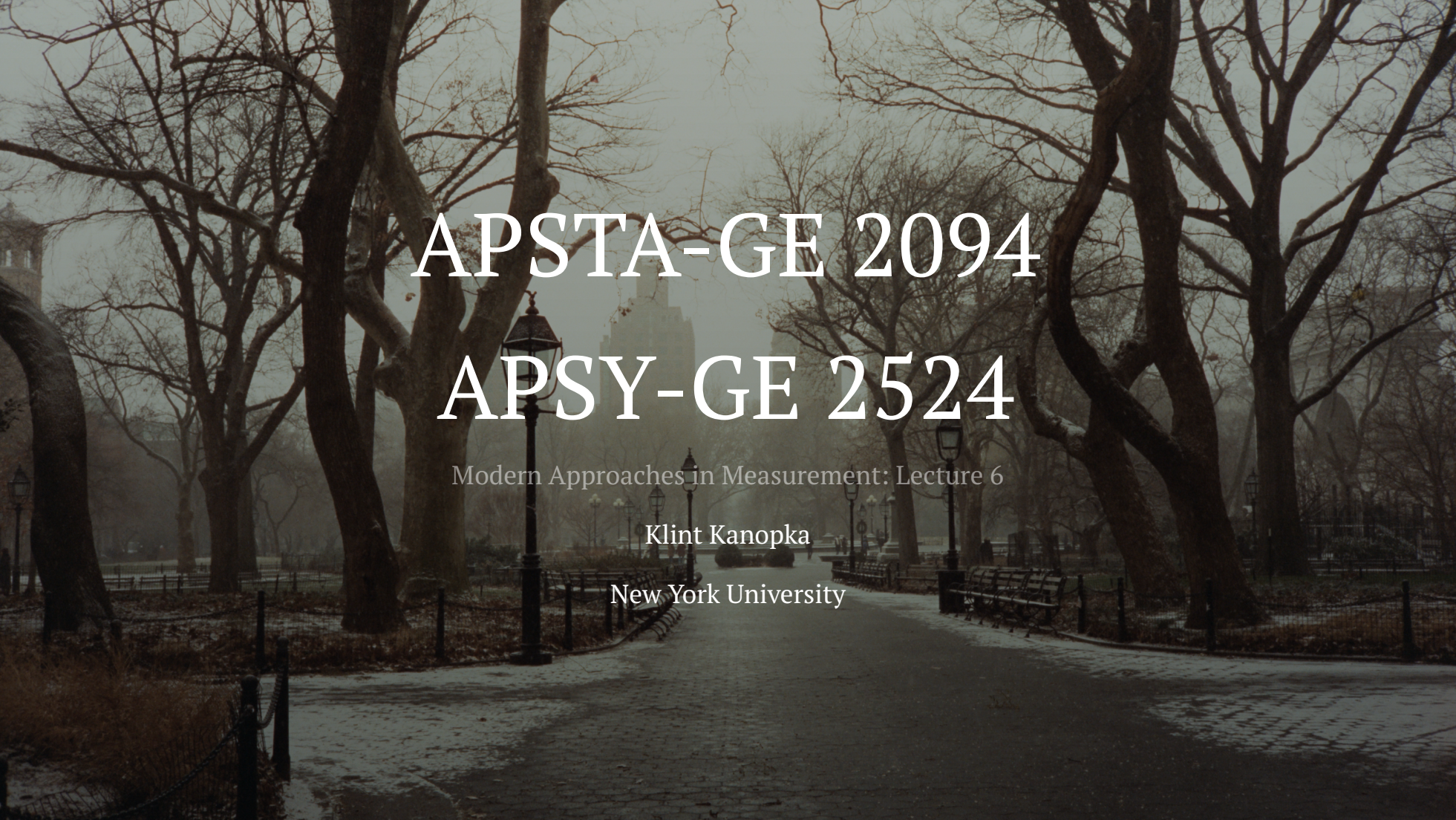


APSTA-GE 2094
APSY-GE 2524

Modern Approaches in Measurement: Lecture 6

Klint Kanopka

New York University

Check-In

- PollEv.com/klintkanopka

Table of Contents

Announcements

- Old slide `.pdf` links are currently broken, since I reorganized the github repo I host my slides from
- PS2 is due tomorrow (2.13 @ 11.59p)
- PS0 and PS1 grades are posted
  - These were good
  - If I made mistakes, submit a regrade request through Gradescope
- PS0 and PS1 solutions are posted
  - These are available on the Problem Sets tab on Brightspace
- PS3 is posted (due 3.20 @ 11.59p)
- PS4 will be due 4.3 @ 11.59p

