midterm

Christopher Huong

2024-03-14

```
library(dagitty)
library(rethinking)
library(dplyr)
library(splines)
library(ggplot2)
```

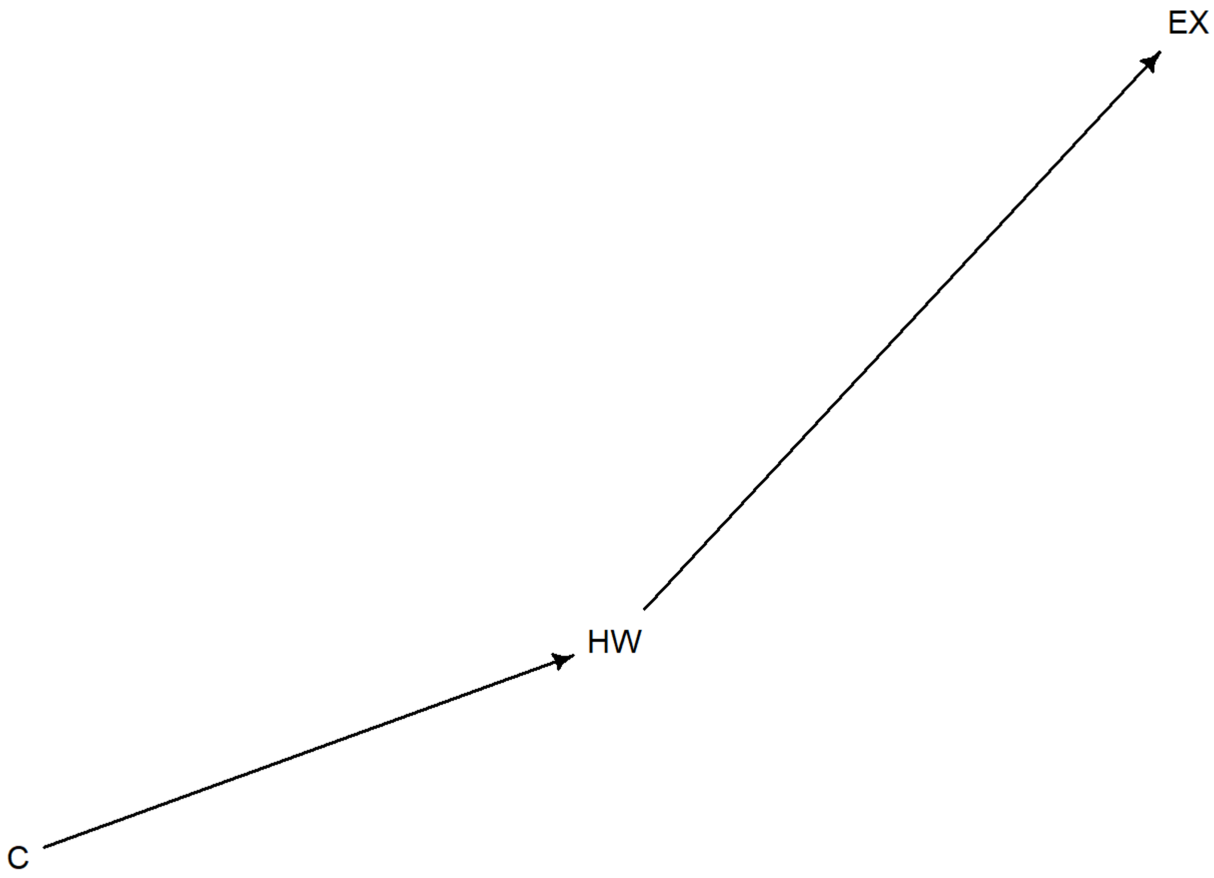
Question 1

1 Conscientiousness; Exam time; Homework scores

As conscientiousness (C) is a relatively stable personality trait, we can assume no causal paths enter C. Given that C predicts GPA, we can posit a path from C to homework score (HW), probably due to more time spent on the course overall. Higher homework scores indicate understanding of the material, which would lead to faster exam times (EX). Thus we can hypothesize a causal path from HW to EX.

This gives us the DAG:

```
dag1 <- dagitty( "dag{
C -> HW -> EX }" )

drawdag(dag1)
```



```
impliedConditionalIndependencies(dag1)
```

```
## C _||_ EX | HW
```

2

For the first DAG; Assuming all variables are standardized

HW ~ N(mu, sigma)
mu = a + b1 * C

EX ~ N(mu, sigma)
mu = a + b2 * HW

3 & 4

```
d1 <- read.csv("HuongIER.csv")
d1$C <- standardize(d1$Conscientiousness)
d1$HW <- standardize(d1$Homework)
d1$EX <- standardize(d1$Duration)
```

```
m1 <- quap(alist(
  # HW <- C
  HW ~ dnorm(mu_HW, sigma_HW),
  mu_HW <- a_HW + b1*C,
  a_HW ~ dnorm(0, 0.2),
  b1 ~ dnorm(0, 0.5),
  sigma_HW ~ dexp(1),

  # EX <- HW
  EX ~ dnorm(mu_EX, sigma_EX),
  mu_EX <- a_EX + b2*HW,
  a_EX ~ dnorm(0, 0.2),
  b2 ~ dnorm(0, 0.5),
  sigma_EX ~ dexp(1)

 ),data=d1)
```

```
precis(m1)
```

```
##                   mean         sd        5.5%       94.5%
## a_HW      6.905000e-06 0.04881911 -0.07801547  0.07802928
## b1        9.469843e-01 0.05094026  0.86557192  1.02839668
## sigma_HW  2.847768e-01 0.03540160  0.22819819  0.34135537
## a_EX     -2.176478e-05 0.08850541 -0.14147051  0.14142698
## b2       -7.890414e-01 0.09860633 -0.94663337 -0.63144945
## sigma_EX  5.583043e-01 0.06910002  0.44786909  0.66873943
```

The coefficients support the hypothesis, with a large positive association between C and HW (b1=0.95, 89% CI: 0.87 to 1.03), and large negative association between HW and EX (b2=-0.79, 89% CI: -0.95 to -0.63). We can also test the conditional independency implied by the DAG

- CI = credibility interval of the posterior

```
m2 <- quap(alist(
  # EX ~ C + HW
  EX ~ dnorm(mu, sigma),
  mu <- a + bC*C + bHW*HW,
  a ~ dnorm(0, 0.2),
  c(bC, bHW) ~ dnorm(0, 0.5),
  sigma ~ dexp(1)
),data=d1)
```

```
precis(m2)
```

```
##                 mean         sd       5.5%      94.5%
## a     -3.895769e-07 0.08886643 -0.1420261  0.1420253
## bC    -1.609330e-01 0.25281895 -0.5649865  0.2431205
## bHW   -6.407442e-01 0.25328282 -1.0455391 -0.2359494
## sigma  5.611404e-01 0.06976911  0.4496359  0.6726449
```

The zero-order correlation matrix:

```
d1 %>% select(C, HW, EX) %>% cor()
```

```
##             C         HW         EX
## C   1.0000000  0.9568923 -0.7805966
## HW  0.9568923  1.0000000 -0.8207747
## EX -0.7805966 -0.8207747  1.0000000
```

It seems the strong negative correlation between C and EX is explained by HW, as evidenced by the uncertainty of the posterior distribution of the bC slope in the multiple regression.

Thus, the proposed DAG is supported by the data.

