APSTA-GE 2094

APSY-GE 2524

Modern Approaches in Measurement: Lecture 3

Klint Kanopka

New York University

Table of Contents

Announcements

- PS1 is due next week (2.13 @ 11.59p)
- A few people joined the course late, so PS0 solutions and grades are delayed slightly

Check-In

- PollEv.com/klintkanopka

More PCA and Factor Analysis

What is PCA doing?

- PCA serves to rotate the coordinate axis of your data so that a smaller number of dimensions (variables) can be used to approximate your data
- The first dimension (called a *principal component*) is the single line that explains the most variance in your data
- You find this by minimizing the distance of each point to that line
- To find the next principal component, you subtract out the information captured from that line and find the next best principal component
- Think about this subtraction as squashing the data along your principal component

Simulating some data

```r
N <- 100
```

Simulating some data

```
N <- 100
beta_2 <- 1
```

Simulating some data

```r
N <- 100
beta_2 <- 1

x <- rnorm(N, mean=0, sd=3)
y <- beta_2*x + rnorm(N)
```

Simulating some data

```r
N <- 100
beta_2 <- 1

x <- rnorm(N, mean=0, sd=3)
y <- beta_2*x + rnorm(N)

d <- data.frame(x=x, y=y)
```

Simulating some data

```r
N <- 100
beta_2 <- 1

x <- rnorm(N, mean=0, sd=3)
y <- beta_2*x + rnorm(N)

d <- data.frame(x=x, y=y)

d$x_2 <- (x + beta_2 * y) / (beta_2^2 + 1)
d$y_2 <- beta_2 * (x + beta_2 * y) / (beta_2^2 + 1)
```

Simulating some data

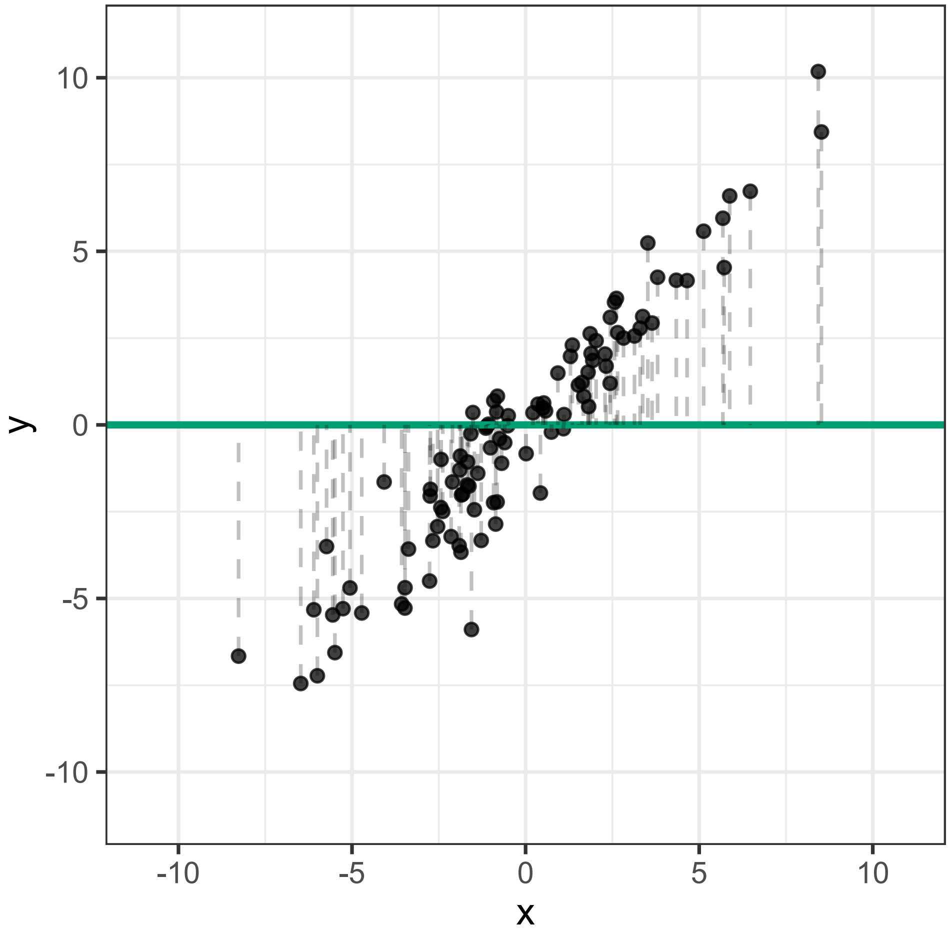
```r
N <- 100
beta_2 <- 1

x <- rnorm(N, mean=0, sd=3)
y <- beta_2*x + rnorm(N)

d <- data.frame(x=x, y=y)

d$x_2 <- (x + beta_2 * y) / (beta_2^2 + 1)
d$y_2 <- beta_2 * (x + beta_2 * y) / (beta_2^2 + 1)

beta_1 <- beta_2 / 2
```

Simulating some data

```
N <- 100
beta_2 <- 1

x <- rnorm(N, mean=0, sd=3)
y <- beta_2*x + rnorm(N)

d <- data.frame(x=x, y=y)

d$x_2 <- (x + beta_2 * y) / (beta_2^2 + 1)
d$y_2 <- beta_2 * (x + beta_2 * y) / (beta_2^2 + 1)

beta_1 <- beta_2 / 2

d$x_1 <- (x + beta_1 * y) / (beta_1^2 + 1)
d$y_1 <- beta_1 * (x + beta_1 * y) / (beta_1^2 + 1)
```

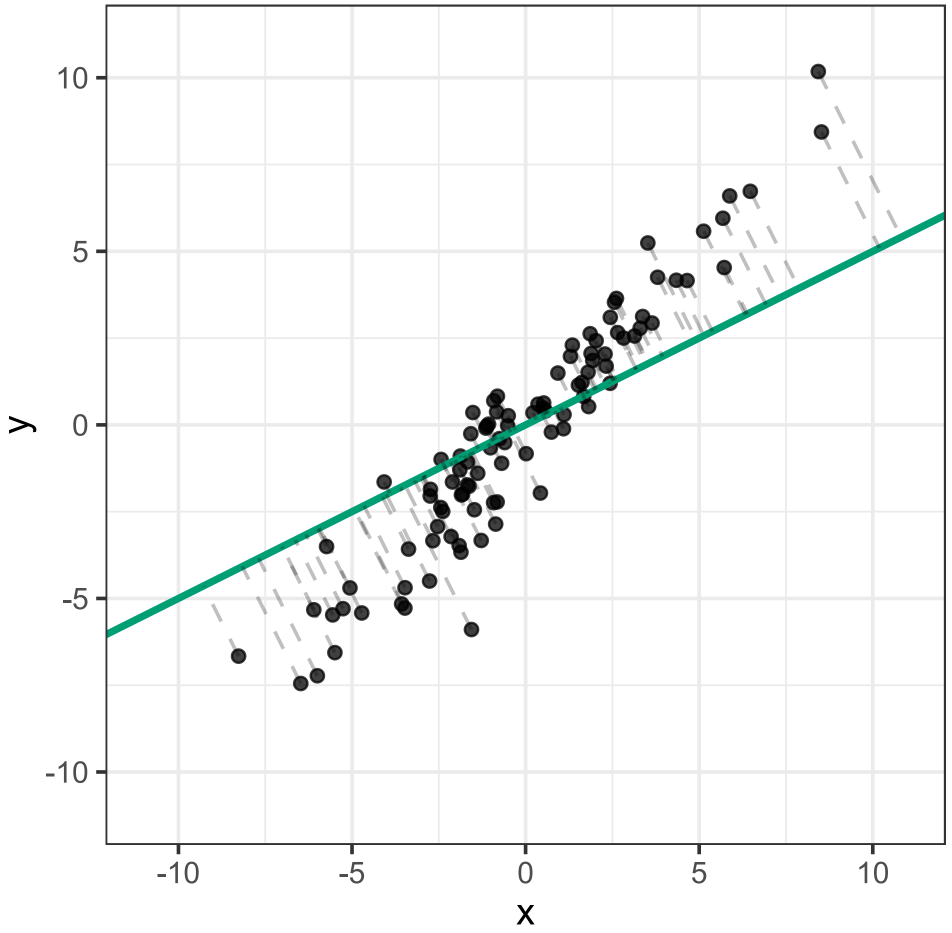
Unrotated x-axis

```r
ggplot(d, aes(x = x, y = y)) +
  geom_point(alpha = 0.75) +
  geom_hline(aes(yintercept = 0),
             linewidth = 1,
             color = okabeito_colors(3)) +
  geom_segment(
    aes(x = x,
        y = y,
        xend = x,
        yend = 0),
        linetype = 2,
        alpha = 0.25
  ) +
  scale_x_continuous(limits = c(-11, 11)) +
  scale_y_continuous(limits = c(-11, 11)) +
  coord_equal() +
  theme_bw()
```