Modeling Response Processes

Response Processes

- The "response process" is the cognitive and physical process that individuals go through when they respond to an item
  - Physical processes include pushing buttons, filling in bubbles, sorting cards, etc.
  - Cognitive processes include understanding a question, understanding the response options, and going through a mental decision tree, etc.
- Think *metacognitively* about your own response process as you answer these questions:

Response Processes

- The "response process" is the cognitive and physical process that individuals go through when they respond to an item
  - Physical processes include pushing buttons, filling in bubbles, sorting cards, etc.
  - Cognitive processes include understanding a question, understanding the response options, and going through a mental decision tree, etc.
- Think *metacognitively* about your own response process as you answer these questions:

1. I am a hard worker.

Response Processes

- The "response process" is the cognitive and physical process that individuals go through when they respond to an item
  - Physical processes include pushing buttons, filling in bubbles, sorting cards, etc.
  - Cognitive processes include understanding a question, understanding the response options, and going through a mental decision tree, etc.
- Think *metacognitively* about your own response process as you answer these questions:

1. I am a hard worker.

- (*Strongly Disagree, Disagree, Agree, Strongly Agree*)

Response Processes

- The "response process" is the cognitive and physical process that individuals go through when they respond to an item
  - Physical processes include pushing buttons, filling in bubbles, sorting cards, etc.
  - Cognitive processes include understanding a question, understanding the response options, and going through a mental decision tree, etc.
- Think *metacognitively* about your own response process as you answer these questions:

1. I am a hard worker.

- (*Strongly Disagree, Disagree, Agree, Strongly Agree*)

2. $8 \div 2(1 + 3) =$

Response Processes

- The "response process" is the cognitive and physical process that individuals go through when they respond to an item
  - Physical processes include pushing buttons, filling in bubbles, sorting cards, etc.
  - Cognitive processes include understanding a question, understanding the response options, and going through a mental decision tree, etc.
- Think *metacognitively* about your own response process as you answer these questions:

1. I am a hard worker.

- (*Strongly Disagree, Disagree, Agree, Strongly Agree*)

2. $8 \div 2(1 + 3) =$

- ($A: 1,\ B: 4,\ C: 8,\ D: 16$)

Response Processes

- The "response process" is the cognitive and physical process that individuals go through when they respond to an item
  - Physical processes include pushing buttons, filling in bubbles, sorting cards, etc.
  - Cognitive processes include understanding a question, understanding the response options, and going through a mental decision tree, etc.
- Think *metacognitively* about your own response process as you answer these questions:

1. I am a hard worker.

- (*Strongly Disagree, Disagree, Agree, Strongly Agree*)

2. $8 \div 2(1+3) =$

- ($A: 1,\ B: 4,\ C: 8,\ D: 16$)

3. What is your gender?

Response Processes

- The "response process" is the cognitive and physical process that individuals go through when they respond to an item
  - Physical processes include pushing buttons, filling in bubbles, sorting cards, etc.
  - Cognitive processes include understanding a question, understanding the response options, and going through a mental decision tree, etc.
- Think *metacognitively* about your own response process as you answer these questions:

1. I am a hard worker.

- (*Strongly Disagree, Disagree, Agree, Strongly Agree*)

2. $8 \div 2(1+3) =$

- ($A: 1,\ B: 4,\ C: 8,\ D: 16$)

3. What is your gender?

- (*Male, Female*)

IRTrees

- IRTrees is the idea that we write out a decision tree associated with the expected response process and use IRT to model each decision point
- Really flexible approach!
  - Leverages IRT's resilience to missing data
  - Can incorporate multiple latent traits!
  - Each node can use any type of IRT model

IRTrees

- Let’s build decision trees for each of these:
  - I am a hard worker. (*Strongly Disagree, Disagree, Agree, Strongly Agree*)
  - What is your gender? (*Male, Female*)