Question 1

```
rm(list=ls())
d2 <- read.csv("HuongYerkes.csv")
str(d2)
```

```
## 'data.frame':    40 obs. of  3 variables:
##  $ Subject   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Motivation: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Score     : int  80 64 71 88 69 70 64 69 76 76 ...
```

```
d2$M <- standardize(d2$Motivation)
d2$S <- standardize(d2$Score)
d2$M_sq <- d2$M^2
```

We will test and compare a linear model, a quadratic model, and a spline while using regularizing (narrow) priors to reduce risk of overfitting

Linear model:

S ~ N(mu, sigma)
mu = a + b * M

Quadratic model:
S ~ N(mu, sigma)
mu = a + b1*M* + *b2*M^2

B-Spline:
S ~ N(mu, sigma)
mu = a + summation k=1 to k (w(k) * b(k,i))

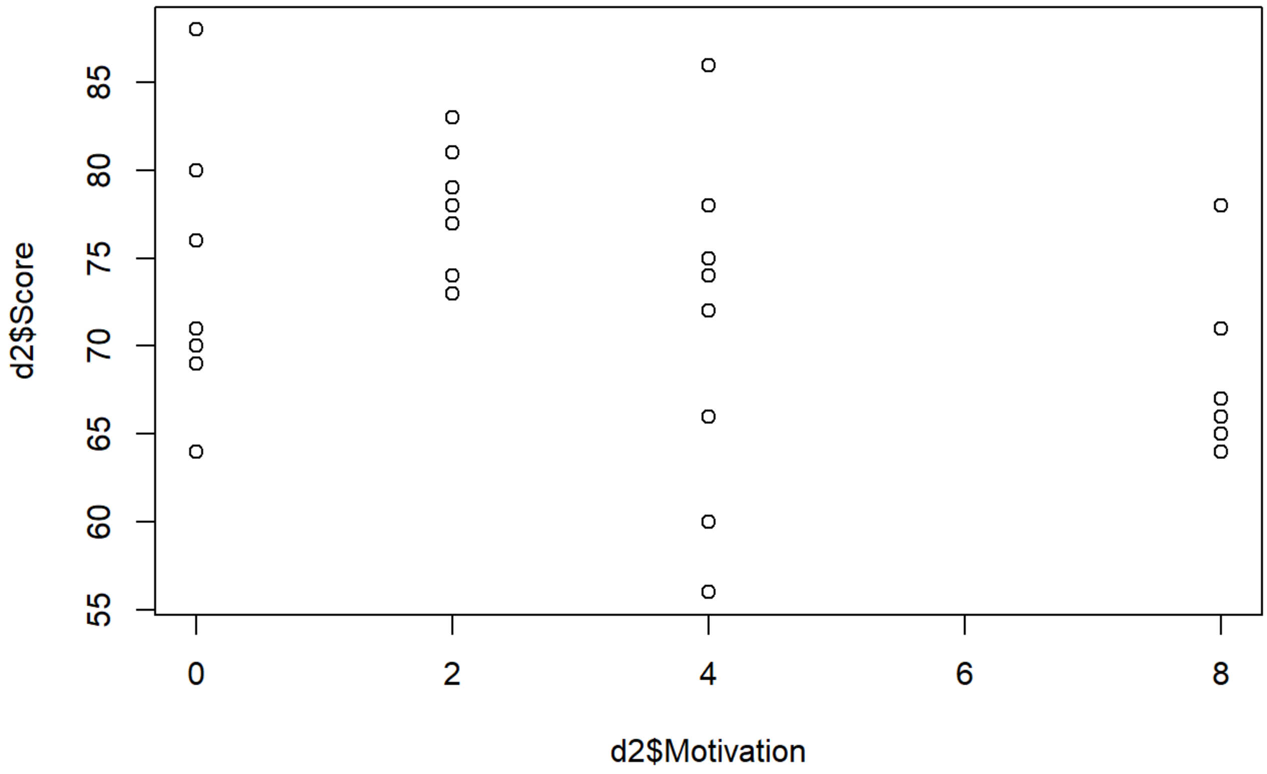
Q2 & Q3

Model syntax

```
lm <- quap(alist(
  S ~ dnorm(mu, sigma),
  mu <- a + b*M,
  a ~ dnorm(0, 0.2),
  b ~ dnorm(0, 0.2),
  sigma ~ dexp(1)
),data=d2)


qm <- quap(alist(
  S ~ dnorm(mu, sigma),
  mu <- a + b*M + b_2*M_sq,
  a ~ dnorm(0, 0.2),
  c(b, b_2) ~ dnorm(0, 0.2),
  sigma ~ dexp(1)
),data=d2)



num_knots <- 5
knot_list <- quantile(d2$M, probs=seq(0, 1, length.out=5))


B <- bs(d2$M,
        knots = knot_list[-c(1, num_knots)],
        degree=3, intercept=T)

sm <- quap(
  alist(
    S ~ dnorm(mu, sigma),
    mu <- a + B %*% w,
    a ~ dnorm(0, 0.2),
    w ~ dnorm(0, 5),  #influences wigglyness
    sigma ~ dexp(1)
  ),data=list(S=d2$S, B=B),
  start=list(w=rep(0, ncol(B)))
)
```

Plot data and inspect results

```
plot(x=d2$Motivation, y=d2$Score)
```



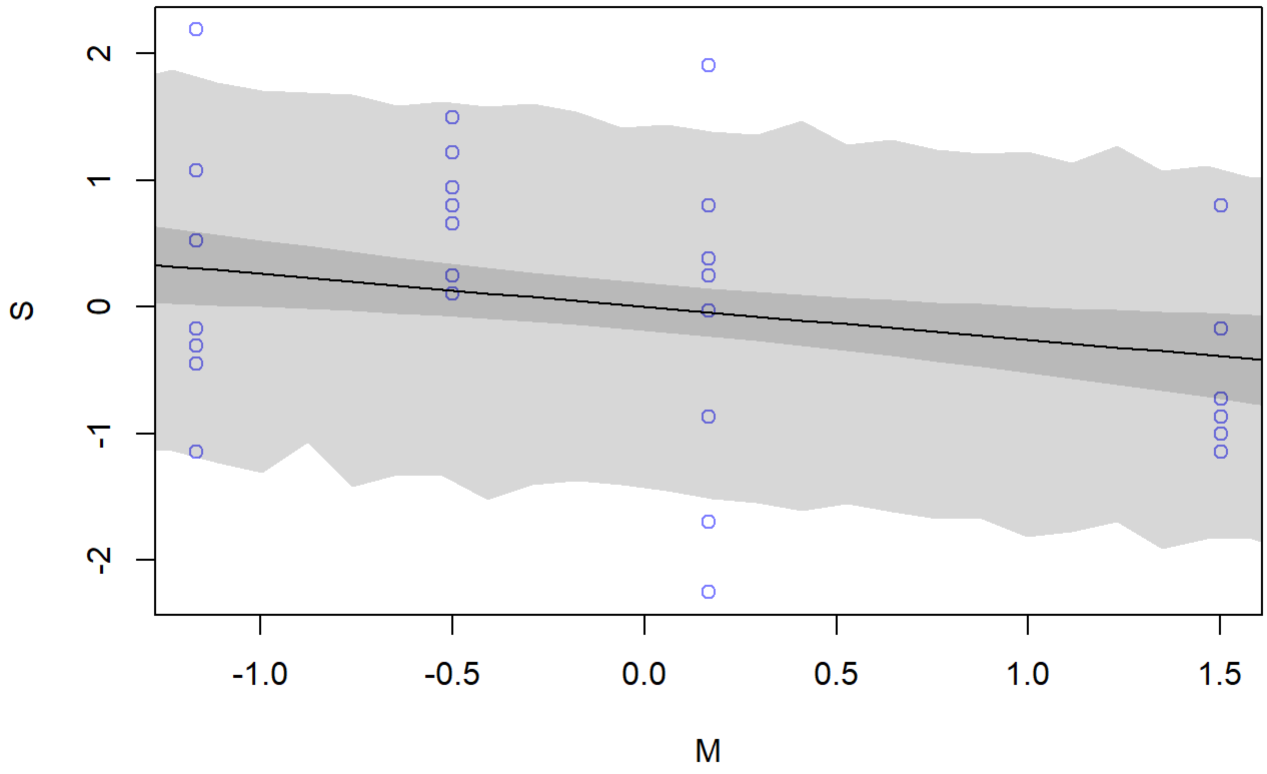
```
precis(lm)
```

```
##                 mean        sd       5.5%       94.5%
## a     -1.107761e-05 0.1165024 -0.1862044  0.18618227
## b     -2.593468e-01 0.1191597 -0.4497870 -0.06890649
## sigma  9.065033e-01 0.1011006  0.7449250  1.06808160
```

Conditional on the linear model and data, 89% of the posterior probability infers a regression slope coefficient between -0.45 and -0.07.