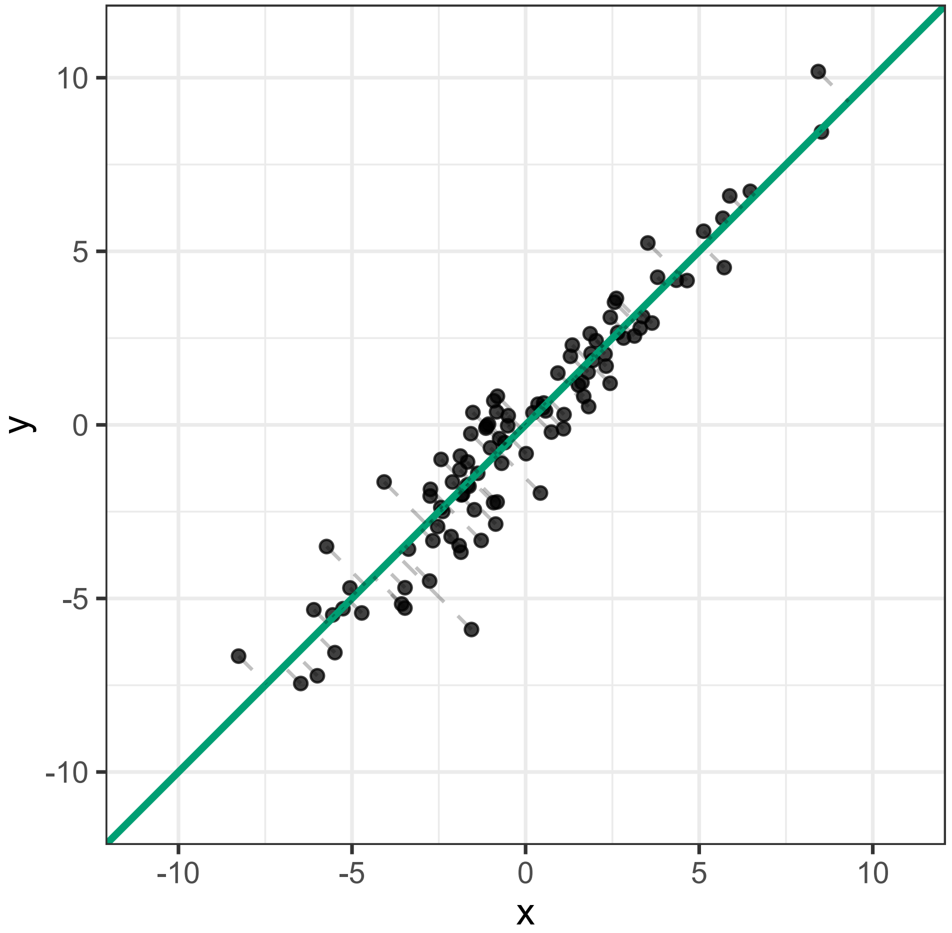
Partially rotated x-axis

```r
ggplot(d, aes(x = x, y = y)) +
  geom_point(alpha = 0.75) +
  geom_abline(
    aes(intercept = 0, slope = beta_1),
    linewidth = 1,
    color = okabeito_colors(3)
  ) +
  geom_segment(
    aes(x = x,
        y = y,
        xend = x_1,
        yend = y_1),
        linetype = 2,
        alpha = 0.25
  ) +
  scale_x_continuous(limits = c(-11, 11)) +
  scale_y_continuous(limits = c(-11, 11)) +
  coord_equal() +
  theme_bw()
```



Optimally rotated x-axis

```r
ggplot(d, aes(x = x, y = y)) +
  geom_point(alpha = 0.75) +
  geom_abline(
    aes(intercept = 0, slope = beta_2),
    linewidth = 1,
    color = okabeito_colors(3)
  ) +
  geom_segment(
    aes(x = x,
        y = y,
        xend = x_2,
        yend = y_2),
        linetype = 2,
        alpha = 0.25
  ) +
  scale_x_continuous(limits = c(-11, 11)) +
  scale_y_continuous(limits = c(-11, 11)) +
  coord_equal() +
  theme_bw()
```



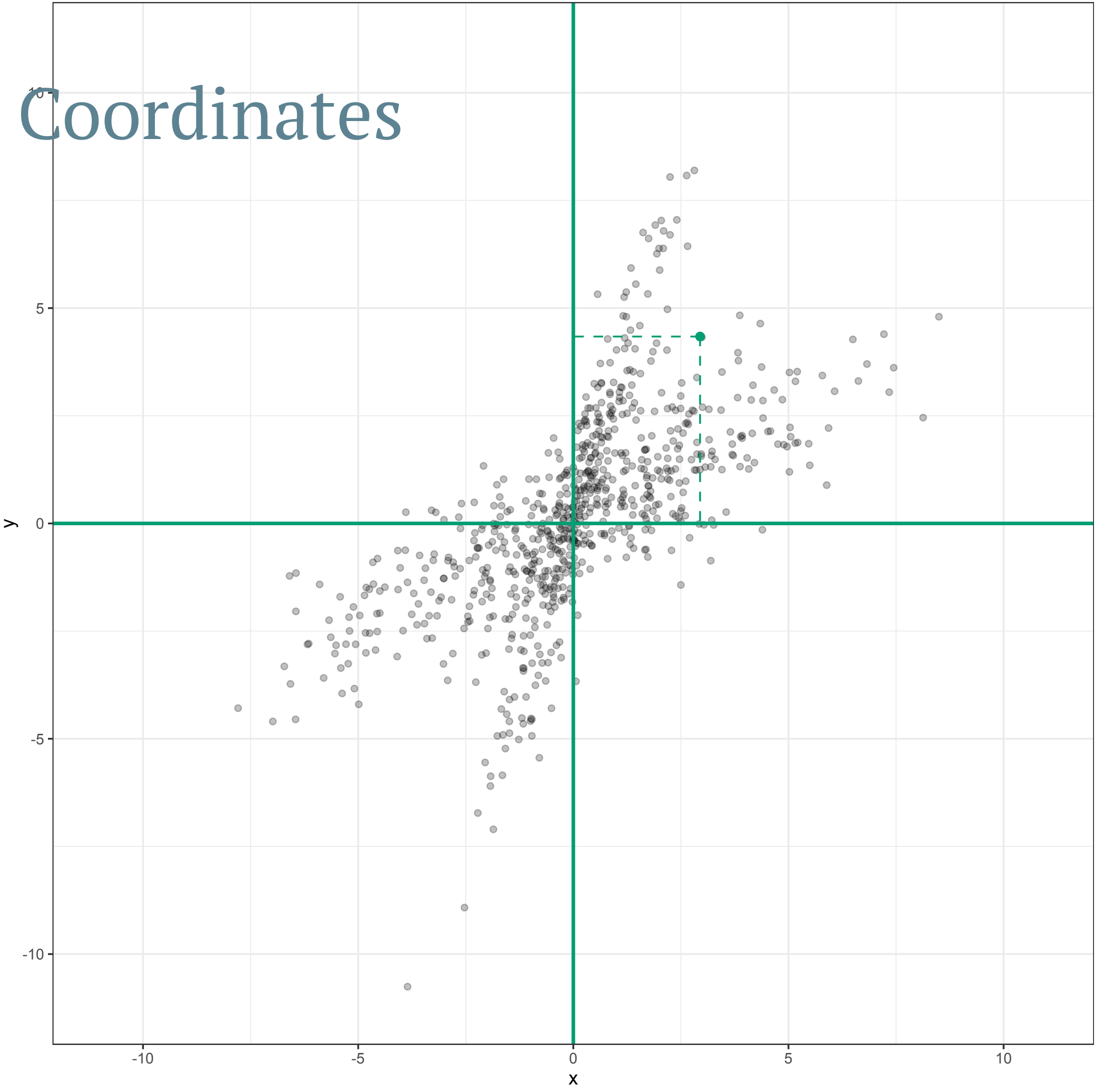
Why do we do this?

- When you project a point onto a principal component, you're creating a weighted sum of the variables you started with
- If the weights and patterns of those variables are meaningful, you can interpret what that principal component captures
- Additionally, the first principal components are more informative than any single variable in your original data
    - The *eigenvalues* tell you how much variance relative to one of the original variables the principal component captures
    - If the eigenvalue of the first principal component is $\lambda_1 = 10$, that single principal component explains $10\times$ the variance of one of your original variables
    - This is why we often use the criterion of keeping principal components with eigenvalues greater than one, as they're more informative than your unrotated variables

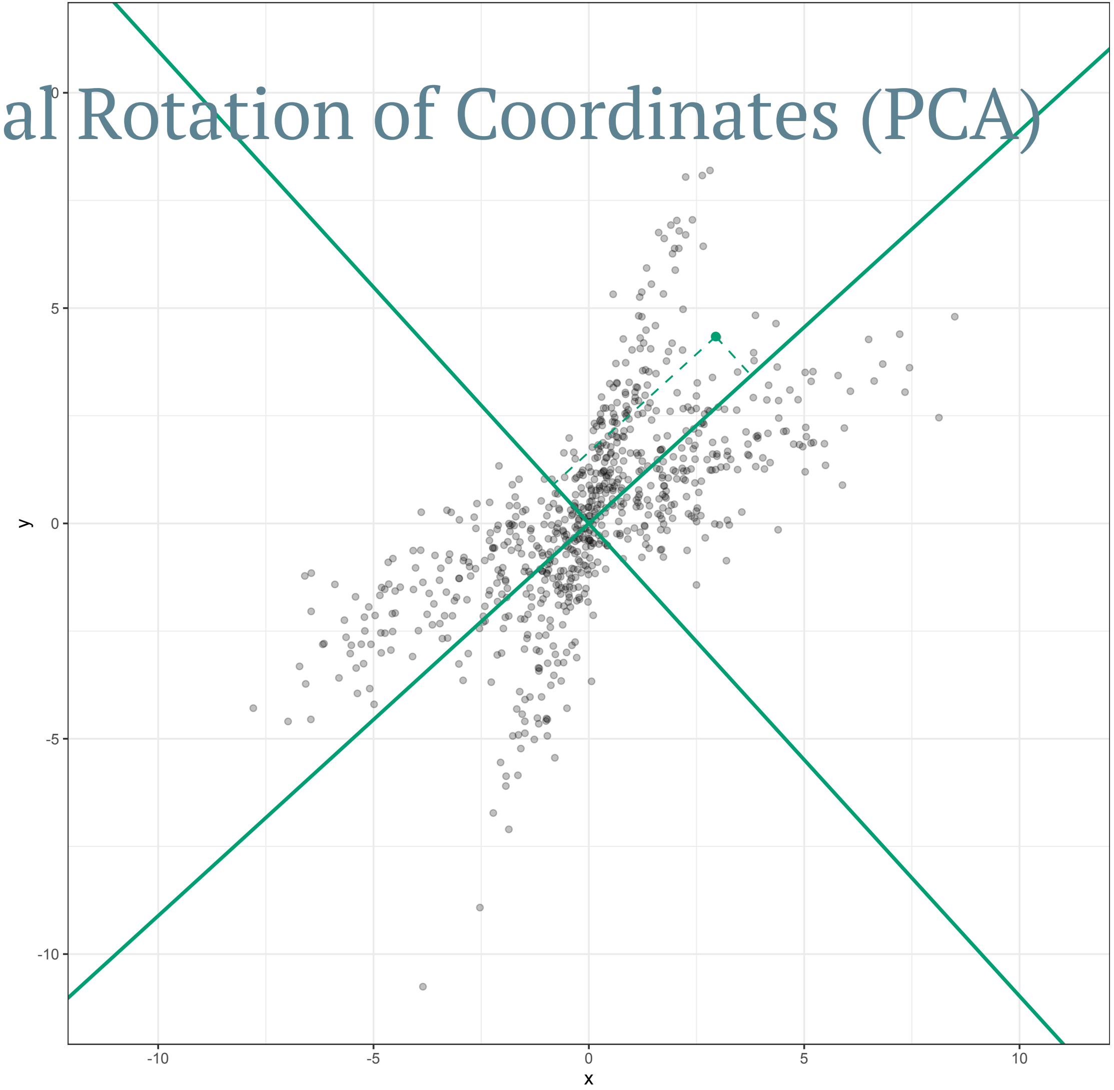
PCA vs. Factor Analysis

- The point of PCA is rotate your coordinates so that you can capture most of your data's information on a smaller number of variables
  - Sometimes we say we are *projecting* our data onto fewer dimensions
- Factor analysis does basically the same job, but in a *really* different way
  - A factor is like a dimension or a principal component, it's a new axis to project your higher dimensional data onto
  - Because of the types of available rotations, factors are not restricted to being orthogonal

