Bayesian Cognitive Modeling: Drift-Diffusion Models

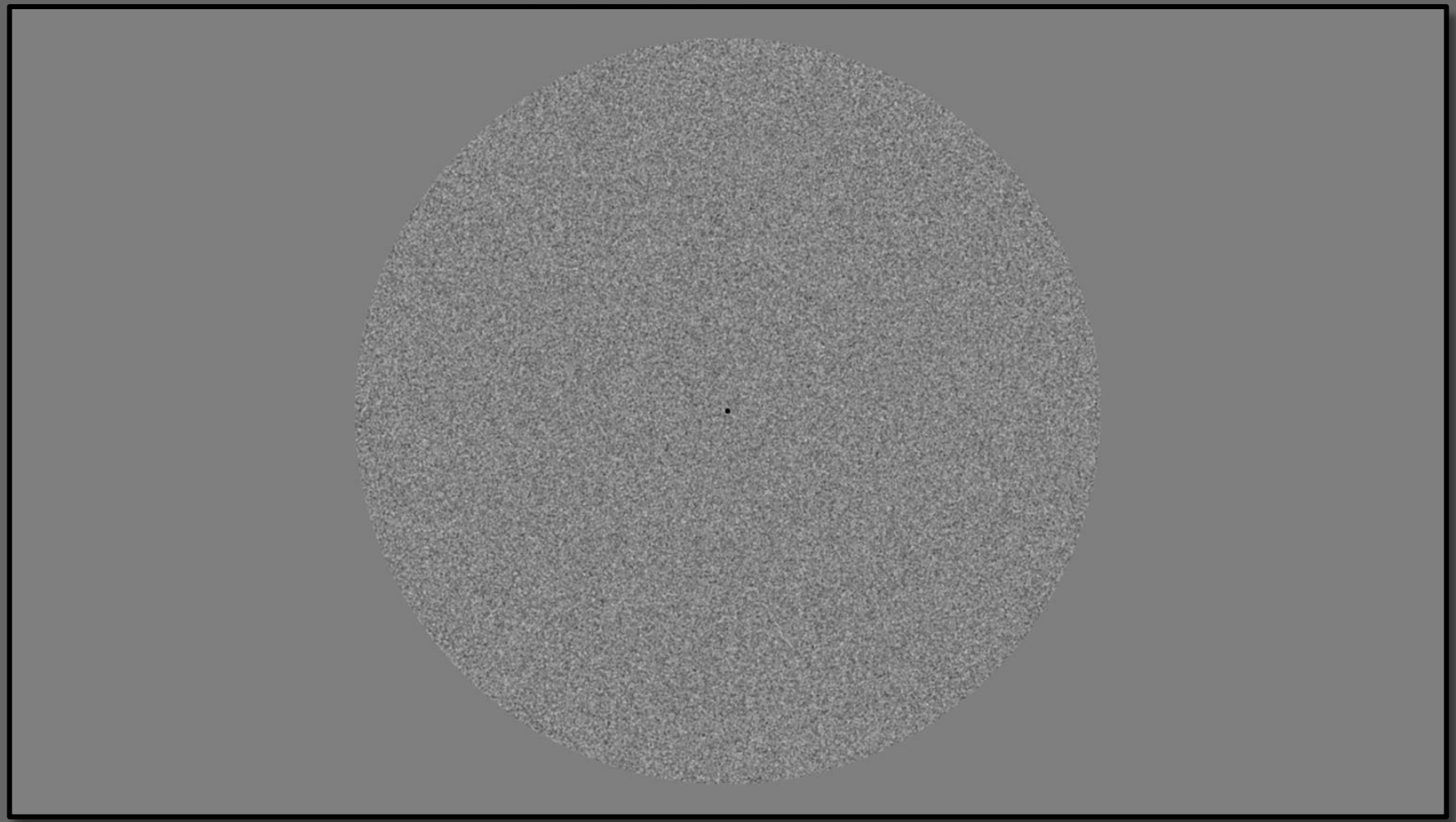
Michael D. Nunez

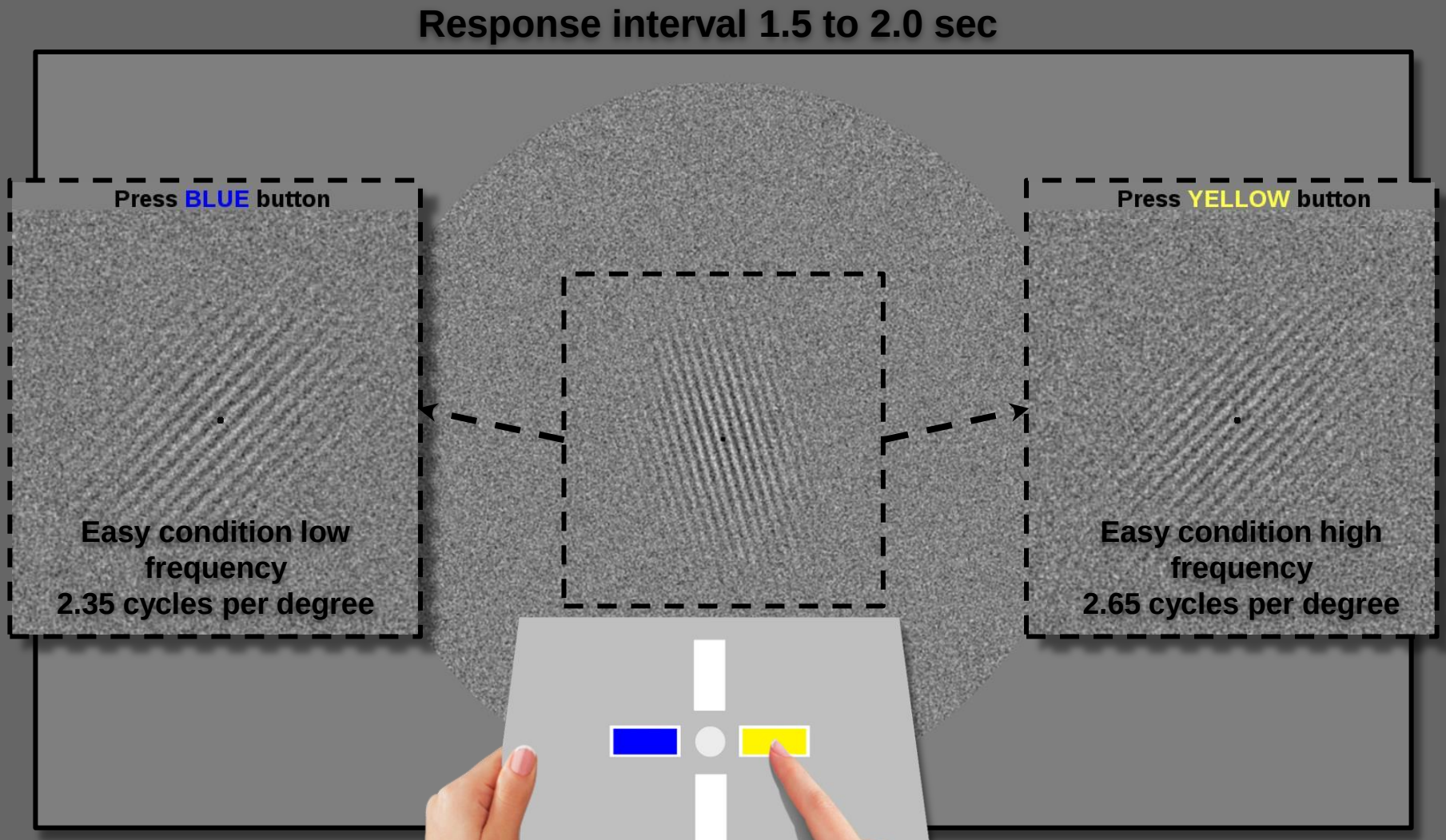
29-August-2022

Bayesian Modeling in brms

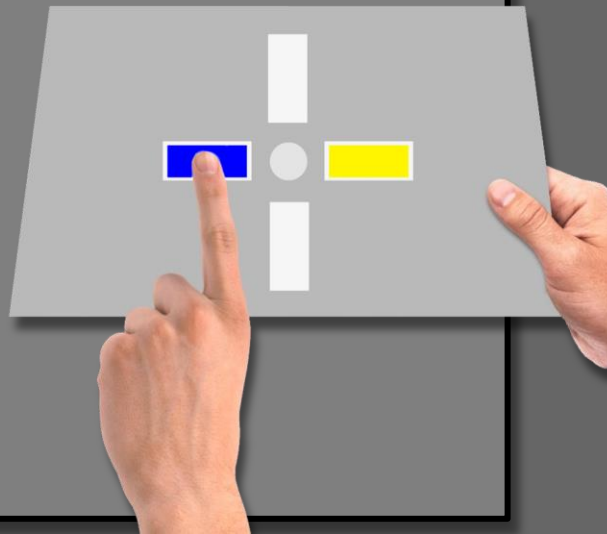
An example experiment

Cue interval 0.5 to 1.0 sec




Response interval 1.5 to 2.0 sec
Press BLUE button
Easy condition low frequency
2.35 cycles per degree
Press YELLOW button
Easy condition high frequency
2.65 cycles per degree