Plot posterior predictions of the linear model against the data

```r
# extract samples from the posterior probability distribution
lm_post <- extract.samples(lm, n=10000)
# link function to generate a distribution of mu values for each value of M
mu.link <- function(M) lm_post$a + lm_post$b*M
# new horizontal axis to plot posterior predictions
m.seq <- seq(from=-1.7, to=1.7, length.out=30)
# apply link function to new horizontal axis of M values
mu <- sapply(m.seq , mu.link)
# distribution of posterior mu values of S for 30 values of M

# summarize the posterior predictions with mean and 89% percentile intervals
mu.mean <- apply(mu, 2, mean)
mu.PI <- apply(mu, 2, PI, prob=0.89)

# simulate Score values for each value of M, and get 89% PI
sim.scores <- sim(lm, data=list(M=m.seq))
scores.PI <- apply(sim.scores, 2, PI, prob=0.89)
# plot posterior predictions of mu, and simulated Scores again raw data
plot(S~M, data=d2, col=rangi2)
lines(m.seq, mu.mean)
shade(mu.PI, m.seq)
shade(scores.PI, m.seq)
```



The bulk of the posterior predictive distribution describes a negative slope. The regression slopes seem to reliably underestimate Scores for (standardized) Motivation values of -0.5. Further, 4 data points lie outside the 89% probability interval for simulated scores

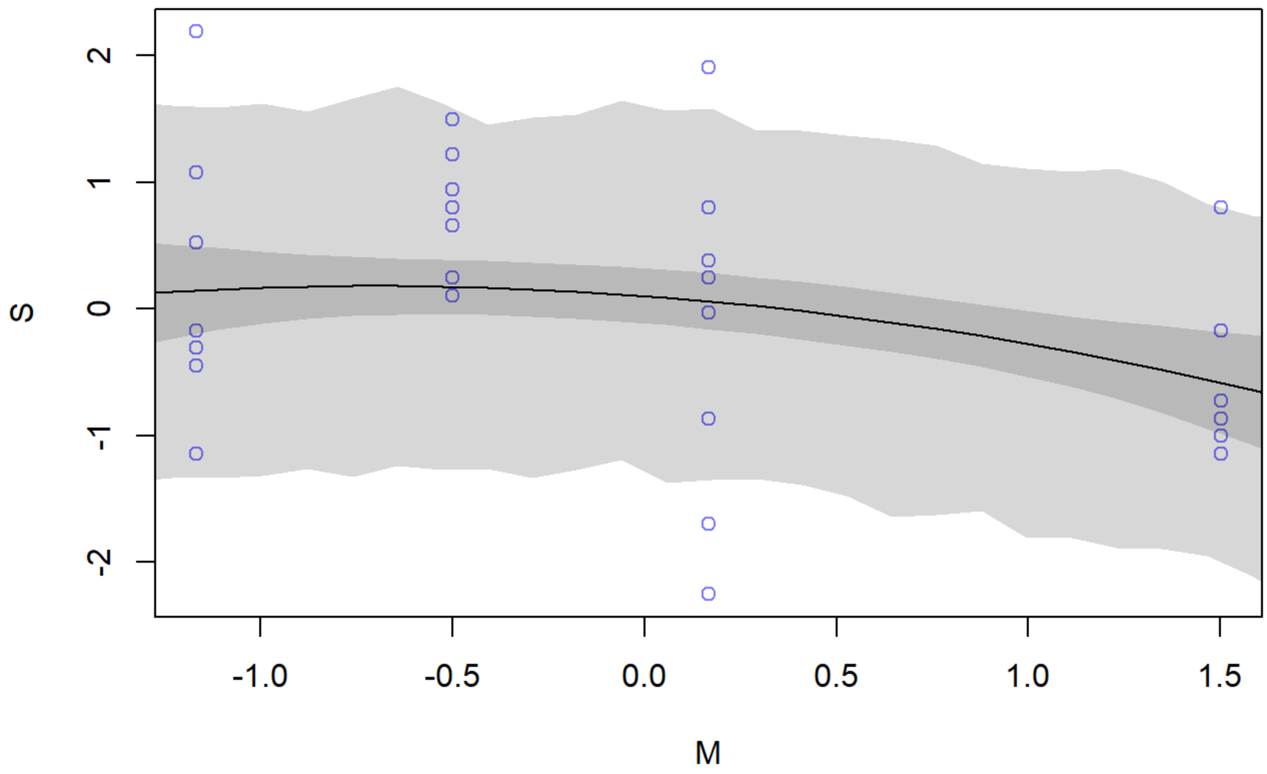
Now for quadratic model

```
precis(qm)
```

```
##             mean         sd       5.5%       94.5%
## a      0.1019033 0.13778356 -0.1183015  0.32210804
## b     -0.2195140 0.12073054 -0.4124647 -0.02656328
## b_2   -0.1554496 0.11524601 -0.3396350  0.02873578
## sigma  0.8830168 0.09880565  0.7251063  1.04092728
```

Not sure if the coefficients are interpretable. Let’s just plot

```
qm_post <- extract.samples(qm, n=10000)
pred_dat <- list(M=m.seq, M_sq=m.seq^2)
mu.link <- function(M) qm_post$a + qm_post$b*M + qm_post$b_2*M^2
mu <- sapply(m.seq , mu.link)


mu.mean <- apply(mu, 2, mean)
mu.PI <- apply(mu, 2, PI, prob=0.89)


sim.scores <- sim(qm, data=pred_dat)
scores.PI <- apply(sim.scores, 2, PI, prob=0.89)
# plot posterior predictions of mu, and simulated Scores again raw data
plot(S~M, data=d2, col=rangi2)
lines(m.seq, mu.mean)
shade(mu.PI, m.seq)
shade(scores.PI, m.seq)
```

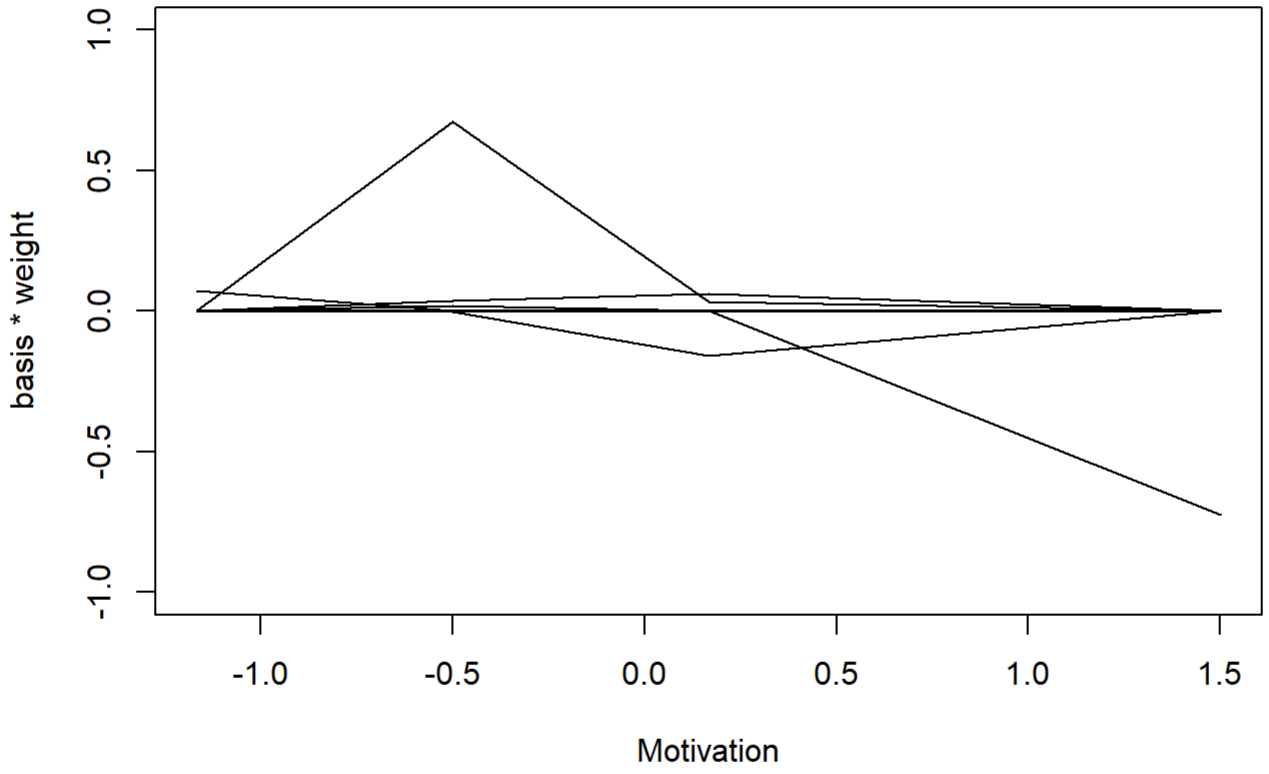


Does not look too different from the linear model. The same 4 points lie outside the 89% interval of simulated Scores.