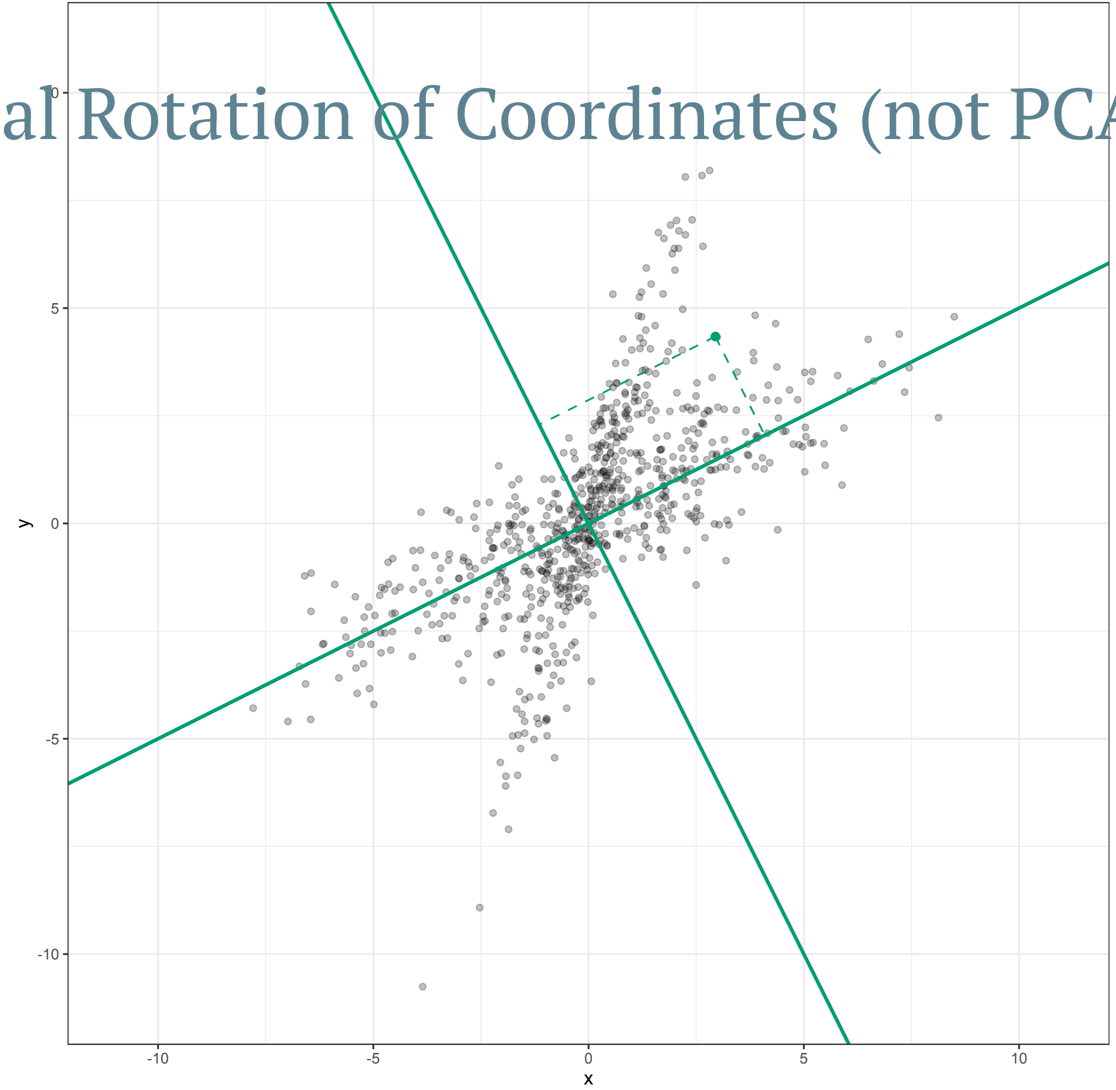
Unrotated Coordinates



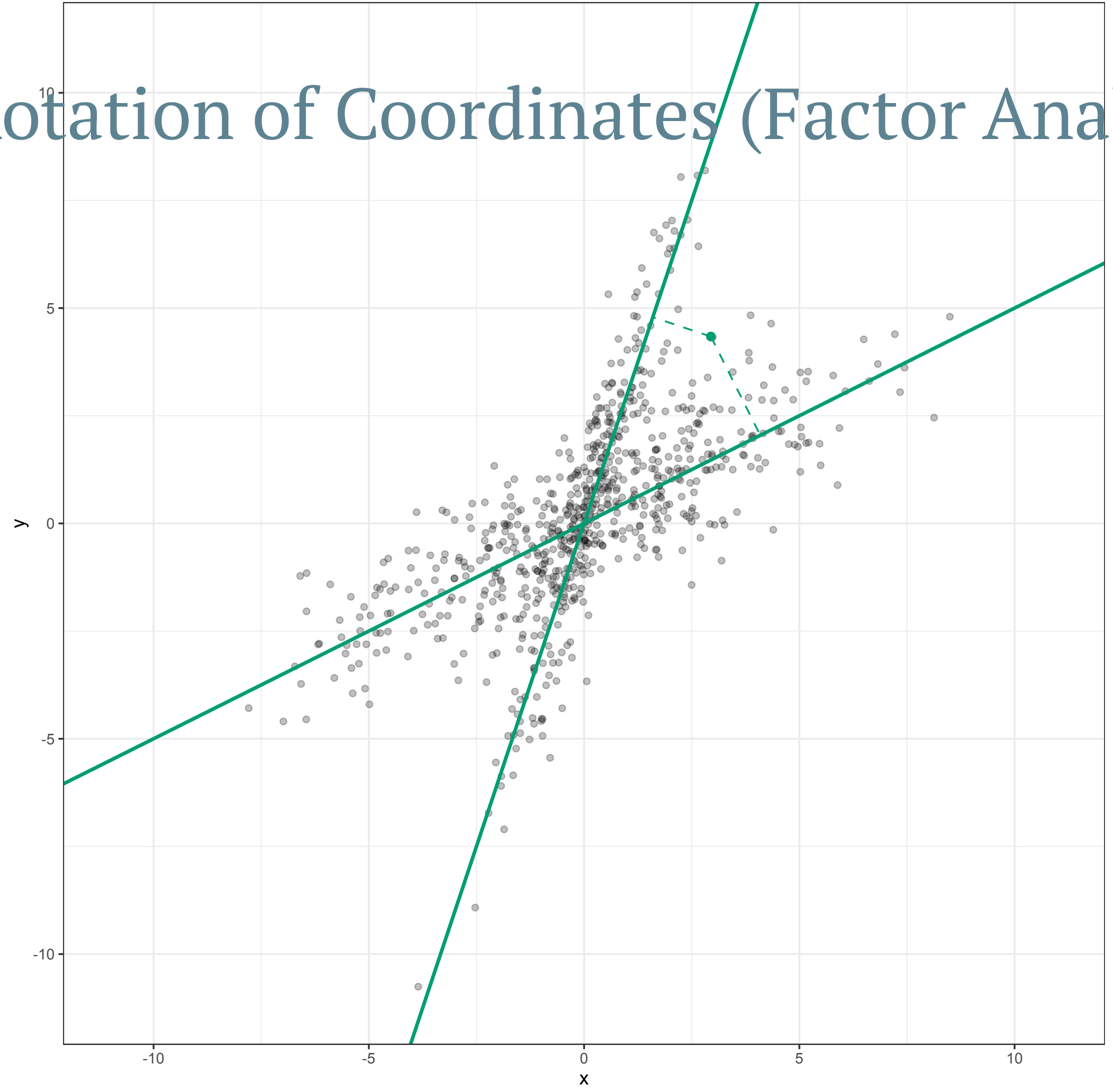
Orthogonal Rotation of Coordinates (PCA)



Orthogonal Rotation of Coordinates (not PCA)



Oblique Rotation of Coordinates (Factor Analysis)



What is Factor Analysis actually?

- Factor analysis lays out a model:

$$
\begin{aligned}
X_{1i} =& \lambda_{11}\theta_{1i} + \lambda_{12}\theta_{2i} + \cdots + \lambda_{1K}\theta_{Ki} + e_{1i} \\
X_{2i} =& \lambda_{21}\theta_{1i} + \lambda_{22}\theta_{2i} + \cdots + \lambda_{2K}\theta_{Ki} + e_{2i} \\
&\vdots \\
X_{Mi} =& \lambda_{m1}\theta_{1i} + \lambda_{m2}\theta_{2i} + \cdots + \lambda_{MK}\theta_{Ki} + e_{Mi}
\end{aligned}
$$

- X_{mi} is individual i's observed value of variable m
- λ_{mk} is the *loading* of factor k onto variable m
  - These is the weight of that variable on the dimension defined by that factor
- θ_{ki} is individual i's *factor score* on latent factor k
  - These are the projections of individuals onto the dimensions defined by the factors
- e_{mi} is an error term
- The only things you observe are the X_{mi}, so how do you even estimate this?

