Mathematical Modelling

Signal Detection Model Extensions

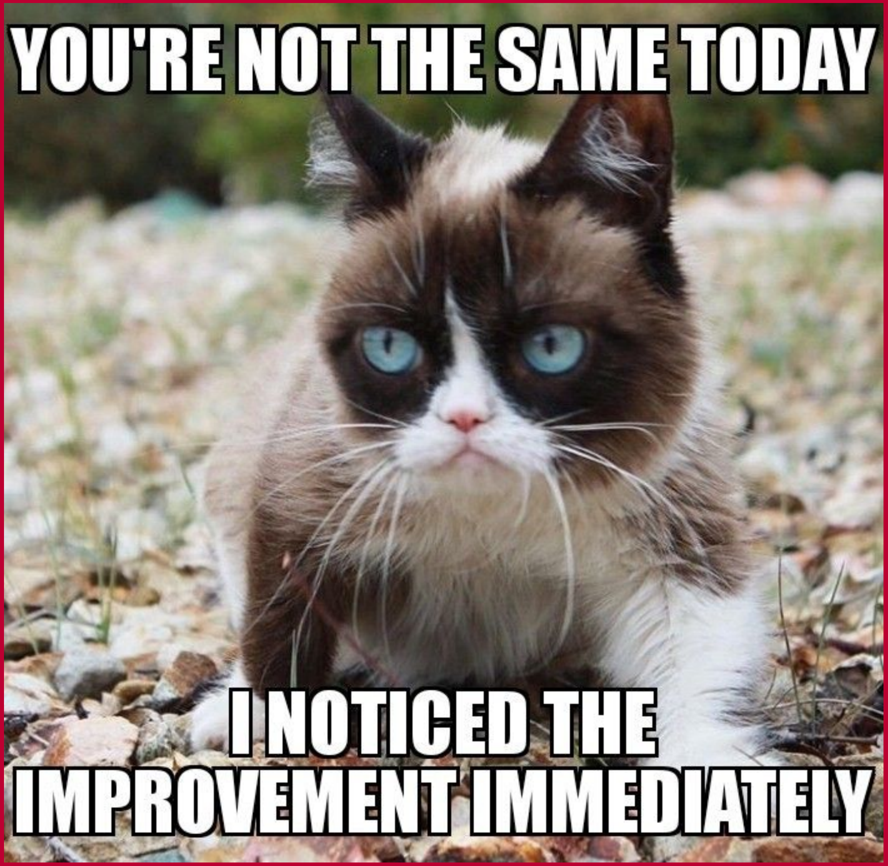
Julia Haaf & Frederik Aust

February/March 2021

Agenda

1. Improving optimization
   - Transformation of d'
   - Convergence
   - Nested optimization
2. Extensions of SDT models
   - Confidence ratings
   - Unequal variance
3. Dealing with a hierarchical data structure
   - Fitting models to aggregated data
   - Fitting models to individual data
   - Accounting for the hierarchical structure

Improving optimization

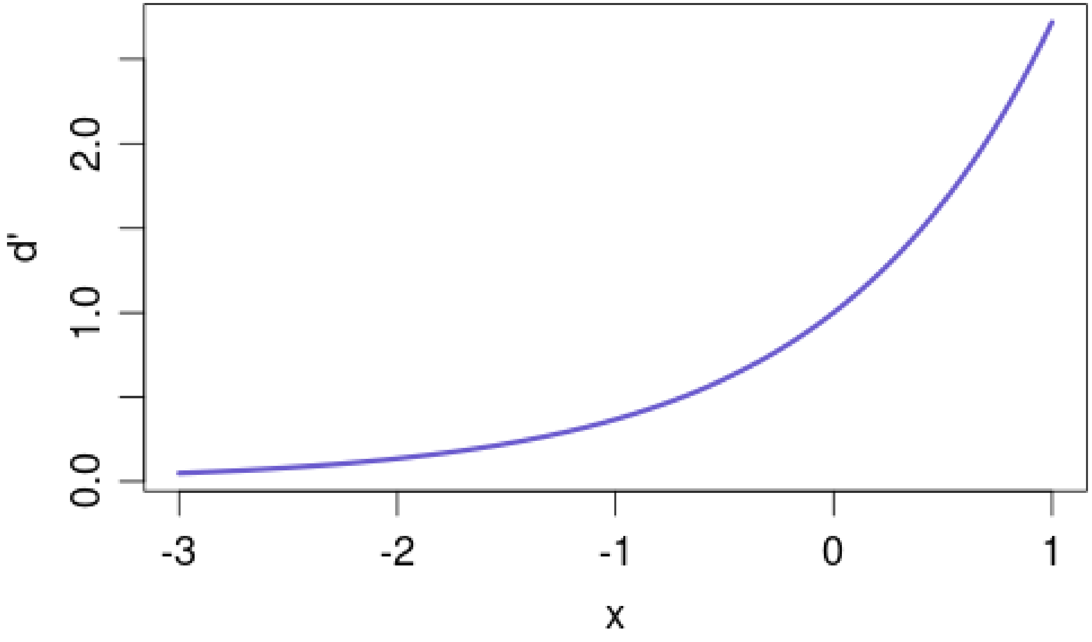


Transformation of d'

- The default algorithm in optim() assumes that parameters can take on any real value.
- For the signal detection model, $c \in (-\infty, \infty)$, $d' \in (0, \infty)$.

Transformation of d'

- The default algorithm in optim() assumes that parameters can take on any real value.
- For the signal detection model, $c \in (-\infty, \infty), d' \in (0, \infty)$.
- Solution: Transform the parameter d' in the function, for example using $\exp(x)$ transformation.



Transformation of d' in R

```r
sdtm <- function(par, y, n) { #par = c(d', c) y = c(hit, fa)
  # transform d' to parameter space
   dprime <- exp(par[1])
  # no transformation needed for criterion
   crit <- par[2]
  p1 <- 1 - pnorm(crit - dprime)
  p2 <- 1 - pnorm(crit)
  -2 * (dbinom(y["hit"], n["signal"], p1, log = TRUE) +
    dbinom(y["fa"], n["noise"], p2, log = TRUE))
}
out <- optim(par = c(d = 0, c = 0), fn = sdtm, y = y, n = n)
out$par
```

```
##          d          c
## -0.8649711 -0.2533813
```

Transformation of d' in R

```r
sdtm <- function(par, y, n) { #par = c(d', c) y = c(hit, fa)
  # transform d' to parameter space
   dprime <- exp(par[1])
  # no transformation needed for criterion
   crit <- par[2]
  p1 <- 1 - pnorm(crit - dprime)
  p2 <- 1 - pnorm(crit)
  -2 * (dbinom(y["hit"], n["signal"], p1, log = TRUE) +
    dbinom(y["fa"], n["noise"], p2, log = TRUE))
}
out <- optim(par = c(d = 0, c = 0), fn = sdtm, y = y, n = n)
out$par
```

```
##          d          c
## -0.8649711 -0.2533813
```

```r
c(exp(out$par[1]), out$par[2]) ## transform back
```

```
##          d          c
##  0.4210637 -0.2533813
```

Convergence

Did the optimization algorithm reach successful completion?