0.25 sec to collect additional responses & 1.5 to 2.0 sec ISI



Obtaining the real data

```
# install.packages('curl')
library(curl)

# See https://github.com/mdnunez/encodingN200
pdmdat <-curl("https://tinyurl.com/PDMdataESCOP2022")
pdm <- read.csv(pdmdat)

colnames(pdm) <- c('N200_latencies', 'N200_amplitudes', 'RT', 'accuracy', 'condition',
'EEG_session', 'experiment', 'session', 'subject')

pdm <- pdm[pdm$experiment == 1, ]

pdm$N200_latencies <- pdm$N200_latencies/1000
pdm$RT <- pdm$RT/1000

head(pdm)
```

Real data from this experiment

	Roos	Eduardo	Yasemin
Accuracy	67.7%	76.4%	79.2%
Mean Response Time (Mean RT)	769 ms	896 ms	938 ms

Which participant is ***better*** at the task? Roos? Eduardo? Yasemin?

Roos is the fastest at this task!

...But **Yasemin** gave the most correct responses!

...But **Eduardo** gave many accurate responses and was somewhat fast!

Real data from this experiment

	Roos	Eduardo	Yasemin
Accuracy	67.7%	76.4%	79.2%
Mean Response Time (Mean RT)	769 ms	896 ms	938 ms

What if each participant had been told to respond as fast as possible?

What if each participant had been told to respond as accurately as possible?

What if participants were awarded € based on a **Scoring Rule**, such that 10c was awarded for 1% point of accuracy, only if the mean RT was less than 900 ms?

Individual Differences and The Speed-Accuracy Tradeoff

	Roos	Eduardo	Yasemin
Accuracy	67.7%	76.4%	79.2%
Mean Response Time (Mean RT)	769 ms	896 ms	938 ms

- We suspect there are individual differences in ability.
- But we also suspect that Roos ignored instructions and decided to answer quickly, while Yasemin preferred to answer accurately.
- How do we compare the ***ability*** of Roos and Yasemin given that they had two different ***strategies*** related to the **speed-accuracy tradeoff**?

Response time distribution (RT) shapes

Normal (Gaussian) distribution?

No, RT distributions have a right skew

"Inverse-Gaussian" distributions are better

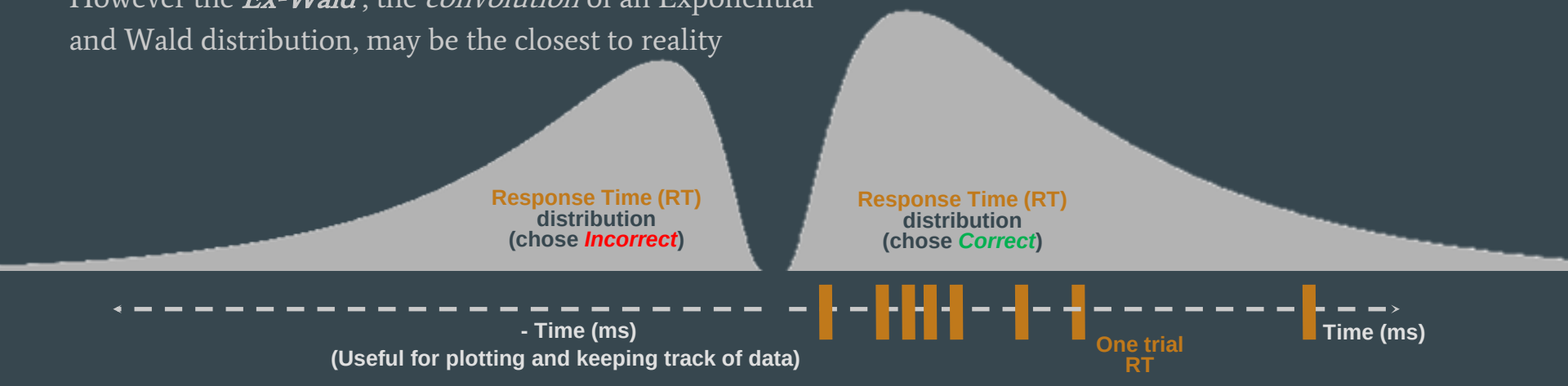
Python:

stats.invgauss?

