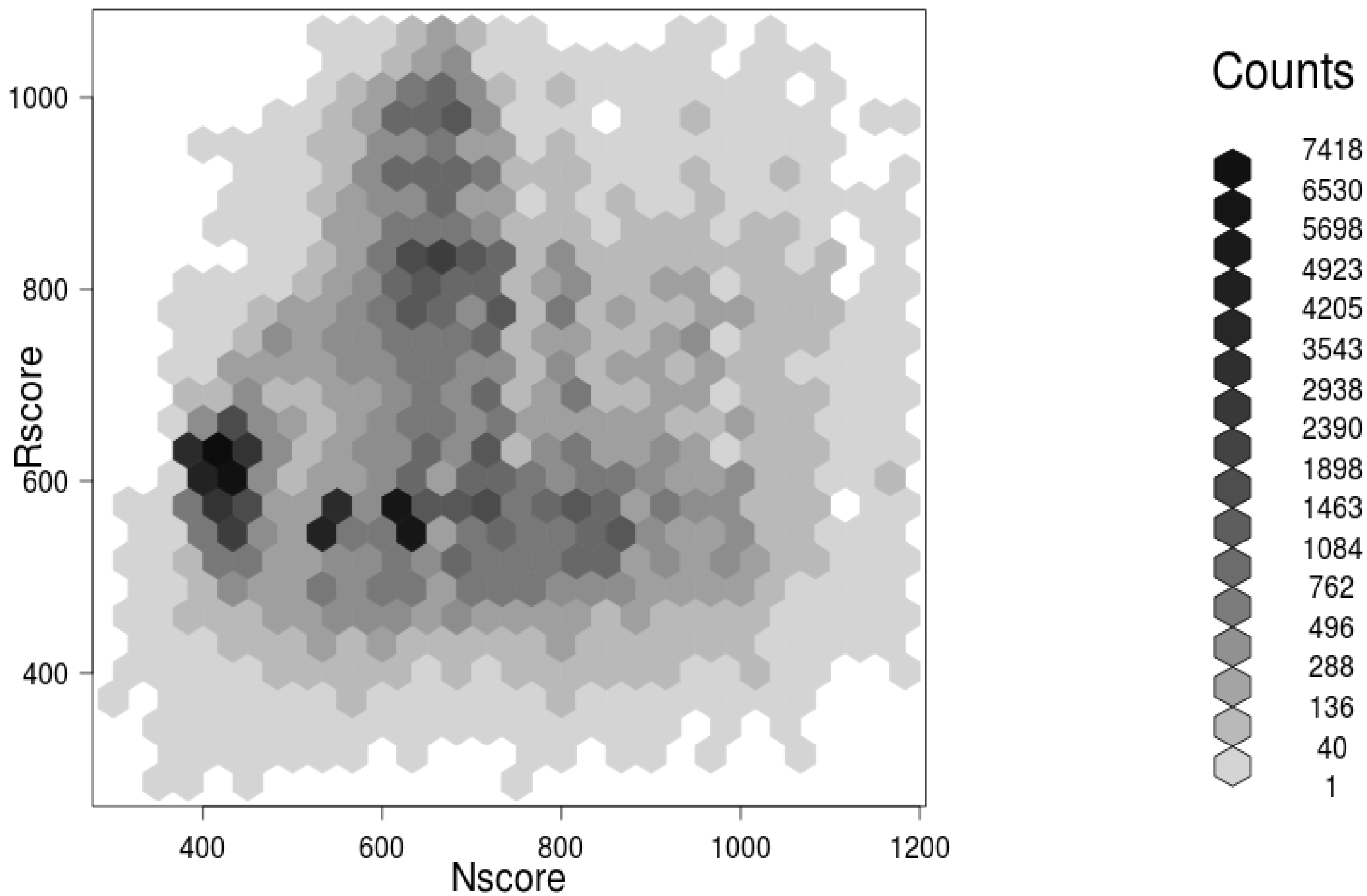
Distribution of Nscore (high res.)



Joint distribution of Nscore & Rscore (linear breaks)



Joint distribution of Nscore & Rscore (square root breaks)



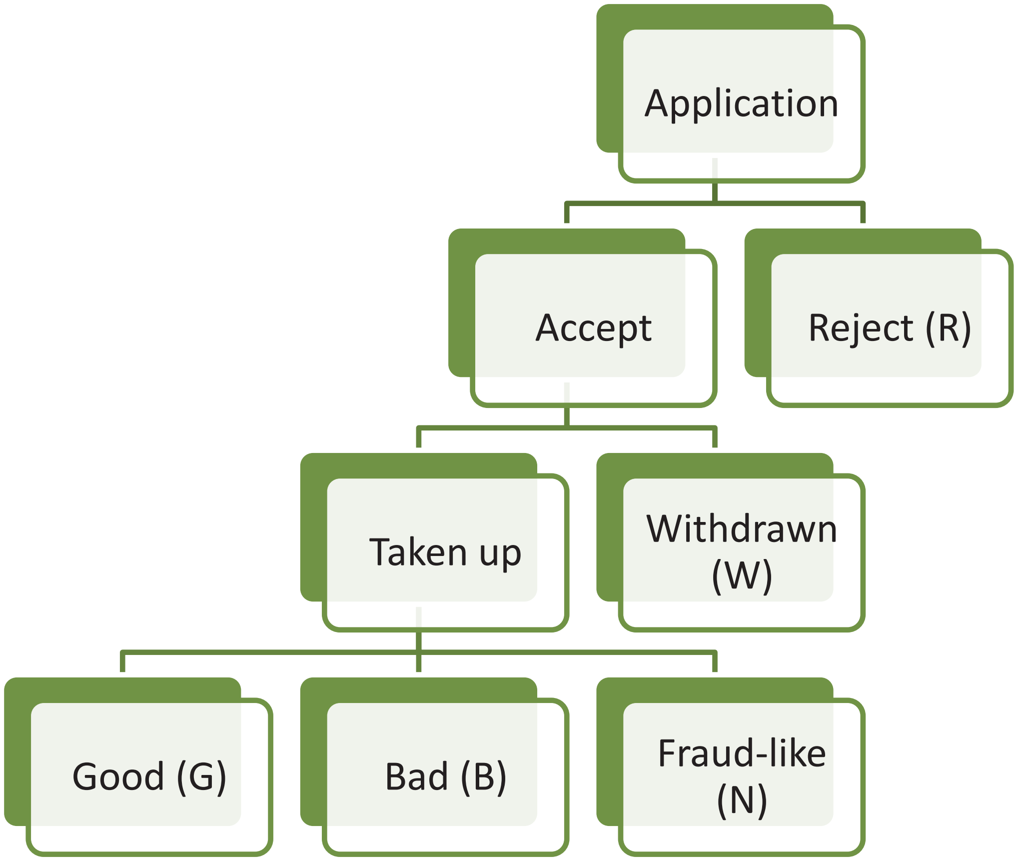
Joint distribution

- Support over most of the rectangular joint space
  - Matrix strategy approach should work well
  - Estimating Pr(outcome) ~ s(Nscore,Rscore)
    - density(Nscore,Rscore) not relevant to estimation
    - Is relevant to strategy setting (lumps)

- Joint distribution is more sparse than marginal distributions
  - Curse of dimensionality at work
- Future aim for smoother marginal distributions
  - Continuous predictors in scores

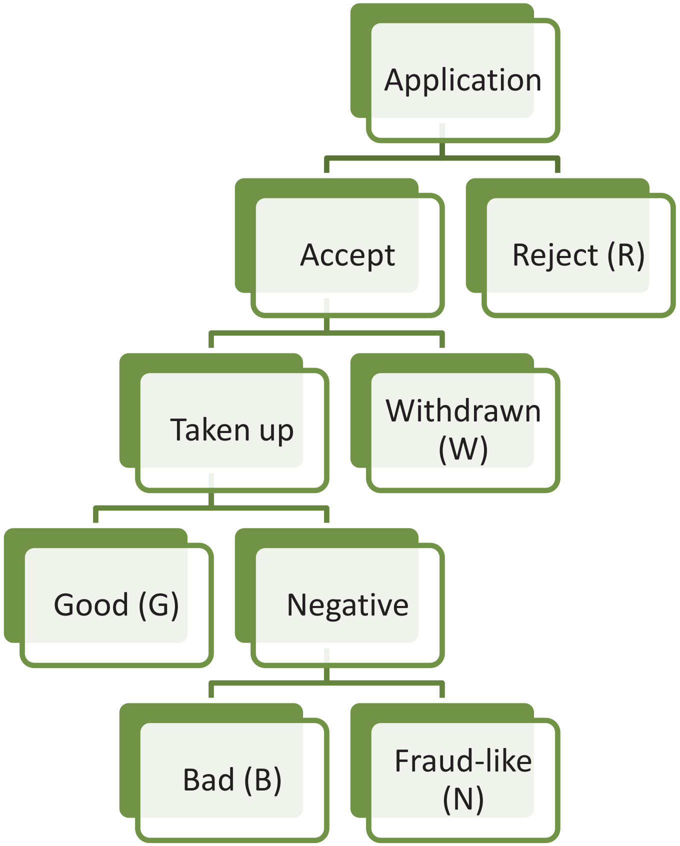
Range of outcomes

- There are more outcomes than I mentioned initially



Reconceptualised range of outcomes

- Modify the structure to reflect the subsequent (binary outcome) analyses
- Sensitivity of the results to this reconceptualisation?