Convergence of `optim()`

> *convergence:* An integer code. 0 indicates successful completion (which is always the case for "SANN" and "Brent"). Possible error codes are
>
> 1: indicates that the iteration limit maxit had been reached.
>
> 10: indicates degeneracy of the Nelder-Mead simplex.
>
> 51: indicates a warning from the "L-BFGS-B" method; see component message for further details.
>
> 52: indicates an error from the "L-BFGS-B" method; see component message for further details.

Checking convergence

$$h = \sum_{i=1}^{n} (y_i - \theta_i)^2$$

Checking convergence

$$h = \sum_{i=1}^{n} (y_i - \theta_i)^2$$

```
y <- 1:20

h <- function(theta, y) sum((theta - y)^2)
par <- rep(10, 20) #starting values
optim(par, h, y = y)$convergence
```

```
## [1] 1
```

Checking convergence

$$h = \sum_{i=1}^{n} (y_i - \theta_i)^2$$

```
y <- 1:20

h <- function(theta, y) sum((theta - y)^2)
par <- rep(10, 20) #starting values
optim(par, h, y = y)$convergence
```

```
## [1] 1
```

```
head(optim(par, h, y = y)$par)
```

```
## [1] 0.3535526 1.7091429 4.1538558 3.0865141 5.8710791 5.2499118
```

```
optim(par, h, y = y)$value
```

```
## [1] 19.91514
```

Nested optimization

- Frame the analysis so that it involves several separate optimizations with a smaller number of parameters each.
- For the function h: Estimate of θ_1 only dependent on y_1, not the other observations.

Nested optimization

- Frame the analysis so that it involves several separate optimizations with a smaller number of parameters each.
- For the function h: Estimate of θ_1 only dependent on y_1, not the other observations.

```
par.est <- rep(0,20) #reserve space
min <- 0
for (i in 1:20){
est <- optimize(h, interval = c(-100, 100), y = y[i])
min <- min + est$objective
par.est[i] <- est$minimum #store the estimates
}
min; head(par.est)
```

```
## [1] 3.786532e-29
```

```
## [1] 1 2 3 4 5 6
```

Nested optimization, an example

	Reward signal trial	Reward noise trial	Hit	Miss	FA	CR
Cond A	10 cents	1 cents	404	96	301	199
Cond B	7 cents	3 cents	348	152	235	265
Cond C	5 cents	5 cents	287	213	183	317
Cond D	3 cents	7 cents	251	249	102	398
Cond E	1 cents	10 cents	148	352	20	480

Nested optimization, an example

- Optimization of parameters specific to conditions, b_A, b_B, ..., b_E is nested within parameters common across all conditions, d.

Nested optimization, an example

- Optimization of parameters specific to conditions, b_A, b_B, ..., b_E is nested within parameters common across all conditions, d.

Steps:

1. Assume the true value of d.
2. Then b_A only depends on condition A, b_B only on condition B, etc.
3. Compute the likelihood for b in a single condition given a fixed value of d.
4. Make a function that separately optimizes b for each condition.
5. Optimize that function for d.

Nested optimization, an example

```
#negative log likelihood for high-threshold model
nll.ht.given.d <- function(b, y, n, d){ #y = c(hit, fa) for one cond
p1 <- d + (1 - d) * b # probability of a hit
p2 <- b # probability of a false alarm
return(-sum(dbinom(y[1], n[1], p1, log = T)
            , dbinom(y[2], n[2], p2, log = T)))
}
```

Nested optimization, an example

```
#negative log likelihood for high-threshold model
nll.ht.given.d <- function(b, y, n, d){ #y = c(hit, fa) for one cond
p1 <- d + (1 - d) * b # probability of a hit
p2 <- b # probability of a false alarm
return(-sum(dbinom(y[1], n[1], p1, log = T)
            , dbinom(y[2], n[2], p2, log = T)))
}
```

```
nll.ht <- function(d, y, n) {
  nll.ht.opt <- function(y_i) {
    res <- optimize(nll.ht.given.d, interval = c(0, 1), y = y_i, n = n
    res$objective
  }
  nll.ht.opt(y[1, ]) + nll.ht.opt(y[2, ]) + nll.ht.opt(y[3, ]) +
    nll.ht.opt(y[4, ]) + nll.ht.opt(y[5, ])
}
```

Nested optimization, an example

```r
#negative log likelihood for high-threshold model
nll.ht.given.d <- function(b, y, n, d){ #y = c(hit, fa) for one cond
p1 <- d + (1 - d) * b # probability of a hit
p2 <- b # probability of a false alarm
return(-sum(dbinom(y[1], n[1], p1, log = T)
            , dbinom(y[2], n[2], p2, log = T)))
}
```

```r
nll.ht <- function(d, y, n) {
  nll.ht.opt <- function(y_i) {
    res <- optimize(nll.ht.given.d, interval = c(0, 1), y = y_i, n = n
    res$objective
  }
  nll.ht.opt(y[1, ]) + nll.ht.opt(y[2, ]) + nll.ht.opt(y[3, ]) +
    nll.ht.opt(y[4, ]) + nll.ht.opt(y[5, ])
}
```

```r
g <- optimize(nll.ht, interval = c(0, 1), y = y, n = n)
```

Nested optimization, output

```
g$minimum; g$objective
```

```
## [1] 0.3315705
```

```
## [1] 43.53011
```

Nested optimization, output

```
g$minimum; g$objective
```

```
## [1] 0.3315705
```

```
## [1] 43.53011
```

```
optimize(nll.ht.given.d
         , interval = c(0, 1)
         , y = y[1,]
         , n = n
         , d = g$minimum)$minimum
```

```
## [1] 0.6419589
```

Extensions to the SDT model

Confidence ratings

Confidence ratings

- So far we have focused on models for data with two response options (Binomial models).
- How about more than two options?