Now for the spline model

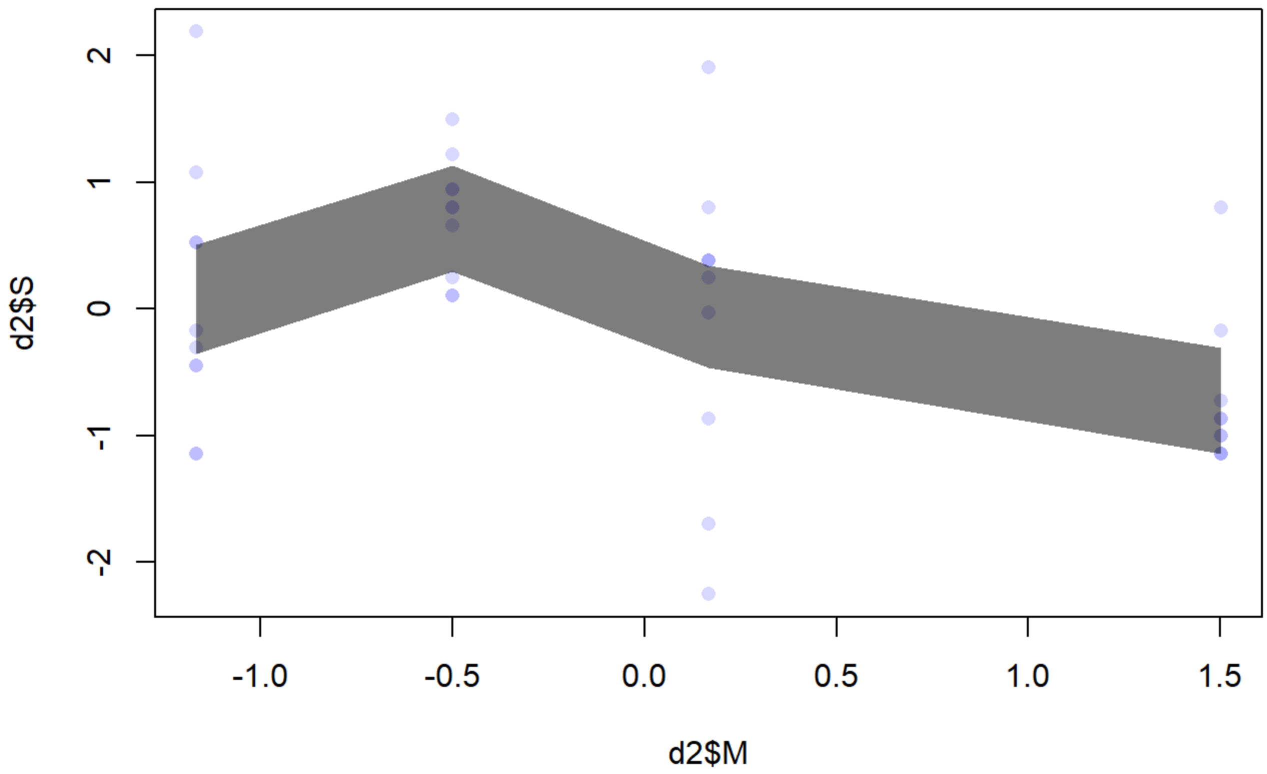
Plot weights for basis functions

```
sm_post <- extract.samples(sm, n=10000)

w <- apply(sm_post$w , 2 , mean )
plot( NULL , xlim=range(d2$M) , ylim=c(-1,1) ,
xlab="Motivation" , ylab="basis * weight" )
for ( i in 1:ncol(B) ) lines( d2$M , w[i]*B[,i] )
```



Plot spline

```
mu <- link(sm)
mu_PI <- apply(mu,2,PI,0.89)
plot(d2$M , d2$S , col=col.alpha(rangi2,0.3) , pch=16 )
shade(mu_PI, d2$M, col=col.alpha("black",0.5) )
```



Compare the predictive accuracy between models

Predictive accuracy is assessed by the average log-probability of a model to estimate relative divergence from the “true” data generating model. The average log-probability is estimated by summing the models log-probability of each observation.

To compute this for a Bayesian model, we calculate the log pointwise predictive density (lppd):

First, we compute log=probabilities for each observation:

1. Draw 10,000 samples for each parameter from the posterior probability distribution.

2. Use those 10,000 parameter samples to estimate 10,000 (standardized) Score values for each 40 observed value of (standardized) Motivation using the model specification relating Motivation to Score (linear, quadratic, spline).

3. Return log-probabilities of each simulated observation (conditional on the model)

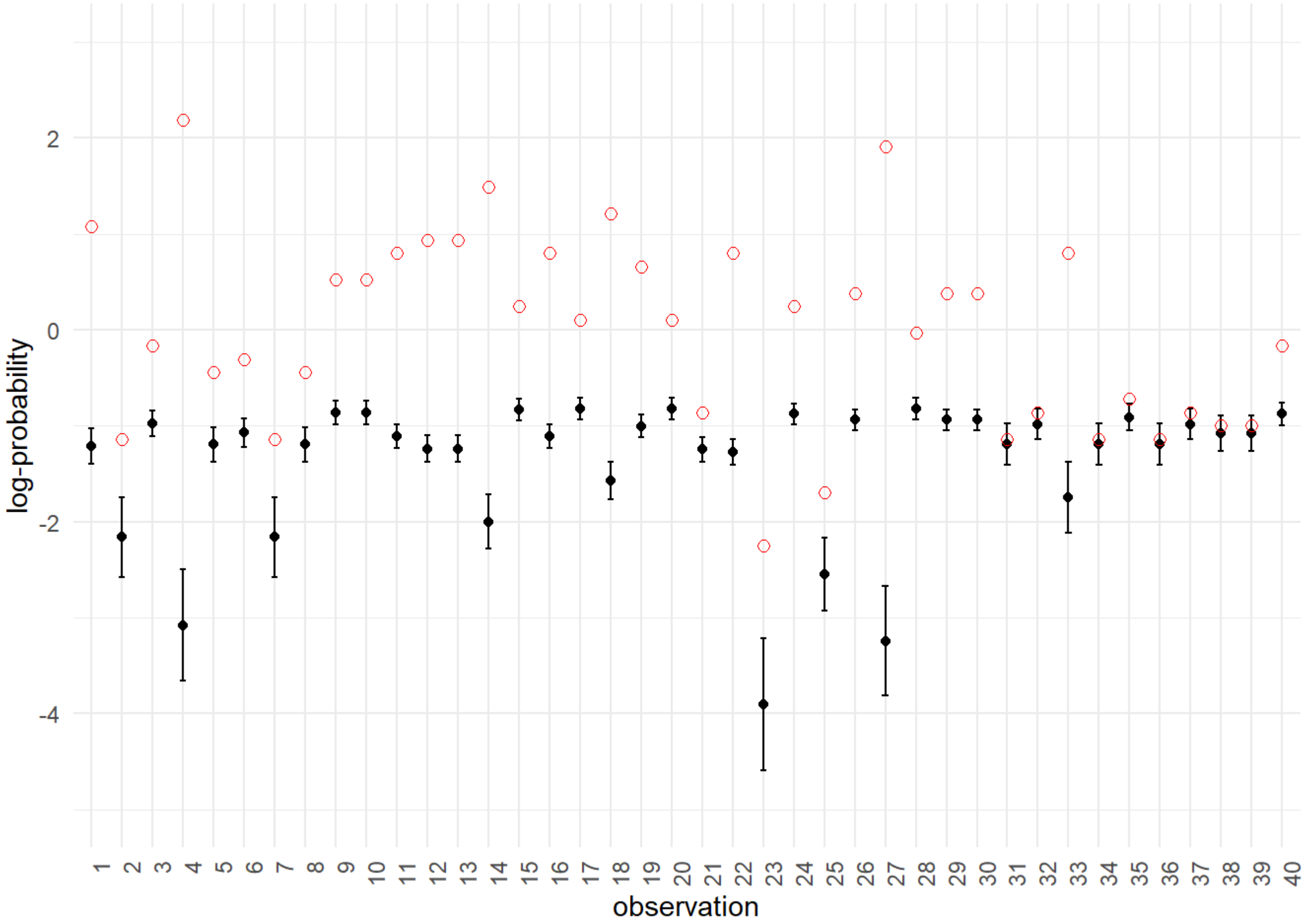
4. Prepare data for plotting

```
logprob_lm <- sim(lm, ll=T, n=1e4)
logprob_lm_plot <- data.frame(
  observation = as.factor(seq(1:40)),
  means = apply(logprob_lm, 2, mean),
  sd = apply(logprob_lm, 2, sd)
)


logprob_qm <- sim(qm, ll=T, n=1e4)
logprob_qm_plot <- data.frame(
  observation = as.factor(seq(1:40)),
  means = apply(logprob_qm, 2, mean),
  sd = apply(logprob_qm, 2, sd)
)


logprob_sm <- sim(sm, ll=T, n=1e4)
logprob_sm_plot <- data.frame(
  observation = as.factor(seq(1:40)),
  means = apply(logprob_sm, 2, mean),
  sd = apply(logprob_sm, 2, sd)
)
```