IRTrees

- Let's build decision trees for each of these:
  - I am a hard worker. (*Strongly Disagree, Disagree, Agree, Strongly Agree*)
  - What is your gender? (*Male, Female*)
- Let's write down the data structure
  - Follow an individual's response through the decision tree
  - Consider each node as its own item
  - If a respondent doesn't reach a node, mark it as `NA`

IRTrees

- Let's build decision trees for each of these:
  - I am a hard worker. (*Strongly Disagree, Disagree, Agree, Strongly Agree*)
  - What is your gender? (*Male, Female*)
- Let's write down the data structure
  - Follow an individual's response through the decision tree
  - Consider each node as its own item
  - If a respondent doesn't reach a node, mark it as `NA`
- You can fit these models in `mirt` with clever data recoding or using the `irtrees` package
- You'll try both in PS3

Fitting an IRTree model in `mirt`

- Let's fit an IRTree model to an empathizing–systemizing instrument
- Each respondent sees 120 statements that they rate on the following scale:
  - (1) strongly disagree
  - (2) slightly disagree
  - (3) slightly agree
  - (4) strongly agree
- Download the data here (redistributed from the Item Response Warehouse)

```
d <- read_csv('empathizing_systemizing.csv')

resp <- d |>
  pivot_wider(
    id_cols = id,
    names_prefix = 'item_',
    names_from = item,
    values_from = resp
  ) |>
  select(-id)
```

Fitting an IRTree model in `mirt`

- We'll use the same decision tree we developed earlier that includes non-response
- We will model nonresponse, agreement, and extreme responses as separate latent traits
- We make responses for the nonresponse nodes first:

```
1
```

Fitting an IRTree model in `mirt`

- We'll use the same decision tree we developed earlier that includes non-response
- We will model nonresponse, agreement, and extreme responses as separate latent traits
- We make responses for the nonresponse nodes first:

```
1 tree_resp_node_1 <- resp
```

Fitting an IRTree model in `mirt`

- We'll use the same decision tree we developed earlier that includes non-response
- We will model nonresponse, agreement, and extreme responses as separate latent traits
- We make responses for the nonresponse nodes first:

```
tree_resp_node_1 <- resp
tree_resp_node_1[is.na(resp)] <- 0
```

Fitting an IRTree model in `mirt`

- We'll use the same decision tree we developed earlier that includes non-response
- We will model nonresponse, agreement, and extreme responses as separate latent traits
- We make responses for the nonresponse nodes first:

```
tree_resp_node_1 <- resp
tree_resp_node_1[is.na(resp)] <- 0
tree_resp_node_1[!is.na(resp)] <- 1
```

Fitting an IRTree model in `mirt`

- We'll use the same decision tree we developed earlier that includes non-response
- We will model nonresponse, agreement, and extreme responses as separate latent traits
- We make responses for the nonresponse nodes first:

```
tree_resp_node_1 <- resp
tree_resp_node_1[is.na(resp)] <- 0
tree_resp_node_1[!is.na(resp)] <- 1
colnames(tree_resp_node_1) <- paste0('F1_', colnames(tree_resp_node_1))
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
1
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
1  tree_resp_node_2 <- resp
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
tree_resp_node_2 <- resp
tree_resp_node_2[resp <= 2] <- 0
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
tree_resp_node_2 <- resp
tree_resp_node_2[resp <= 2] <- 0
tree_resp_node_2[resp >= 3] <- 1
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
tree_resp_node_2 <- resp
tree_resp_node_2[resp <= 2] <- 0
tree_resp_node_2[resp >= 3] <- 1
colnames(tree_resp_node_2) <- paste0('F2_', colnames(tree_resp_node_2))
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
1
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
1  tree_resp_node_3 <- resp
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
tree_resp_node_3 <- resp
tree_resp_node_3[resp %in% c(2, 3)] <- 0
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
tree_resp_node_3 <- resp
tree_resp_node_3[resp %in% c(2, 3)] <- 0
tree_resp_node_3[resp %in% c(1, 4)] <- 1
```