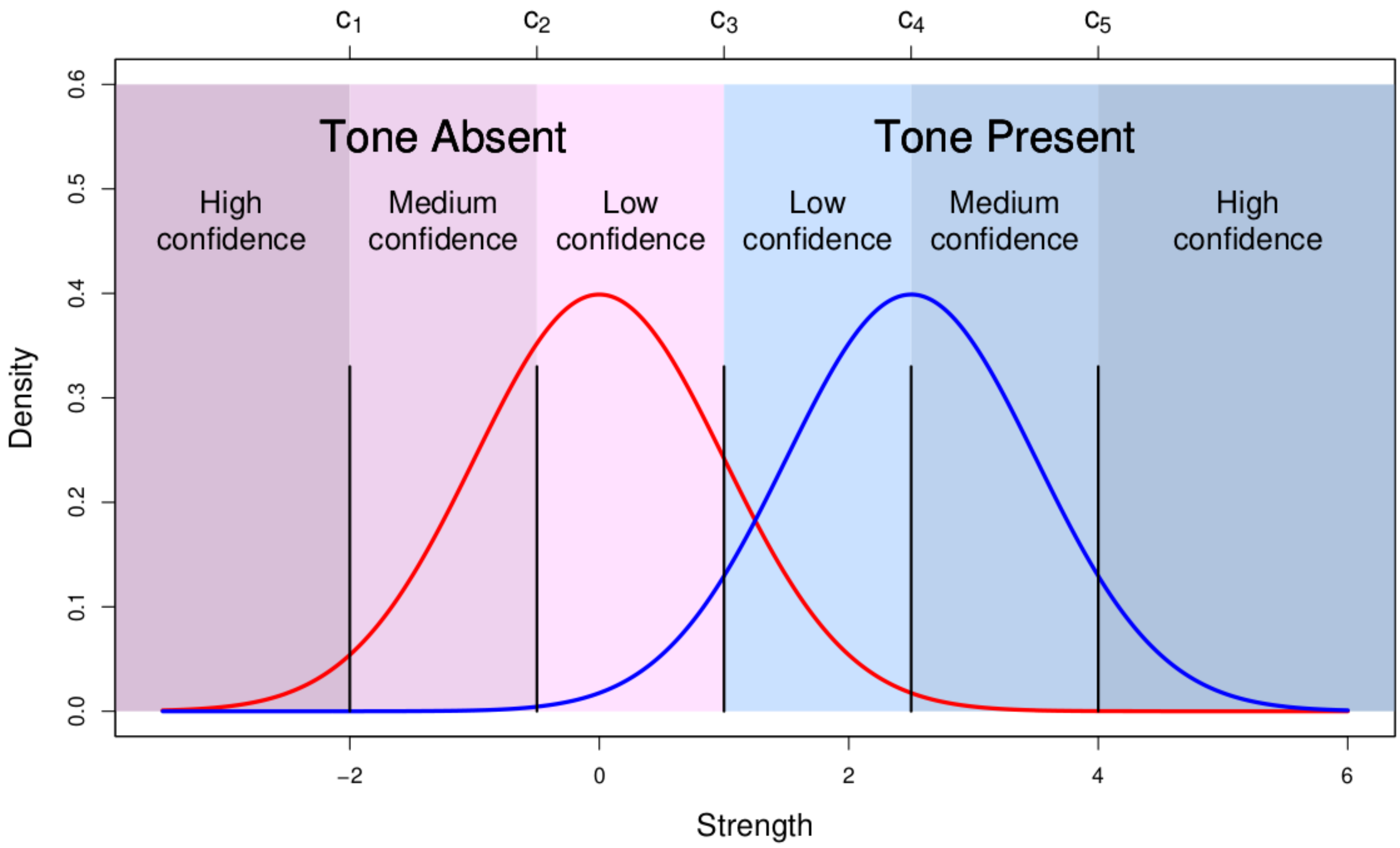
Confidence ratings

- So far we have focused on models for data with two response options (Binomial models).
- How about more than two options?
- Confidence tasks: Participants indicate confidence in their judgments by choosing an option such as "I have low confidence."
- For confidence ratings we need to use a multinomial model (>2 responses).