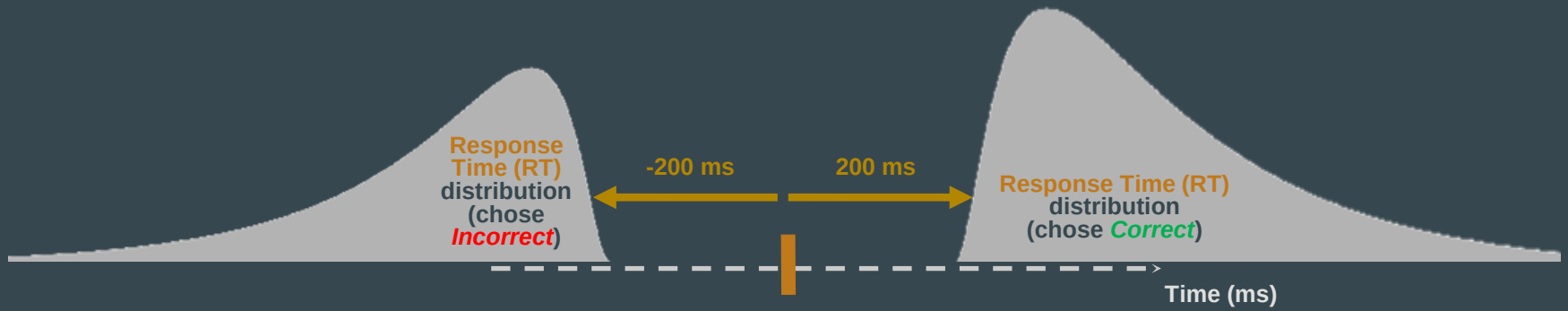
R:

install.packages('actuar'); **library('actuar');** help(rinvgauss)

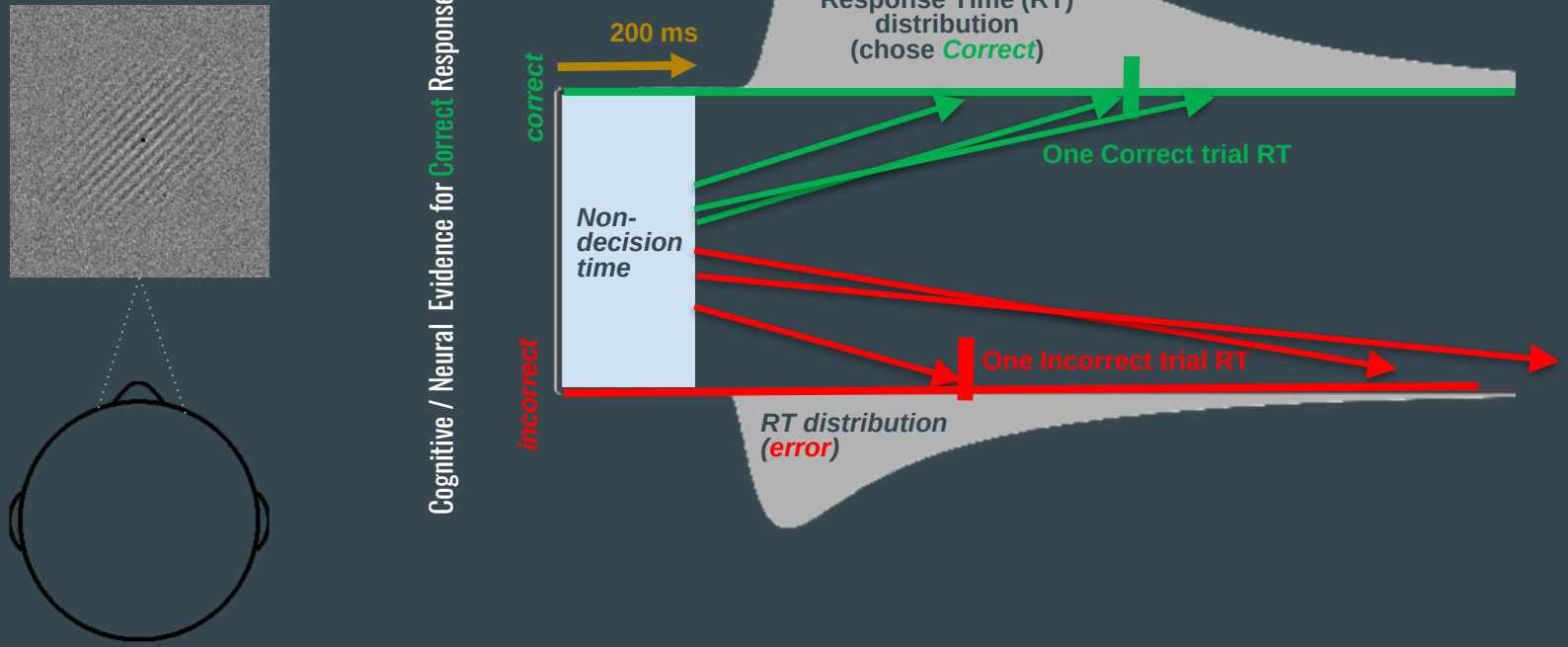
However the ***Ex-Wald***, the *convolution* of an Exponential and Wald distribution, may be the closest to reality



