Computational Cognitive Science 2018-2019

Tutorial 1: Basic Model Building

Experiment

In this tutorial we will implement our first computational model. We will explore a model that was proposed 50 years ago to describe how people act in speeded-choice tasks. To get an idea of what a speeded-choice task is and what data the model is supposed to capture, you might want to try our implementation of Smith & Vickers. In the original experiment (see Smith and Vickers 1988) participants had to perform many trials under controlled conditions, so this application is not a completely accurate representation of the experiment. If you would like to see how you performed, write the id number at the bottom of the page, e.g., in "Your id is ab12345c-6def-7gh8-ij99-9k9l999m999n".

Getting Started

Please make sure you have the files associated with this tutorial; they can be downloaded from the "materials" column on the tutorials page. The tutorials in this course will be using the R programming language. We recommend you to use the RStudio environment to interact with the scripts and code assignments in the tutorials. RStudio environment offers a very convenient way to access help, documentation, as well as the output of your scripts. You can access RStudio on your DICE account by executing `rstudio` in your command prompt. RStudio and R are generally well-documented online, but you have a question you can't easily answer, feel free to post it to the piazza forum.

The document you are reading is itself based on an R-markdown file, which you can find in the associated file (`tutorial1.Rmd`). If you want to follow along with this document in R, open the `tutorial1.Rmd` file in RStudio. You can execute code lines with `CTR + Enter` or whole code chunks with `CTR + Shift + Enter`. Alternatively, you can also copy and paste the code snippets in this tutorial in your own script file or try them directly in the R-console. **Please make sure that the R is set to the correct file path, otherwise it might complain about not finding the scripts and data files.** One way to make sure that all paths are correct is to start a new project which contains all files for each tutorial.

In the first part of the tutorial we will explore how the random-walk model can account for speeded response tasks like the one in Smith and Vickers (1988). We will step you through all parts of the first model in this document. To follow along, please execute the code chunks in this `tutorial1.Rmd` file, or follow along by reading and modifying the corresponding `rw.r` file. This first part covers much of the topics already introduced in lecture 2 of the course. If you already feel familiar with R and the topics covered in lecture 2 feel free to skim this part.

In part two of the tutorial you will be provided a version of the random-walk model that allows trial-to-trial variability. Please try to answer the questions and tasks provided in this document before the tutorial session. Questions and exercises will be presented in **bold** subsections.

We will finish the tutorial by having a look at the data from our online experiment. We will return to this data set in future tutorials.

Part 1: The Random-Walk Model

To define the random walk model we first assign two variables (`nreps` and `nsamples`) that determine the general behavior of the simulation. The number of random walks (`nreps`) corresponds to the number of decisions that are performed in the task. In the web version of the experiment this number varied, but in the original experiment there was a fixed number of trials. The number of samples (`nsamples`) determines how much evidence is sampled for each of those decisions.

```
nreps <- 10000
nsamples <- 2000
```

These variables are considered *fixed* for this model, since the number of decisions is fully determined by the number of trials in the experiment and the amount of evidence sampled is assumed to be constant.

Psychological Variables

The drift rate (`drift`) determines how much evidence systematically influences the walk at each time step, i.e., the larger the drift rate is the more the random walk is nudged towards one of the boundaries. A drift rate of zero would mean that no evidence is available and decisions are completely random.

```
drift <- 0.0
```

We then specify how noisy the samples for the evidence are via the standard deviation (`sdrw`).

```
sdrw <- 0.3
```

Finally, we set a response criterion (`criterion`). This value determines how far the two decision boundaries are from the origin (the point of zero evidence).

```
criterion <- 3
```

Storing Model Results

We will need to store the results of our simulations for further analysis and plotting. For this we create 2 vectors (`latencies, responses`) and a two-dimensional matrix (`evidence`). The evidence matrix will contain all simulated decisions (`nreps`), where each decision contains all samples for these decisions (`nsamples`).

```
latencies <- rep(0,nreps)
responses <- rep(0,nreps)
evidence <- matrix(0, nreps, nsamples+1)
```

Running the Model

Now it's time to run the model. Here we loop through the number of decisions (`nreps`). For each decision we first calculate the evidence by using `rnorm` command in `R`. We concatenate the results of `rnorm` to a vector with a leading zero `c(0, ...)`, representing that at the beginning of the simulation we have no evidence. And calculate the evidence for the decisions.

The result of these computations in `[1]` (below) is a random walk towards (and beyond) one of the evidence boundaries. We now use the random walk to simulate a decision `[2]` (below). We need to determine when the random walk first crossed a response boundary. For this we use `which(abs(evidence[i,])>criterion)[1]`[1]. This returns the index at which the simulated decision is taken (`p`).

[1] We calculate *all* positions where the absolute evidence is larger than the criterion and then discard all but the first element.

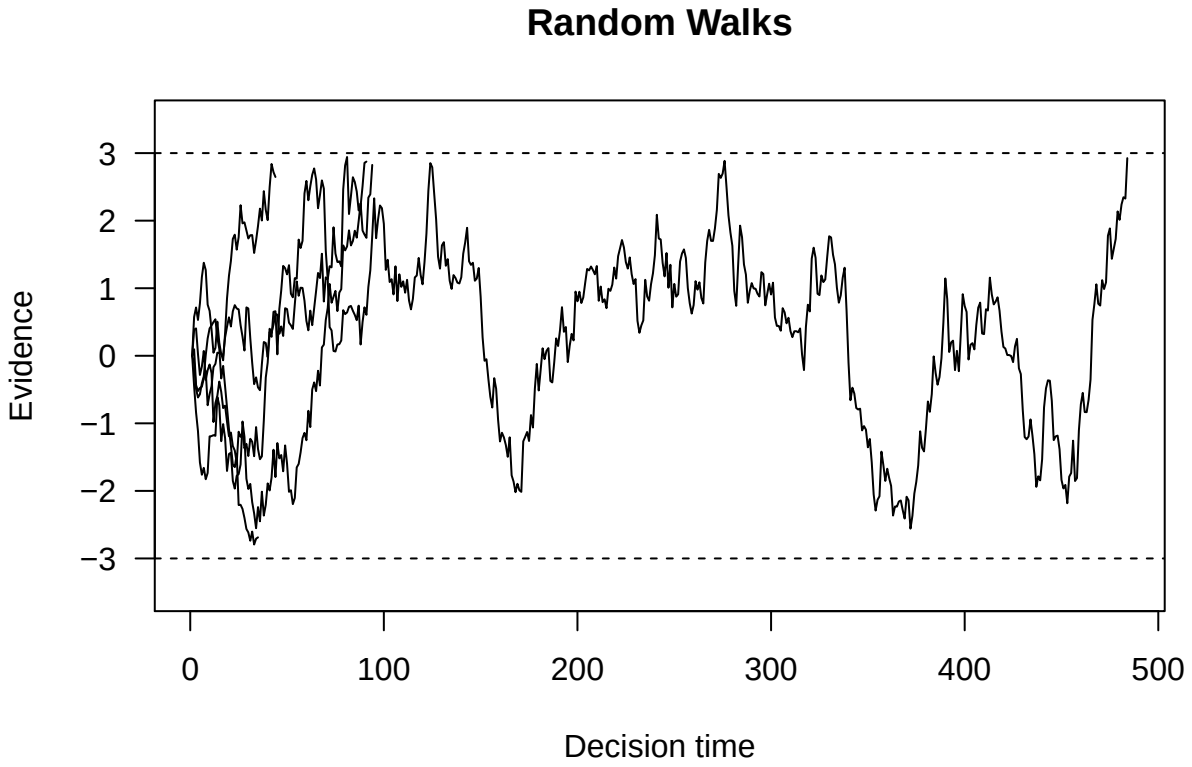
We store the index of the decision in our `latencies` and `responses` vectors. Finally, we need to determine what decision was made, i.e. which decision boundary was crossed (left or right choice in Smith and Vickers (1988)). We get the `sign` of the value at the index at which we first crossed the decision boundary.