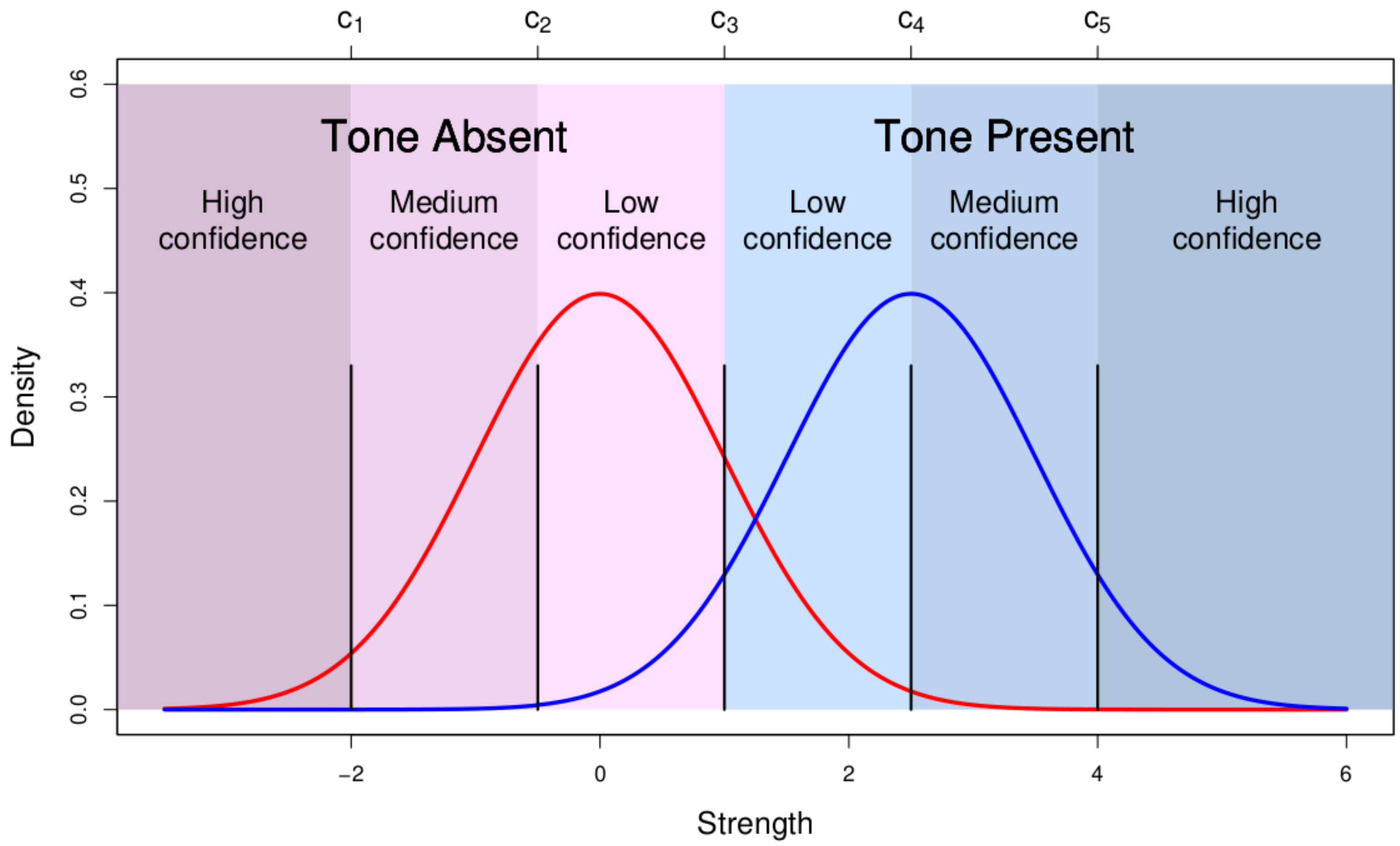
Confidence ratings

- So far we have focused on models for data with two response options (Binomial models).
- How about more than two options?
- Confidence tasks: Participants indicate confidence in their judgments by choosing an option such as "I have low confidence."
- For confidence ratings we need to use a multinomial model (>2 responses).
- Q: Any idea how confidence ratings could be implemented as an SDT model?

Confidence ratings

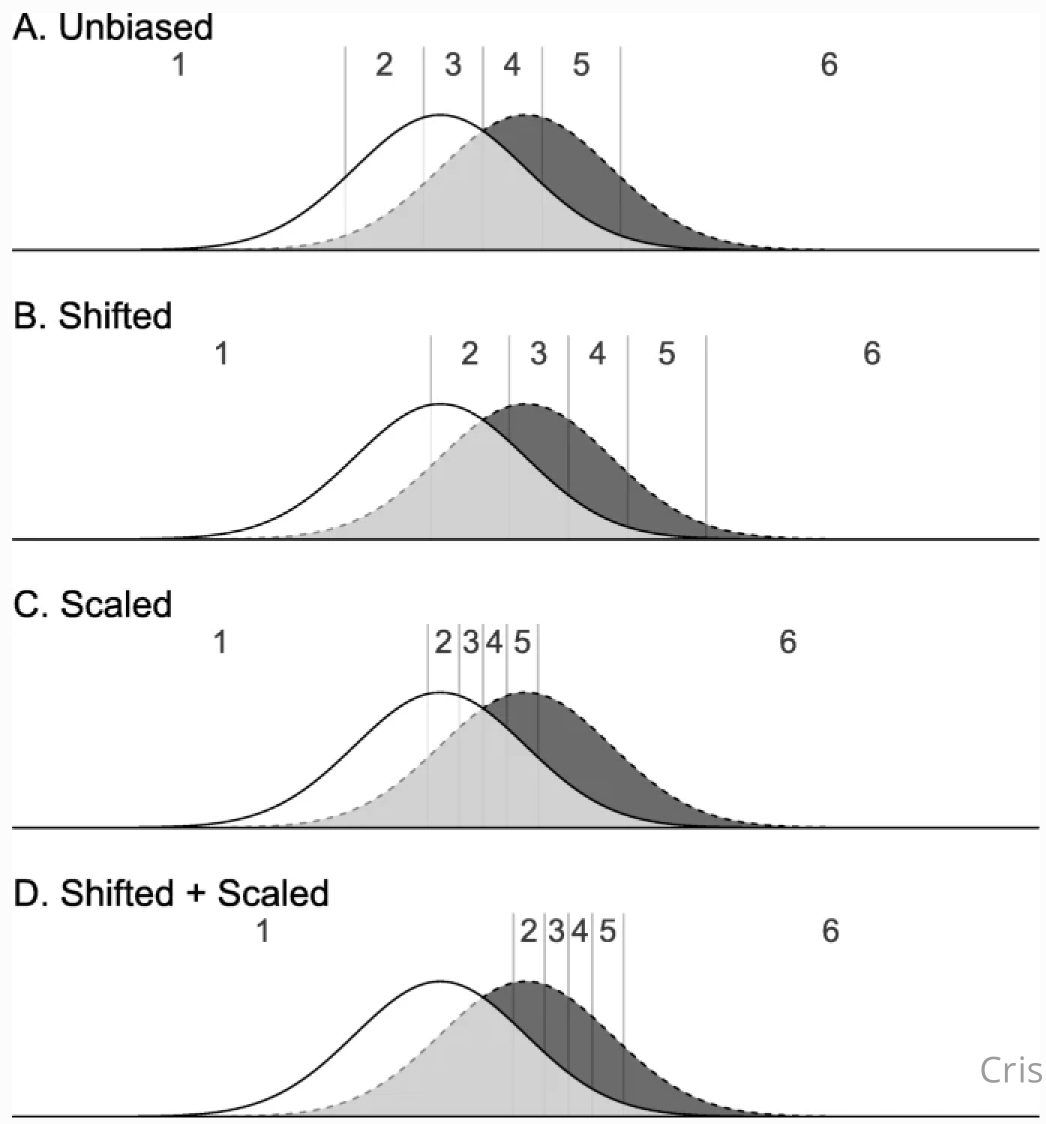


Confidence ratings



- For K rating options there are $k - 1$ criteria.