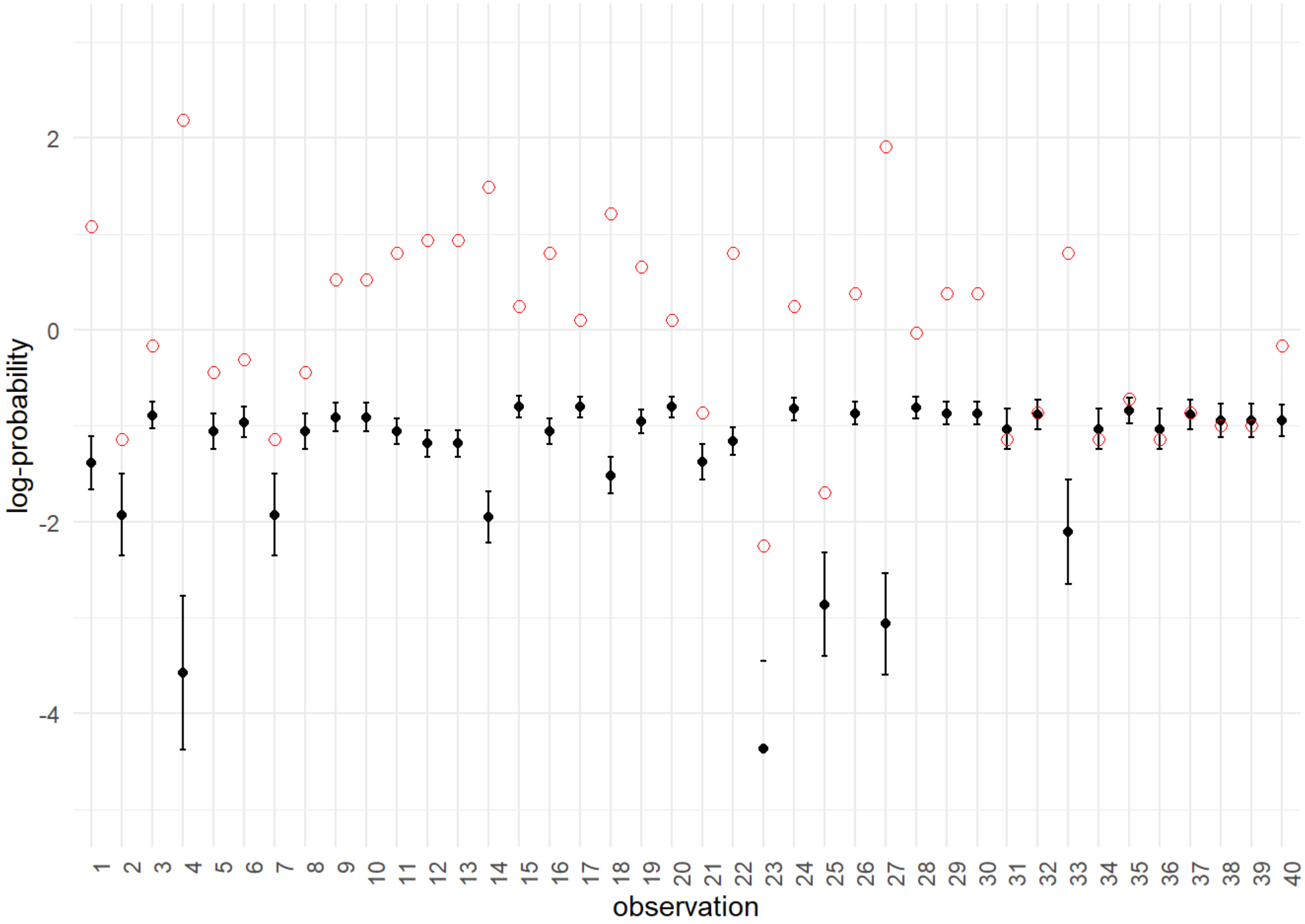
Plot simulated log-probabilities (points = mean, bars = standard deviations) of each observed Score

```
logprob_plot <- function(logprob_d) {
  ggplot(logprob_d, aes(x = observation, y = means)) +
   geom_point() +
   geom_errorbar(aes(ymin = means - sd, ymax = means + sd), width = 0.2) +
    geom_point(aes(y = d2$S), color = "red", size = 2, shape = 1) +
    labs(x = "observation", y = "log-probability") +
    ylim(-5, 3) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
}
```

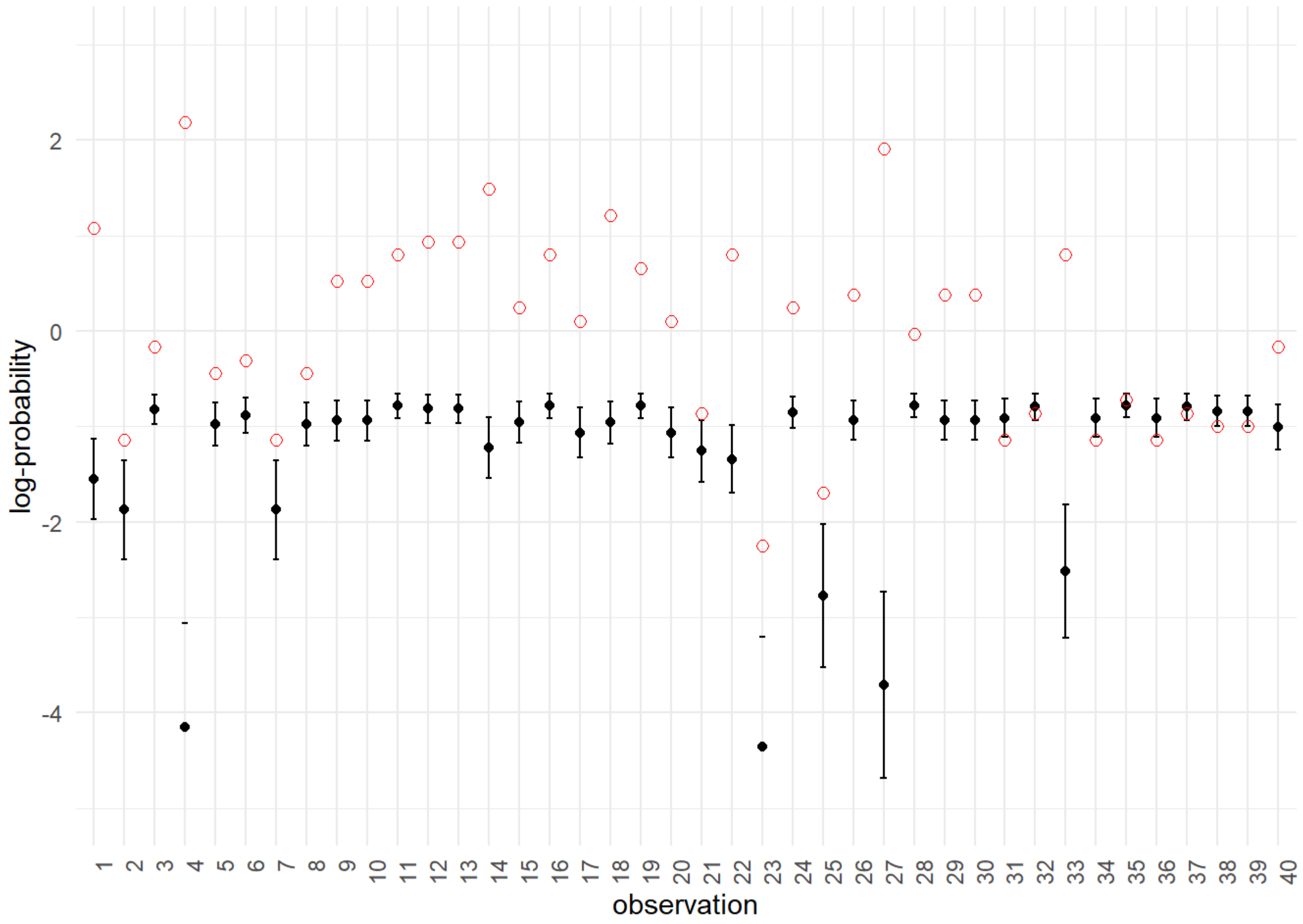
```
logprob_plot(logprob_lm_plot)
```



```
logprob_plot(logprob_qm_plot)
```



```
logprob_plot(logprob_sm_plot)
```