Our full model looks like this:

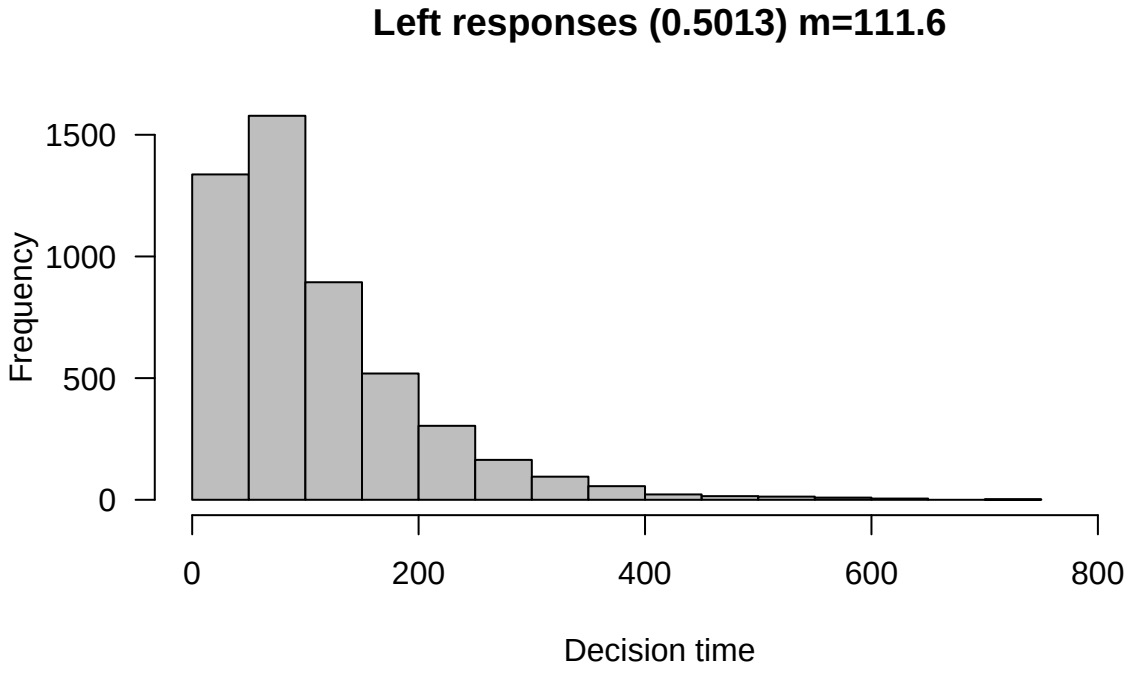
```
for (i in c(1:nreps)) {
  evidence[i,] <- cumsum(c(0,rnorm(nsamples,drift,sdrw)))  # [1] calculating the evidence
  p <-  which(abs(evidence[i,])>criterion)[1]              # [2] taking the decision
  latencies[i]  <- p                                       # storing the time
  responses[i] <- sign(evidence[i,p])                      # storing the direction
}
```

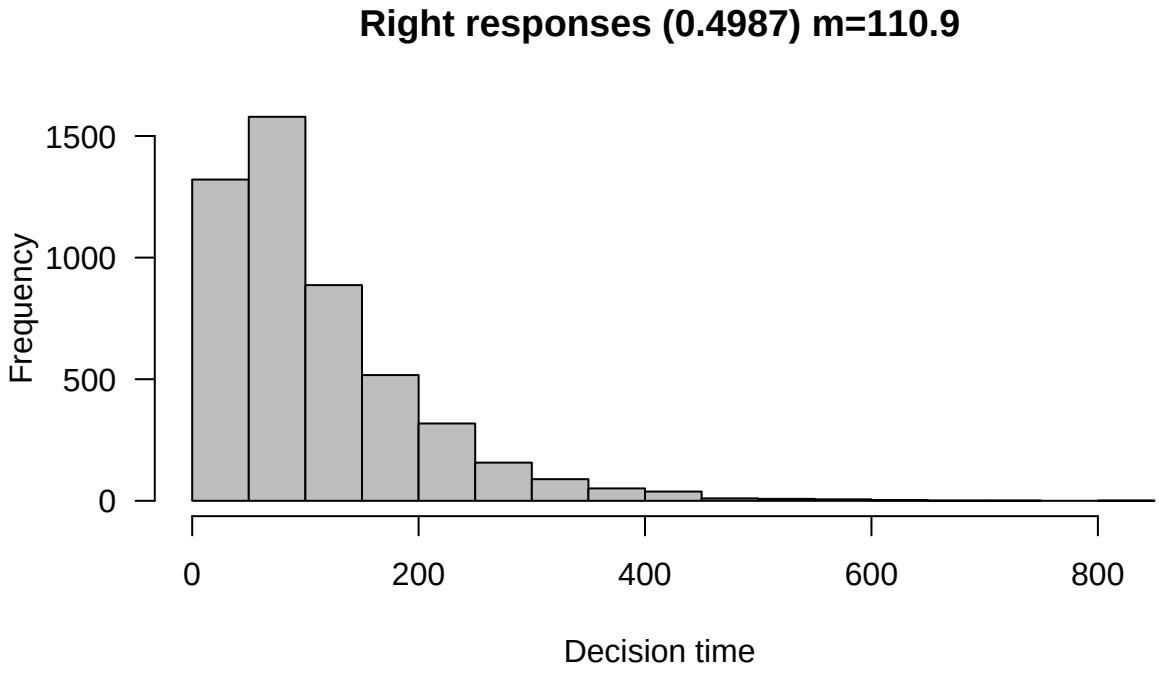
Examining the Model

Let's see what the model does. First we visualize the random walks for 5 trials.



Now let's summarize the responses the model generates. We will plot histograms of the response times for both left and right responses.





Part 2: Trial-to-Trial Variability

Before introducing trial-to-trial variability, let's get an sense of how the model predictions change for different parameters.

Exercise: Change the drift rate for the model in Part 1 to 0.03. What is the result of the new drift rate on decision times?

Solution/notes:

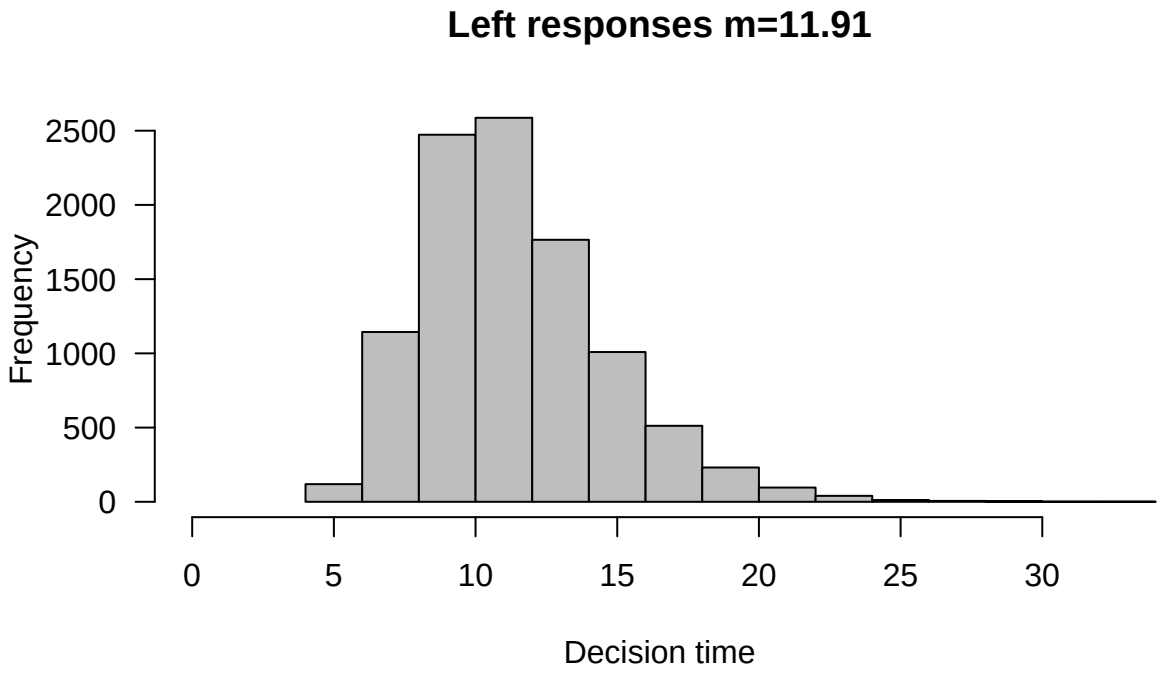
The reaction times are much faster (around 12). This drift is so extreme that there might not be any 'right' responses. In that case, the histogram code you used above may throw an error about an invalid number of breaks, because there are more breaks than items in the second one. The code below uses a function to check the length and plots an empty the histogram if there isn't anything to plot.