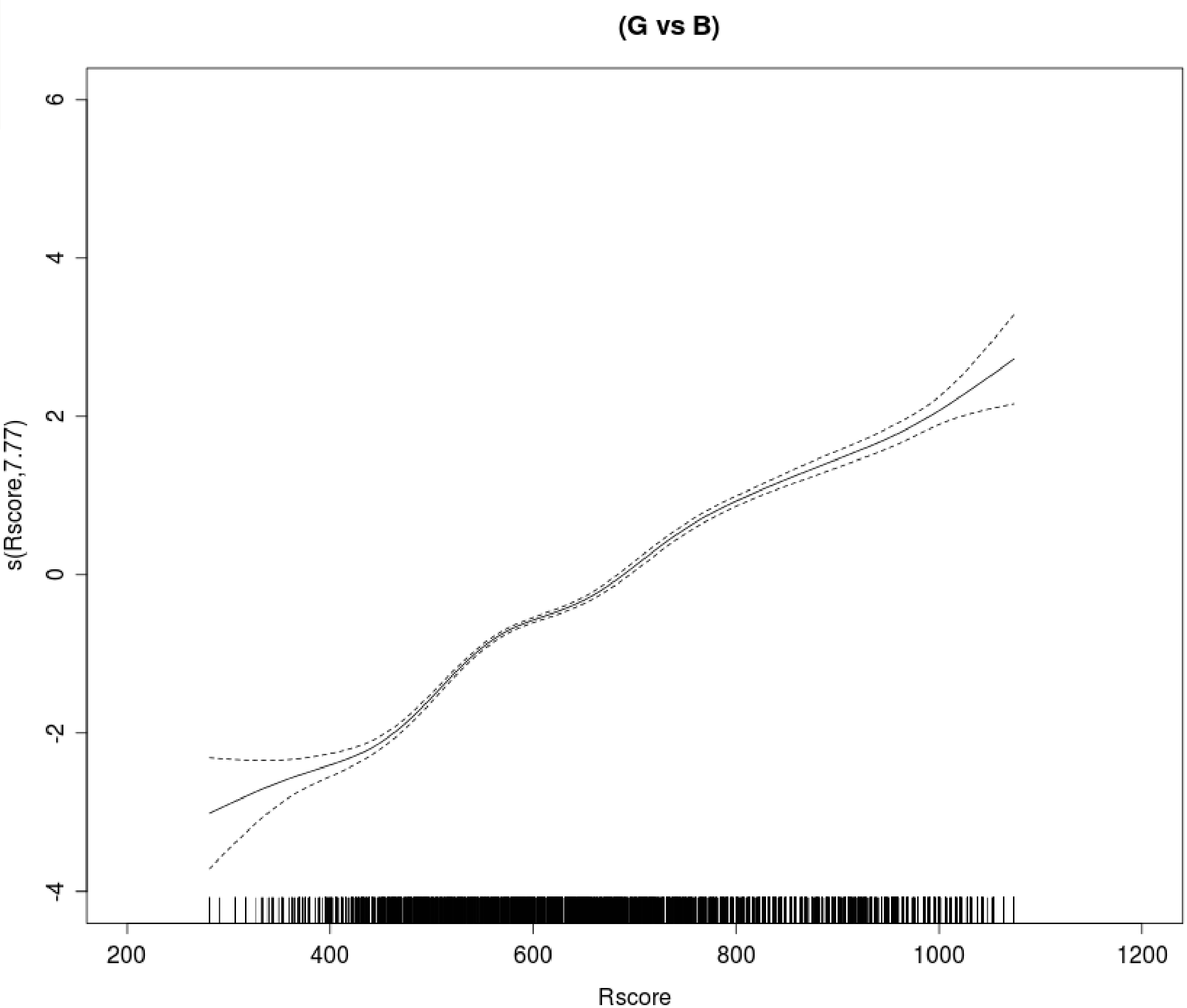


Separate scores

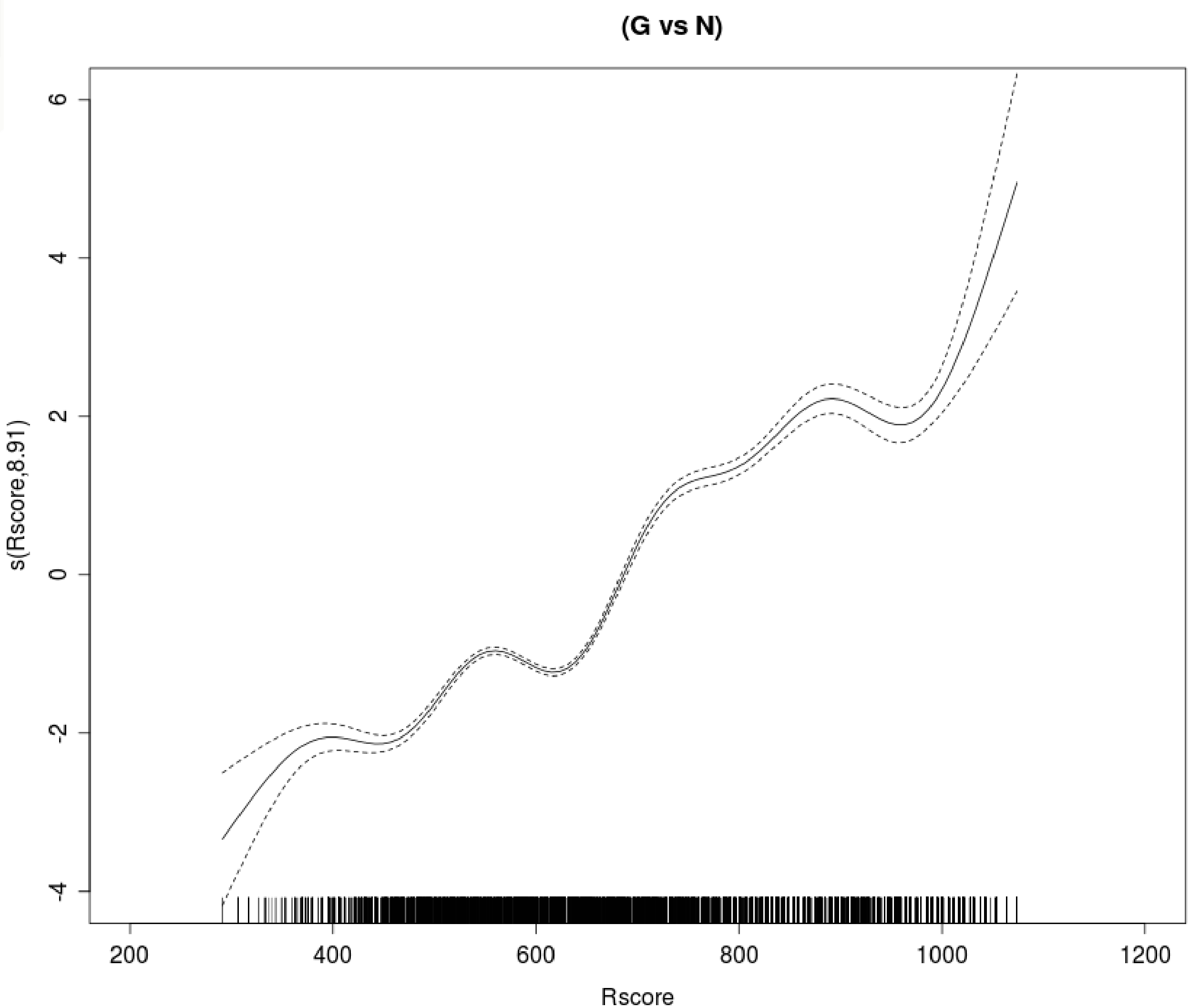
- How well do each of the scorecards work at predicting the various outcomes?
- Look at calibration curves

Rscore calibration

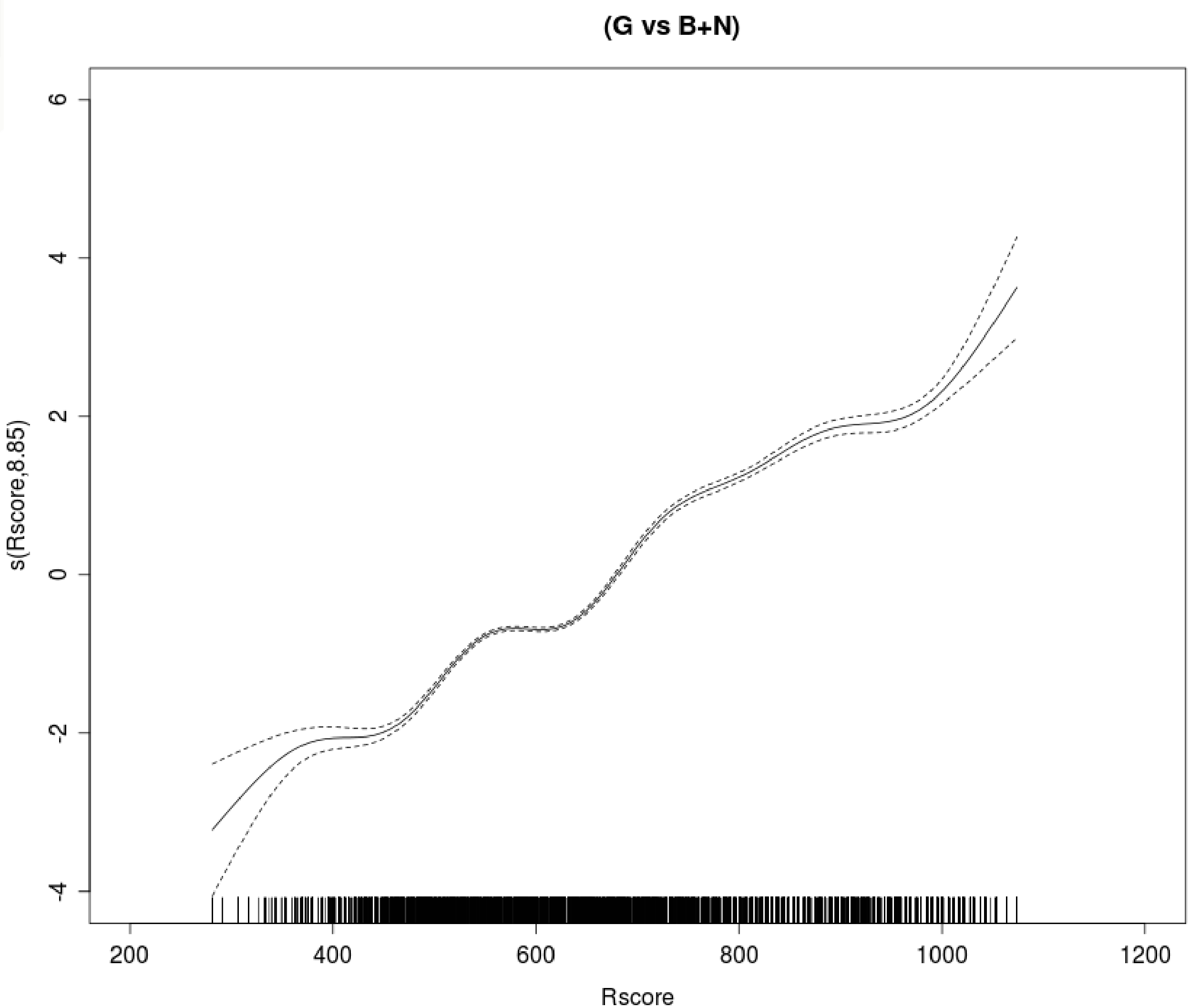
G vs B ~ s(Rscore)



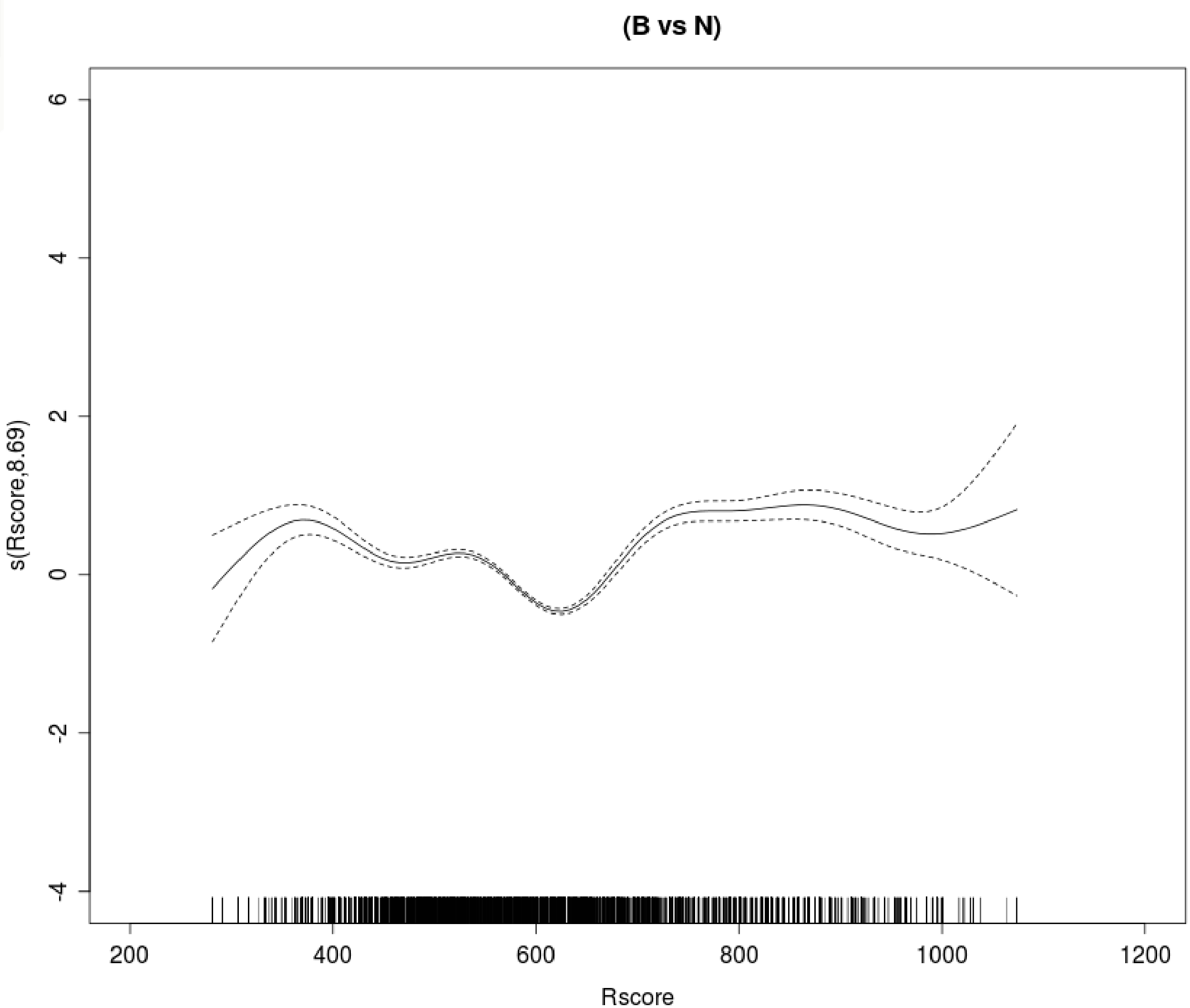
G vs N ~ s(Rscore)