Estimation Methods

The joy of coin flipping

- I flip a coin fifteen times and get the results: $X = \{H, H, H, T, H, T, H, T, H, H, H, T, T, H, H\}$
  - What do you think about this result?

The joy of coin flipping

- I flip a coin fifteen times and get the results: $X = \{H, H, H, T, H, T, H, T, H, H, H, T, T, H, H\}$
  - What do you think about this result?
- What if I told you I'm a *prolific* cheater with an extensive collection of weighted coins?
  - Given this information, what do you think $P(H)$ is for the coin I'm flipping?
  - Do this now!

The joy of coin flipping

- I flip a coin fifteen times and get the results: $X = \{H, H, H, T, H, T, H, T, H, H, H, T, T, H, H\}$
  - What do you think about this result?
- What if I told you I'm a *prolific* cheater with an extensive collection of weighted coins?
  - Given this information, what do you think $P(H)$ is for the coin I'm flipping?
  - Do this now!
- How would you construct your *best* estimate of $P(H)$?
  - Maximum likelihood estimation (MLE) to the rescue!

Maximum Likelihood Estimation (MLE)

- Let's start by saying describing heads as an individual observation $x_i = 1$ and we seek to estimate the parameter, p, that captures this probability

Maximum Likelihood Estimation (MLE)

- Let's start by saying describing heads as an individual observation $x_i = 1$ and we seek to estimate the parameter, p, that captures this probability
- We need to estimate p from the data we have, X
  - Formally, we put hats on estimated quantities: $\hat{p}$

Maximum Likelihood Estimation (MLE)

- Let's start by saying describing heads as an individual observation $x_i = 1$ and we seek to estimate the parameter, p, that captures this probability
- We need to estimate p from the data we have, X
  - Formally, we put hats on estimated quantities: $\hat{p}$
- We first write down the *likelihood function*, $P(\theta|X)$
  - Often written $\mathcal{L}(\theta|X)$

Maximum Likelihood Estimation (MLE)

- For an individual coin flip:

$$\mathcal{L}(p|X = x) = p^x(1 - p)^{1-x}$$

  - When $x = 1$, we get p (the probability of flipping H)
  - When $x = 0$, we get $1 - p$ (the probability of flipping T)

Maximum Likelihood Estimation (MLE)

- For an individual coin flip:

$$\mathcal{L}(p|X=x) = p^x(1-p)^{1-x}$$

  - When $x = 1$, we get p (the probability of flipping H)
  - When $x = 0$, we get $1 - p$ (the probability of flipping T)
- For a bunch of coin flips $X = \{x_1, x_2, \ldots, x_n\}$:

$$\mathcal{L}(p|X) = \prod_{x_i \in X} p^{x_i}(1-p)^{1-x_i}$$

Maximizing the likelihood

- If we flip n heads and m tails, we can rewrite the likelihood:

$$\mathcal{L}(p|X) = p^n(1-p)^m$$

- To maximize, set $\frac{d\mathcal{L}}{dp} = 0$:

Maximizing the likelihood

- If we flip n heads and m tails, we can rewrite the likelihood:

$$\mathcal{L}(p|X) = p^n(1-p)^m$$

- To maximize, set $\frac{d\mathcal{L}}{dp} = 0$:

$$\frac{d\mathcal{L}}{dp} = np^{n-1}(1-p)^m - mp^n(1-p)^{m-1}$$

Maximizing the likelihood

- If we flip n heads and m tails, we can rewrite the likelihood:

$$\mathcal{L}(p|X) = p^n(1-p)^m$$

- To maximize, set $\frac{d\mathcal{L}}{dp} = 0$:

$$\frac{d\mathcal{L}}{dp} = np^{n-1}(1-p)^m - mp^n(1-p)^{m-1}$$

$$\frac{d\mathcal{L}}{dp} = p^{n-1}(1-p)^{m-1}\big(n(1-p) - mp\big)$$

Maximizing the likelihood

- If we flip n heads and m tails, we can rewrite the likelihood:

$$\mathcal{L}(p|X) = p^n(1-p)^m$$

- To maximize, set $\frac{d\mathcal{L}}{dp} = 0$:

$$\frac{d\mathcal{L}}{dp} = np^{n-1}(1-p)^m - mp^n(1-p)^{m-1}$$

$$\frac{d\mathcal{L}}{dp} = p^{n-1}(1-p)^{m-1}\big(n(1-p) - mp\big)$$

$$n(1-\hat{p}_{MLE}) - m\hat{p}_{MLE} = 0$$

Maximizing the likelihood

- If we flip n heads and m tails, we can rewrite the likelihood:

$$\mathcal{L}(p|X) = p^n(1-p)^m$$

- To maximize, set $\frac{d\mathcal{L}}{dp} = 0$:

$$\frac{d\mathcal{L}}{dp} = np^{n-1}(1-p)^m - mp^n(1-p)^{m-1}$$

$$\frac{d\mathcal{L}}{dp} = p^{n-1}(1-p)^{m-1}\big(n(1-p) - mp\big)$$

$$n\big(1-\hat{p}_{MLE}\big) - m\hat{p}_{MLE} = 0$$

$$n = \hat{p}_{MLE}(n+m)$$

Maximizing the likelihood

- If we flip n heads and m tails, we can rewrite the likelihood:

$$\mathcal{L}(p|X) = p^n(1-p)^m$$

- To maximize, set $\frac{d\mathcal{L}}{dp} = 0$:

$$\frac{d\mathcal{L}}{dp} = np^{n-1}(1-p)^m - mp^n(1-p)^{m-1}$$

$$\frac{d\mathcal{L}}{dp} = p^{n-1}(1-p)^{m-1}\big(n(1-p) - mp\big)$$

$$n(1-\hat{p}_{MLE}) - m\hat{p}_{MLE} = 0$$

$$n = \hat{p}_{MLE}(n+m)$$

$$\hat{p}_{MLE} = \frac{n}{n+m}$$

Circling back:

- The maximum likelihood estimator for the probability of success (heads) in a Bernoulli trial (coinflip) is:

$$\hat{p}_{MLE} = \frac{n}{n+m}$$

- For my original example this is:

$$\hat{p}_{MLE} = \frac{10}{10+5} \approx 0.667$$

- We are going to think about dichotomous responses to items as coin flips and then model the underlying probability!
- But first, a new challenger has appeared…

A new, scummier, question

- As a known cheater with weighted coins, I am going to pull *two* out, with unknown true probabilities p_A, p_B

A new, scummier, question

- As a known cheater with weighted coins, I am going to pull *two* out, with unknown true probabilities p_A, p_B
- I will select a coin at random and flip it ten times
  - I will only tell you the results of the ten flips, but **not** which coin produced them
  - I will, however, generate multiple sequences, selecting a coin at random each time
  - Each sequence of coin flips has a *latent* coin assignment, $\theta_i \in \{A, B\}$, that dictates what coin generated the data
  - We *never* get to observe this information. The θ_is are completely unrecoverable

A new, scummier, question

- As a known cheater with weighted coins, I am going to pull *two* out, with unknown true probabilities p_A, p_B
- I will select a coin at random and flip it ten times
  - I will only tell you the results of the ten flips, but **not** which coin produced them
  - I will, however, generate multiple sequences, selecting a coin at random each time
  - Each sequence of coin flips has a *latent* coin assignment, $\theta_i \in \{A, B\}$, that dictates what coin generated the data
  - We *never* get to observe this information. The θ_is are completely unrecoverable
- What is your best guess at the weights of these two coins?
  - Specifically, compute $\hat{p}_A, \hat{p}_B$
  - What could you try here?

Try something!

Here are data from five sets of ten coin flips:

1. $H, T, T, T, H, H, H, H, T, T$
2. $H, H, H, H, T, H, H, H, H, H$
3. $H, T, H, H, H, H, H, T, H, H$
4. $H, T, H, T, T, T, H, H, T, T$
5. $T, H, H, H, T, H, H, H, T, H$

Estimate $\hat{p}_A, \hat{p}_B$!

The Expectation-Maximization Algorithm

- This is a new type of missing data problem
  - We never get the latent coin assignments, $\theta_1, \cdots, \theta_5$
  - With these, the problem is *super* easy
- The strategy is to estimate both the parameters we care about, p_A, p_B, and the latent coin assignments, $\theta_1, \cdots, \theta_5$, *at the same time*
  - We do this using the Expectation-Maximization (EM) Algorithm
- Iterative algorithm
  - Treats the latent assignments as a *probability distribution*
  - Allows us to create a weighted set of data that represents all possible coin assignments (E-step)
  - Then we estimate our parameters $\hat{p}_A, \hat{p}_B$ from that (M-step)
  - Repeat until convergence!