Confidence ratings



Selker, van den Bergh, Criss, & Wagenmakers, 2019

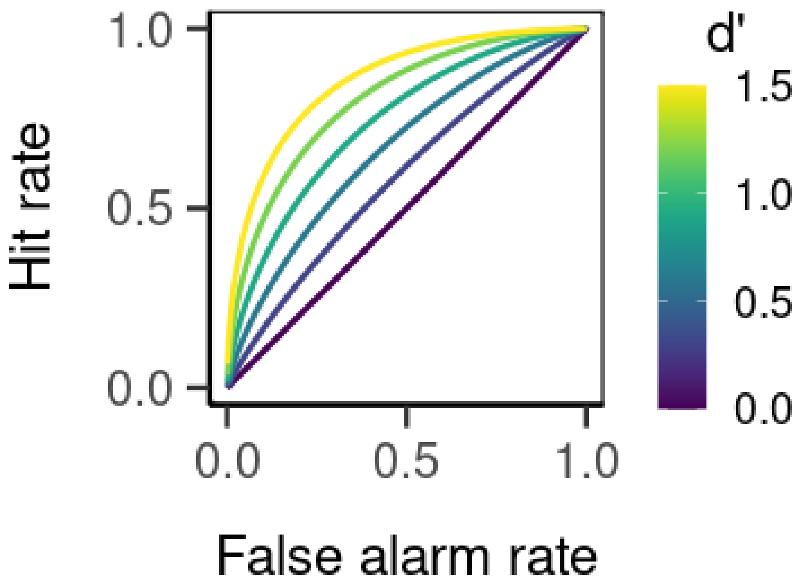
Unequal variance extension

Equal variance assumption

- In the standard SDT model the variances for the noise and signal distributions are fixed to 1.

Equal variance assumption

- In the standard SDT model the variances for the noise and signal distributions are fixed to 1.
- Consequence for ROC curves:



Unequal variance

- Equal variance assumption can be relaxed:

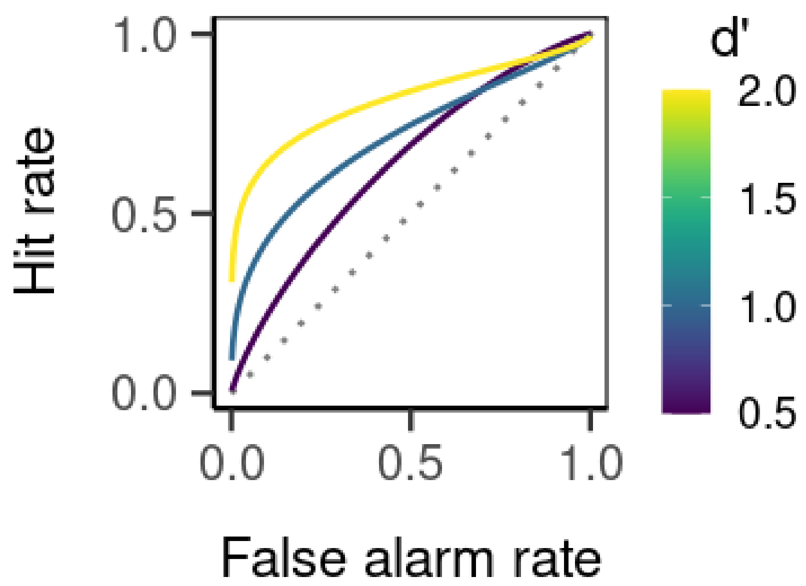
$$S \sim \begin{cases} \mathrm{Normal}(\mu = d', \sigma^2), & \text{for signal-present trials,} \\ \mathrm{Normal}(\mu = 0, 1), & \text{for signal-absent trials.} \end{cases}$$

Unequal variance

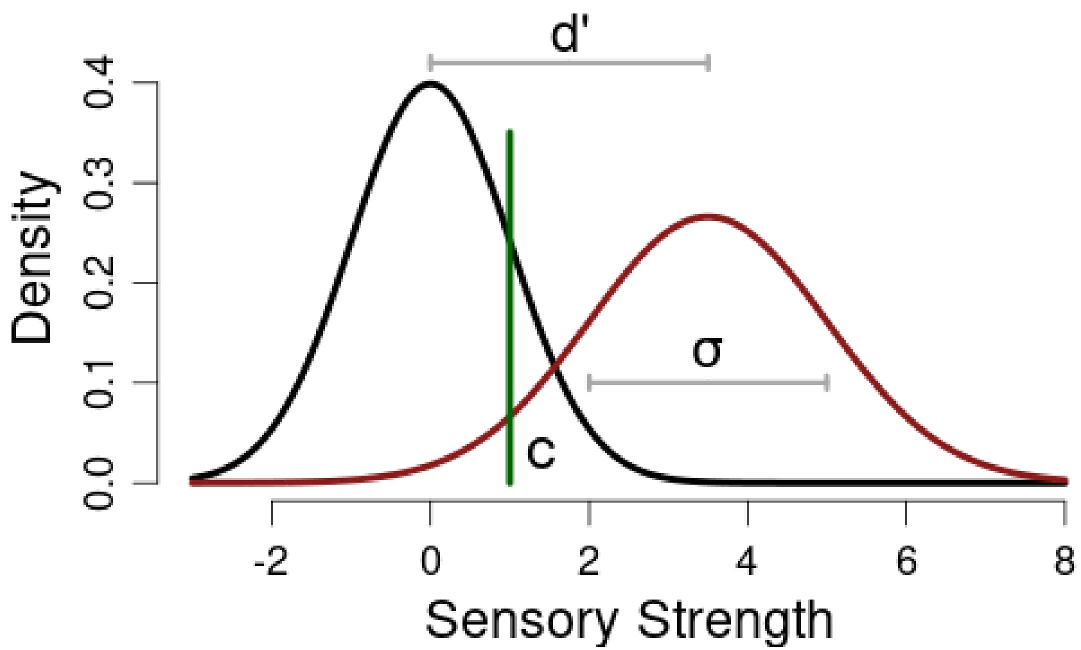
- Equal variance assumption can be relaxed:

$$S \sim \begin{cases} \text{Normal}(\mu = d', \sigma^2), & \text{for signal-present trials,} \\ \text{Normal}(\mu = 0, 1), & \text{for signal-absent trials.} \end{cases}$$