Fitting an IRTree model in `mirt`

- Next we construct responses for the agreement nodes:

```
tree_resp_node_3 <- resp
tree_resp_node_3[resp %in% c(2, 3)] <- 0
tree_resp_node_3[resp %in% c(1, 4)] <- 1
colnames(tree_resp_node_3) <- paste0('F3_', colnames(tree_resp_node_3))
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
1
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- ''
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = '
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND =
      AGREE = '
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND =
      AGREE =
      STRONG = '
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE =
      STRONG = '
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = '
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model()
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = , itemnames = )
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = )
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))

m <- mirt()
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))

m <- mirt(data = , model = , itemtype = )
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))

m <- mirt(data = tree_resp, model = , itemtype = )
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))

m <- mirt(data = tree_resp, model = tree_model, itemtype = )
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))

m <- mirt(data = tree_resp, model = tree_model, itemtype = 'Rasch')
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))

m <- mirt(data = tree_resp, model = tree_model, itemtype = 'Rasch')

thetas <- fscores(m)
```

Fitting an IRTree model in `mirt`

- Next we combine the responses, free up some memory, and build out the `mirt.model` object

```
tree_resp <- bind_cols(tree_resp_node_1, tree_resp_node_2, tree_resp_node_3)

rm('tree_resp_node_1', 'tree_resp_node_2', 'tree_resp_node_3')

s <- 'RESPOND = F1_item_1-F1_item_120
      AGREE = F2_item_1-F2_item_120
      STRONG = F3_item_1-F3_item_120'

tree_model <- mirt.model(input = s, itemnames = names(tree_resp))

m <- mirt(data = tree_resp, model = tree_model, itemtype = 'Rasch')

thetas <- fscores(m)

tree_pars <- coef(m, IRTpars = TRUE, simplify = TRUE)
```

Non-Monotonic Item Response Functions

Different Shapes

- Until now, we've dealt with *monotonoic* IRFs
  - Monotonic means probability of response is always increasing (or decreasing) with ability
  - More θ means more $P(X = 1|\theta)$

Different Shapes

- Until now, we've dealt with *monotonoic* IRFs
  - Monotonic means probability of response is always increasing (or decreasing) with ability
  - More θ means more $P(X = 1|\theta)$
- This isn't the only reasonable IRF, though!
  - What about a Goldilocks situation?
  - $P(X = 1|\theta)$ is maximized when θ is "just right"

Let’s buy jeans!

Buying Jeans

- We all have different body types and fit preferences!
- If the cut of the jeans are skinnier or baggier than you like, you're less likely to buy them $\big(P(X = 0)\big)$

Buying Jeans

- We all have different body types and fit preferences!
- If the cut of the jeans are skinnier or baggier than you like, you're less likely to buy them $\big(P(X = 0)\big)$
- Is there something we could build an item response function out of?

Ideal Point Models

- We start from a normal pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right]$$

Ideal Point Models

- We start from a normal pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right]$$

- Notice that the the mean of the distribution, μ, is a location parameter, which acts like a difficulty, b

Ideal Point Models

- We start from a normal pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma} \right)^2 \right]$$

- Notice that the the mean of the distribution, μ, is a location parameter, which acts like a difficulty, b
- We also see that the inverse of the standard deviation, $\frac{1}{\sigma}$, functions like a discrimination, a

Ideal Point Models

- We start from a normal pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right]$$

- Notice that the the mean of the distribution, μ, is a location parameter, which acts like a difficulty, b
- We also see that the inverse of the standard deviation, $\frac{1}{\sigma}$, functions like a discrimination, a
- The stuff up front is a normalization constant to make sure the pdf integrates to 1

Ideal Point Models

- We start from a normal pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

- Notice that the the mean of the distribution, μ, is a location parameter, which acts like a difficulty, b
- We also see that the inverse of the standard deviation, $\frac{1}{\sigma}$, functions like a discrimination, a
- The stuff up front is a normalization constant to make sure the pdf integrates to 1
- Let's write an example IRF:

Ideal Point Models

- We start from a normal pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

- Notice that the the mean of the distribution, μ, is a location parameter, which acts like a difficulty, b
- We also see that the inverse of the standard deviation, $\frac{1}{\sigma}$, functions like a discrimination, a
- The stuff up front is a normalization constant to make sure the pdf integrates to 1
- Let's write an example IRF:

$$P(X = 1|\theta) = c \cdot \exp\left[-a(\theta - b)^2\right]$$

Ideal Point Models

- We start from a normal pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

- Notice that the the mean of the distribution, μ, is a location parameter, which acts like a difficulty, b
- We also see that the inverse of the standard deviation, $\frac{1}{\sigma}$, functions like a discrimination, a
- The stuff up front is a normalization constant to make sure the pdf integrates to 1
- Let's write an example IRF:

$$P(X = 1|\theta) = c \cdot \exp\left[-a(\theta - b)^2\right]$$

- b is the location of the ideal point
- $a \in [0, \infty)$ is the discrimination
- $c \in [0, 1]$ is the upper bound on $P(X = 1)$

Unfolding Models

- Check out an unfolding IRF on Desmos
- In general, these are used for modeling preference data
- Often applied to Likert scale data, as well
- More generally called *unfolding models*, which is a reference to the theorized cognitive process that governs making choices
- In `mirt`, you can use `itemtype='ideal'` with dichotomous data
- `itemtype='ggum'` works with both dichotomous and polytomous data
  - `ggum` stands for "generalized graded unfolding model"

Competitions

A Problem (and a Break)

- Download the `squash-matches.rds` data here
- The `player_1` and `player_2` columns have an integer that indicates which player is in the match
- The `result` column tells you if `player_1` won the match
- Using this data, rank the players from strongest to weakest

Elo Models

- Used in chess and other competitive sports

Elo Models

- Used in chess and other competitive sports
- We think the probability of winning is related to the difference in two players' abilities

$$P(A \text{ defeats } B|\theta_A, \theta_B) = \frac{1}{1 + \exp\big(-k(\theta_A - \theta_B)\big)}$$

Elo Models

- Used in chess and other competitive sports
- We think the probability of winning is related to the difference in two players' abilities

$$P(A \text{ defeats } B|\theta_A, \theta_B) = \frac{1}{1 + \exp\big(-k(\theta_A - \theta_B)\big)}$$

- Here, k is just a scaling factor that is preselected to work with the desired θ scale

Elo Models

- Used in chess and other competitive sports
- We think the probability of winning is related to the difference in two players' abilities

$$P(A \text{ defeats } B|\theta_A, \theta_B) = \frac{1}{1 + \exp\big(-k(\theta_A - \theta_B)\big)}$$

- Here, k is just a scaling factor that is preselected to work with the desired θ scale
- This functional form allows for making updates to individual abilities after each competition like this:

$$\theta_A^{i+1} = \theta_A^i + k\big(X_{AB} - P(x_{AB} = 1)\big)$$

Elo Models

- Used in chess and other competitive sports
- We think the probability of winning is related to the difference in two players' abilities

$$P(A \text{ defeats } B|\theta_A, \theta_B) = \frac{1}{1 + \exp\big(-k(\theta_A - \theta_B)\big)}$$

- Here, k is just a scaling factor that is preselected to work with the desired θ scale
- This functional form allows for making updates to individual abilities after each competition like this:

$$\theta_A^{i+1} = \theta_A^i + k\big(X_{AB} - P(x_{AB} = 1)\big)$$

- Note that k sets the scale and adjustments to abilities are small if the win was a "sure thing" and huge if the win was an upset!

Back to Squash

- Turns out, `mirt` doesn't have an item type for Elo models
- We can, however, estimate this ourselves!
- The plan:
  - Write out a log likelihood function
  - Use `optim()` to find the solutions
  - Feel free to download my code here

```
1
```

Back to Squash

- Turns out, `mirt` doesn't have an item type for Elo models
- We can, however, estimate this ourselves!
- The plan:
  - Write out a log likelihood function
  - Use `optim()` to find the solutions
  - Feel free to download my code here

```
LogLikMatch <- function() {
}
```

Back to Squash

- Turns out, `mirt` doesn't have an item type for Elo models
- We can, however, estimate this ourselves!
- The plan:
  - Write out a log likelihood function
  - Use `optim()` to find the solutions
  - Feel free to download my code here

```
LogLikMatch <- function() {
  return(ll)
}
```