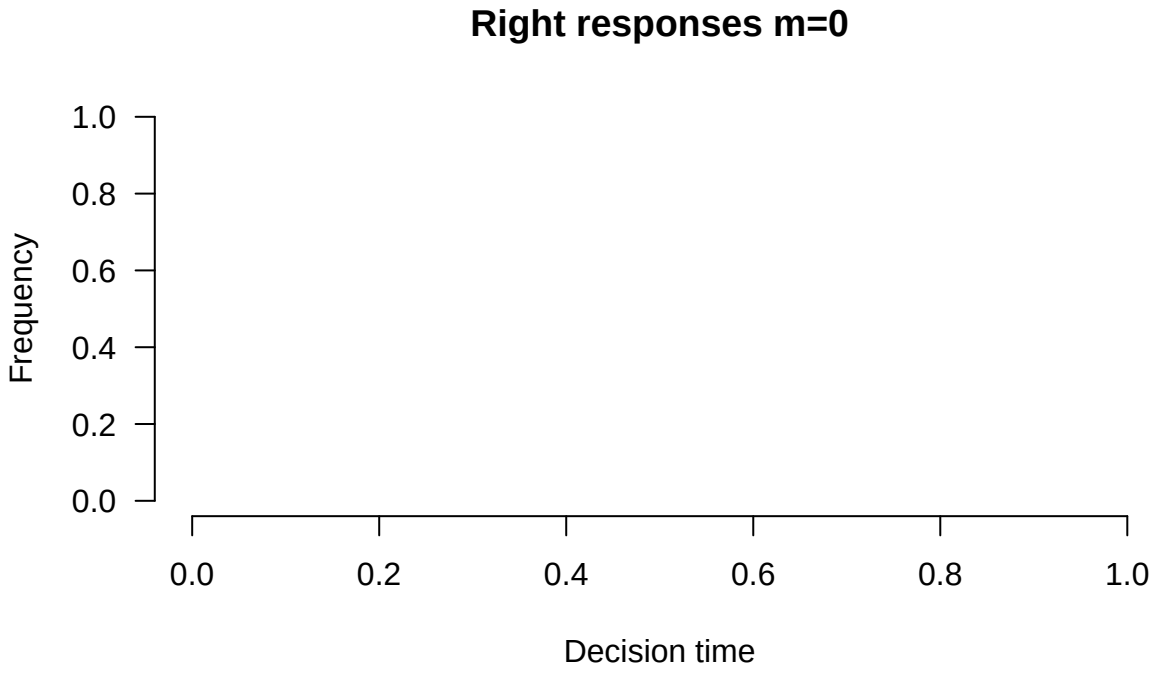
```
drift <- .3

for (i in c(1:nreps)) {
  evidence[i,] <- cumsum(c(0,rnorm(nsamples,drift,sdrw)))  # [1] calculating the evidence
  p <-  which(abs(evidence[i,])>criterion)[1]              # [2] taking the decision
  latencies[i]  <- p                                       # storing the time
  responses[i] <- sign(evidence[i,p])                      # storing the direction
}

rtHist <- function(rts,latencies,sideLabel) {
  nreps <- length(latencies)
  if(length(rts) > 0) {
    hist(rts,col="gray",
      xlab="Decision time", xlim=c(0,max(latencies)),
      main=paste(sideLabel," responses m=",as.character(signif(mean(rts),4)),
          sep=""),las=1)
  } else {
    #No responses on this side, so plot an empty histogram instead
    plot(NULL,type="h",bty="n", xlim=c(0,1), ylim=c(0,1),
         ylab="Frequency", xlab="Decision time",
         main=paste(sideLabel," responses m=0",sep=""),las=1)
  }
}

par(mfrow=c(2,1))
toprt <- latencies[responses>0]
rtHist(toprt,latencies,"Left")
botrt <- latencies[responses<0]
rtHist(botrt,latencies,"Right")
```