Resume computing lppd

```
n <- 40
ns <- 10000


# for the 10,000 simulated log-probabilities of each observation (conditional on the model), exponentiate each value, sum them, then take the log. subtract the log of the sample size from each (this divides the sum by the number of samples)

lppd_func <- function(i, logprob_d) {
  log_sum_exp(logprob_d[, i]) - log(ns)
}
```

Apply the function to compute lppd for each model (equivalent results to lppd())

```
sapply(1:n, function(i) lppd_func(i, logprob_lm)) |> sum()
```

```
## [1] -53.37836
```

```
sapply(1:n, function(i) lppd_func(i, logprob_qm)) |> sum()
```

```
## [1] -52.19257
```

```
sapply(1:n, function(i) lppd_func(i, logprob_sm)) |> sum()
```

```
## [1] -49.75263
```

The lppd score computed from training data improves with model complexity. Ideally, the lppd for each model can be computed for data the model was not trained on, and those scores can be compared to rank models by predictive accuracy. Information criterion and cross-validation are techniques to estimate this hypothetical out-of-sample lppd/deviance score.

Cross-validation re-fits the model on subsets of the data, and computes lppd (or deviance? I don't see a -2 multiplier in the formula, yet the magnitudes are very similar to WAIC which does estimate deviance) scores of each model on the observations that were left out. The lppd is then averaged for a final out-of-sample predictive accuracy estimate

PSIS approximates a LOOCV score without having to refit the model by weighing observations by their relative likelihood.

```
compare(lm, qm, sm, func=PSIS)
```

```
## Some Pareto k values are high (>0.5). Set pointwise=TRUE to inspect individual points.
## Some Pareto k values are high (>0.5). Set pointwise=TRUE to inspect individual points.
## Some Pareto k values are high (>0.5). Set pointwise=TRUE to inspect individual points.
```

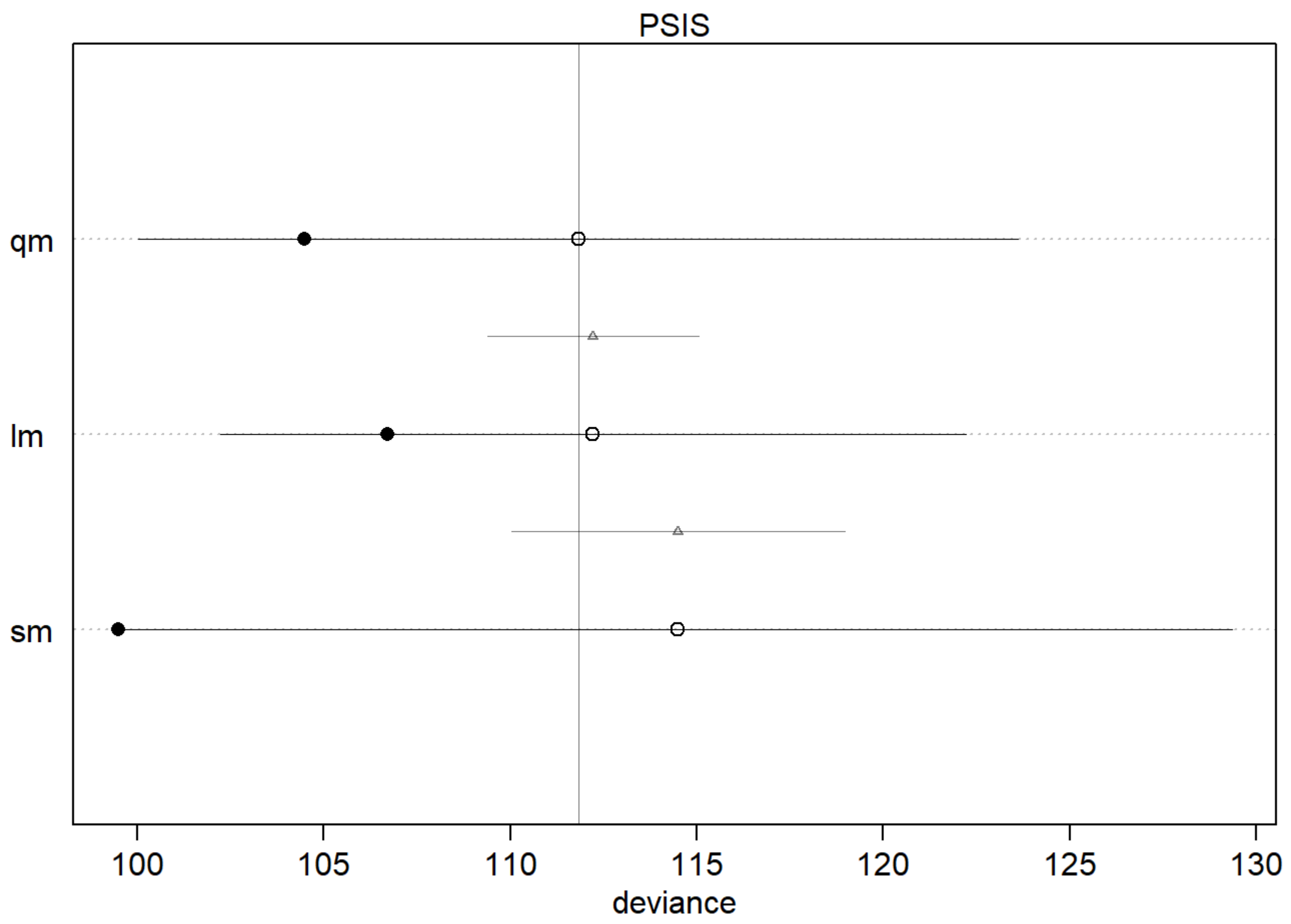
```
##        PSIS        SE      dPSIS      dSE    pPSIS    weight
## lm 111.8378  9.922361 0.00000000       NA 2.564240 0.4137051
## qm 111.8661 11.808815 0.02830084 2.928991 3.744569 0.4078922
## sm 113.5200 14.324100 1.68222007 5.999466 6.992820 0.1784027
```

Also can plot

```
plot(compare(lm, qm, sm, func=PSIS))
```

```
## Some Pareto k values are high (>0.5). Set pointwise=TRUE to inspect individual points.
## Some Pareto k values are high (>0.5). Set pointwise=TRUE to inspect individual points.
```

```
## Some Pareto k values are very high (>1). Set pointwise=TRUE to inspect individual points.
```



The models are not reliably different in their predictive accuracy according to PSIS, given the largest dPSIS value is smaller than its corresponding dSE value (the standard error of the difference is larger than the difference itself). Though as expected, the penalty term (pPSIS) increases with model complexity