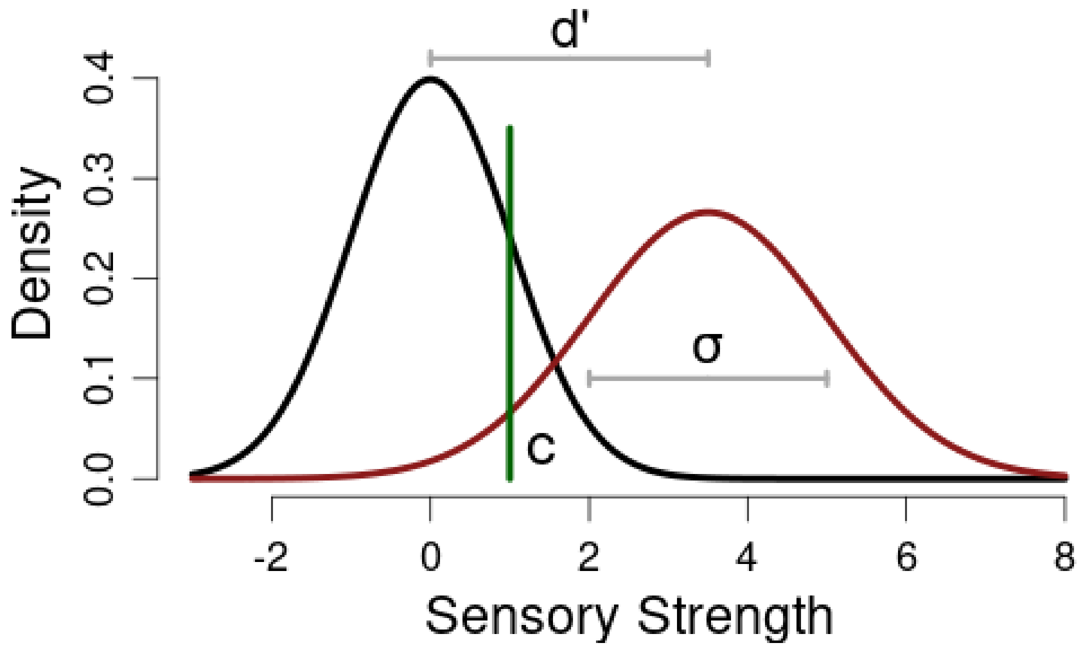
- Consequence for ROC curves:



The unequal variance signal detection model (UVSD)



The unequal variance signal detection model (UVSD)



$$p_1 = p(\text{"Signal"} \mid \text{Signal}) = 1 - \Phi\left(\frac{c - d'}{\sigma}\right),$$

$$p_2 = p(\text{"Signal"} \mid \text{Noise}) = 1 - \Phi(c)$$

UVSD and estimation

Identifiability?

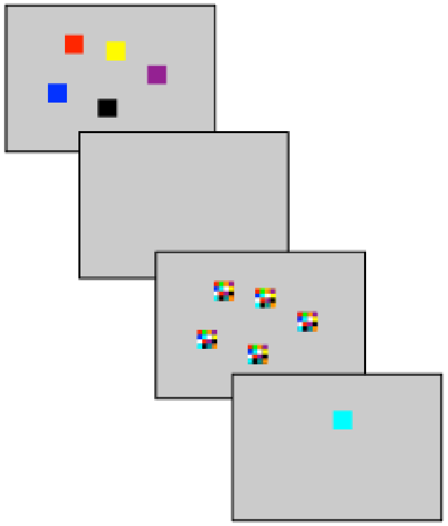
UVSD and estimation

Identifiability?

For standard signal detextion experiments or memory experiments the UVSD model is not identified.

- Three parameters (d', c, σ).
- Two independent observations (hits and false alarms).

Let's get some more data



- Is there a fixed capacity limit to visual working memory?
- Manipulated: Set size (number of items to remember): 2, 5, or 8

Rouder et al., 2008

Let's get some more data

	cond2	cond5	cond8
hits	666	593	529
fas	47	193	268

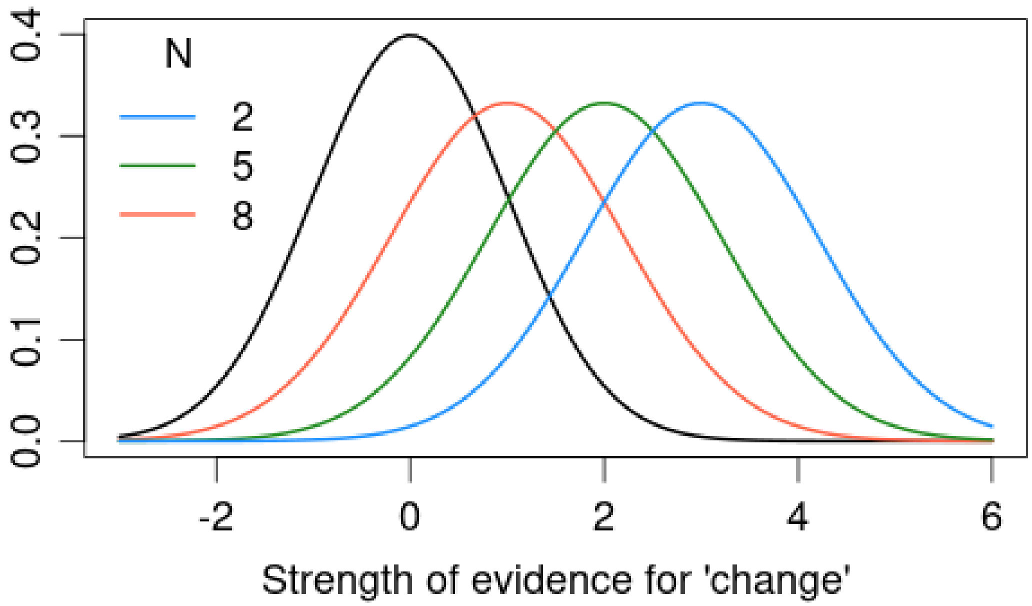
Fitting the UVSD model

- Three conditions, 6 independent data points.
- How many parameters if we let everything vary across conditions?

Fitting the UVSD model

- Three conditions, 6 independent data points.
- How many parameters if we let everything vary across conditions?
- Restrict the model: Assume only d' is affected.