The E-Step

- We start from some initial parameter estimates: $\hat{p}_A^{(0)} = 0.6, \hat{p}_B^{(0)} = 0.5$
- We compute the likelihood of observing each string of flips under each coin assignment

$$\mathcal{L}(p|X) = p^n(1-p)^m$$

- Normalize the likelihoods to find the distribution $P(\theta_i)$

$$P\left(\theta_i^{(1)} = A\right) = \frac{\mathcal{L}(\hat{p}_A^{(0)}, X)}{\mathcal{L}(\hat{p}_A^{(0)}, X) + \mathcal{L}(\hat{p}_B^{(0)}, X)}$$

The E-Step

H, T	$\mathcal{L}(\hat{p}_A^{(0)}, X)$	$\mathcal{L}(\hat{p}_B^{(0)}, X)$	$P\left(\theta_i^{(1)} = A\right)$	$P\left(\theta_i^{(1)} = B\right)$
$5, 5$	$0.6^5 \cdot 0.4^5$	$0.5^5 \cdot 0.5^5$	0.45	0.55
$9, 1$	$0.6^9 \cdot 0.4^1$	$0.5^9 \cdot 0.5^1$	0.80	0.20
$8, 2$	$0.6^8 \cdot 0.4^2$	$0.5^8 \cdot 0.5^2$	0.73	0.27
$4, 6$	$0.6^4 \cdot 0.4^6$	$0.5^4 \cdot 0.5^6$	0.35	0.65
$7, 3$	$0.6^7 \cdot 0.4^3$	$0.5^7 \cdot 0.5^3$	0.65	0.35

The M-Step

- We use the weights from the previous step to weight the observations and compute new parameter estimates: $\hat{p}_A^{(1)}, \hat{p}_B^{(1)}$

H, T	Coin A	Coin B
$5, 5$	$2.2H, 2.2T$	$2.8H, 2.8T$
$9, 1$	$7.2H, 0.8T$	$1.8H, 0.2T$
$8, 2$	$5.9H, 1.5T$	$2.1H, 0.5T$
$4, 6$	$1.4H, 2.1T$	$2.6H, 3.9T$
$7, 3$	$4.5H, 1.9T$	$2.5H, 1.1T$

$$\hat{p}_A^{(1)} \approx \frac{21.3}{21.3 + 8.6} \approx 0.71;\ \hat{p}_B^{(1)} \approx \frac{11.7}{11.7 + 8.4} \approx 0.58$$

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```r
# Results go here
```

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```
[1] 0.71 0.58
```

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```
[1] 0.71 0.58
[1] 0.76 0.48
```

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```
[1] 0.71 0.58
[1] 0.76 0.48
[1] 0.78 0.52
```

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```
[1] 0.71 0.58
[1] 0.76 0.48
[1] 0.78 0.52
[1] 0.79 0.50
```

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```
[1] 0.71 0.58
[1] 0.76 0.48
[1] 0.78 0.52
[1] 0.79 0.50
[1] 0.79 0.51
```

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```
[1] 0.71 0.58
[1] 0.76 0.48
[1] 0.78 0.52
[1] 0.79 0.50
[1] 0.79 0.51
[1] 0.80 0.51
```

Doing it in code

```r
flips <- matrix(c(5, 9, 8, 4, 7, 5, 1, 2, 6, 3),
                nrow=5, ncol=2, byrow=F)
p_new <- c(0.6, 0.5)
p_old <- c(0,0)

while(!identical(round(p_old,2), round(p_new,2))){
  p_old <- p_new
  p <- matrix(p_new, nrow=5, ncol=2, byrow=T)
  likelihood <- p^flips * (1-p)^(10-flips)
  theta <- likelihood / rowSums(likelihood)
  theta_A <- theta[,1]*flips
  theta_B <- theta[,2]*flips
  p_new <- c(sum(theta_A[,1])/sum(theta_A),
             sum(theta_B[,1])/sum(theta_B))
  print(round(p_new,2))
}
```

```
[1] 0.71 0.58
[1] 0.76 0.48
[1] 0.78 0.52
[1] 0.79 0.50
[1] 0.79 0.51
[1] 0.80 0.51
[1] 0.80 0.51
```

Break

Item Response Theory

Why not sum scores?

- Not all items are created equal!

Why not sum scores?

- Not all items are created equal!

$$7 \times 8 = ?$$

Why not sum scores?

- Not all items are created equal!

$$7 \times 8 = ?$$

$$9 - 3 \div \frac{1}{3} + 1 = ?$$

Why not sum scores?

- Not all items are created equal!

$$7 \times 8 = ?$$

$$9 - 3 \div \frac{1}{3} + 1 = ?$$

$$\int_0^{\pi} \sin x \, dx = ?$$

Item Response Theory

- At its core, a probabilistic model that describes how individuals generate responses to individual items
- Produces parameters that describe specific items *independent* of the sample of respondents used to calibrate them
- Estimates person ability and item difficulty on a common scale
- Produces ability estimates *independent* of the specific items on a test form
- Can produce interval scales (this is a Rasch thing)

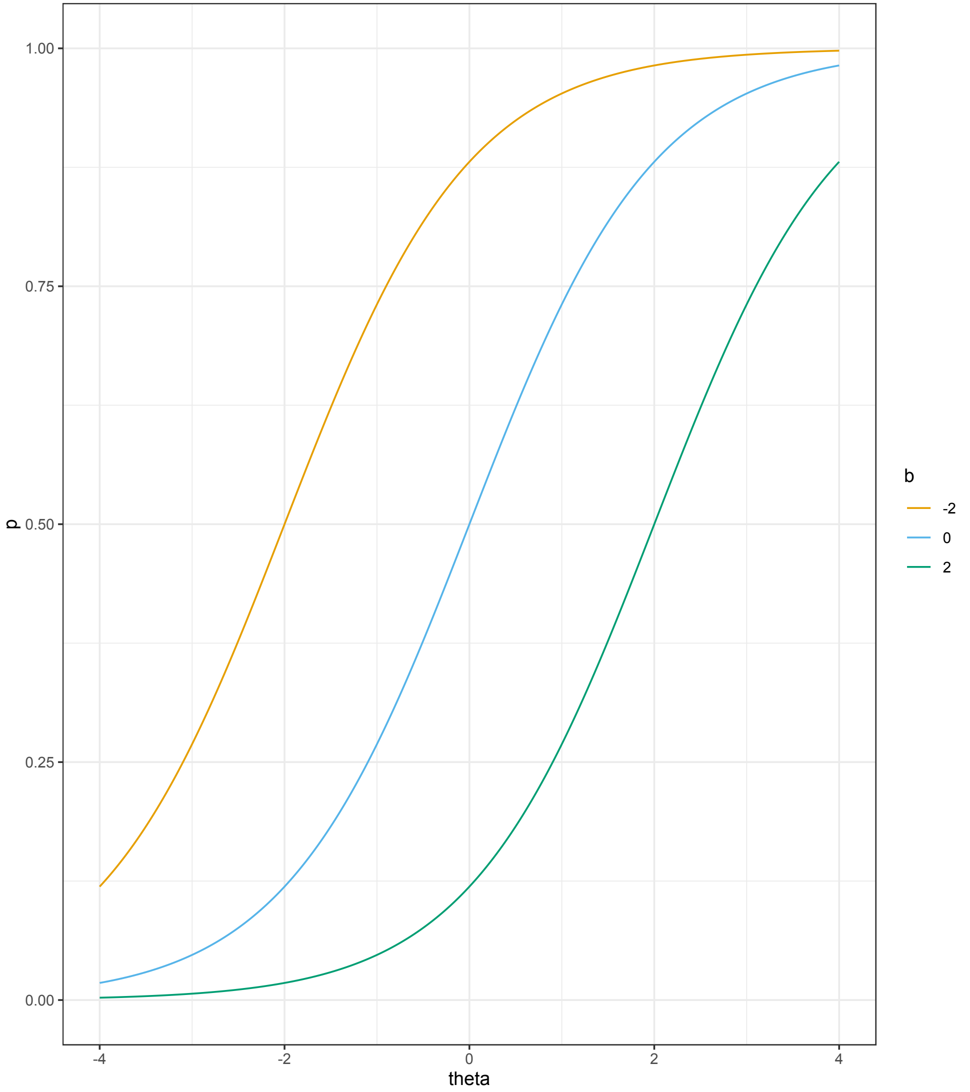
The most basic IRT model

$$P(X_{ij} = 1|\theta_i) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

- We refer to this as a *one parameter logistic model*, or 1PL
- This is the core model used in Rasch measurement
- Model parameters:
  - X_{ij} is individual i's response to item j
  - θ_i is individual i's latent ability
  - b_j is the difficulty of item j
- Note that the only thing the probability depends on is the *difference* $(\theta_i - b_j)$
  - When $\theta_i = b_j, P(X_{ij} = 1) = 0.5$
  - When $\theta_i < b_j, P(X_{ij} = 1) \in (0, 0.5)$
  - When $\theta_i > b_j, P(X_{ij} = 1) \in (0.5, 1)$

The 1PL item response function (IRF)