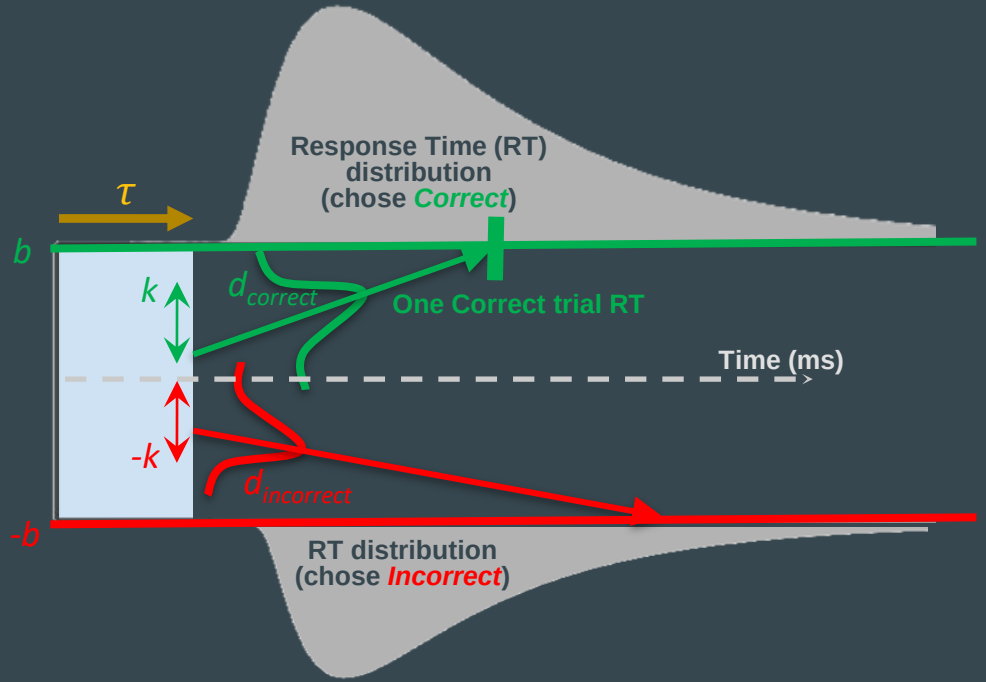
Can one model describe both accuracies and response times?



Simulating a race between 2 linear evidence accumulators…



Simulating 2 Linear Ballistic Accumulators (LBA model)



- b : boundary
- k : start point
- $d_{correct}$: drift rate to correct
- $d_{incorrect}$: drift rate to incorrect
- τ: non-decision time
- Simple geometry:
  - $d = (b - k) / (RT - \tau)$
  - $RT = \tau + (b - k) / d$

Try at home: Simulating 2 Linear Ballistic Accumulators (LBA model)

```
boundary = 1.2  # Boundary in evidence units
k_max = .5  # Start-point in evidence units
ntrials = 500
set.seed(1)
ndt = 0.2  # Non-decision time, 200 ms
rts = rep(0, ntrials); correct = rep(1, ntrials)
for(n in 1:ntrials)
{
    drift_correct = rnorm(1,1.5, .5)  # drift rate to correct, evidence units per second
    drift_incorrect = rnorm(1,1, .5)  # drift rate to incorrect, evidence units per second
    k_correct = runif(1, 0, k_max)
    k_incorrect = runif(1, 0, k_max)
    # Correct for linear ballistic being strange when all drift rates < 0
    if((drift_correct < 0) & (drift_incorrect < 0))
    {
        correct[n] = NA
        rts[n] = NA
    } else {
        rt_correct = ndt + (boundary - k_correct) / drift_correct
        rt_incorrect = ndt + (boundary - k_incorrect) / drift_incorrect
        # Only valid first boundary passes
        if(((rt_incorrect < rt_correct) & (drift_incorrect > 0)) || (drift_correct < 0))
        {
            rts[n] = rt_incorrect
            correct[n] = 0  # Incorrect response
        } else {
            rts[n] = rt_correct
        }
    }
}
x11()
hist((correct * 2 - 1) * rts, breaks=20)  # Typical method of plotting choice-RTs
mean(correct, na.rm=TRUE)  # Ignores NAs
mean(rts, na.rm=TRUE)
```