UVSD for change detection



UVSD in R

```r
#Log likelihood for UVSD
uvsd <- function(dprime, crit, sigma, y, n) {
   p1 <- 1 - pnorm(crit, mean = dprime, sd = sigma)
   p2 <- 1 - pnorm(crit)
  - (dbinom(y["hits"], n["old"], p1, log = TRUE) +
    dbinom(y["fas"], n["new"], p2, log = TRUE))
}

# nested optimization
nll.uvsd <- function(par, y, n){ #par = c(crit, sigma)
  par[2] <- exp(par[2])
  out <- matrix(ncol = 2, nrow = ncol(y))
  for(i in 1:ncol(y)){
    tmp <- optimize(uvsd, interval = c(0,10)
                    , y = y[, i], n = n
                    , crit = par[1], sigma = par[2])
    out[i, 1] <- tmp$objective
    out[i, 2] <- tmp$minimum
  }
  return(sum(out[, 1], out[,2]))
}
```

UVSD in R

```r
g <- optim(par = c(crit = 0, sigma = log(1)), nll.uvsd, y = y, n = n)

crit <- g$par["crit"]
sigma <- exp(g$par["sigma"])
print(c(crit, sigma))
```

```
##      crit     sigma
## 0.6867411 0.3778104
```

UVSD in R

```r
g <- optim(par = c(crit = 0, sigma = log(1)), nll.uvsd, y = y, n = n)

crit <- g$par["crit"]
sigma <- exp(g$par["sigma"])
print(c(crit, sigma))
```

```
##      crit     sigma
## 0.6867411 0.3778104
```

```r
out <- c()
for(i in 1:ncol(y)){
    tmp <- optimize(uvsd, interval = c(0,10)
                    , y = y[, i], n = n
                    , crit = crit, sigma = sigma)
    out[i] <- tmp$minimum
}
print(c(out, crit, sigma))
```

```
##                                          crit     sigma
## 1.3723718 1.0939115 0.9617541 0.6867411 0.3778104
```

Dealing with a hierarchical data structure

Nested models, nested optimization, now nested data?!

Data so far:

	Reward signal trial	Reward noise trial	Hit	Miss	FA	CR
Cond A	10 cents	1 cents	404	96	301	199
Cond B	7 cents	3 cents	348	152	235	265
Cond C	5 cents	5 cents	287	213	183	317
Cond D	3 cents	7 cents	251	249	102	398
Cond E	1 cents	10 cents	148	352	20	480

Nested models, nested optimization, now nested data?!

Actual data:

N2_0.5_H	N2_0.5_M	N2_0.5_Fa	N2_0.5_Cr	N5_0.5_H	N5_0.5_M	N5_0.5_Fa
29	1	1	29	25	5	7
28	2	0	30	25	5	6
27	3	7	23	23	7	10
30	0	0	30	27	3	7
30	0	0	30	27	3	4
29	1	0	30	23	7	7
29	1	3	27	23	7	9
28	2	5	25	29	1	12
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3	11

Nested models, nested optimization, now nested data?!

- It is common in experimental psychology to have a nested data structure.
- Each participant responds to several trials, typically across several experimental conditions.

Nested models, nested optimization, now nested data?!

- It is common in experimental psychology to have a nested data structure.
- Each participant responds to several trials, typically across several experimental conditions.
- Also called within-subjects design (duh!).

Nested models, nested optimization, now nested data?!

- It is common in experimental psychology to have a nested data structure.
- Each participant responds to several trials, typically across several experimental conditions.
- Also called within-subjects design (duh!).
- Typical analysis: Aggregate across participants.

Nested models, nested optimization, now nested data?!

- It is common in experimental psychology to have a nested data structure.
- Each participant responds to several trials, typically across several experimental conditions.
- Also called within-subjects design (duh!).
- Typical analysis: Aggregate across participants.
- Is this appropriate for mathematical modeling?

Fitting models to group data

- Sum data across participants, fit the model once.

Advantages

- Simple.
- One (model comarison) result that can be interpreted immediately.

Disadvantages

- Ignores individual differences.
- Masks individual differences.

Fitting models to individual data

- If you have 30 participants, fit the model 30 times.

Advantages

- Improved understanding of each person's processing structure.
- Allows for a more accurate assessment of robustness of model comparison results.