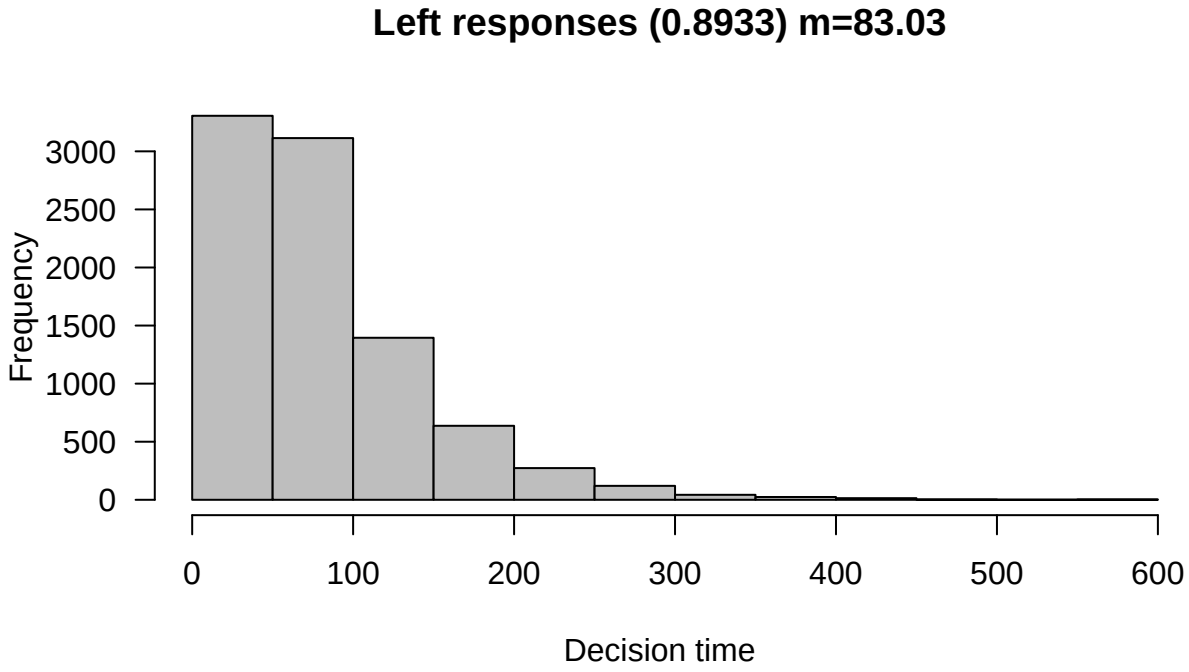


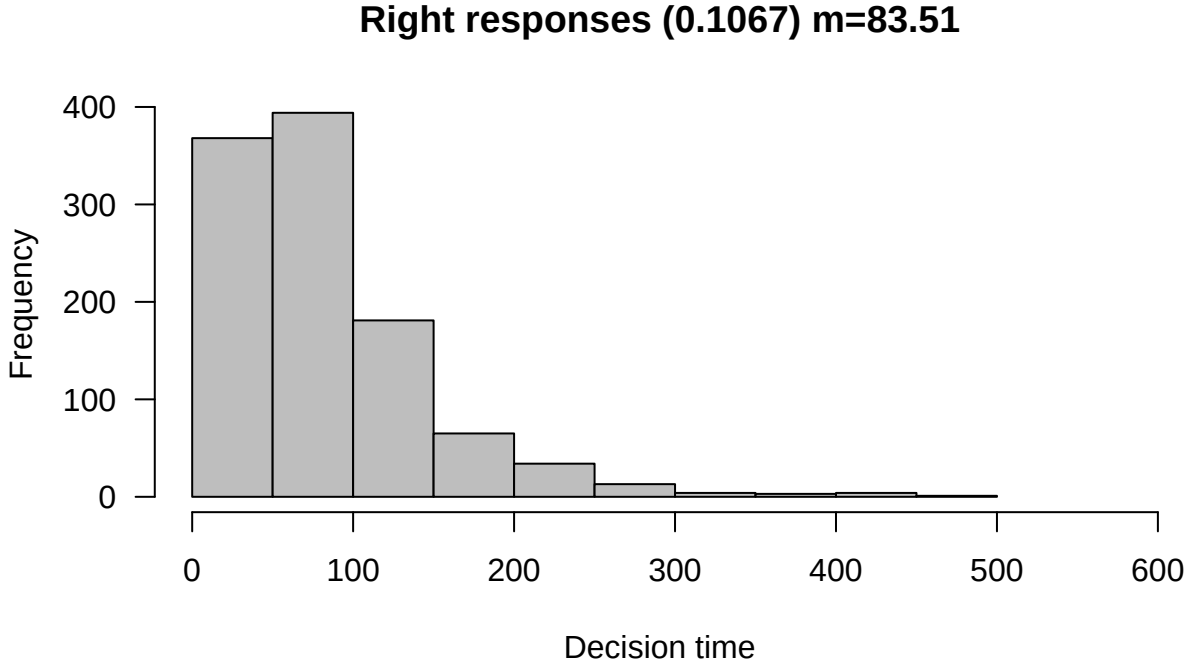


Exercise Updated: Change the drift rate for the model in Part 1 to 0.03. What is the result of the new drift rate on decision times?

Solution/notes:

The proportion of left to right responses is now strongly in favour of left responses. However, the reaction times do not differ much.





Our model predicts that both errors and correct reposes exhibit the same reaction times. Try to remember when you did the experiment earlier - do you think that the decision times for correct vs. incorrect responses were similar? How do you think did your response times reflect the difficulty of the task?

It has been found that response times in speeded response tasks exhibit both fast and slow error patterns. Under time pressure, participants often exhibit fast errors when the discriminability is high. When the task is more difficult and more time is available errors can be slow.

"When you need to decide whether a traffic light is red or green, the few errors you commit

will likely be fast. When you need to differentiate between a breaburn and a corland apple, by contrast, you may commit numerous errors and they will tend to be slow" (F&L) referring to (Luce 1963)

Both slow and fast errors can be explained by our model if we allow trial-to-trial variability. This variability is different from the noise(`sdrw`), as it allows parameters to change between different trials. This variability is based on the assumption that physical or psychological circumstances in an experiment are not constant, but can differ considerably.

Question: Did you experience any variability while doing the experiment?

Solution/notes:

People's experiences may be different, if they've tried the experiment. One pattern is for there to be occasional fast errors even in the condition where the evidence is unambiguous. This admits mulitiple explanations, including the "starting point" one offered in F&L.

Two parameters have been found to have a powerful impact on the model's prediction:

- Variability in the starting value of the random walk.
- Variability in the drift rate of the model.

We will now explore a version of the random-walk model that permits variability for both drift rate and starting values. We only have to make minor changes to our previous code to allow trial to trial variability. We introduce two standard deviations for starting values and drift rate and store them in an array (`t2tsd <- c(...)`).

```
nreps <- 1000           # Same as before
nsamples <- 2000        # Same as before

drift <- 0.00           # Same as before
sdrw <- 0.3             # Same as before
criterion <- 3          # Same as before
t2tsd  <- c(0.1,0.01)
```

Then, when we run the model instead of using a fixed drift or starting value we sample values from a normal distribution with the corresponding standard deviation (e.g. for the starting position `sp <- rnorm(1,0, t2tsd[1])`).

```
latencies <- rep(0,nreps)   # Same as before
responses <- rep(0,nreps)   # Same as before
evidence <- matrix(0, nreps, nsamples+1)  # Same as before

for (i in c(1:nreps)) {
  sp <- rnorm(1,0,t2tsd[1])
  dr <- rnorm(1,drift,t2tsd[2])
  evidence[i,] <- cumsum(c(sp,rnorm(nsamples,dr,sdrw))) # Same as before
  p <-  which(abs(evidence[i,])>criterion)[1]           # Same as before
  responses[i] <- sign(evidence[i,p])                   # Same as before
  latencies[i]  <- p                                    # Same as before
}
```

The Effect of Varying Starting Points and Drift

Now that you have seen the new model, explore the effects of different starting points on the model output. To answer the questions below, copy the `rwt2.r` file from the tutorial files and modify the code accordingly.

Exercise: What happens if we introduce variability for the starting point? Set the standard devation for the starting point to `0.8` and keep the drift rate fixed: `t2tsd <- c(0.8,0.0)`. Run the `rwt2.r` script and inspect the output of the plots. Can you explain why incorrect responses are now faster than correct responses?

Solution/notes:

The histogram suggests that there are some fast errors – the distribution falls off more quickly as time increases in the incorrect-response histogram. If you make the starting place sd very high (e.g., 3) this becomes more obvious, as many of the decisions are made before the random walk begins. The textbook describes this phenomenon reasonably well.

Exercise: Now consider variable drift rates for fixed starting points. Create a new copy of `rwt2.r` and modify your code with `t2tsd <- c(0.0,0.025)`. What are the results on response times for correct and error responses?

Solution/notes:

The incorrect responses are slower on average in this case. When drift is lower, the overall time is likely to be longer, and the probability of an incorrect response increases.

Looking at Real Data

Let's finish this tutorial by having a look at the data from our online version of Smith and Vickers (1988). If you run the `load_data.r` script provided in the tutorial files you will get access to the dataset `data` in your global environment. Have a look at the shape of the data by calling `View(data)` in the console. As you can see the data contains 6 columns:

- `sessionId`: identifier for participants.
- `mean`: Mean angle of the lines displayed, negative numbers indicate tilt to the left, positive numbers tilt to the right.
- `participantChoice`: "left" or "right"
- `RT`: The reaction time in milliseconds
- `orientation`: Tilt of the mean lines displayed, either "left" or "right"
- `correct`: Was the participant choice correct (1) or false (0)

Exercise: Explore and plot the data. You might want to look at some simple features of the data set. How many trials did the participants complete on average? How does average performance change for different values of `mean`? How about the reaction times? How do reaction times vary between participants? How do RT and accuracies differ for correct and incorrect trials?