Back to Squash

- Turns out, `mirt` doesn't have an item type for Elo models
- We can, however, estimate this ourselves!
- The plan:
  - Write out a log likelihood function
  - Use `optim()` to find the solutions
  - Feel free to download my code here

```
LogLikMatch <- function(data, theta) {
  return(ll)
}
```

Back to Squash

- Turns out, `mirt` doesn't have an item type for Elo models
- We can, however, estimate this ourselves!
- The plan:
  - Write out a log likelihood function
  - Use `optim()` to find the solutions
  - Feel free to download my code here

```
LogLikMatch <- function(data, theta) {
  p <- plogis(theta[data$player_1] - theta[data$player_2])
  return(ll)
}
```

Back to Squash

- Turns out, `mirt` doesn't have an item type for Elo models
- We can, however, estimate this ourselves!
- The plan:
  - Write out a log likelihood function
  - Use `optim()` to find the solutions
  - Feel free to download my code here

```
LogLikMatch <- function(data, theta) {
  p <- plogis(theta[data$player_1] - theta[data$player_2])
  ll <- sum(data$result * log(p) + (1 - data$result) * log(1 - p))
  return(ll)
}
```

Back to Squash

- Turns out, `mirt` doesn't have an item type for Elo models
- We can, however, estimate this ourselves!
- The plan:
  - Write out a log likelihood function
  - Use `optim()` to find the solutions
  - Feel free to download my code here

```
LogLikMatch <- function(data, theta) {
  p <- plogis(theta[data$player_1] - theta[data$player_2])
  ll <- sum(data$result * log(p) + (1 - data$result) * log(1 - p))
  return(ll)
}
```

Estimating the Model

- Using a single call to `optim()`, we can estimate all of our model parameters:

```
sol <- optim(
  starting_theta,
  LogLikMatch,
  data = d,
  method = 'L-BFGS-B',
  lower = -3,
  upper = 3,
  control = list(fnscale = -1, maxit = 1e6)
)
```

Build an Output Object

```
1
```

Build an Output Object

```
tmp <- d |>
  mutate(win = if_else(result == 1, player_1, player_2)) |>
  mutate(loss = if_else(result == 0, player_1, player_2))
```

Build an Output Object

```
tmp <- d |>
  mutate(win = if_else(result == 1, player_1, player_2)) |>
  mutate(loss = if_else(result == 0, player_1, player_2))

wins <- tmp |>
  count(win) |>
  select(player = win, wins = n)
```

Build an Output Object

```
tmp <- d |>
  mutate(win = if_else(result == 1, player_1, player_2)) |>
  mutate(loss = if_else(result == 0, player_1, player_2))

wins <- tmp |>
  count(win) |>
  select(player = win, wins = n)

losses <- tmp |>
  count(loss) |>
  select(player = loss, losses = n)
```

Build an Output Object

```
tmp <- d |>
  mutate(win = if_else(result == 1, player_1, player_2)) |>
  mutate(loss = if_else(result == 0, player_1, player_2))

wins <- tmp |>
  count(win) |>
  select(player = win, wins = n)

losses <- tmp |>
  count(loss) |>
  select(player = loss, losses = n)

tot <- full_join(wins, losses, by = 'player') |>
  arrange(player)
```

Build an Output Object

```r
tmp <- d |>
  mutate(win = if_else(result == 1, player_1, player_2)) |>
  mutate(loss = if_else(result == 0, player_1, player_2))

wins <- tmp |>
  count(win) |>
  select(player = win, wins = n)

losses <- tmp |>
  count(loss) |>
  select(player = loss, losses = n)

tot <- full_join(wins, losses, by = 'player') |>
  arrange(player)

tot[is.na(tot)] <- 0
```