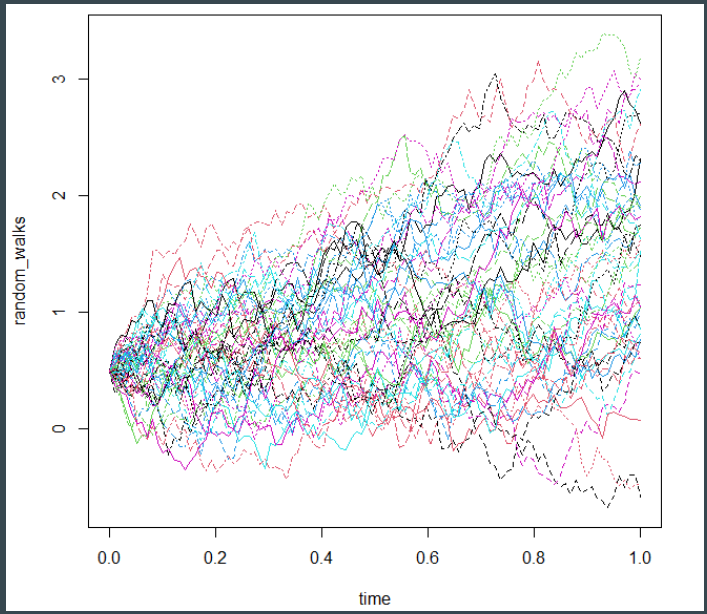
Random Walks (R code)

A random walk process is a model in which the next step of a process comes from a distribution which is then added to the result of the previous step.

Let us assume that a participant collects evidence in one experimental trial for the low or high spatial frequency target that follows a random walk.

```
set.seed(1)
nwalks = 50; nsteps = 100; step_length = .01
time = seq(0, 1, length.out=nsteps)
random_walks = matrix(0, nrow=nsteps, ncol=nwalks)
for(w in 1:nwalks)
{
   this_random_walk = 0.5
   for(s in 2:nsteps)
   {
      evidence_units = rnorm(1, 0.01, 0.1)
      this_random_walk[s] = this_random_walk[s-1] + evidence_units
   }
   random_walks[,w] = this_random_walk
}

matplot(time, random_walks,type='l')
```