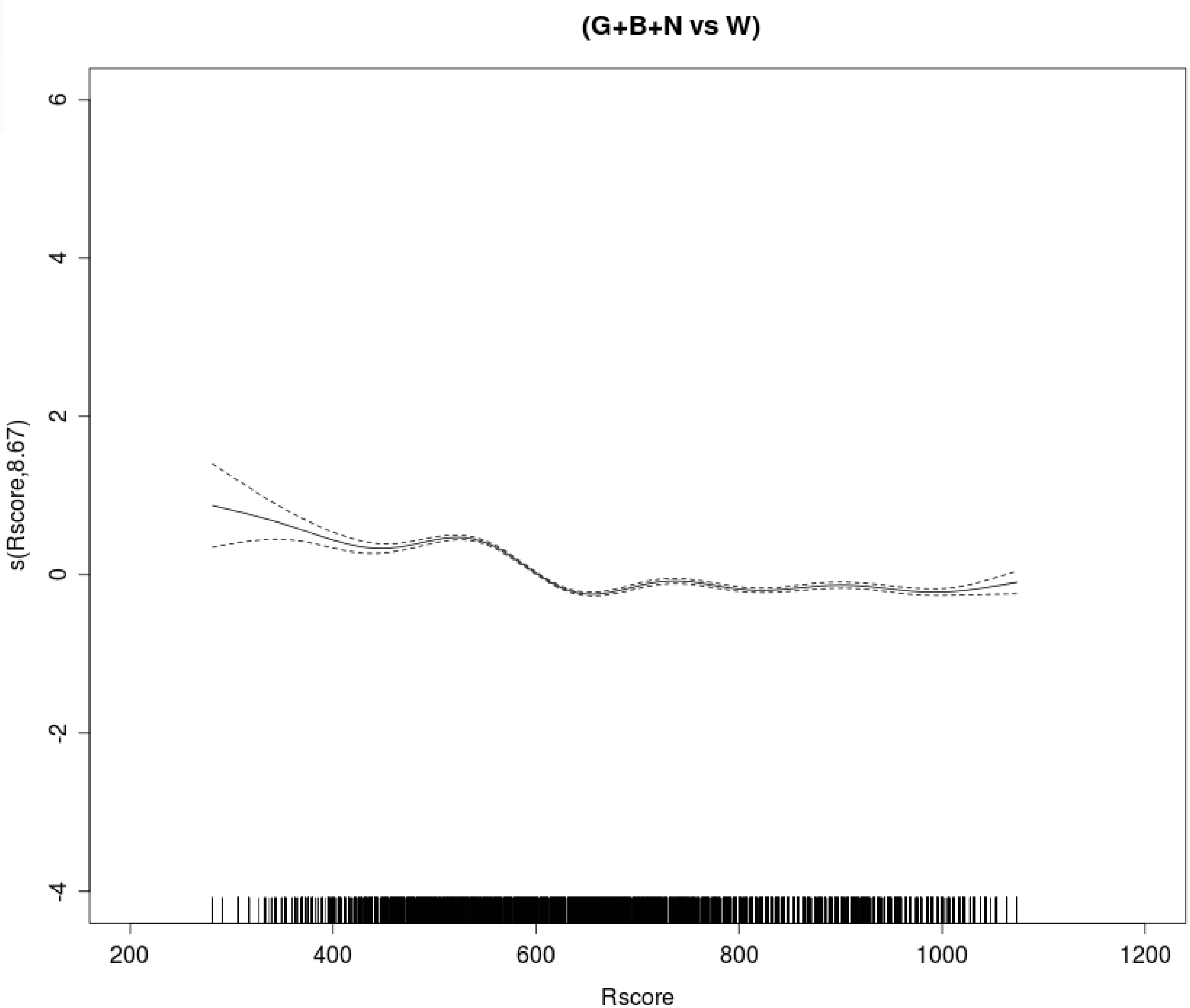
G vs B+N ~ s(Rscore)



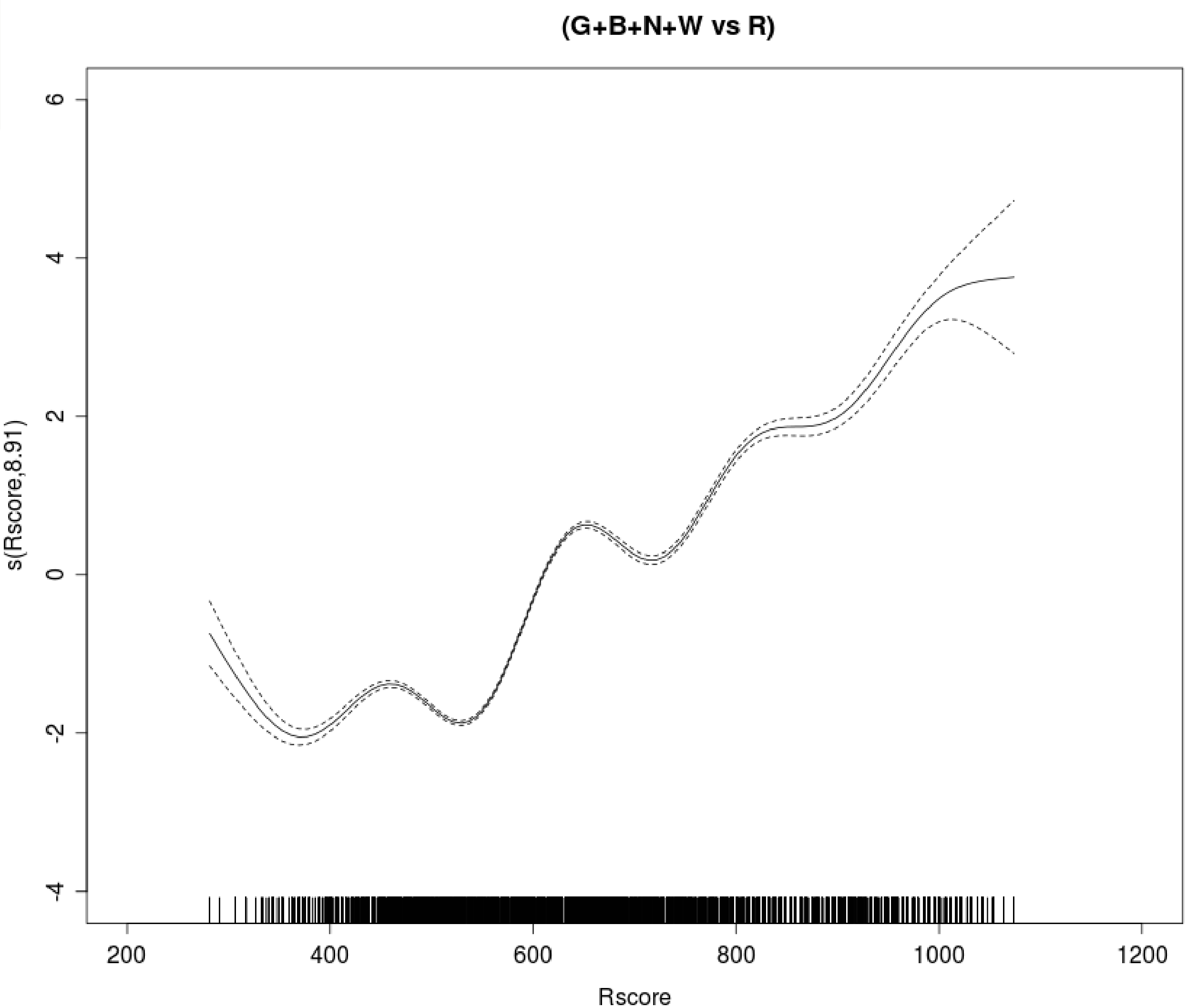
B vs N ~ s(Rscore)



G+B+N vs W ~ s(Rscore)

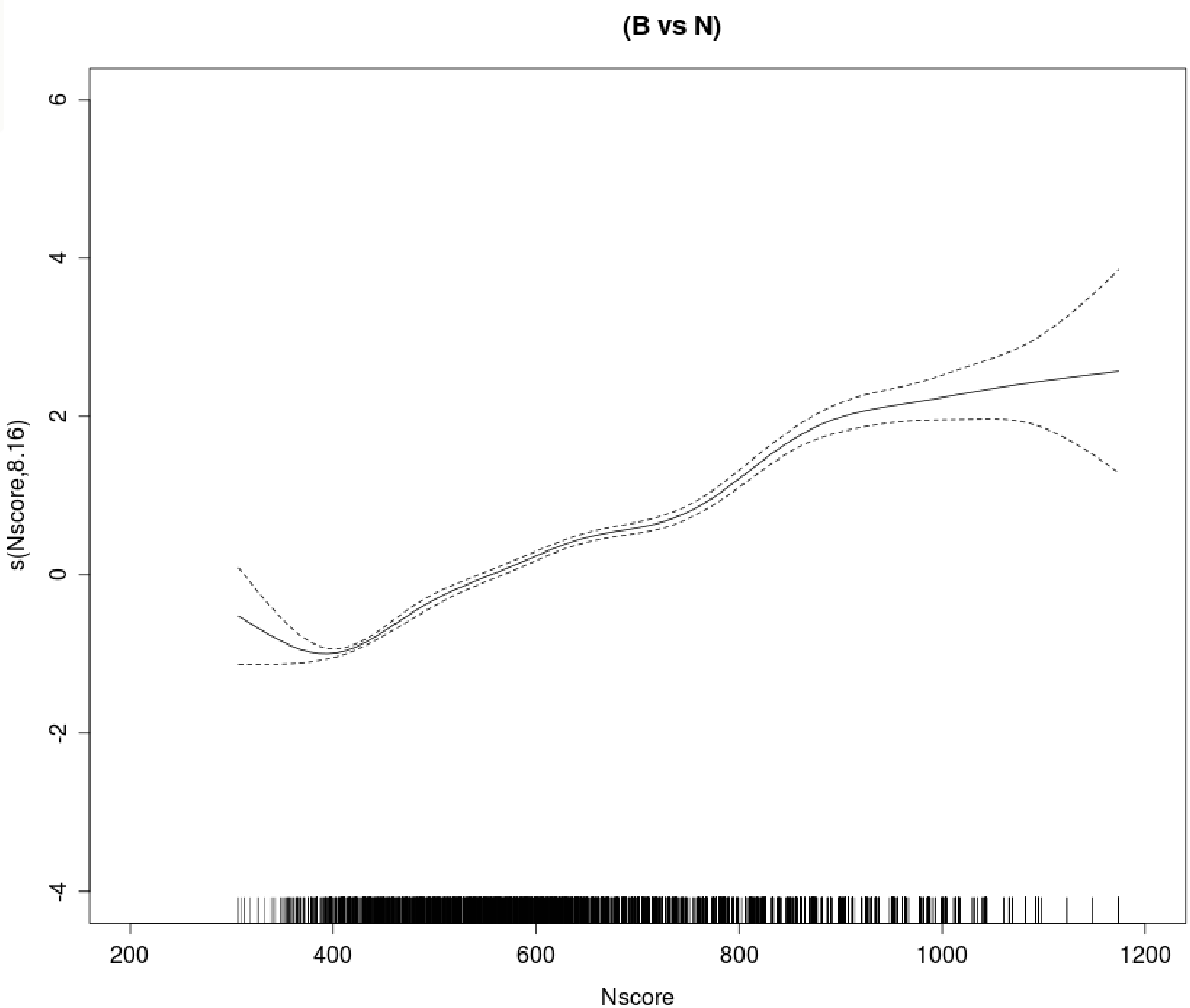


G+B+N+W vs R ~ s(Rscore)