Solution/notes:

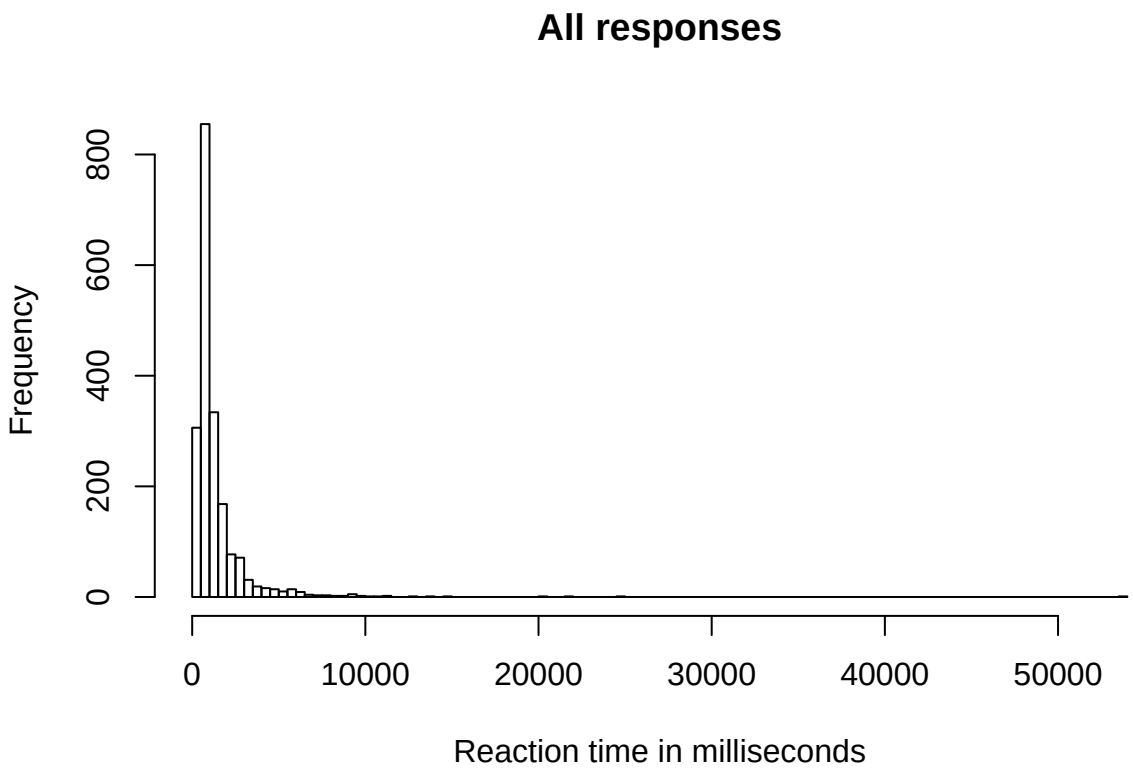
We can start by looking at some high-level statistics of the data set. For example we can compare the number of correct and total responses (`sum(data$correct)` and `length(data$correct)`). R provides a simple tool to get lots of summary statistics, via `summary(data)`.

```
source('load_data.r')

summary(data)
```

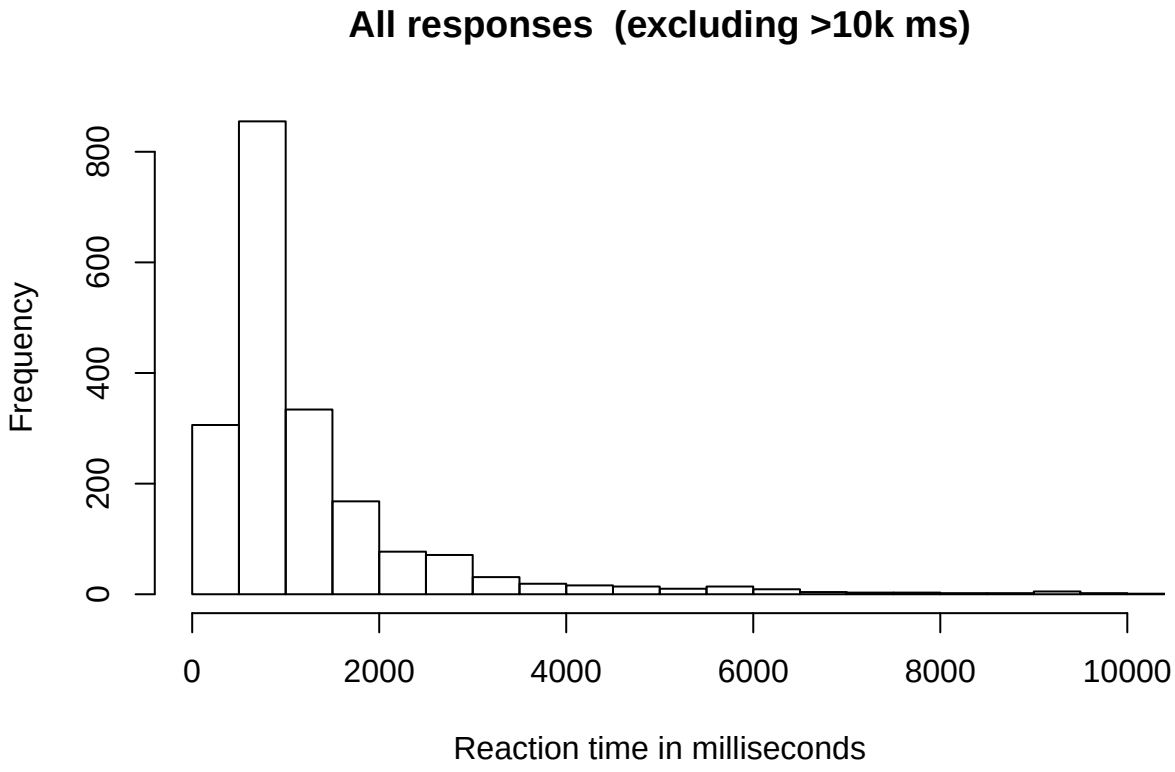
```
##   sessionId              mean          participantChoice        RT
##  Length:1956        Min.   :-22.500   Length:1956        Min.   :   22
##  Class :character   1st Qu.:-12.500   Class :character   1st Qu.:  551
##  Mode  :character   Median :  2.500   Mode  :character   Median :  832
##                     Mean   :  0.409                      Mean   : 1358
##                     3rd Qu.: 12.500                      3rd Qu.: 1424
##                     Max.   : 22.500                      Max.   :53676
##  orientation           correct
##  Length:1956        Min.   :0.0000
##  Class :character   1st Qu.:1.0000
##  Mode  :character   Median :1.0000
##                     Mean   :0.8594
##                     3rd Qu.:1.0000
##                     Max.   :1.0000
```

```
hist(data$RT,breaks=100,xlab="Reaction time in milliseconds",
     main="All responses")
```



Someone took 50 seconds! What about responses that take <10 seconds?

```
hist(data$RT,breaks=100,xlab="Reaction time in milliseconds",
  main="All responses  (excluding >10k ms)",xlim=c(0,10000))
```

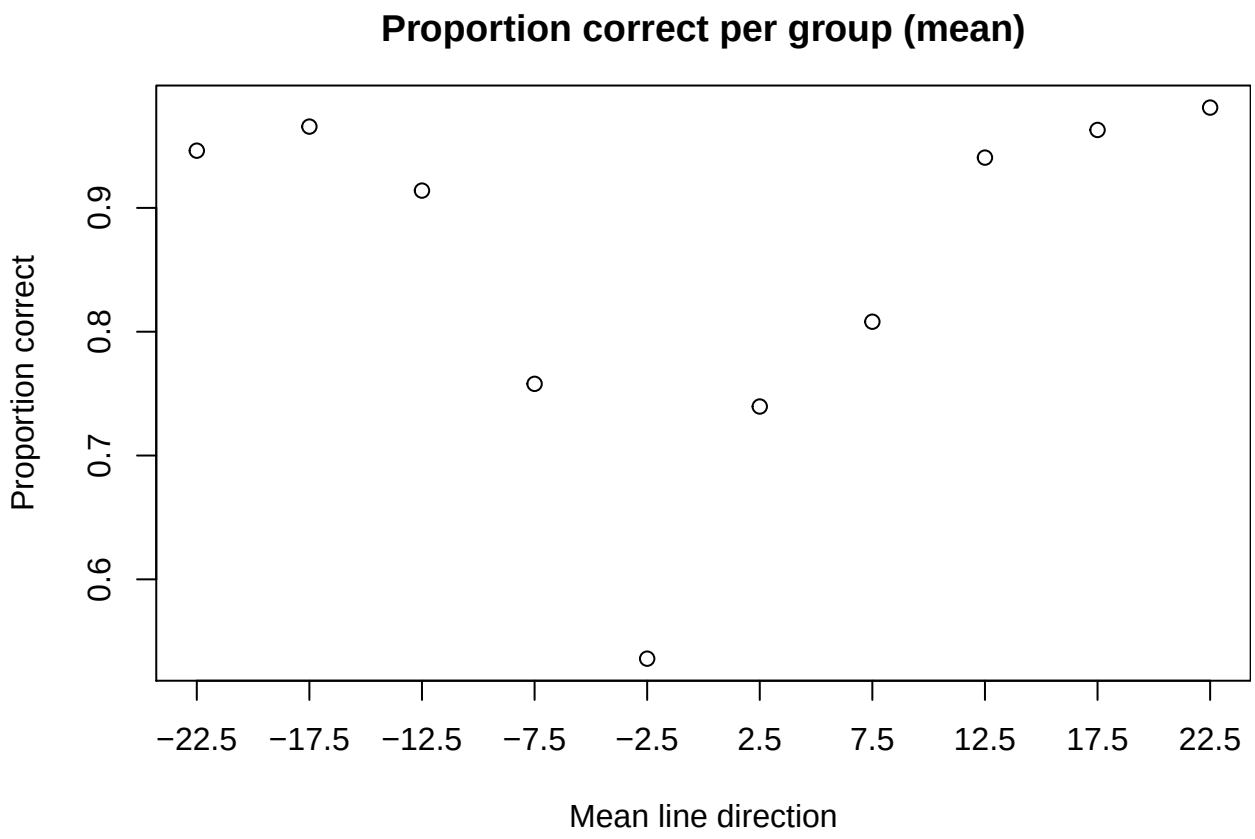



```
nParticipants <- length(unique(data$sessionId))
print(paste("Average responses per session (rounded):",
            round(length(data$RT)/nParticipants)))
```