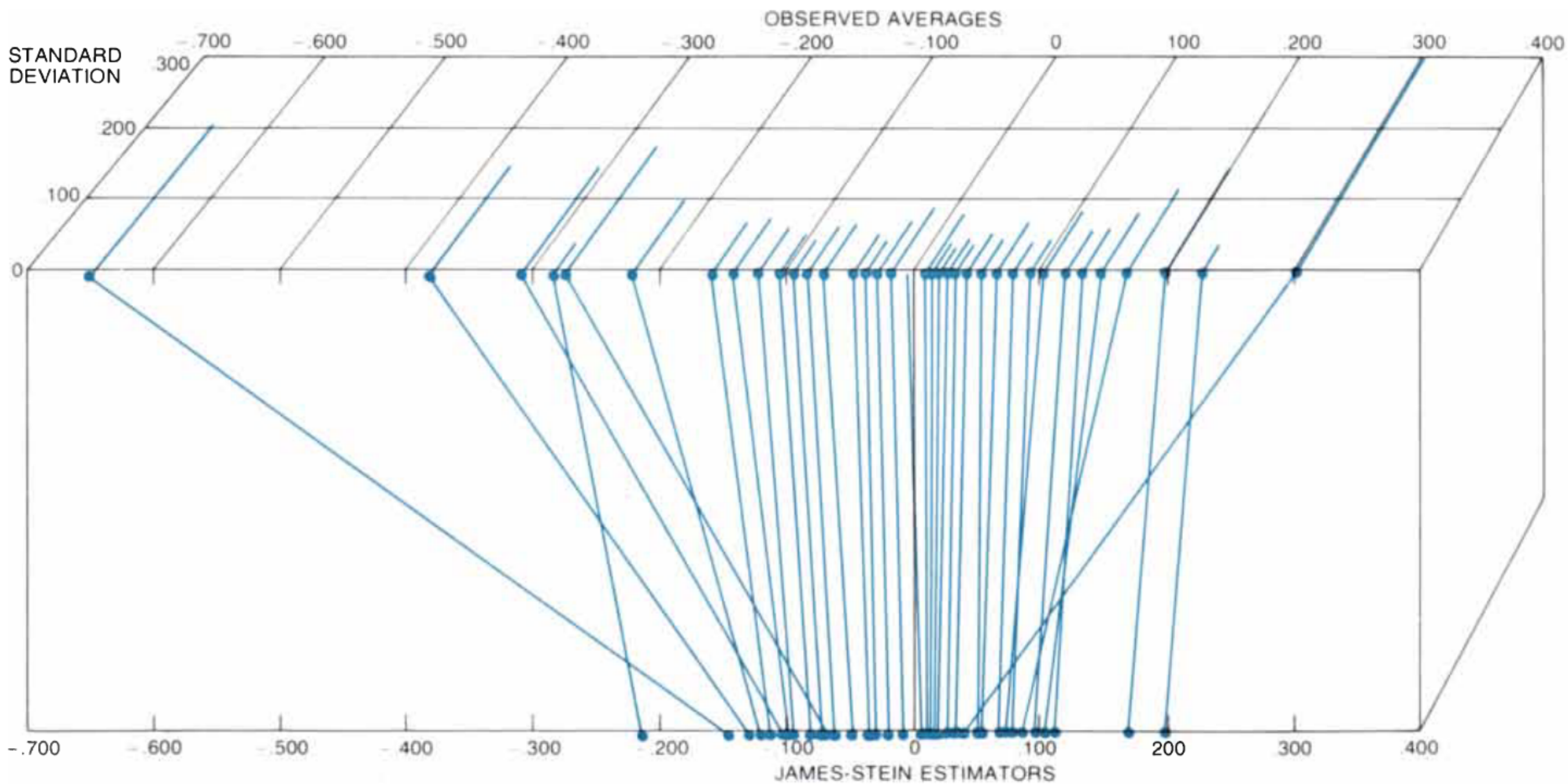
Disadvantages

- Difficult to interpret (e.g., "HTM was preferred over SDM for 60% of participants.")
- May overstate individual differences.

Accounting for the hierarchical structure

- Hierarchical modeling allows to optimally combine individual participant data.
- Models dependencies between observations within a person.
- Results in overall estimates across people *and* individual estimates.
- Much much more difficult.

Accounting for the hierarchical structure



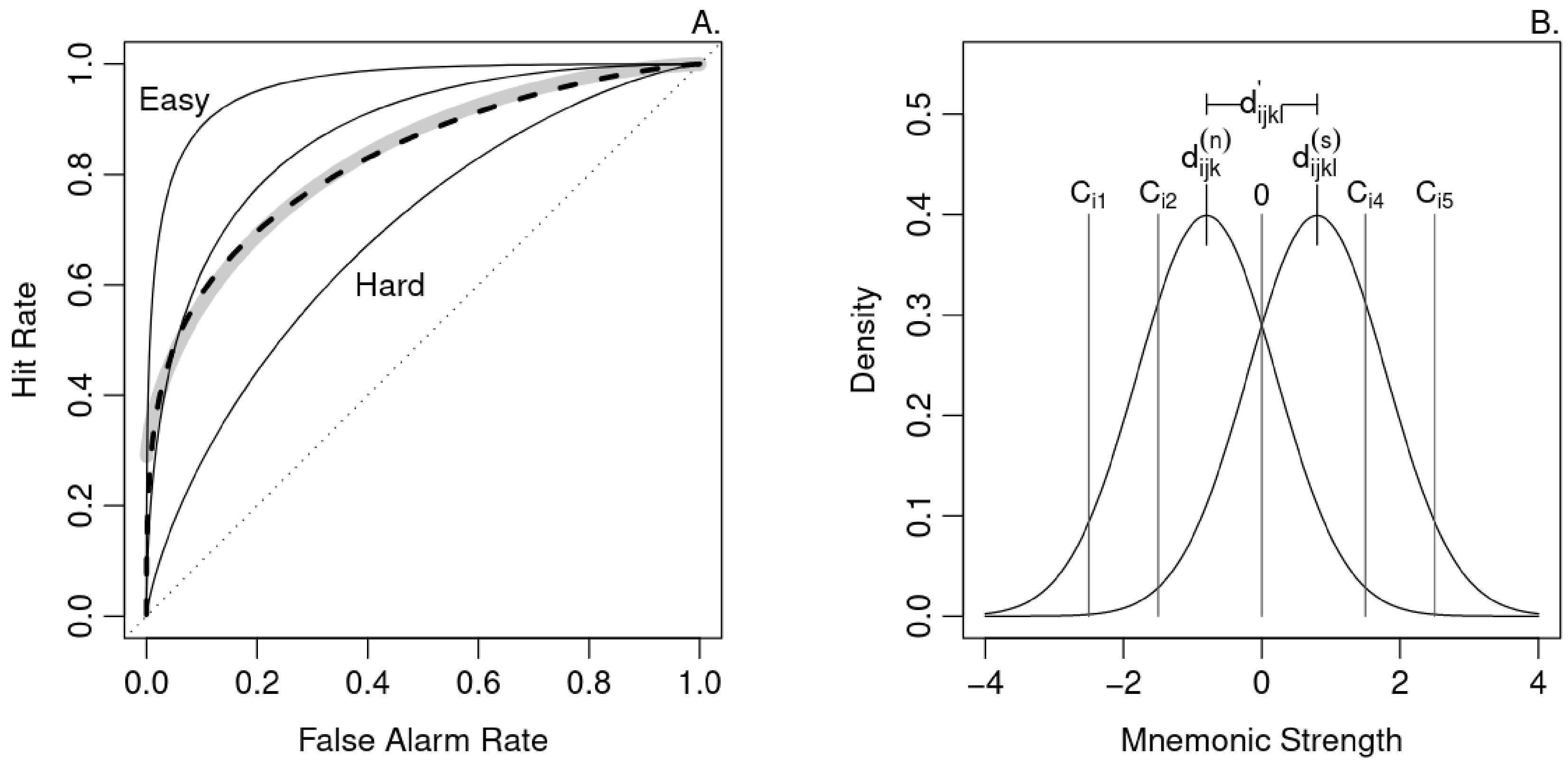
Efron & Morris, 1977

Accounting for the hierarchical structure

- More and more common for mathematical models.
- Introductory chapter: Rouder, Morey & Pratte (2017). *Bayesian hierarchical models of cognition.*

Accounting for the hierarchical structure

- More and more common for mathematical models.
- Introductory chapter: Rouder, Morey & Pratte (2017). *Bayesian hierarchical models of cognition.*



Wrap up

Wrap up

- Many optimization algorithms in R are pretty bad for many parameters.
- There are tricks to help (check convergence, transformation, nested optimization, …).

Wrap up

- Many optimization algorithms in R are pretty bad for many parameters.
- There are tricks to help (check convergence, transformation, nested optimization, …).

Signal Detection models

- can be extended to be even more flexible.
- can be used for rating data.

Wrap up

- Many optimization algorithms in R are pretty bad for many parameters.
- There are tricks to help (check convergence, transformation, nested optimization, ...).

Signal Detection models

- can be extended to be even more flexible.
- can be used for rating data.

Data in exp psy are

- typically hierarchical.
- optimally modeled using hierarchical models.

At least we should fit models to individual participants' data.

:)