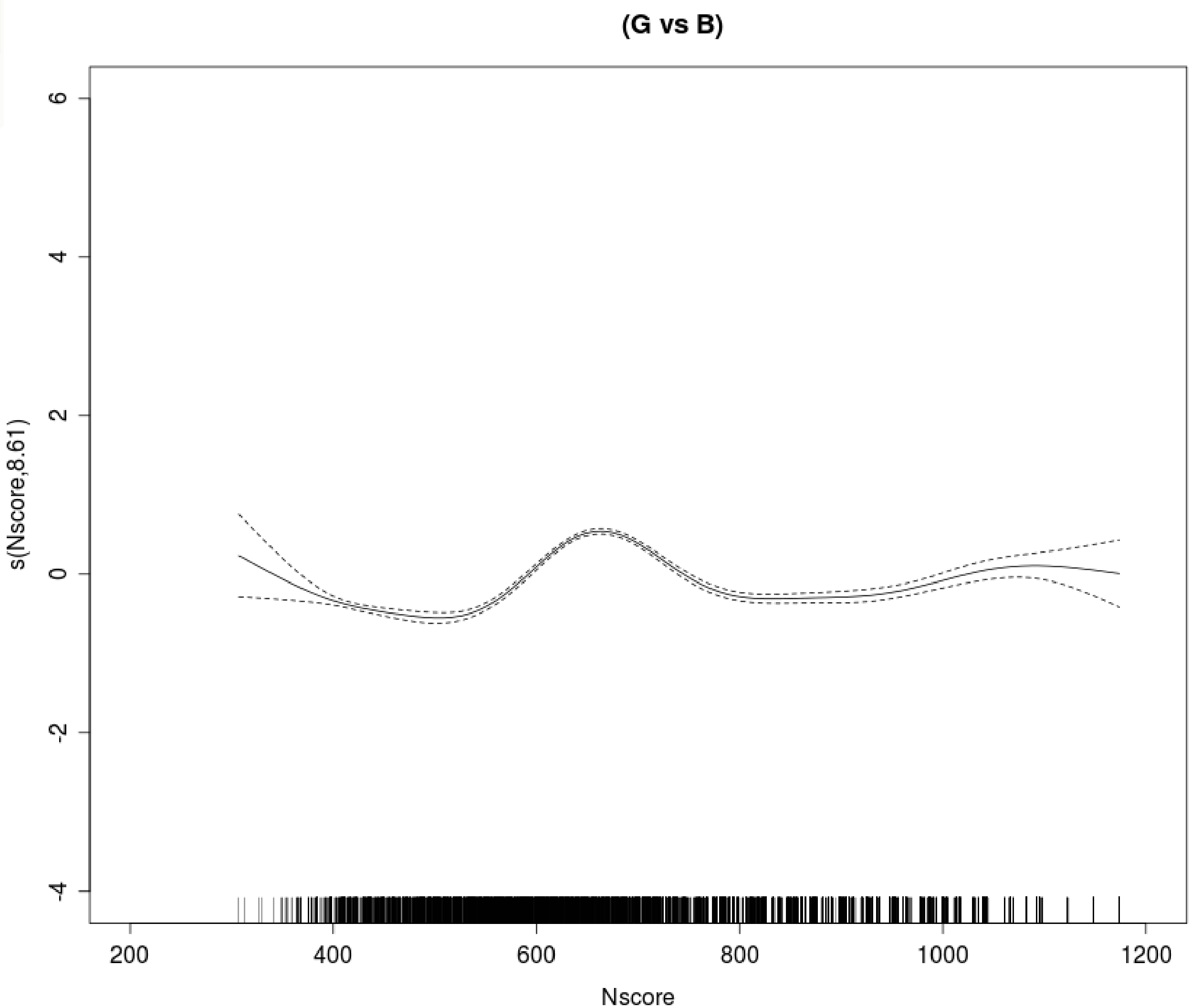


Nscore calibration

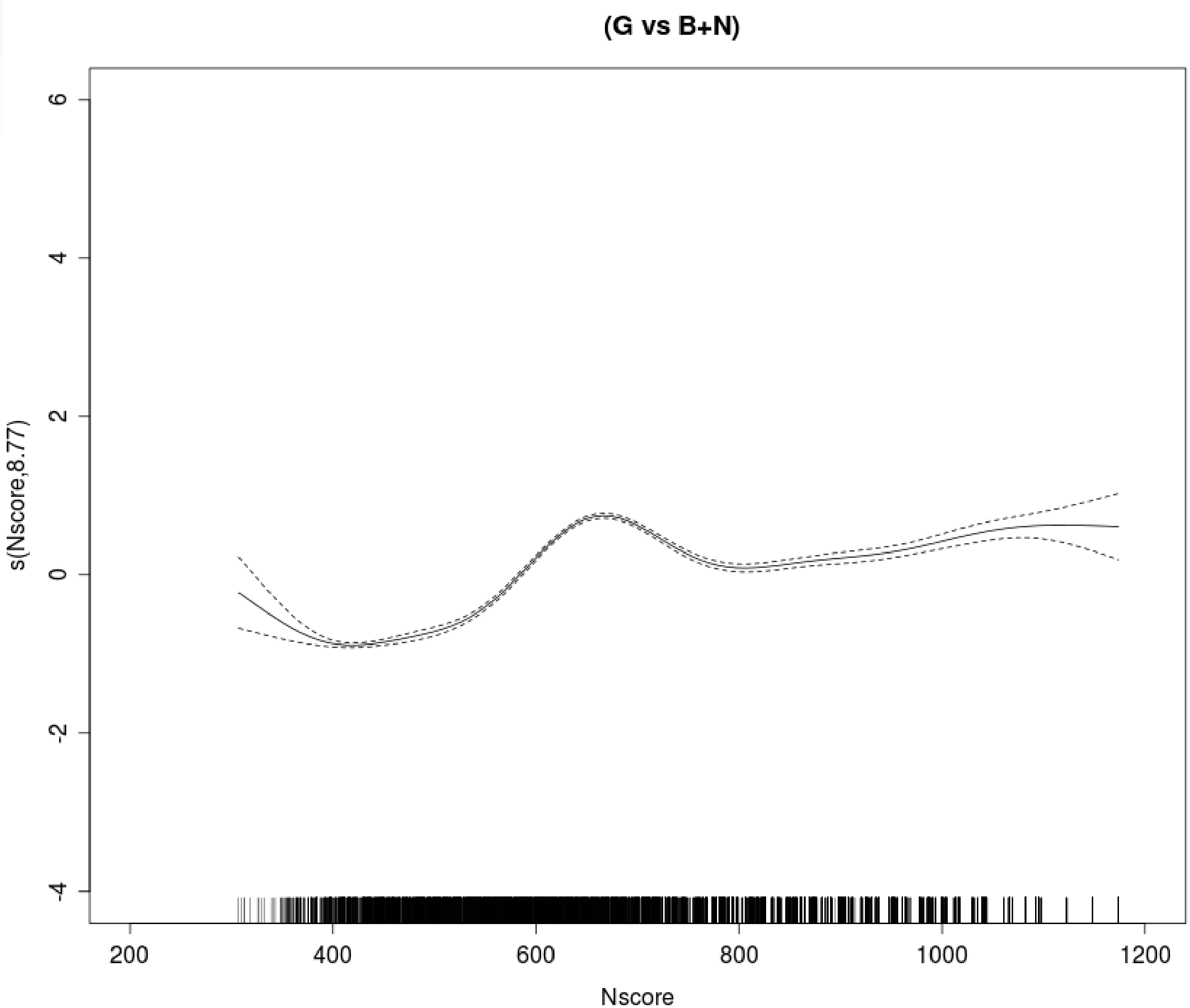
B vs N ~ s(Nscore)



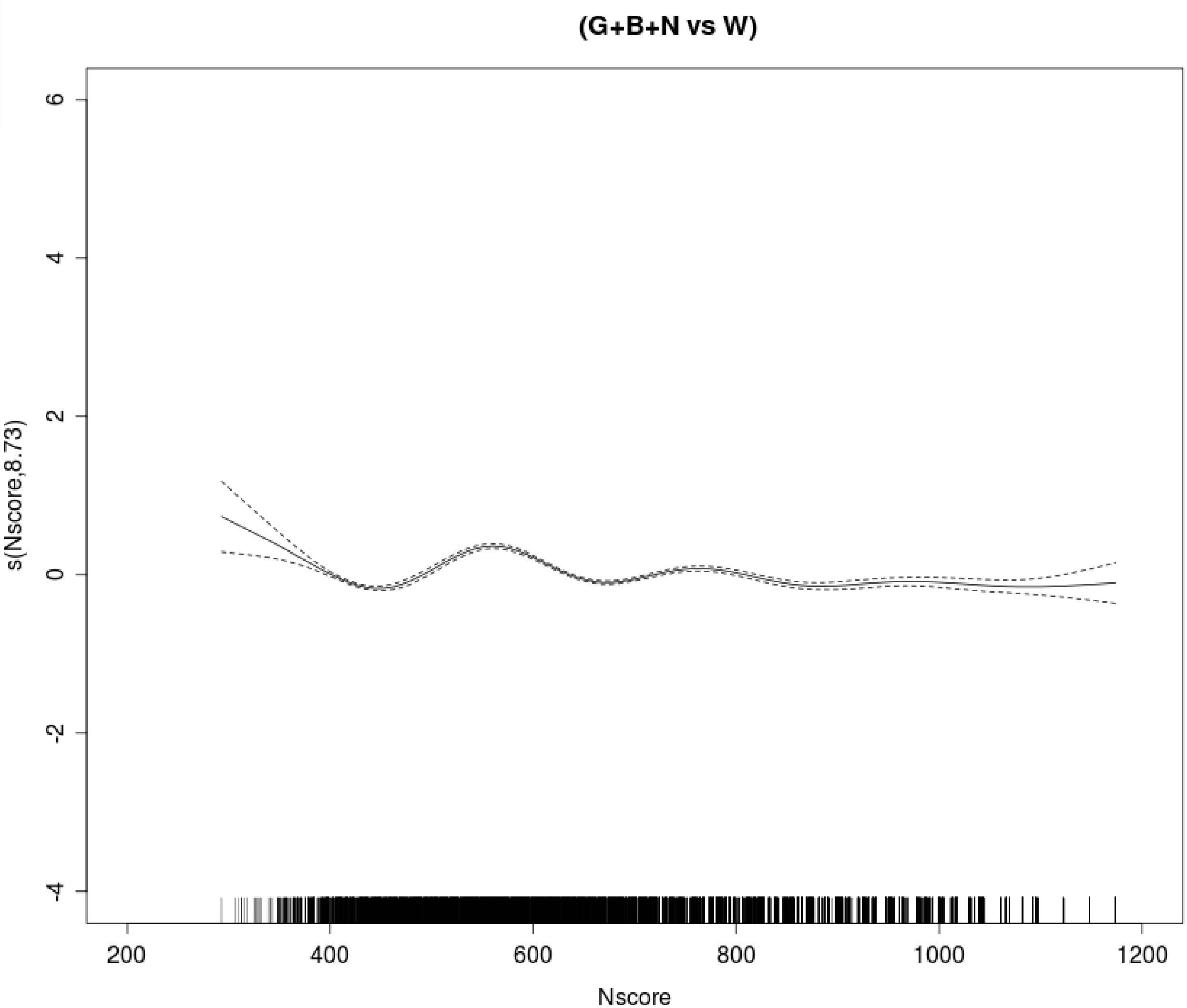
G vs B ~ s(Nscore)



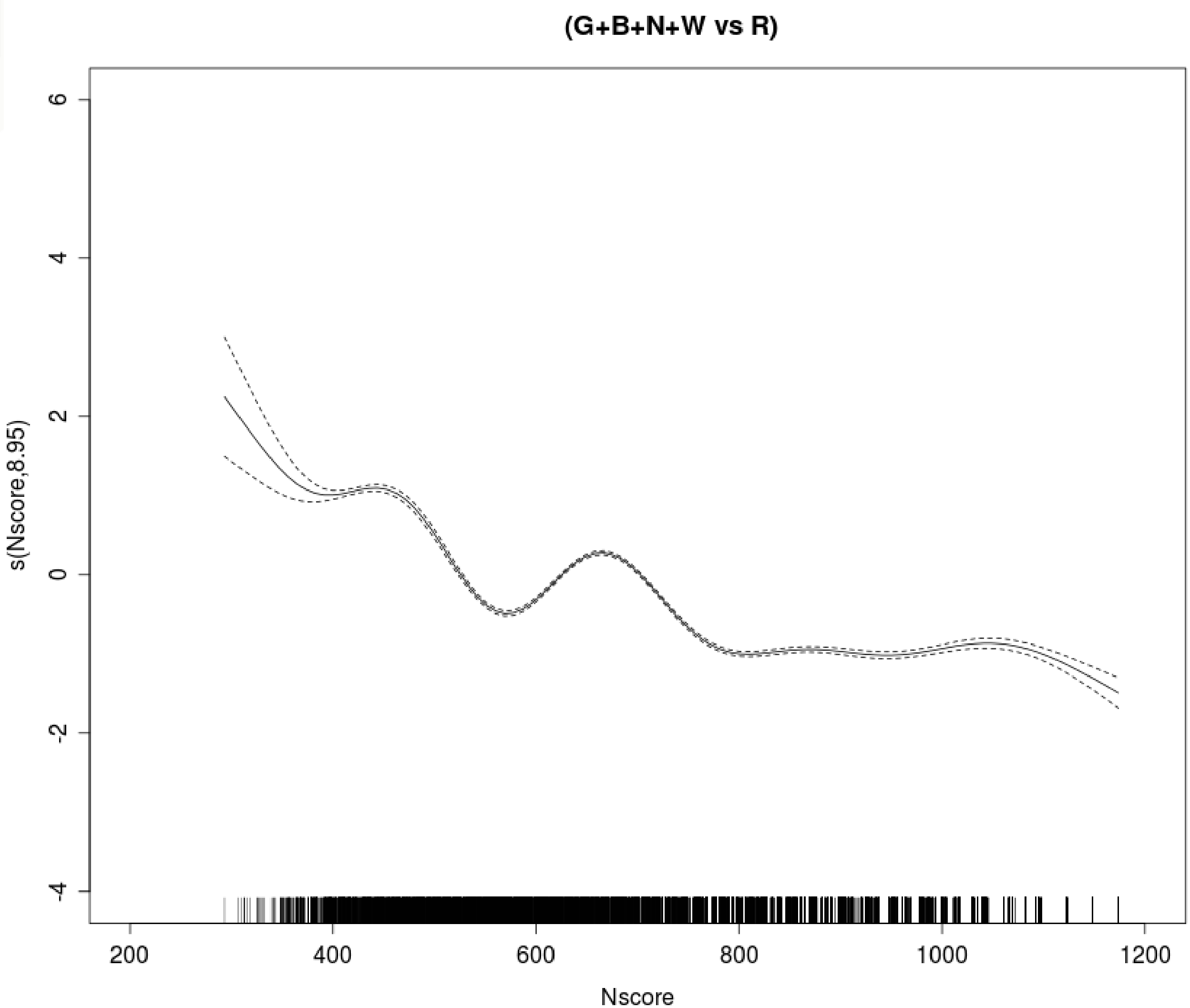
G vs B+N ~ s(Nscore)



G+B+N vs W ~ s(Nscore)

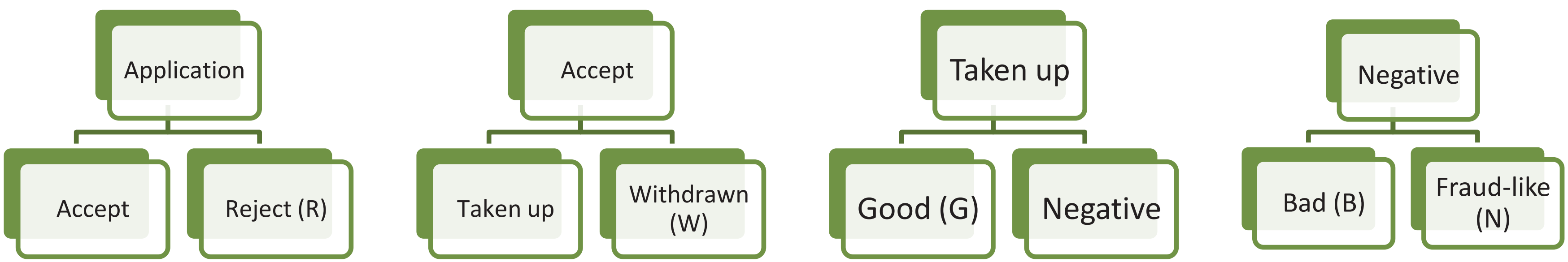


G+B+N+W vs R ~ s(Nscore)