Some outliers may be disproportionately contributing to overfitting, and thus reduced predictive accuracy. We can inspect these with by specifying pointwise=T

```
PSIS_lm <- PSIS(lm, pointwise=T)
print(PSIS_lm)
```

```
##        PSIS       lppd    penalty  std_err             k
## 1  2.458048 -1.2290238 0.03539509 9.952468  1.050560e-01
## 2  4.562749 -2.2813745 0.19530268 9.952468  4.341976e-01
## 3  1.975464 -0.9877318 0.01831714 9.952468  1.496604e-01
## 4  6.568404 -3.2842022 0.34507855 9.952468  3.179761e-01
## 5  2.429411 -1.2147056 0.03553843 9.952468  1.254432e-01
## 6  2.175695 -1.0878474 0.02497772 9.952468  2.086926e-01
## 7  4.562749 -2.2813745 0.19530268 9.952468  4.341976e-01
## 8  2.429411 -1.2147056 0.03553843 9.952468  1.254432e-01
## 9  1.740704 -0.8703521 0.01474302 9.952468 -1.357465e-01
## 10 1.740704 -0.8703521 0.01474302 9.952468 -1.357465e-01
## 11 2.234952 -1.1174762 0.01541323 9.952468 -9.917332e-02
## 12 2.500257 -1.2501283 0.01894482 9.952468  4.499467e-02
## 13 2.500257 -1.2501283 0.01894482 9.952468  4.499467e-02
## 14 4.098498 -2.0492490 0.07535080 9.952468  3.160533e-01
## 15 1.675287 -0.8376434 0.01280221 9.952468 -2.909025e-05
## 16 2.234952 -1.1174762 0.01541323 9.952468 -9.917332e-02
## 17 1.658045 -0.8290224 0.01249244 9.952468 -5.556382e-02
## 18 3.188601 -1.5943005 0.03590804 9.952468  1.420809e-01
## 19 2.020601 -1.0103007 0.01375693 9.952468 -2.836459e-01
## 20 1.658045 -0.8290224 0.01249244 9.952468 -5.556382e-02
## 21 2.496741 -1.2483703 0.01513911 9.952468  1.764493e-01
## 22 2.577313 -1.2886564 0.01584464 9.952468  1.176185e-02
## 23 8.436067 -4.2180337 0.51650664 9.952468  3.048841e-01
## 24 1.763334 -0.8816670 0.01161416 9.952468 -1.519480e-01
## 25 5.256815 -2.6284074 0.15178837 9.952468  2.915764e-01
## 26 1.891682 -0.9458408 0.01099807 9.952468 -2.367892e-01
## 27 6.925632 -3.4628160 0.32690861 9.952468  4.811740e-01
## 28 1.653135 -0.8265674 0.01260027 9.952468 -2.726477e-02
## 29 1.891682 -0.9458408 0.01099807 9.952468 -2.367892e-01
## 30 1.891682 -0.9458408 0.01099807 9.952468 -2.367892e-01
## 31 2.409167 -1.2045837 0.04358855 9.952468  2.677485e-01
## 32 1.969546 -0.9847728 0.02456427 9.952468  1.658827e-01
## 33 3.710654 -1.8553269 0.14759971 9.952468  3.746881e-01
## 34 2.409167 -1.2045837 0.04358855 9.952468  2.677485e-01
## 35 1.827433 -0.9137163 0.01892561 9.952468  8.327044e-02
## 36 2.409167 -1.2045837 0.04358855 9.952468  2.677485e-01
## 37 1.969546 -0.9847728 0.02456427 9.952468  1.658827e-01
## 38 2.163190 -1.0815950 0.03256985 9.952468  2.341858e-01
## 39 2.163190 -1.0815950 0.03256985 9.952468  2.341858e-01
## 40 1.767506 -0.8837528 0.01468307 9.952468  2.611362e-01
```

```
PSIS_qm <- PSIS(qm, pointwise=T)
```

```
## Some Pareto k values are high (>0.5). Set pointwise=TRUE to inspect individual points.
```

```
print(PSIS_qm)
```

```
##        PSIS       lppd    penalty  std_err            k
## 1  2.889229 -1.4446146 0.07552142 11.60725  0.527923351
## 2  4.001404 -2.0007022 0.17136120 11.60725  0.277701534
## 3  1.787760 -0.8938799 0.01619077 11.60725  0.178638209
## 4  7.973575 -3.9867875 0.63121183 11.60725  0.600954179
## 5  2.132315 -1.0661577 0.03141169 11.60725  0.178325824
## 6  1.932699 -0.9663495 0.02203499 11.60725  0.173221136
## 7  4.001404 -2.0007022 0.17136120 11.60725  0.277701534
## 8  2.132315 -1.0661577 0.03141169 11.60725  0.178325824
## 9  1.862810 -0.9314048 0.02254271 11.60725  0.134252653
## 10 1.862810 -0.9314048 0.02254271 11.60725  0.134252653
## 11 2.136631 -1.0683155 0.01707276 11.60725 -0.020081516
## 12 2.393672 -1.1968361 0.02085843 11.60725  0.159446215
## 13 2.393672 -1.1968361 0.02085843 11.60725  0.159446215
## 14 3.972279 -1.9861397 0.07353197 11.60725  0.506478070
## 15 1.626900 -0.8134498 0.01196954 11.60725 -0.171695557
## 16 2.136631 -1.0683155 0.01707276 11.60725 -0.020081516
## 17 1.627004 -0.8135019 0.01125560 11.60725 -0.252376038
## 18 3.069947 -1.5349734 0.03722432 11.60725  0.258299893
## 19 1.932007 -0.9660033 0.01485077 11.60725 -0.100345803
## 20 1.627004 -0.8135019 0.01125560 11.60725 -0.252376038
## 21 2.814379 -1.4071895 0.03842275 11.60725  0.204466222
## 22 2.334760 -1.1673802 0.02279713 11.60725 -0.004430421
## 23 9.870581 -4.9352905 0.86216858 11.60725  0.670596153
## 24 1.665468 -0.8327340 0.01224136 11.60725 -0.202060658
## 25 6.098108 -3.0490541 0.29539484 11.60725  0.383400608
## 26 1.754854 -0.8774271 0.01320457 11.60725 -0.129248993
## 27 6.427377 -3.2136883 0.30098057 11.60725  0.645402533
## 28 1.640145 -0.8200726 0.01147451 11.60725 -0.235684340
## 29 1.754854 -0.8774271 0.01320457 11.60725 -0.129248993
## 30 1.754854 -0.8774271 0.01320457 11.60725 -0.129248993
## 31 2.140016 -1.0700081 0.04339945 11.60725  0.145994831
## 32 1.801181 -0.9005905 0.02234520 11.60725  0.128054767
## 33 4.482068 -2.2410340 0.28301891 11.60725  0.367047783
## 34 2.140016 -1.0700081 0.04339945 11.60725  0.145994831
## 35 1.713104 -0.8565520 0.01661851 11.60725  0.106453847
## 36 2.140016 -1.0700081 0.04339945 11.60725  0.145994831
## 37 1.801181 -0.9005905 0.02234520 11.60725  0.128054767
## 38 1.943344 -0.9716718 0.03115662 11.60725  0.141716469
## 39 1.943344 -0.9716718 0.03115662 11.60725  0.141716469
## 40 1.903270 -0.9516352 0.02557425 11.60725  0.280339964
```

```
PSIS_sm <- PSIS(sm, pointwise=T)
```

```
## Some Pareto k values are very high (>1). Set pointwise=TRUE to inspect individual points.
```

```
print(PSIS_sm)
```