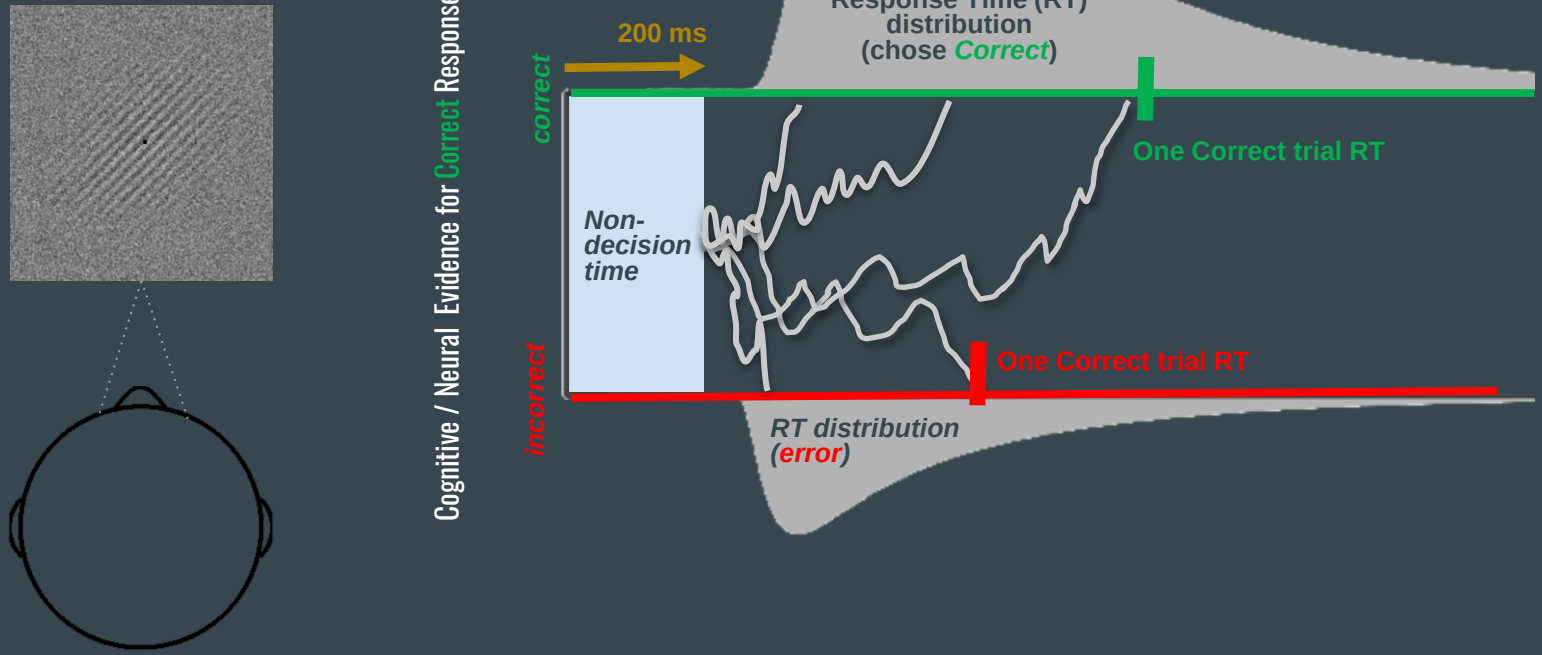
Do it yourself: Random Walks

- Change the standard deviation of the random walk to 0.001, what happens? How would you describe this graph?
- Change the standard deviation of the random walk to 1, what happens? What has changed compared to the original figure?

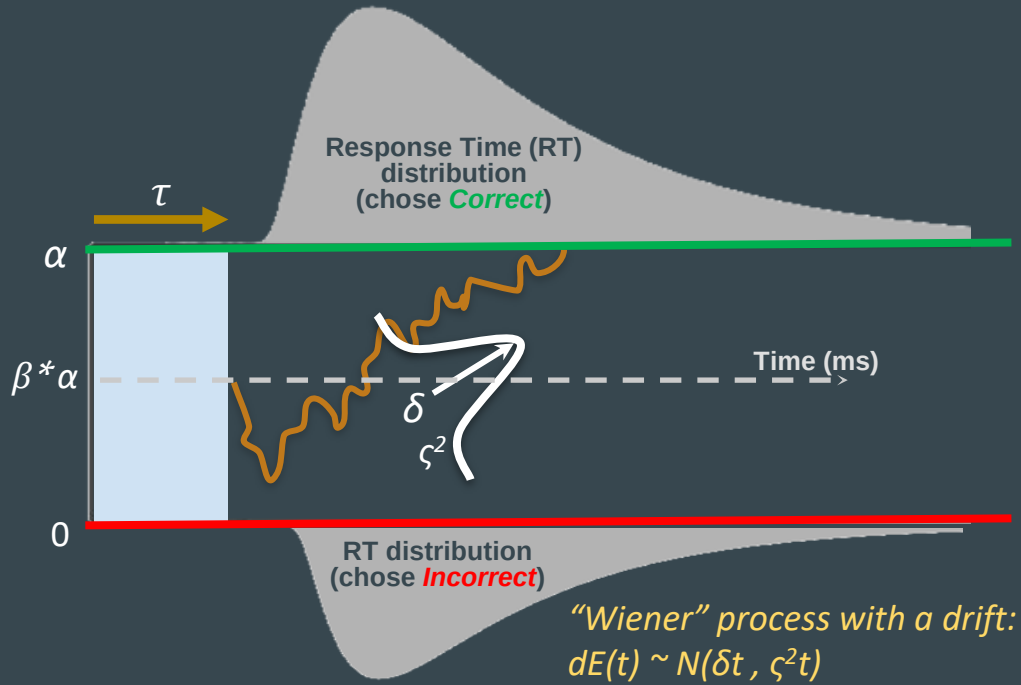
Answers

- Answer: All random walks trend upwards in almost a linear equation. These result in ballistic accumulation
- Answer: The scale of the y-axis is now about 10 times larger than the original graph

Simulating one accumulator with a random walk…



Simulating Drift-Diffusion Models (DDMs)



"Wiener" process with a drift:
$dE(t) \sim N(\delta t, \varsigma^2 t)$

Stochastic Differential Equations, Compare to Lecture 3

- α : boundary (speed-accuracy tradeoff)
- δ: drift rate (ability parameter)
- τ: non-decision time
- β: relative start point
  - Fixed at 0.5 in our simulation
- ς : Diffusion coefficient
  - Fixed at 1 in our simulation.
  - Almost all model *fitting* procedures assume this is fixed