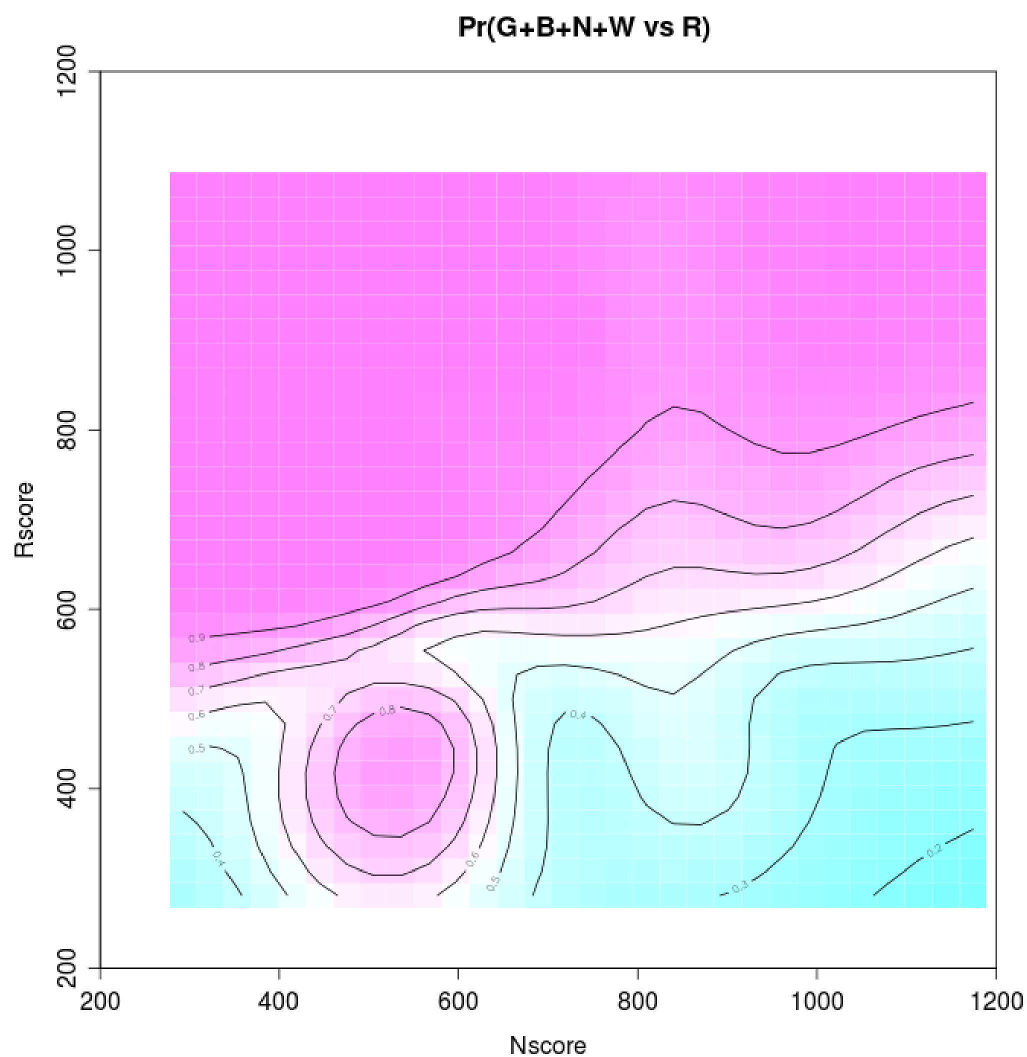


Nested conditional joint-score models

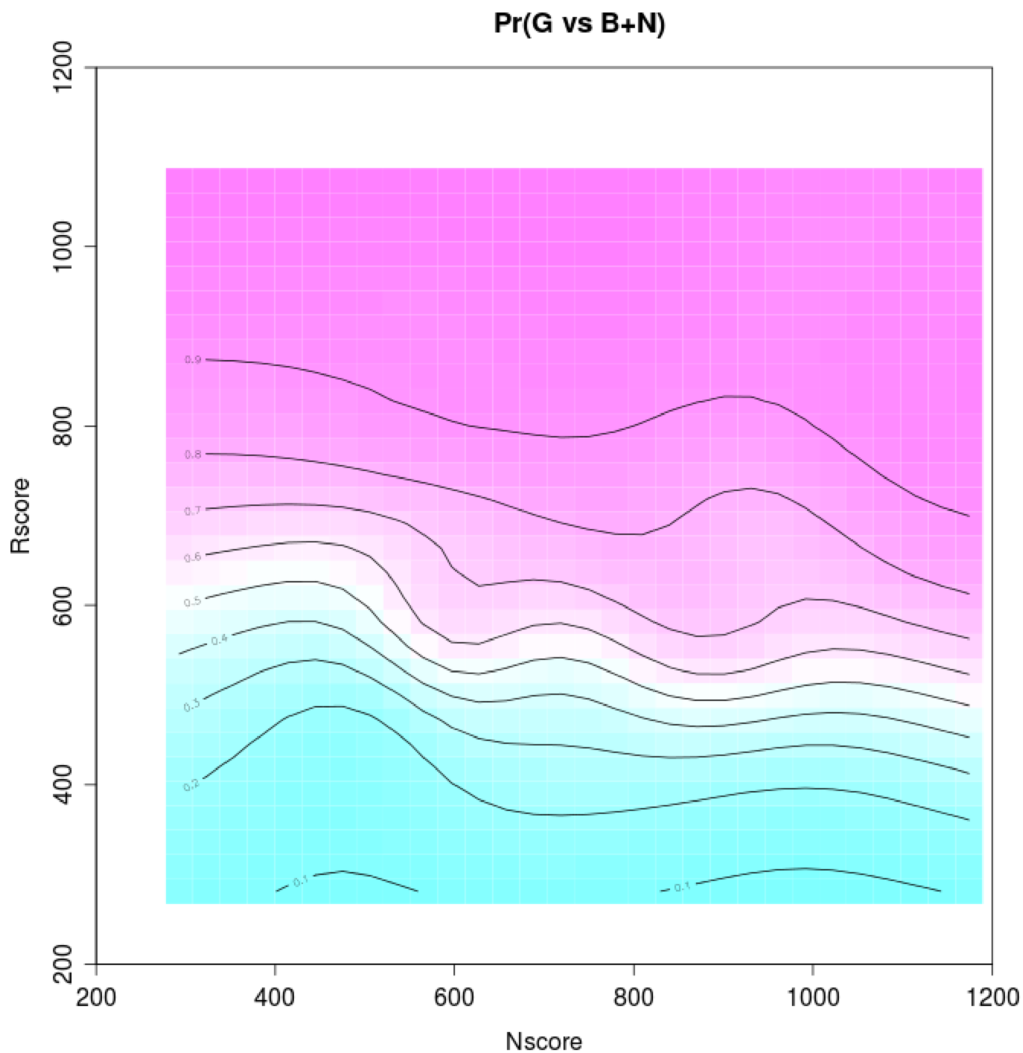
- Build separate models on the population of each parent node with the immediate children as outcomes
- First model – just for interest
- Latter 3 models – for use later



G+B+N+W vs R ~ s(Nscore,Rscore)



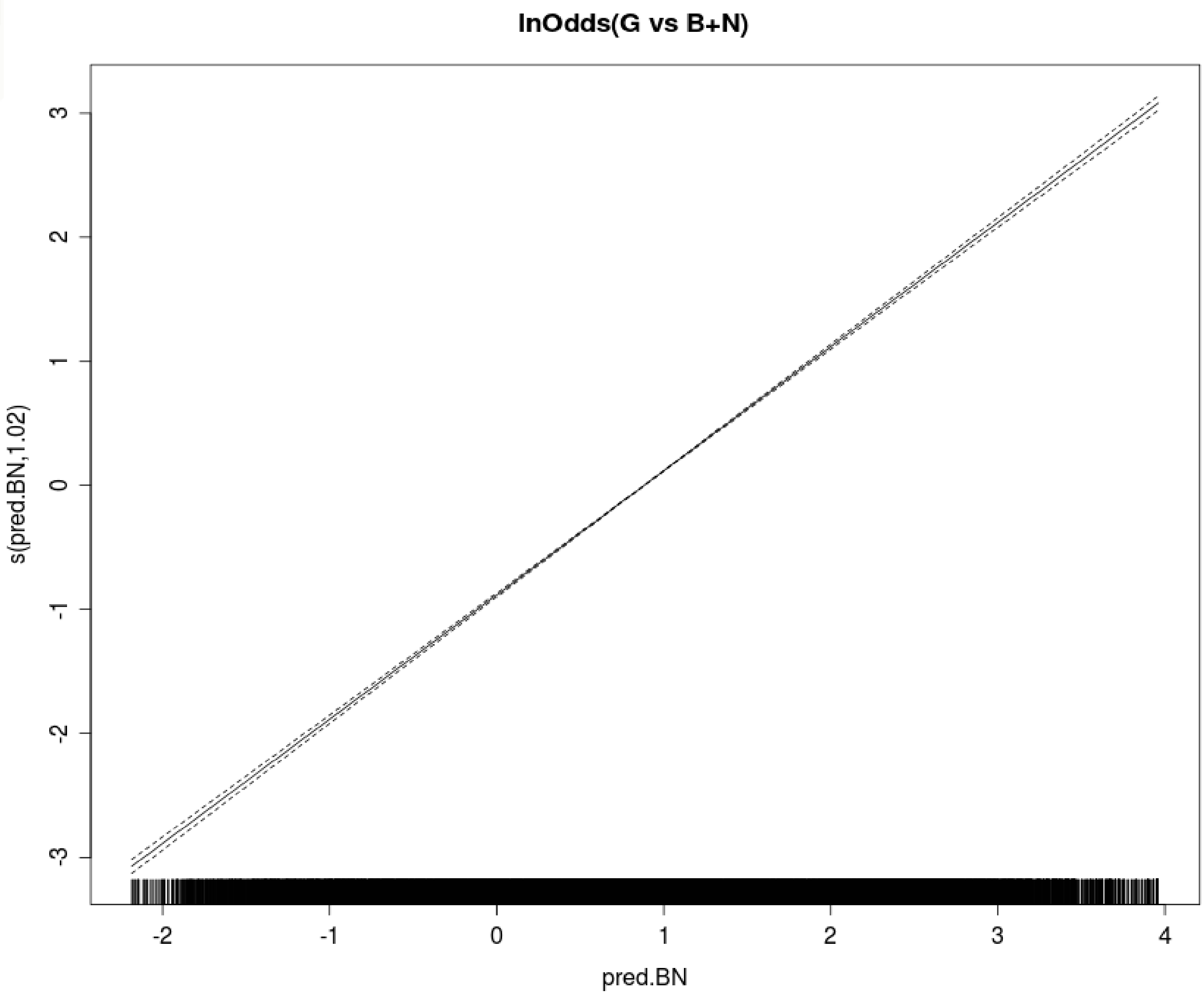
G vs B+N ~ s(Nscore,Rscore)