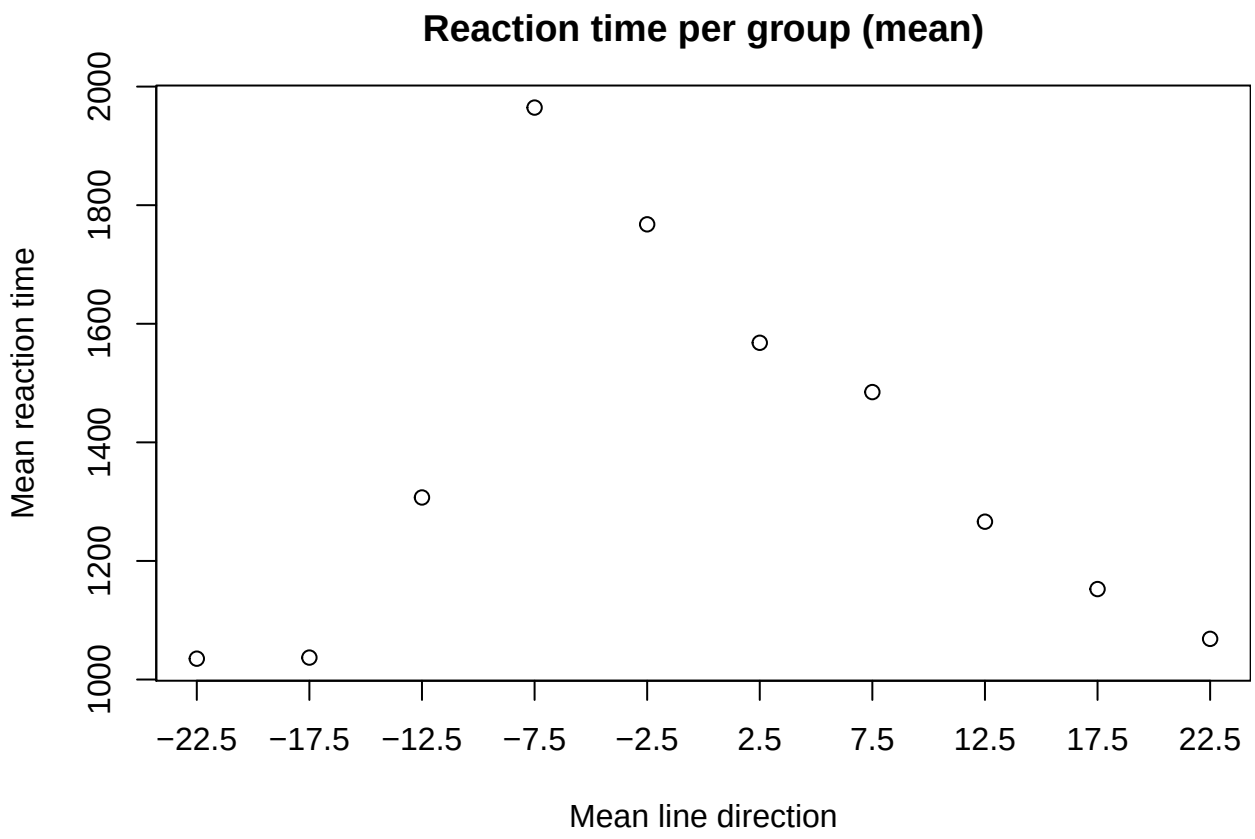
```
## [1] "Average responses per session (rounded): 19"
```

How do we look at average performance per value of "mean"? Googling "R average by" produces a pointer to a stack overflow answer about "aggregate"

```
#data$correct is 1 if correct, 0 if incorrect
#so the average corresponds to proportion correct
correctByMean <- aggregate(data$correct, list(data$mean), mean)
par(mar=c(4,4,3,0))
plot(correctByMean$Group.1,correctByMean$x,xlab="Mean line direction",
     main="Proportion correct per group (mean)",
     ylab="Proportion correct", xaxt="n")
axis(side=1, at=correctByMean$Group.1)
```

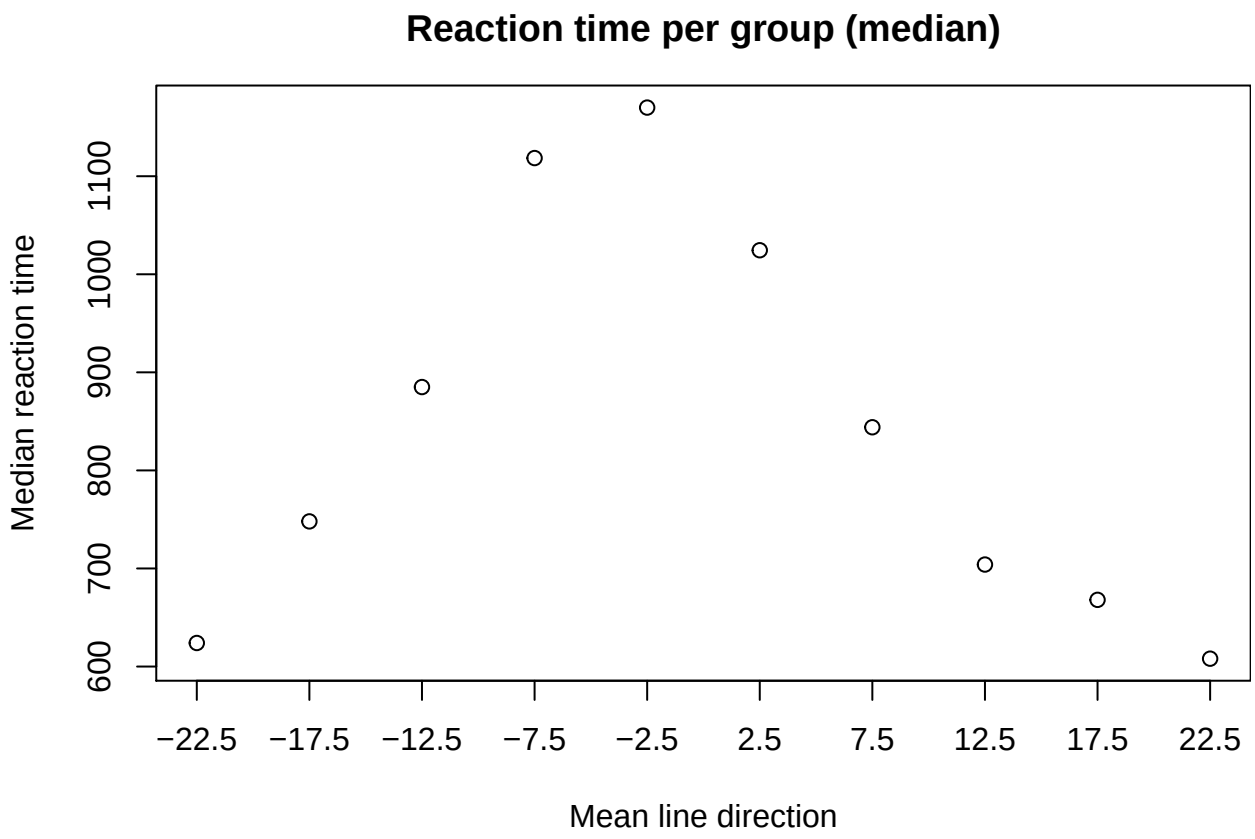



```
averageByMean <- aggregate(data$RT, list(data$mean), mean)
par(mar=c(4,4,3,0))
plot(averageByMean$Group.1,averageByMean$x,xlab="Mean line direction",
     main="Reaction time per group (mean)",
     ylab="Mean reaction time", xaxt="n")
axis(side=1, at=averageByMean$Group.1)
```