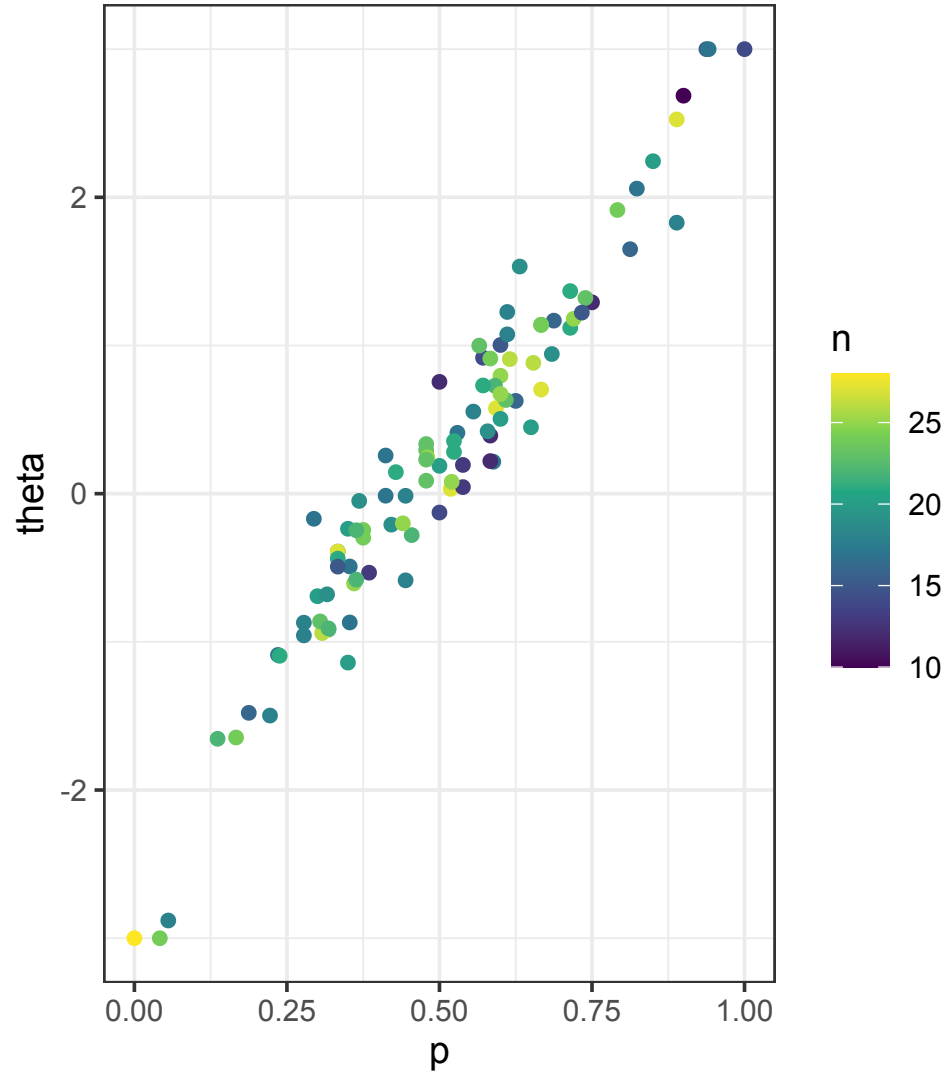
Build an Output Object

```r
tmp <- d |>
  mutate(win = if_else(result == 1, player_1, player_2)) |>
  mutate(loss = if_else(result == 0, player_1, player_2))

wins <- tmp |>
  count(win) |>
  select(player = win, wins = n)

losses <- tmp |>
  count(loss) |>
  select(player = loss, losses = n)

tot <- full_join(wins, losses, by = 'player') |>
  arrange(player)

tot[is.na(tot)] <- 0

tot <- tot |>
  mutate(p = wins / (wins + losses),
         theta = sol$par,
         n = wins + losses)
```

Visualizing the Solution

```
ggplot(tot, aes(x = p, y = theta, color = n))
  scale_color_viridis_c() +
  geom_point() +
  theme_bw()

cor(tot$p, tot$theta)
```

```
[1] 0.9686027
```



Wrap Up

Recap

- IRTrees are *incredibly* flexible and useful for way more than you'd expect
- Unfolding models can be useful for expressing preferences with the Goldilocks property
- Elo-like systems can be useful for head-to-head competition
- This was our last lecture on vanilla-ish IRT!
- Next week we'll take a detour into topic models
  - Or, as we civilized folk say, "factor analysis for words"
- After next week, we'll bust into *categorical* latent variables
  - We'll start with clustering, then get kinda weird
- PS3 is out!
  - Polytomous IRT, IRTrees, and a fun surprise!

Check-Out

- PollEv.com/klintkanopka