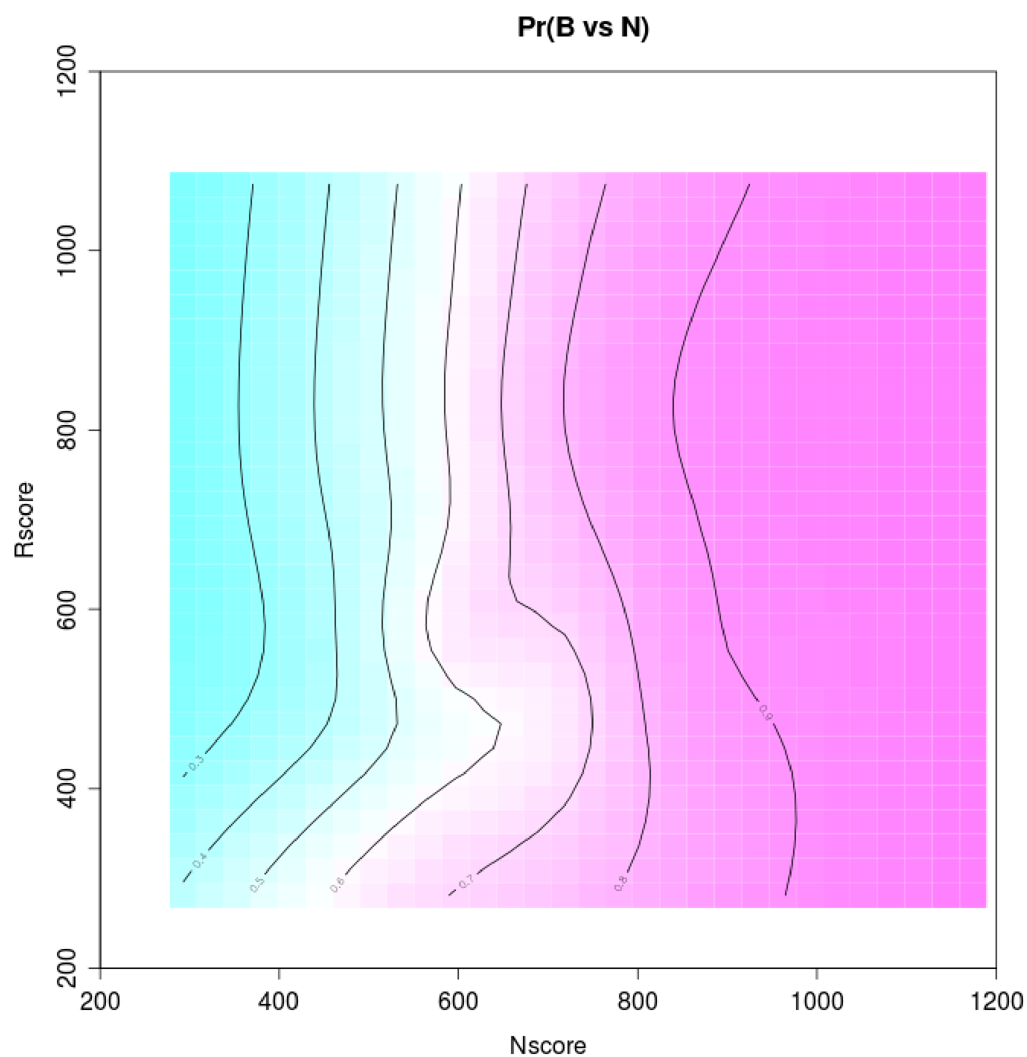
Check the calibration

- Take the predicted log-odds from the model and use that as the predictor of the actual outcome

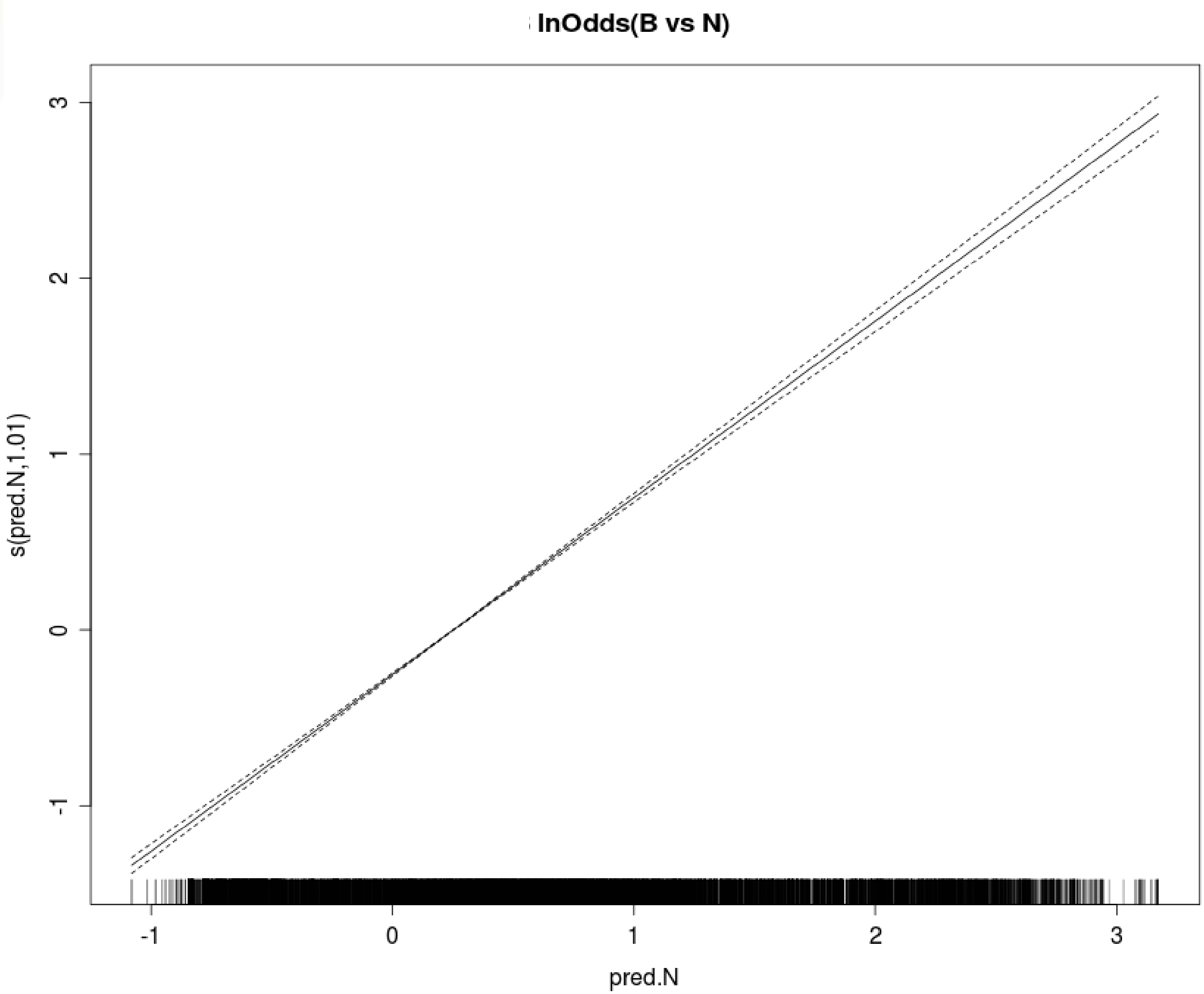
G vs B+N ~ s(pred(G vs B+N))



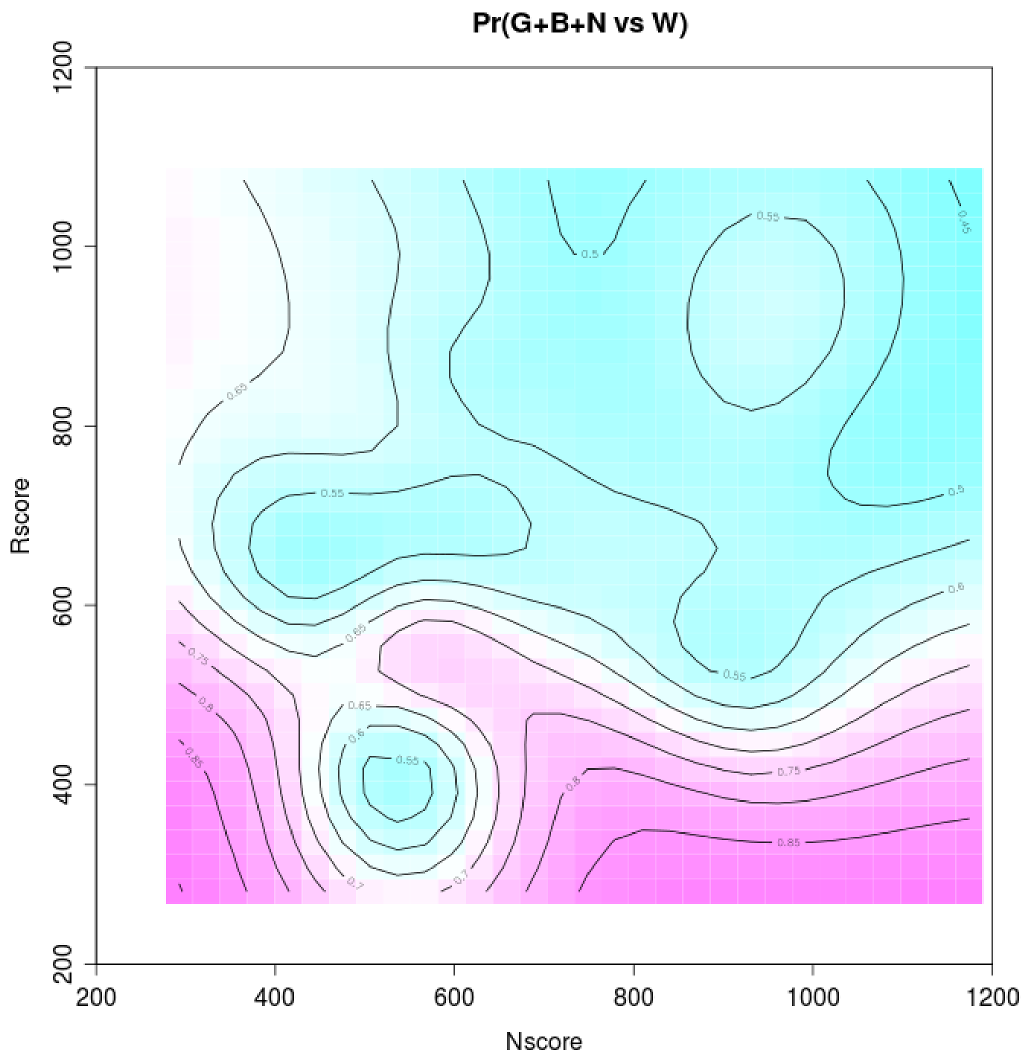
B vs N ~ s(Nscore,Rscore)



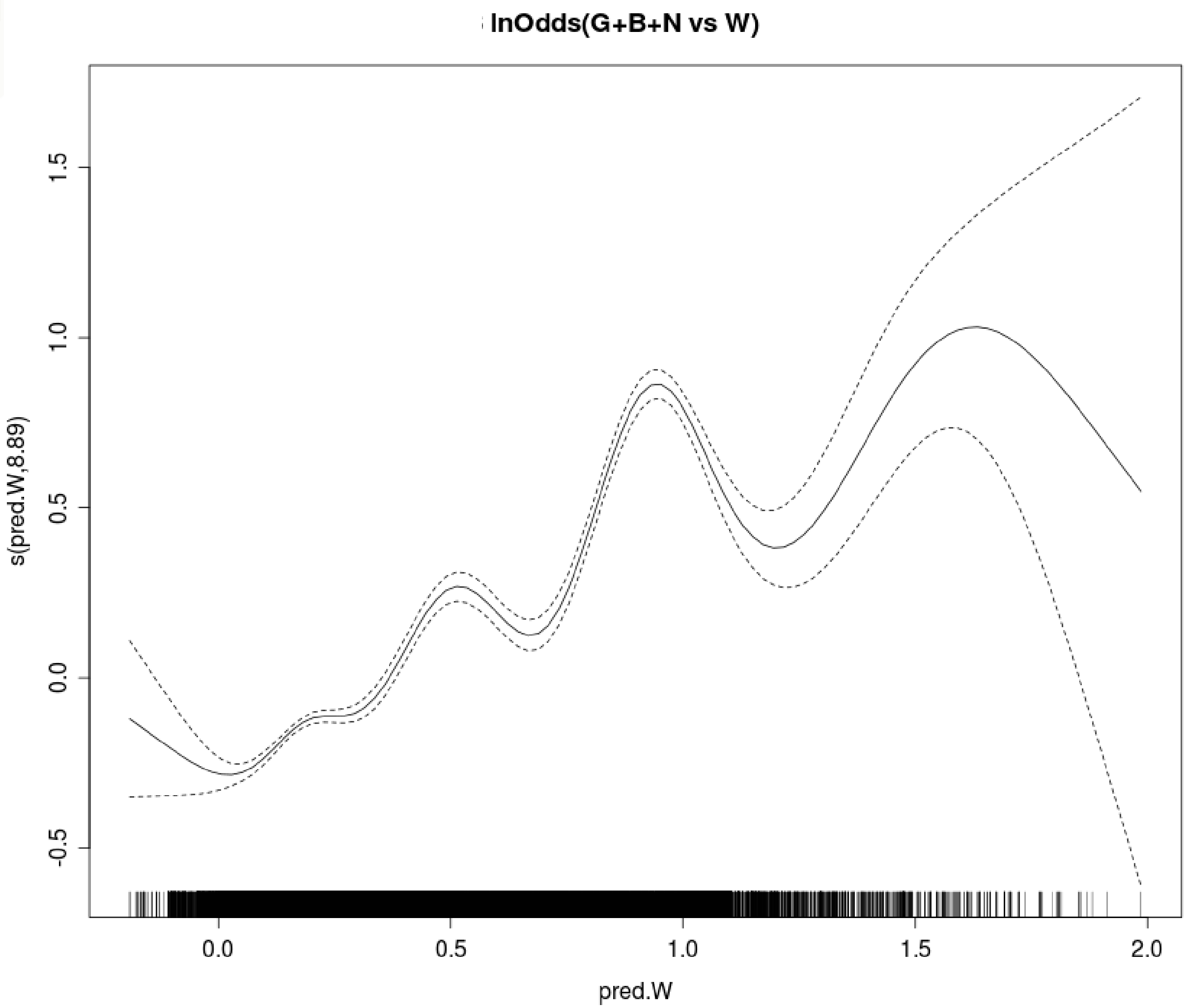
B vs N ~ s(pred(B vs N))