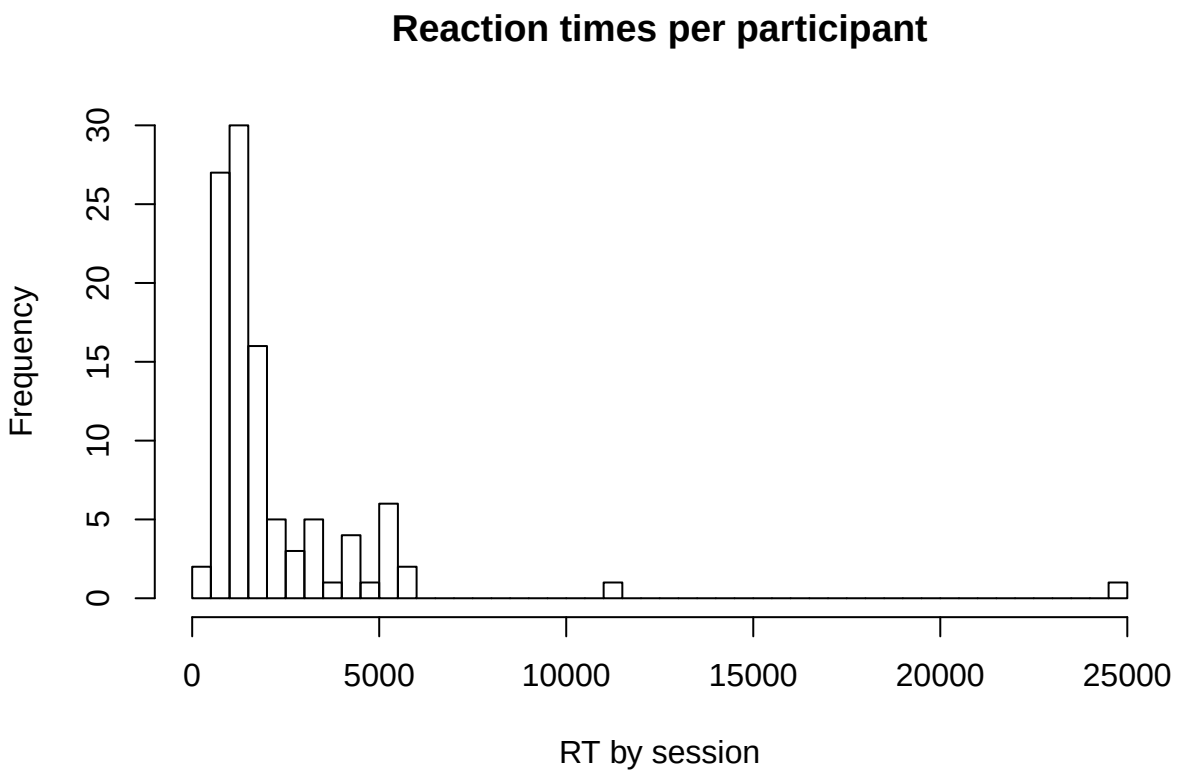
We might also want to look at the median:

```
medianByMean <- aggregate(data$RT, list(data$mean), median)
par(mar=c(4,4,3,0))
plot(medianByMean$Group.1,medianByMean$x,xlab="Mean line direction",
     main="Reaction time per group (median)",
     ylab="Median reaction time", xaxt="n")
axis(side=1, at=medianByMean$Group.1)
```

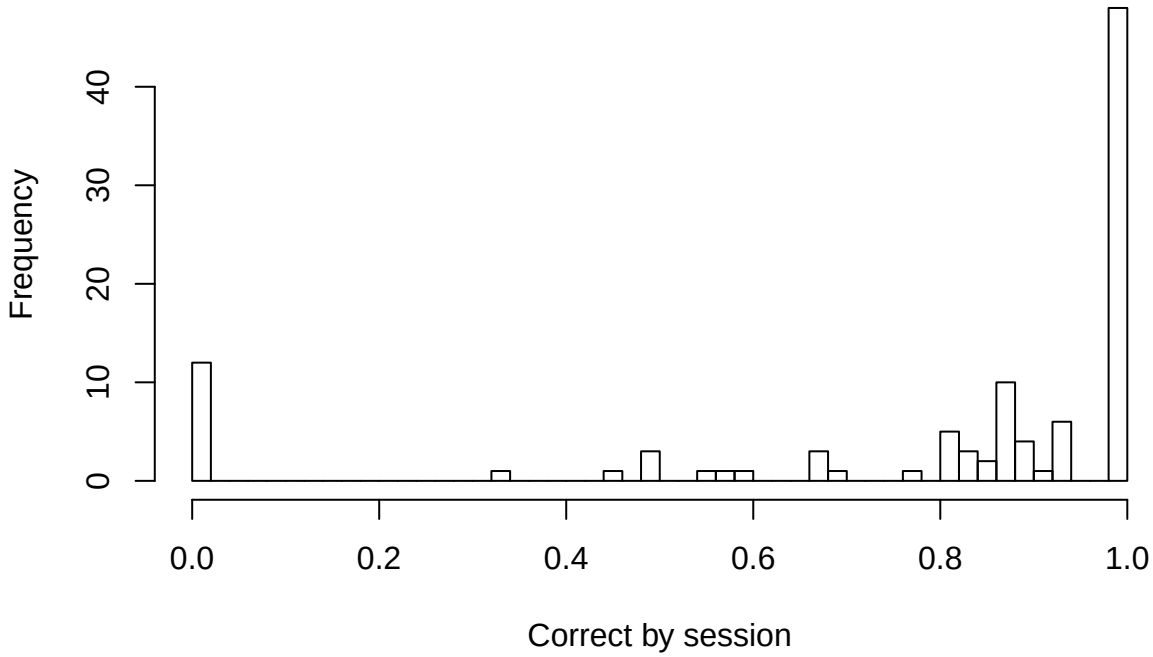


How much do participants vary?