Simulating Drift Diffusion Model (DDM) in R

```
ntrials = 200; nsteps = 300; step_length = .01
time = seq(0, nsteps*step_length, length.out=nsteps)
boundary = 1.2; drift = 1.5; ndt = 0.2; dc = 1
random_walks = matrix(0, nrow=nsteps, ncol=ntrials)
rts = rep(0, ntrials); correct = rep(1, ntrials)
set.seed(1)
for(n in 1:ntrials)
{
   random_walks[1, n] = 0.5*boundary  # relative start point = 0.5
   for(s in 2:nsteps)
   {
      random_walks[s, n] = random_walks[s-1, n] + rnorm(1, drift*step_length, dc*sqrt(step_length))
      if(random_walks[s, n] >= boundary)
      {
         random_walks[-(1:(s-1)), n] = boundary
         rts[n] = s*step_length + ndt
         correct[n] = 1
         break
      } else if (random_walks[s, n] <= 0) {
         random_walks[-(1:(s-1)), n] = 0
         rts[n] = s*step_length + ndt
         correct[n] = 0
         break
      } else if (s == nsteps) {
         correct[n] = NA; rts[n] = NA
         break
      }
   }
}
x11(); matplot(time, random_walks,type='l')
x11(); hist((correct * 2 - 1) * rts, breaks=20)
mean(correct, na.rm=TRUE); mean(rts, na.rm=TRUE)
```

Fitting DDMs to data

- To *fit* a model is to discover a set of parameter estimates (or parameter uncertainties using Bayesian methods) that best describe the *data* given the model architecture

- Fitting models to data is a useful method to test hypotheses by either
  - (1) by directly estimating and then evaluating parameters, like comparing parameter estimates across participants
  - (2) comparing multiple models' ability to describe data.

Fitting DDMs to data using EZ Diffusion

See:
Wagenmakers, van der Maas, Grasman (2007)

```
ezdiff = function(rt, correct, s=1.0)
{
    if (length(rt) <= 0) stop('length(rt) <= 0')
    if (length(rt) != length(correct)) stop('RT and correct unequal lengths')
    if (max(correct, na.rm=TRUE) > 1) stop('Correct values larger than 1')
    if (min(correct, na.rm=TRUE) < 0) stop('Correct values smaller than 0')
    pc = mean(correct, na.rm=TRUE)
    if (pc <= 0) stop('Mean accuracy less than or equal to 0')
    # subtract or add 1/2 an error to prevent division by zero
    if (pc == 1.0) {pc=1 - 1/(2*length(correct))}
    if (pc == 0.5) {pc=0.5 + 1/(2*length(correct))}
    MRT = mean(rt[correct == 1], na.rm=TRUE)
    VRT = var(rt[correct == 1], na.rm=TRUE)
    if (VRT <= 0) stop('RT variance less than or equal to 0')
    r=(qlogis(pc)*(((pc^2)*qlogis(pc)) - pc*qlogis(pc) + pc - 0.5))/VRT
    drift=sign(pc-0.5)*s*(r)^0.25
    boundary=(s^2 * qlogis(pc))/drift
    y=(-1*drift*boundary)/(s^2)
    MDT=(boundary/(2*drift))*((1-exp(y))/(1+exp(y)))
    ndt=MRT-MDT
    return(list(boundary, drift, ndt))
}
est_params = ezdiff(pdm$RT, pdm$accuracy); est_params[1]; est_params[2]; est_params[3]
```

Fitting DDMs to data using brms

For more information see this blog post by Henrik Singmann: http://singmann.org/wiener-model-analysis-with-brms-part-i/

```
#Define the formula for brms
ddm_formula <- bf(RT | dec(accuracy) ~ 1,
                  bs ~ 1,
                  ndt ~ 1,
                  bias = 0.5)
# Remove contaminant RTs
pdm_reduced <- pdm[pdm$RT > 0.2, ]
# Define the initial values for each Markov chain.
# Note that brms and Stan do not have good default start values for DDMs.
# Also note that we enforce the starting value of NDT to be less than the minimum RT
mcmc_initials <- function() {
  list(
    Intercept = runif(1, -4, 4),
    Intercept_bs = runif(1, 0.5, 2),
    Intercept_ndt = runif(1, 0, min(pdm$RT))
  )
}
bayes_ddm <- brm(ddm_formula, family=wiener(link_bs ="identity",link_ndt="identity"), init=mcmc_initials, data=pdm_reduced)
summary(bayes_ddm)
```

Do it yourself: Fitting DDMs

- Fit the real PDM data to both the EZ Diffusion model and the brms (Stan) implementation of the DDM.
- Do you draw the same conclusions? What do you get from the Bayesian DDM that you do not get from the EZ Diffusion?

Bonus:

Fit the DDM such that all parameters change by condition