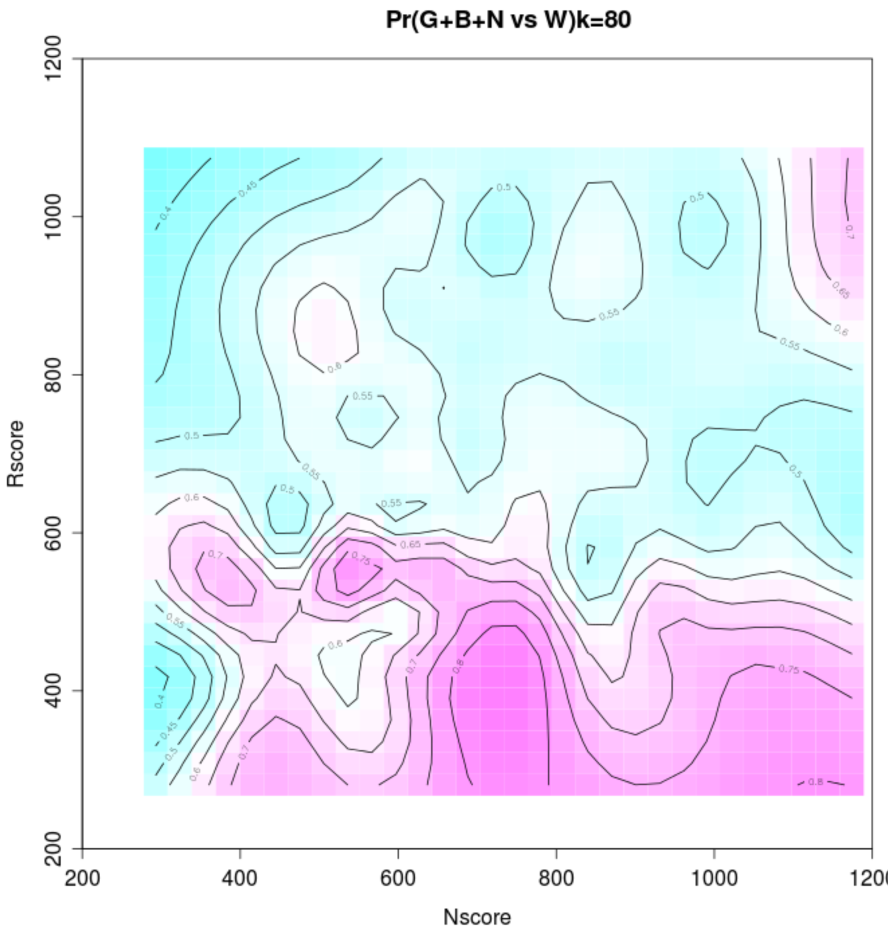
G+B+N vs W ~ s(Nscore,Rscore)



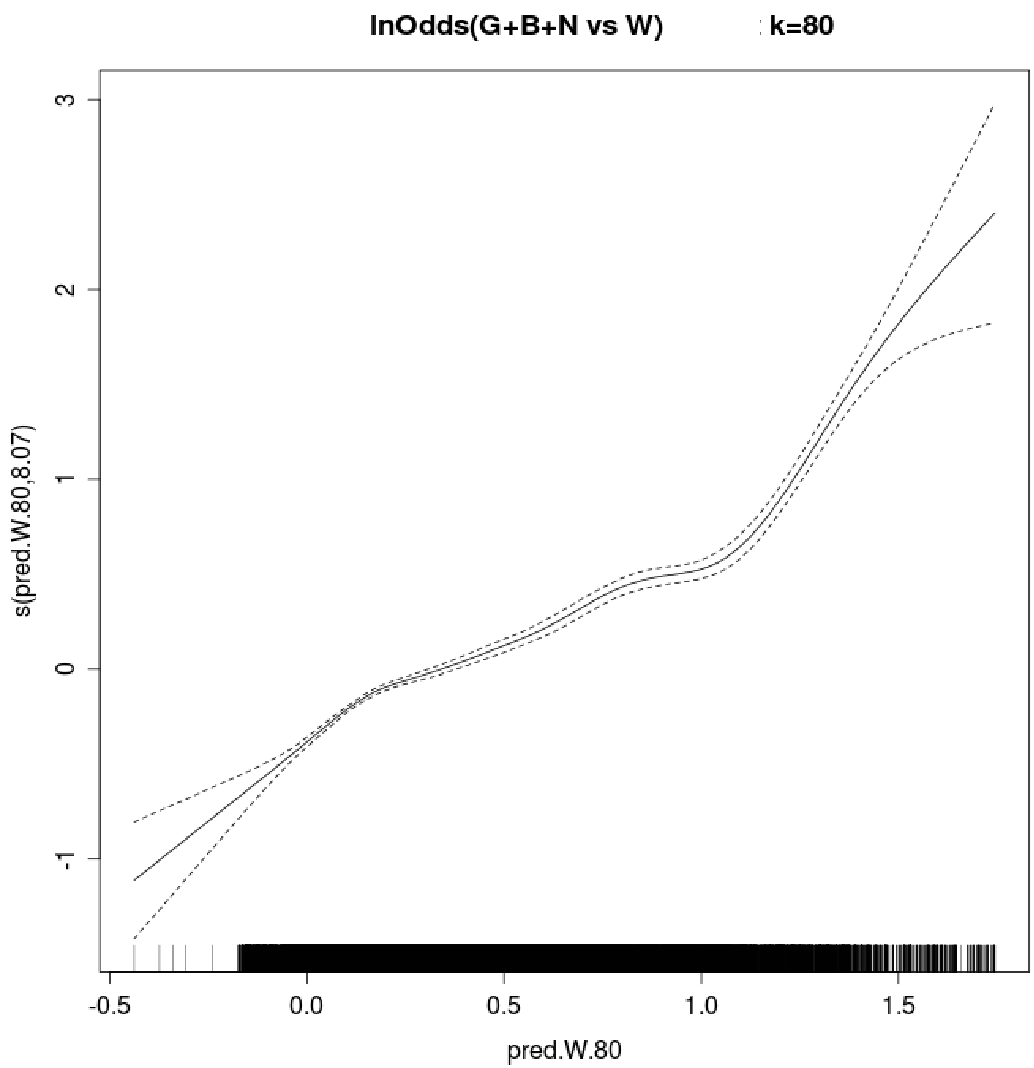
G+B+N vs W ~ s(pred(G+B+N vs W))



G+B+N vs W ~ s(Nscore,Rscore; k=80)



G+B+N vs W ~ s(pred(G+B+N vs W; k=80))

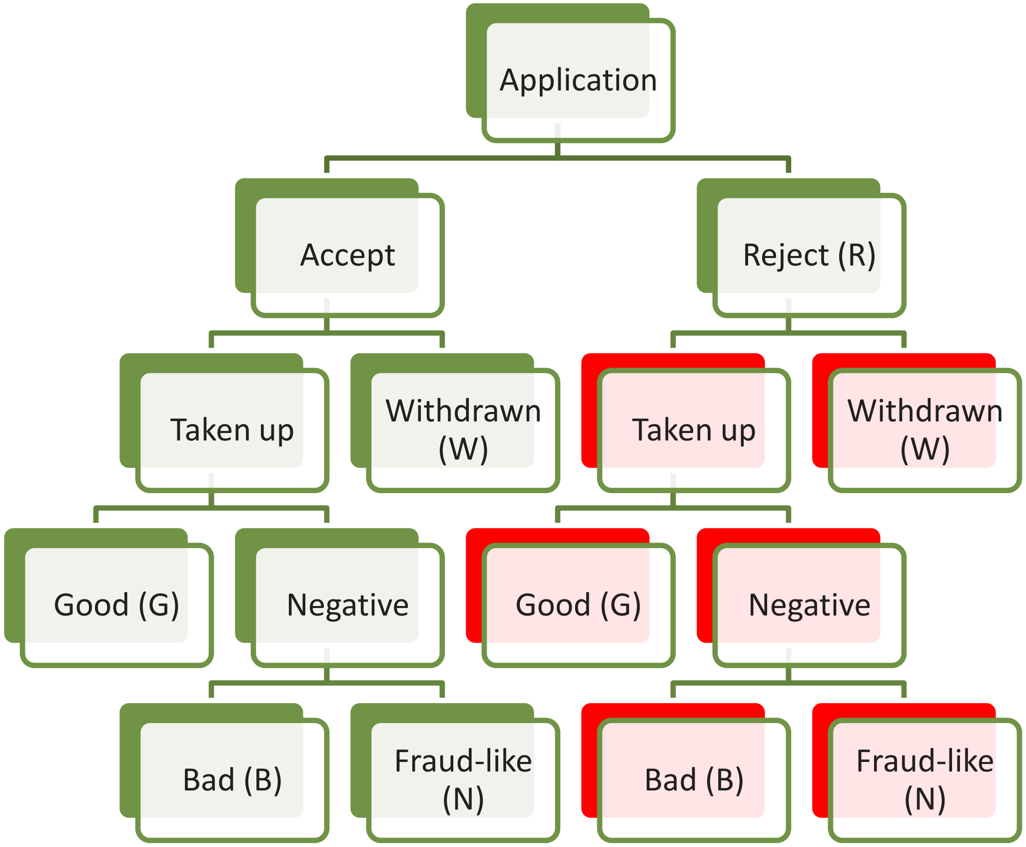


Monotonic calibration

- Monotonic is good enough
- Can take the calibration curve and apply it as a function to monotonically transform the predictions to get a straight calibration line

Reject inference (1)

- Need predicted outcomes for the rejects so that the matrix can be populated with the outcomes conditional on acceptance



Reject inference (2)

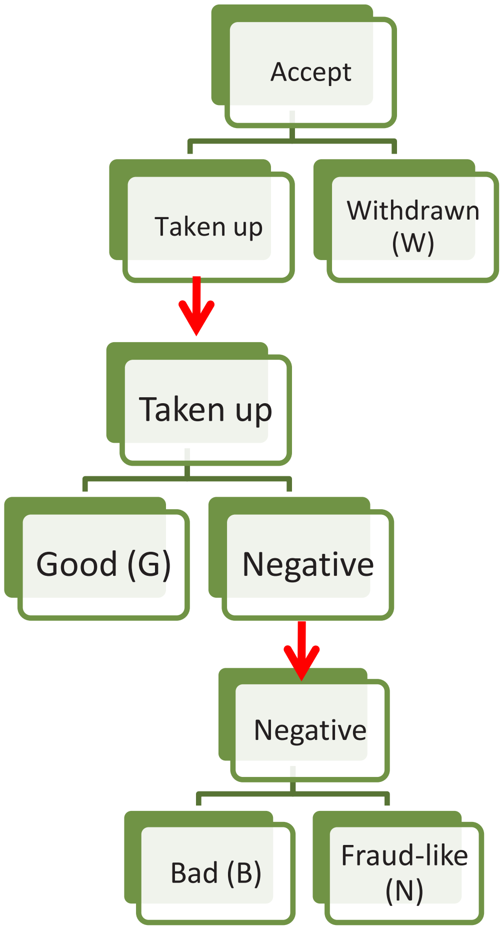
Assume (for the moment) that all predictable variance in the outcomes is captured by the two scores

Assume rejects will show the same pattern of relationships as the accepts

Use the nested conditional models to estimate conditional probabilities for the rejected cases

Combine the conditional estimates to get unconditional probabilities using P(A | B) = P(A & B) / P(B)

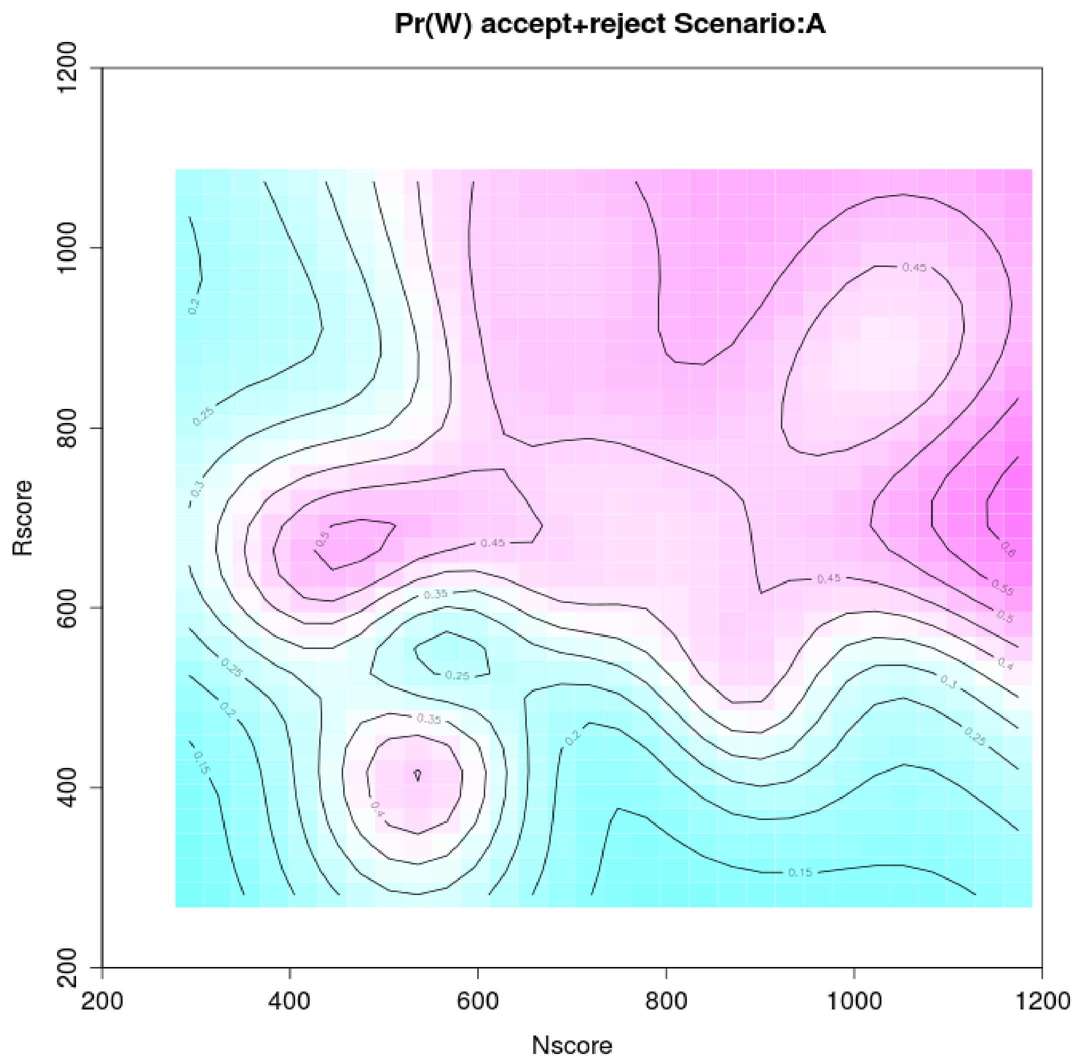
Actually – assume there is some extra information in the fact of rejection and modify the predictions slightly