```r
data.frame(theta=seq(-4, 4,
                     length.out=1e4),
           b_1 = -2, b_2=0, b_3=2) |>
  pivot_longer(starts_with('b_'),
               names_to='param',
               values_to='b') |>
  select(-param) |>
  mutate(p = plogis(theta-b)) |>
  ggplot(aes(x=theta,
             y=p,
             color=as.character(b))) +
  geom_line() +
  labs(color='b') +
  scale_color_okabeito() +
  theme_bw()
```



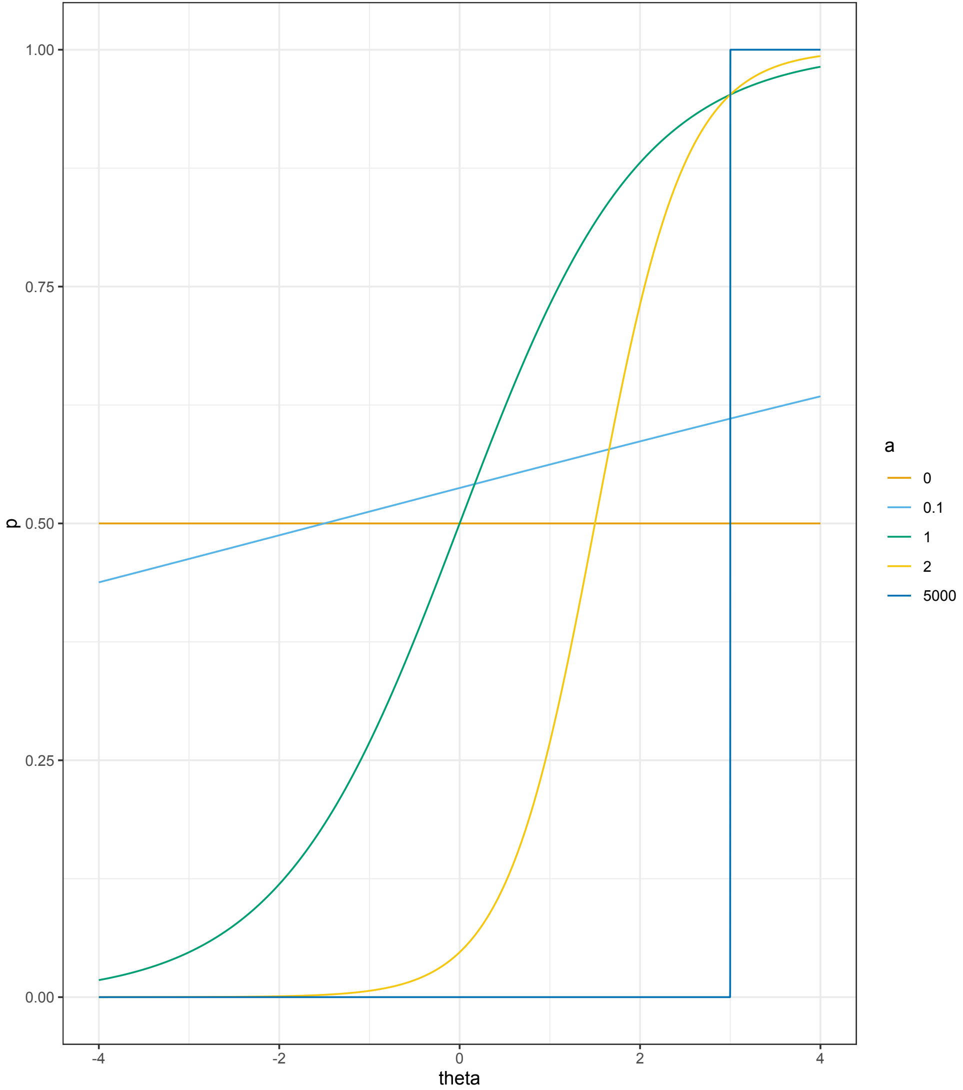
The two parameter logistic IRT model

$$P(X_{ij} = 1|\theta_i) = \frac{1}{1 + e^{-a_j(\theta_i - b_j)}}$$

- IRT models are named for the number of item parameters and the link function
- We add one parameter, a_j
- Nothing else changes!
- a_j is the *discrimination*
    - It controls the slope of the IRF, and steeper slopes are more discriminating
    - When $a_j = 0$, the IRF is flat and response probability does not depend on θ_i
    - As $a_j \to \infty$, the IRF becomes a vertical line and the item becomes *perfectly* discriminating
- In the IRT case, discrimination answers the question, "How well does this item separate respondents below its difficulty on the scale from respondents above its difficulty on the scale?"

The 2PL IRF

```r
data.frame(
  theta = seq(-4, 4, length.out = 1e4),
  b_1 = -1.5, b_2 = 0, b_3 = 1.5,
  b_4 = -3, b_5 = 3) |>
  pivot_longer(starts_with('b_'),
               names_to = 'param',
               values_to = 'b') |>
  select(-param) |>
  mutate(a = case_when(b == -3 ~ 0,
                       b == -1.5 ~ 0.1,
                       b == 0 ~ 1,
                       b == 1.5 ~ 2,
                       b == 3 ~ 5e3)) |>
  mutate(p = plogis(a * (theta - b))) |>
  ggplot(aes(x = theta,
             y = p,
             color = as.character(a))) +
  geom_line() +
  labs(color = 'a') +
  scale_color_okabeito() +
  theme_bw()
```



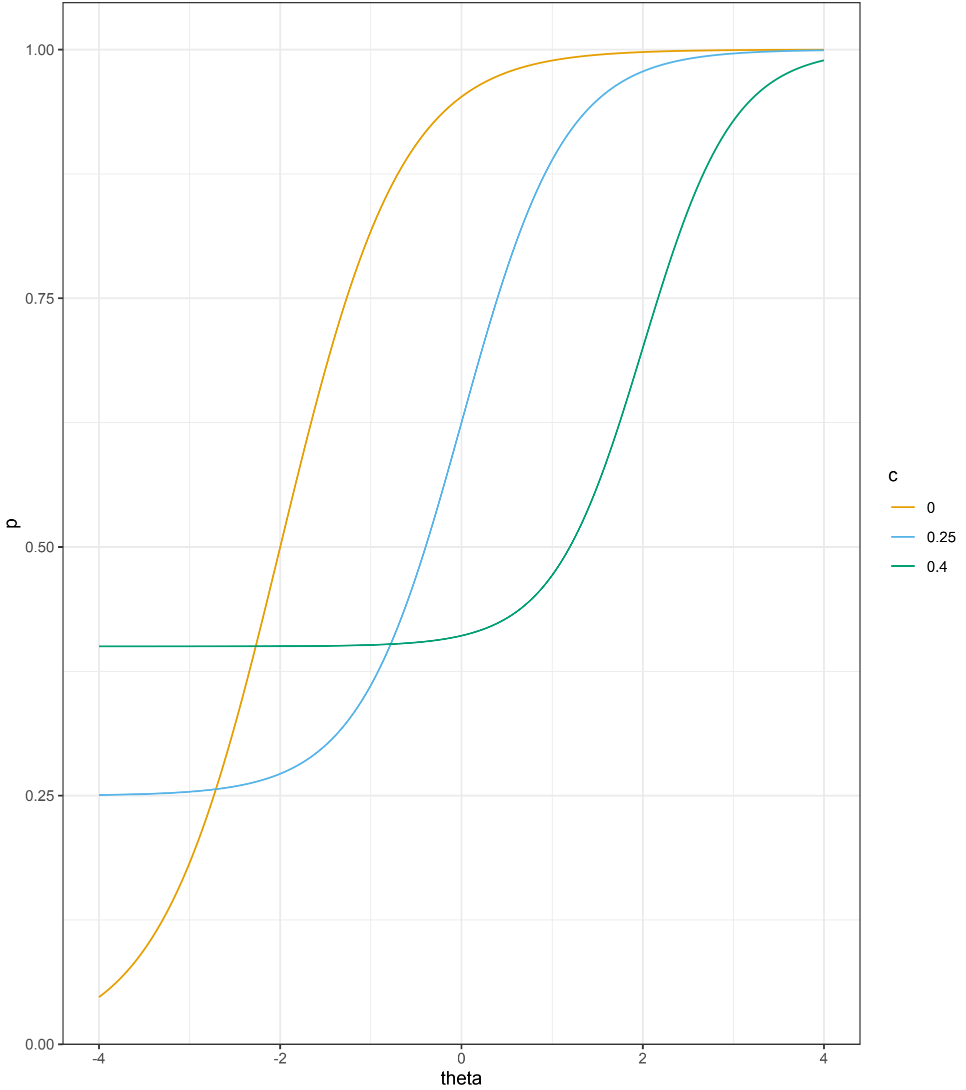
The three parameter logistic IRT model

$$P(X_{ij} = 1|\theta_i) = c_j + \frac{1 - c_j}{1 + e^{-a_j(\theta_i - b_j)}}$$

- We add one more parameter, c_j
- Nothing else changes!
- c_j is the *guessing parameter*
  - It puts a floor on the lowest probability of correct response

The 3PL IRF

```
data.frame(theta = seq(-4, 4,
                       length.out = 1e4),
           b_1 = -2,
           b_2 = 0,
           b_3 = 2) |>
  pivot_longer(starts_with('b_'),
               names_to = 'param',
               values_to = 'b') |>
  select(-param) |>
  mutate(a = case_when(b == -2 ~ 1.5,
                       b == 0 ~ 1.75,
                       b == 2 ~ 2)) |>
  mutate(c = case_when(b == -2 ~ 0,
                       b == 0 ~ 0.25,
                       b == 2 ~ 0.4)) |>
  mutate(p = c+(1-c)*plogis(a*(theta-b))) |>
  ggplot(aes(x = theta,
             y = p,
             color = as.character(c))) +
  geom_line() +
  labs(color = 'c') +
  scale_color_okabeito() +
  theme_bw()
```



Important IRT estimation stuff

- We will use the package `mirt`
  - Short for "multidimensional item response theory"
  - Absolutely the best IRT package in `R`
- The scale that everything gets projected to is not *identified*
- There are lots of different ways to enforce identification, `mirt` does two things:
  - $\bar{b}_j = 0$
  - $a_1 = 1$
- `mirt` also does one other weird thing...

Extracting item parameters from a `mirt` model

- What `mirt` actually estimates for a 2PL is this:

$$P(X_{ij} = 1|\theta_i) = \frac{1}{1 + e^{-a_j\theta_i + b_j}}$$

- Note that instead of $-a_j(\theta_i - b_j)$, it is using $(-a_j\theta_i + b_j)$
  - Called a *slope-intercept* paramterization
  - This is because `mirt` is designed to be flexible with multidimensional (i.e., multiple θs per person) models
  - This means we interpret the default display of the b parameter from `mirt` as an *easiness* parameter
- This is awful, so you can get the parameters in a nicer form
  - Try `params <- coef(model, IRTpars = TRUE, simplify = TRUE)`
  - `IRTpars=TRUE` is doing most of the work here

Applying Item Response Theory

A Familiar Dataset

- Data are available here
- Lecture code is available here

```r
d <- read_rds('animalfights_clean.rds')

# Isolate the item response matrix
resp <- d |>
  select(starts_with('d_'))
```

Fitting a 1PL

```r
# Estimate the model
m_1pl <- mirt(resp, 1, itemtype='Rasch')

# Check default item parameterization
coef(m_1pl)

# Extract something in a better form
params_1pl <- coef(m_1pl, IRTpars = TRUE, simplify = TRUE)

# Generate ability estimates for each person
thetas_1pl <- fscores(m_1pl)
```

1. What do these objects look like? What components do they have?
2. What is the difference between the output from `coef(m_1pl)` and the output when `IRTpars=TRUE`?
3. What comes out of the `fscores()` function?

Fitting a 2PL

```r
m_2pl <- mirt(resp, 1, itemtype='2PL')