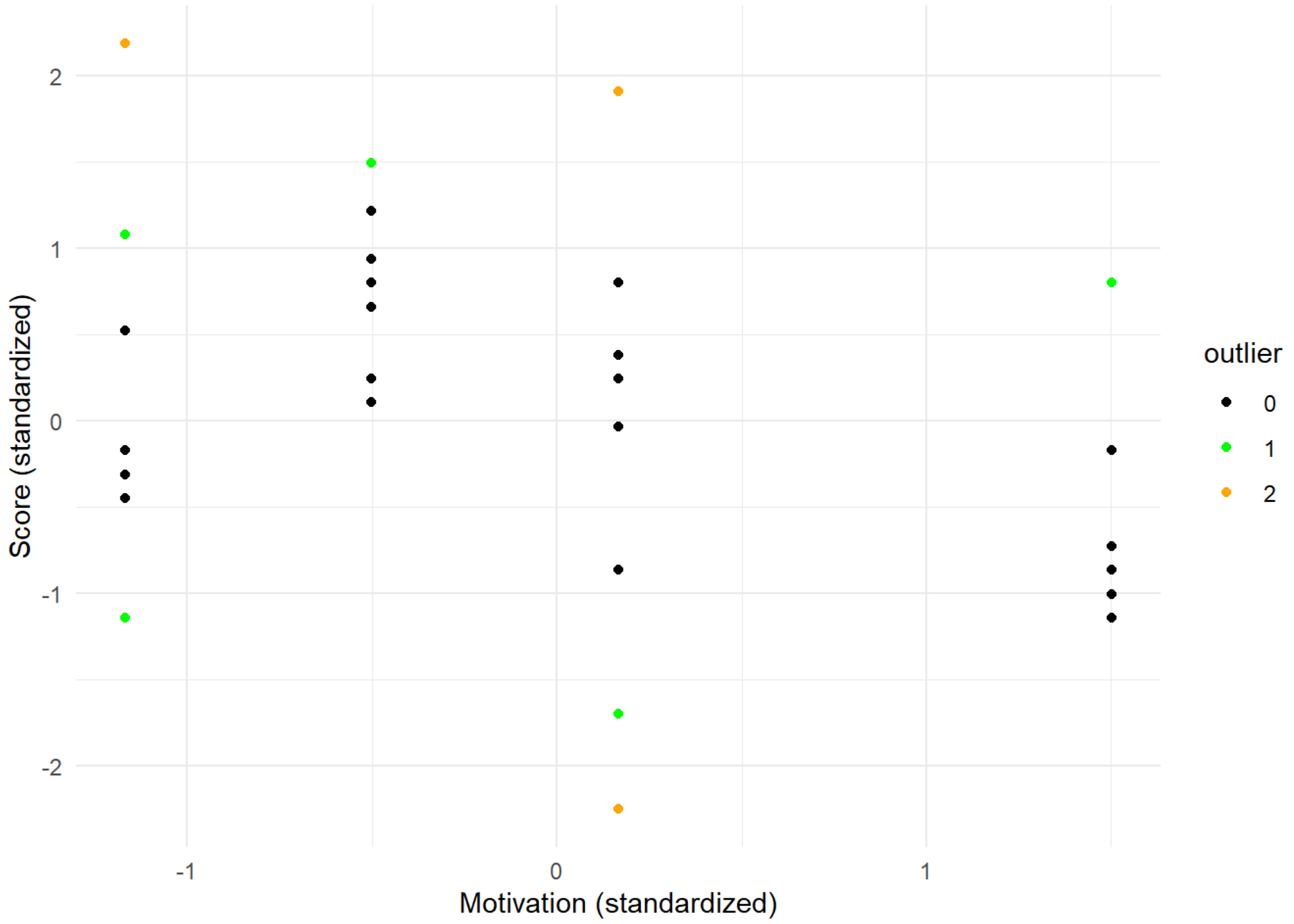
```
##         PSIS       lppd    penalty  std_err          k
## 1   3.348199 -1.6740996 0.19312234 15.83132  0.3019114
## 2   4.120033 -2.0600167 0.31201883 15.83132  0.5509001
## 3   1.665927 -0.8329635 0.02464702 15.83132  0.3296279
## 4  10.985977 -5.4929886 1.77458969 15.83132  1.0604277
## 5   2.010766 -1.0053830 0.05665834 15.83132  0.3403047
## 6   1.805343 -0.9026717 0.03733315 15.83132  0.3236438
## 7   4.120033 -2.0600167 0.31201883 15.83132  0.5509001
## 8   2.010766 -1.0053830 0.05665834 15.83132  0.3403047
## 9   1.932839 -0.9664197 0.04451048 15.83132 -0.0224463
## 10  1.932839 -0.9664197 0.04451048 15.83132 -0.0224463
## 11  1.573328 -0.7866639 0.01759750 15.83132  0.1137148
## 12  1.636803 -0.8184017 0.02180068 15.83132  0.1826908
## 13  1.636803 -0.8184017 0.02180068 15.83132  0.1826908
## 14  2.529502 -1.2647510 0.09056159 15.83132  0.2616447
## 15  1.948204 -0.9741021 0.04337020 15.83132  0.2274368
## 16  1.573328 -0.7866639 0.01759750 15.83132  0.1137148
## 17  2.202024 -1.1010122 0.06305797 15.83132  0.2835409
## 18  1.953793 -0.9768964 0.04422888 15.83132  0.1180806
## 19  1.572377 -0.7861887 0.01739921 15.83132  0.2042313
## 20  2.202024 -1.1010122 0.06305797 15.83132  0.2835409
## 21  2.632978 -1.3164891 0.11193574 15.83132  0.2212944
## 22  2.848291 -1.4241455 0.13415296 15.83132  0.4017957
## 23 11.414518 -5.7072592 1.84338481 15.83132  0.9367735
## 24  1.734389 -0.8671943 0.02809994 15.83132  0.1695657
## 25  6.410359 -3.2051793 0.66793945 15.83132  0.7012296
## 26  1.911771 -0.9558855 0.04301096 15.83132  0.2221699
## 27  9.784468 -4.8922339 1.53252243 15.83132  1.1008131
## 28  1.572227 -0.7861135 0.01502485 15.83132  0.1774367
## 29  1.911771 -0.9558855 0.04301096 15.83132  0.2221699
## 30  1.911771 -0.9558855 0.04301096 15.83132  0.2221699
## 31  1.867635 -0.9338175 0.04158665 15.83132  0.1227378
## 32  1.608570 -0.8042848 0.01854359 15.83132  0.1193239
## 33  5.748140 -2.8740701 0.53607983 15.83132  0.6714781
## 34  1.867635 -0.9338175 0.04158665 15.83132  0.1227378
## 35  1.576288 -0.7881438 0.01594062 15.83132  0.1134536
## 36  1.867635 -0.9338175 0.04158665 15.83132  0.1227378
## 37  1.608570 -0.8042848 0.01854359 15.83132  0.1193239
## 38  1.705285 -0.8526425 0.02682874 15.83132  0.1322557
## 39  1.705285 -0.8526425 0.02682874 15.83132  0.1322557
## 40  2.088625 -1.0443123 0.06036727 15.83132  0.2592917
```

Each model has identified different points as outliers. We will flag all points over 0.50 across models and highlight these in a plot.

```
d2$outlier_lm <- ifelse(PSIS_lm$k > 0.5, 1, 0)
d2$outlier_qm <- ifelse(PSIS_qm$k > 0.5, 1, 0)
d2$outlier_sm <- ifelse(PSIS_sm$k > 0.5, 1, 0)
d2$outlier <- rowSums(d2[, c("outlier_lm", "outlier_qm", "outlier_sm")])
d2$outlier <- d2$outlier |> as.factor()
```

Plot raw standardized data, and color code by how many models a point has been identified as an outlier in (0-3).

```
ggplot(d2, aes(x=M, y=S, color=outlier)) +
  geom_point() +
  labs(x = "Motivation (standardized)", y = "Score (standardized)") +
  scale_color_manual(values = c("0"="black", "1"="green", "2"="orange", "3"="red")) +
  theme_minimal()
```



This helps visualize which points in the data are most contributing to overfitting.

Information criterion estimates the out-of-sample deviance by considering the complexity of the model in a penalty term (in AIC, that is the number of parameters). In WAIC, the penalty term is defined by the variance in log-probabilities of each observation, given the model.

```
compare(lm, qm, sm, func=WAIC)
```

```
##        WAIC        SE     dWAIC      dSE    pWAIC    weight
## qm 111.4046 11.433730 0.0000000       NA 3.527496 0.4329670
## lm 112.0747  9.750818 0.6701453 2.761222 2.665950 0.3096953
## sm 112.4451 13.748585 1.0405453 4.139801 6.465284 0.2573376
```

The models are not reliably different in terms of predictive accuracy, using the WAIC. The PSIS and WAIC results converge, suggesting these results are reliable.

In conclusion, of the 3 models compared, the quadratic model *might* have the best predictive accuracy, but if so, not by much. Thus, there is no support for the inverted-U shape of Performance ~ Motivation, as the quadratic model does not reliably outperform the linear model.

Question 3

Explain Bayes Rule.

Bayes theorem: P(A|B) = P(B|A) * P(A) / P(B)

In the context of Bayesian statistics, Bayes rule provides the logical way to update a model's probability distributions (i.e., your initial hypothesis) based on new evidence. The model initially has a prior probability distribution which, represents your initial belief about the world (a hypothesis; P(H)). The upon observing new data, you update your model accordingly.

The posterior distribution, P(A|B), is the product of the likelihood of the data given the prior distribution, and the prior distribution, over the average probability of the data.

What does McElreath mean when he distinguishes between small-world and large-world?

My interpretation of what McElreath means when he distinguishes between small-world and large-world is analagous to the map versus the territory. The territory / large-world refers to nature, in all her complexity, down to the last subatomic particle. The map / small-world refers to models of nature, which we use to simplify a particular set of phenomena, facilitating comprehension and prediction, and thus informing decision making.

Give an example of the distinction between small-world and large-world based on your area of research interest.

An example of this can be found in psychiatric diagnosis and nosology. The phenomena we call 'mental disorders' refers to a wide and heterogeneous range and collection of experiences and behaviors that revolve around subjective suffering (and many times associated with physiological degradation or neurological dysfunction). These collections of experiences can have complex and diverse phenomenologies, behavioral manifestations, causes, risk factors, severities, interpretations, and timescales. Yet, to facilitate understanding through research, and ultimately treatment, we deploy a simplified diagnostic system to categorize these experiences. The reality of individual suffering and dysfunction constitute the large-world of mental disorders, and the constructed classification system of the DSM constitute the small-world.