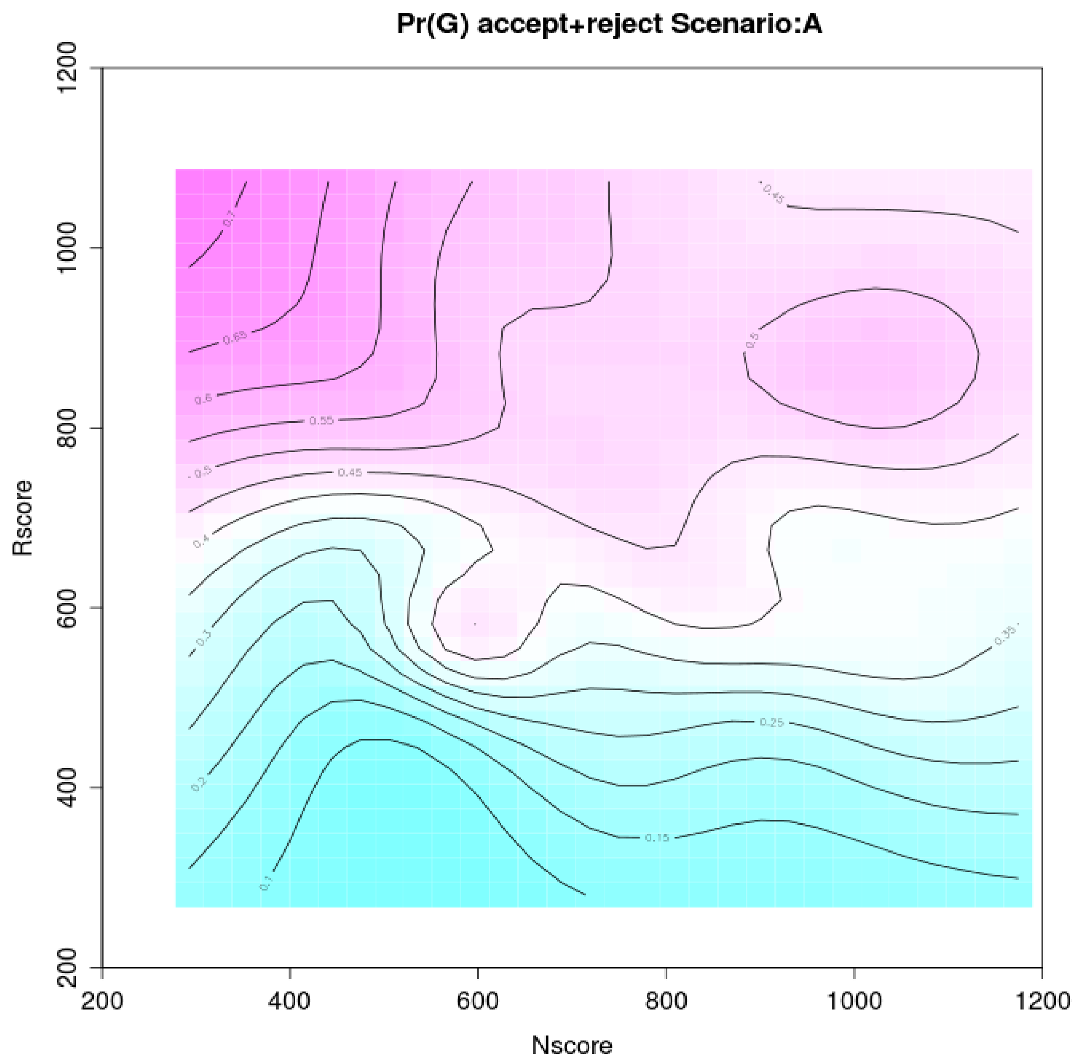
Reject inference (3)

- Randomly realise the inferred outcomes for the rejects according to the predicted unconditional probabilities
- R → {W, G, B, N}
- Pool the known accept cases and the inferred reject cases
- Fit smooth probability surfaces for each of the outcomes as a function of the scores
- These are the smoothed contents of the strategy matrix

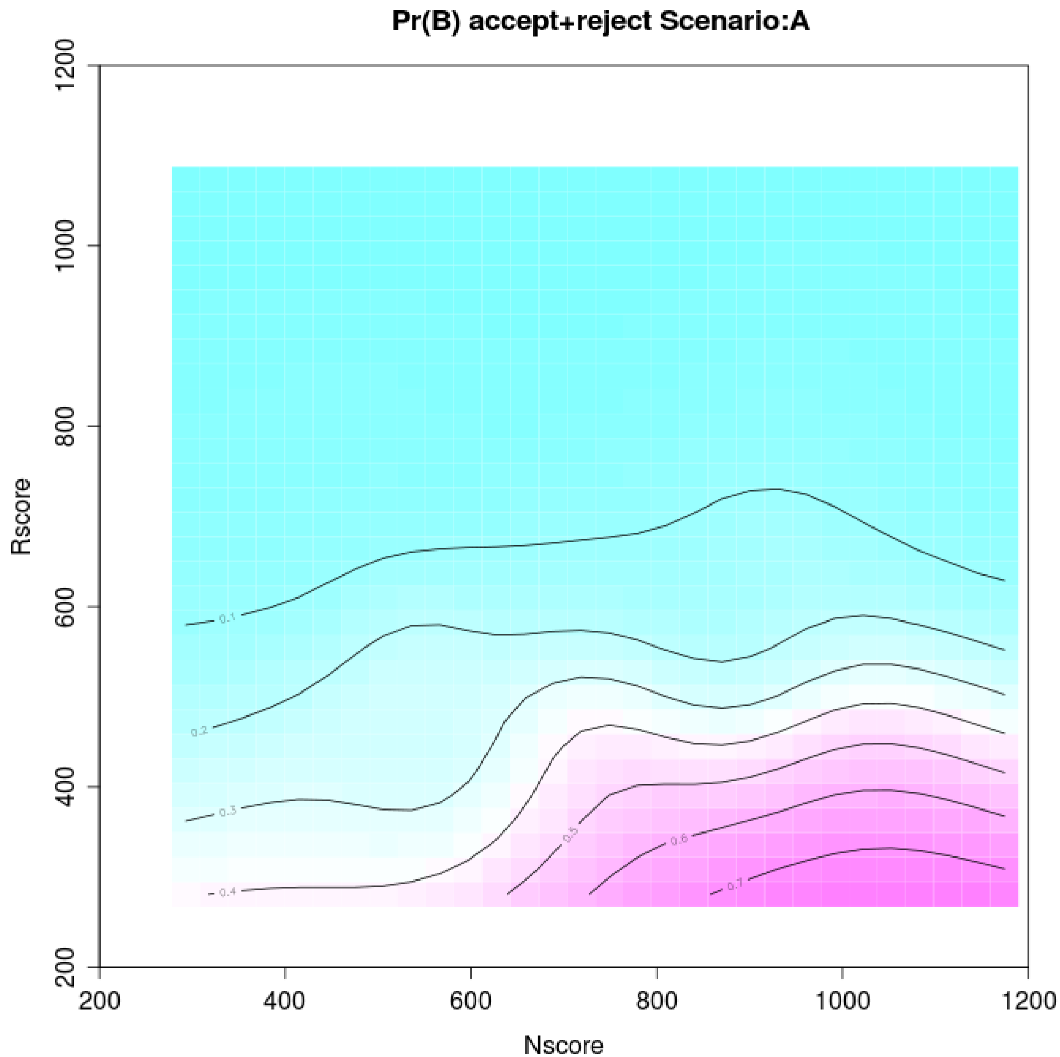
W vs G+B+N ~ s(Nscore,Rscore) (known + inferred)



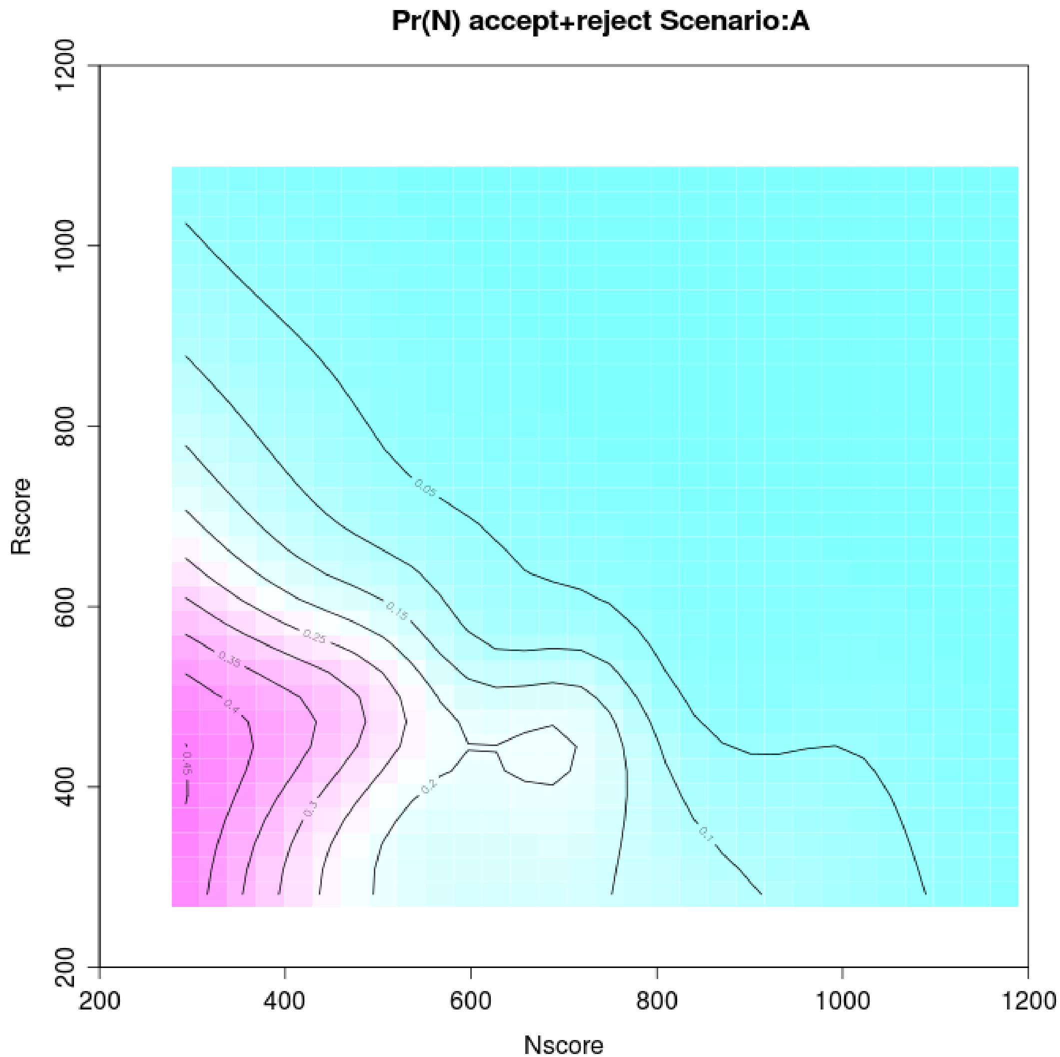
G vs W+B+N ~ s(Nscore,Rscore) (known + inferred)



B vs W+G+N ~ s(Nscore,Rscore) (known + inferred)



N vs W+G+B ~ s(Nscore,Rscore) (known + inferred)



Questions / Comments?