# Check default item parameterization
coef(m_2pl)

# Extract better item parameters
params_2pl <- coef(m_2pl, IRTpars = TRUE, simplify = TRUE)

# Generate ability estimates for each person
thetas_2pl <- fscores(m_2pl)
```

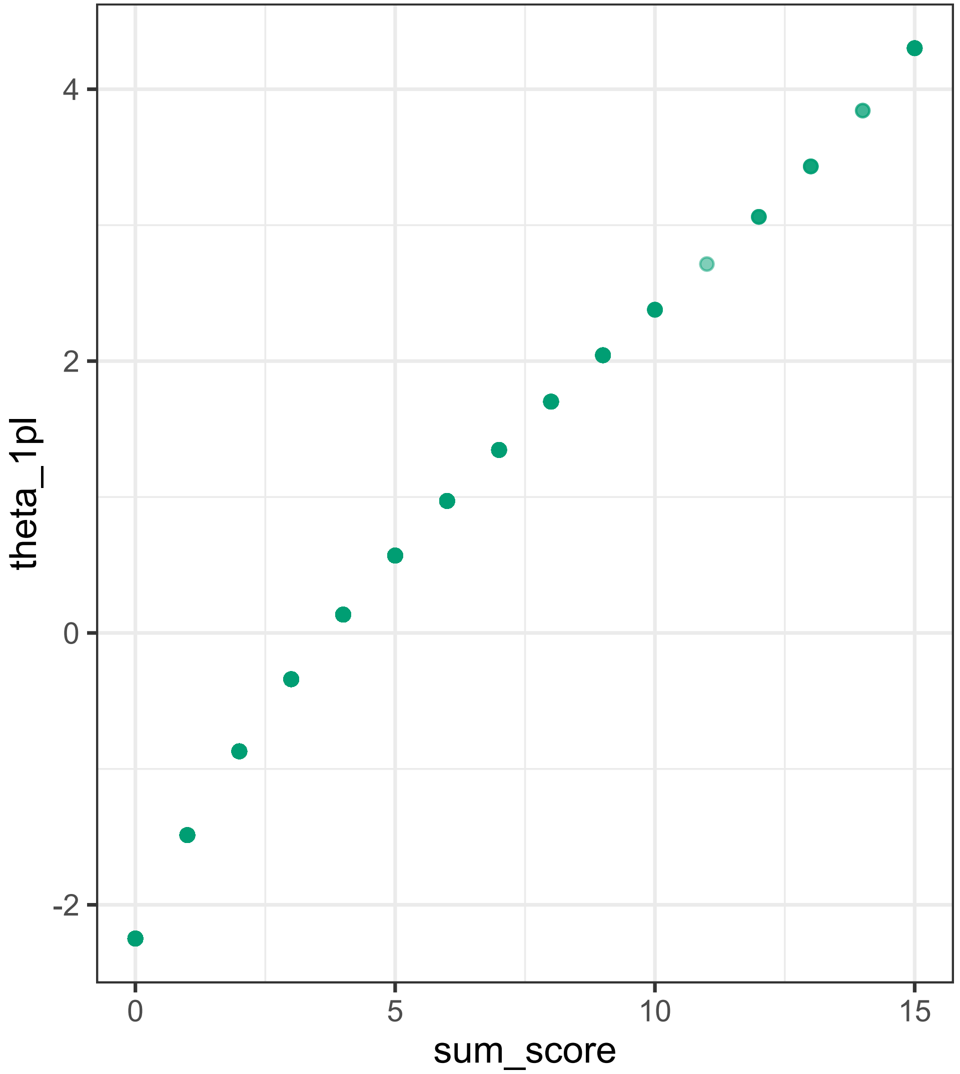
1. What is different between the output from the 1PL and 2PL?
2. What is the difference between the output from `coef(m_2pl)` and the output when `IRTpars=TRUE`?

Combine Output for Plotting

```r
items <- data.frame(item = rownames(params_1pl$items),
                    b_1pl = params_1pl$items[,2],
                    b_2pl = params_2pl$items[,2],
                    a_2pl = params_2pl$items[,1])

persons <- data.frame(sum_score = rowSums(resp),
                      theta_1pl = as.vector(thetas_1pl),
                      theta_2pl = as.vector(thetas_2pl))
```

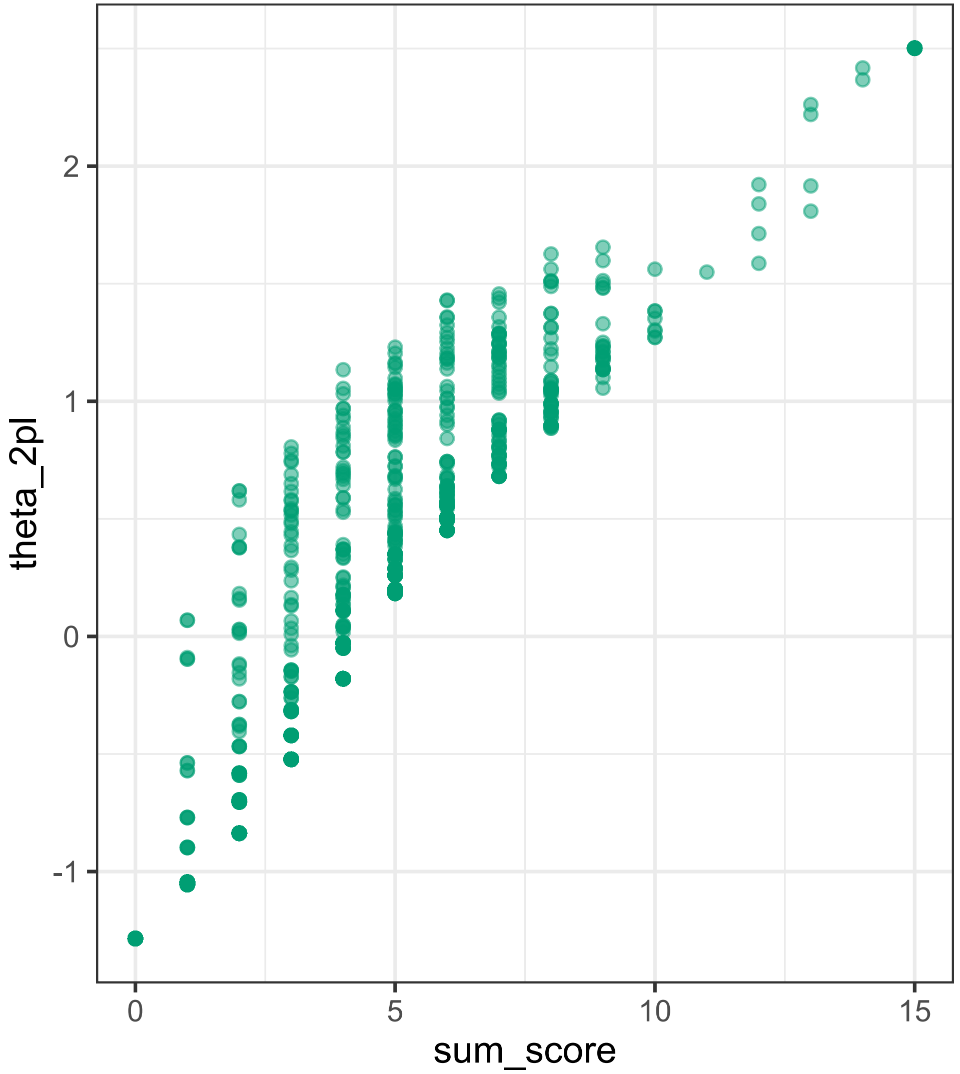
Ability Estimation: Sum Score vs. 1PL

```r
ggplot(persons, aes(x = sum_score,
                    y = theta_1pl)) +
  geom_point(color = okabeito_colors(3),
             alpha = 0.5) +
  theme_bw()
```



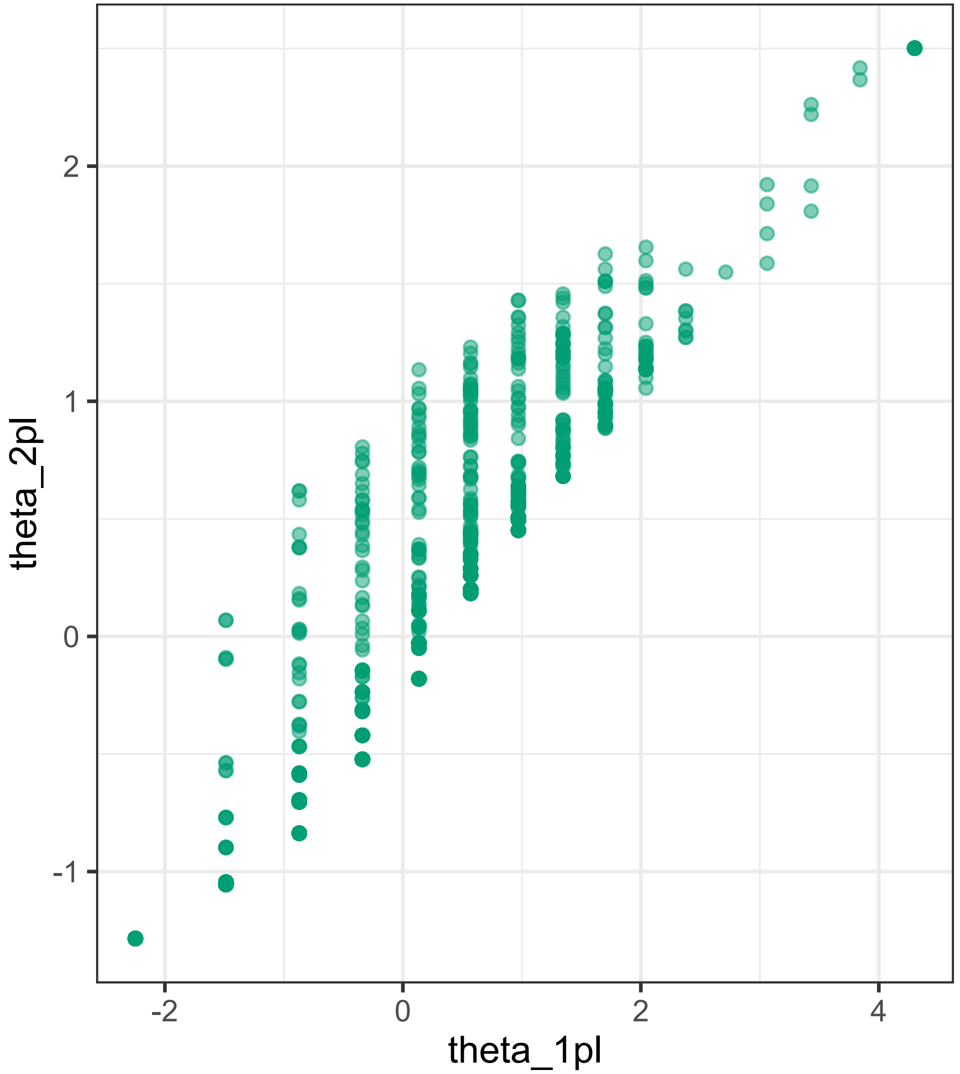
Ability Estimation: Sum Score vs. 2PL

```r
ggplot(persons, aes(x = sum_score,
                    y = theta_2pl)) +
  geom_point(color = okabeito_colors(3),
             alpha = 0.5) +
  theme_bw()
```



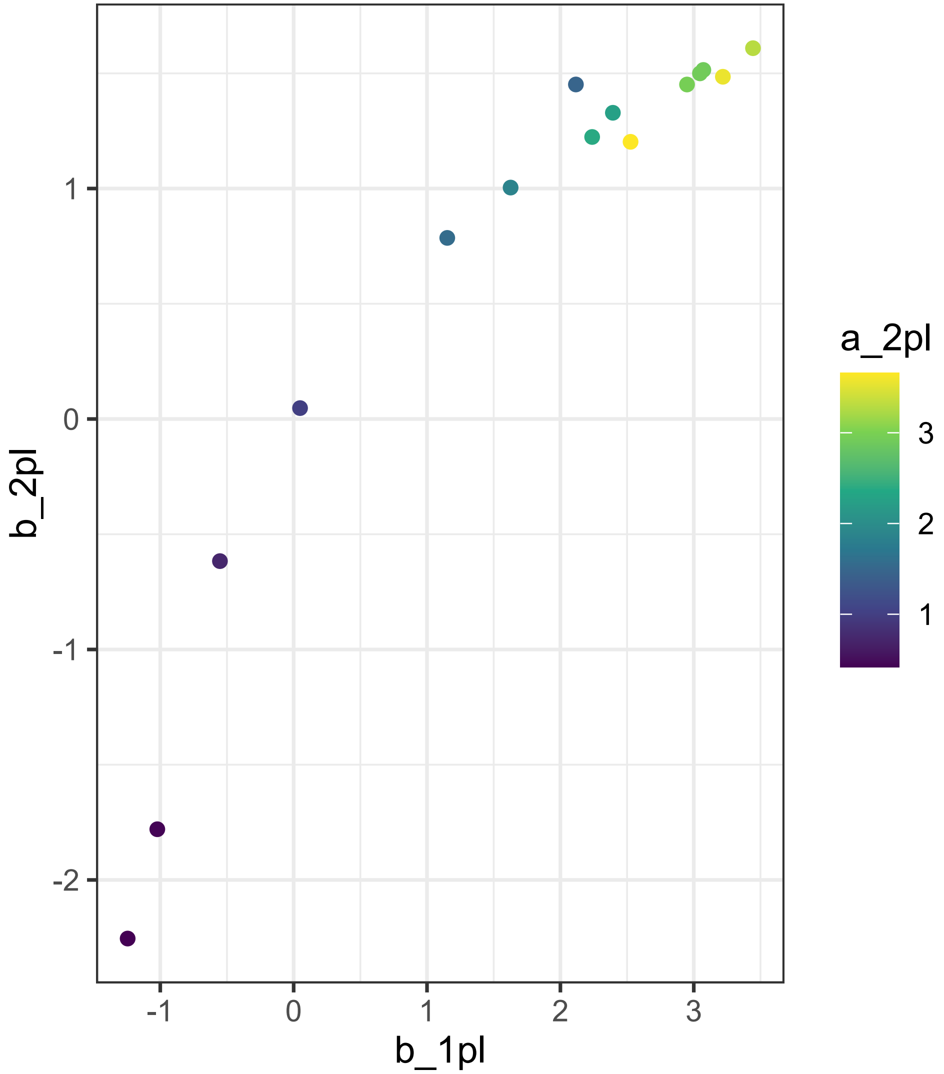
Ability Estimation: 1PL vs. 2PL

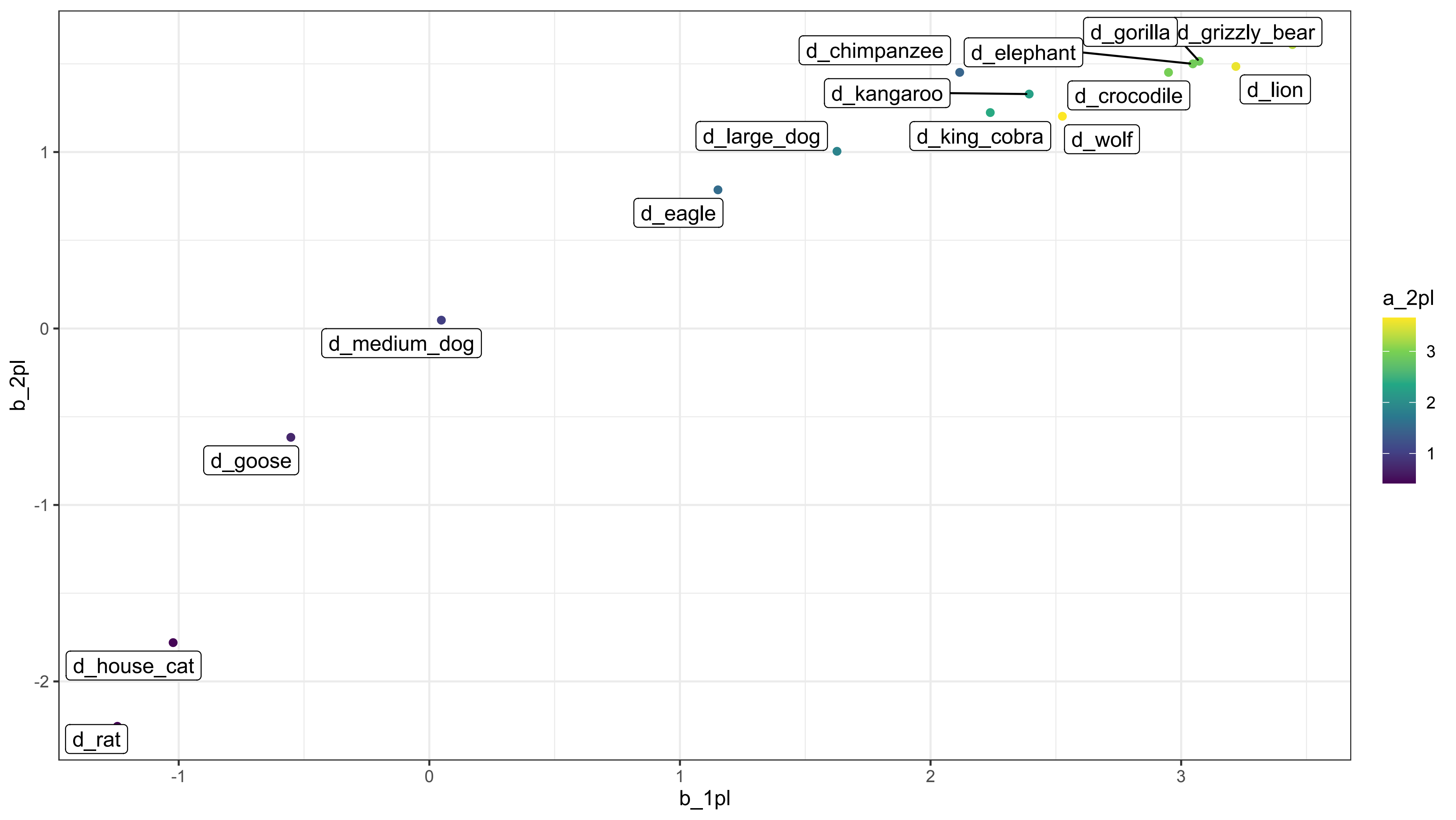
```r
ggplot(persons, aes(x = theta_1pl,
                    y = theta_2pl)) +
  geom_point(color = okabeito_colors(3),
             alpha = 0.5) +
  theme_bw()
```



Item Difficulties

```
items |>
  ggplot(aes(x= b_1pl,
             y= b_2pl,
             color=a_2pl,
             label=item)) +
  geom_point() +
  scale_color_viridis_c() +
  theme_bw()
```




d_gorilla
d_grizzly_bear
d_chimpanzee
d_elephant
d_lion
d_kangaroo
d_crocodile
d_king_cobra
d_wolf
d_large_dog
d_eagle
d_medium_dog
d_goose
d_house_cat
d_rat
b_2pl
b_1pl
a_2pl
1
0
-1
-2
-1
0
1
2
3
3
2
1

Wrap Up

Recap

- Hopefully PCA and factor analysis are a little more clear
  - Readings are helpful here if you need more technical detail!
- Latent variable models pose a really wild estimation challenge!
  - The Expectation Maximization Algorithm can solve it in many cases
- Item Response Theory (IRT) puts people and items onto a common scale
  - It can even produce interval scales!

Loose ends and looking forward

- There is a *ton* of measurement theory that we just handwaved past
- There is also a *ton* of IRT stuff that we also just handwaved past
- The next three weeks are all IRT, and some of the second half of the class is also IRT
  - We'll get the other IRT stuff gradually instead of just dumping a pile of crap into our brains today
  - Measurement theory stuff will come gradually as well
- Next week: more IRT, multidimensional IRT, and shoving non-item response data into IRT models
- The week after: polytomous IRT (multiple response categories, great for you survey weirdos)
- The week after that: weird IRT models (Elo scores, ideal point models) and data that you might not think of as item responses

Check-Out

- PollEv.com/klintkanopka