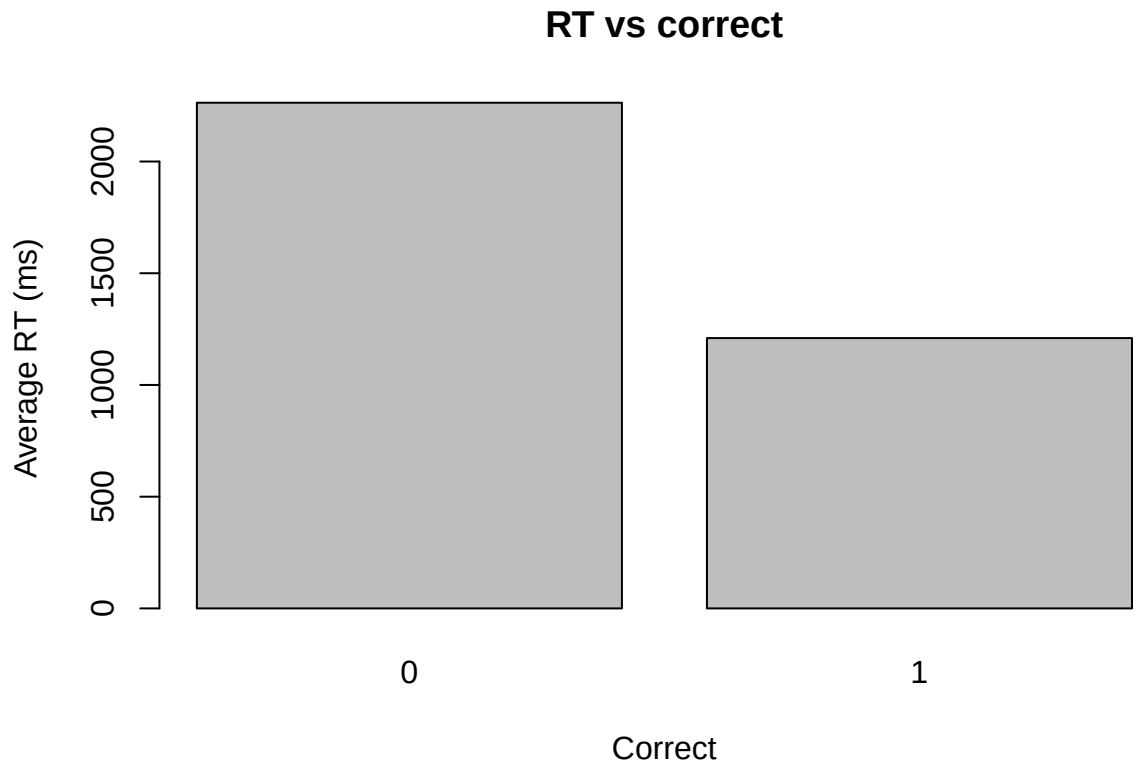
```
rtBySession <- aggregate(data$RT, list(data$sessionId), mean)
hist(rtBySession$x,breaks=40,main="Reaction times per participant", xlab="RT by session")
```




```
correctBySession <- aggregate(data$correct, list(data$sessionId), mean)
hist(correctBySession$x,breaks=40,main="Proportion correct per participant",
     xlab="Correct by session")
```

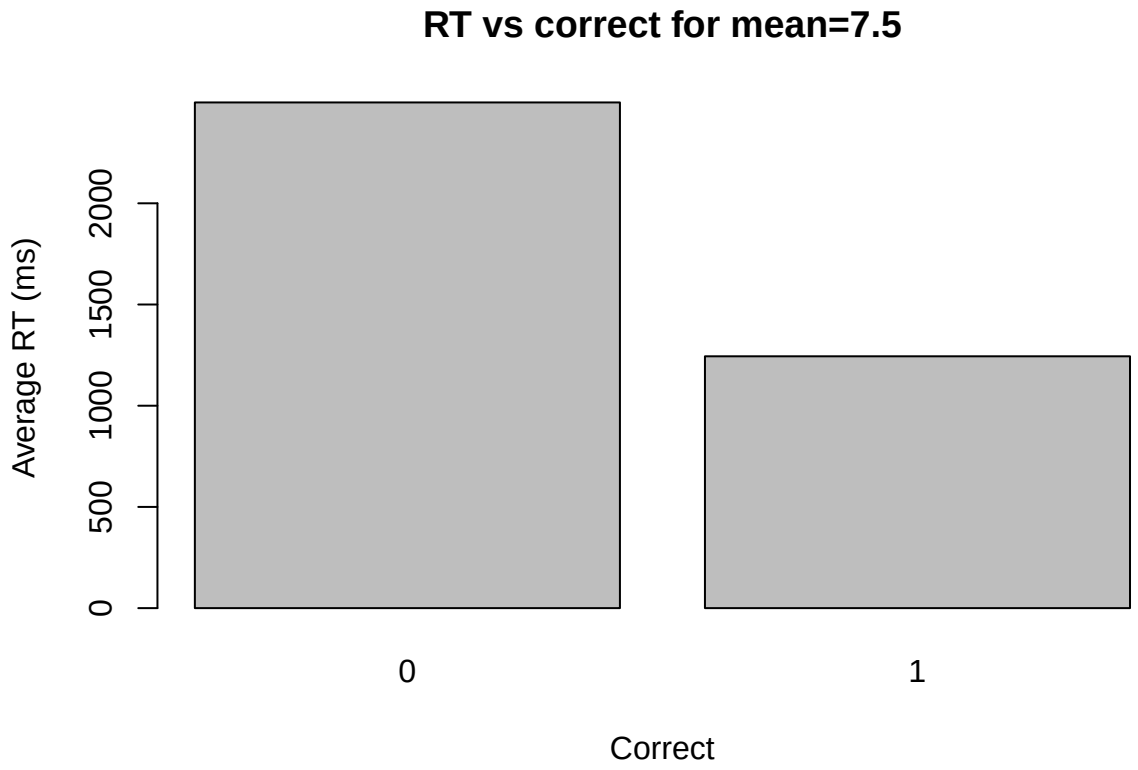


```
rtByCorrect <- aggregate(data$RT, list(data$correct), mean)
barplot(rtByCorrect$x,names.arg=rtByCorrect$Group.1,
        main="RT vs correct",xlab="Correct",ylab="Average RT (ms)")
```



What if we just look at one mean line direction?

```
just75 = data[data$mean==7.5,]
rtByCorrect75 <- aggregate(just75$RT, list(just75$correct), mean)
barplot(rtByCorrect75$x,names.arg=rtByCorrect75$Group.1,
        main="RT vs correct for mean=7.5",xlab="Correct",ylab="Average RT (ms)")
```



Optional: If you want to have a look at the performance and RTs in all subgroups have a look at the provided `extraplots-tut1.r` *script.*

General Cognitive Modeling Concepts

The aim of this part of the tutorial is to discuss key concepts in cognitive modeling with your tutor and your peers, in order to deepen your understanding and to clarify any things that are unclear.

Question: In this tutorial we explored the effects of different parameters on the model's predictions. We chose these parameter values by hand; what are pros and cons of doing this? What's a good alternative? Why?

Solution/notes:

Choosing parameter values and looking at how well the model does with those choices is a good place to start: it forces us to think about what parameter values are plausible, and why. If one then fits parameters and gets dramatically different values, that can shed light on places where our intuitions or assumptions are poorly-matched to our data. If we subsequently want to conduct a Bayesian data analysis, our prior distributions should probably have modes or medians that are close to these parameter choices. At the same time, we rarely want to stop with choosing parameters by hand. Even if we're happy with our initial choice, we probably want to know how much better our model can do if we fit parameters, how well our optimized parameters correspond to our a priori choices, and how sensitive our model is to our choices.

Question: Explain the concept of *goodness of fit* of a model. How does it relate to *parameter estimation*? What role does the *discrepancy function* play in this context?

Solution/notes:

Goodness of fit is a measure of how well the predictions (or post-hoc "predictions") of our model match our data. In choosing a discrepancy function, we are choosing a specific way to quantify goodness of fit. A common convention in discrepancy functions is that they are equal to zero when the fit is perfect. For example, RMSD is zero when all predictions are exactly equal to all observed data, and negative log likelihood is zero when the probability of the data set is 1 under our model. Parameter estimation involves choosing parameters values that minimize the discrepancy.

Question: Farrell and Lewandowsky (2018) distinguish between data descriptions and process models. If we didn't already have a process model, how would you go about quantitatively describing the data we have? How would this be useful?

Solution/notes:

There isn't a single right answer to this question; students are likely to have interesting and complementary ideas about this. One general approach could be to visualize the data and see whether the data have a familiar form. For example, there are several probability distributions that resemble the empirical distributions in our data set, e.g., the gamma distribution, the ex-Gaussian distribution, and the log-normal. If we find that one of these distributions matches our data very well, we can think about other phenomena that have these distributions and draw analogies to them to construct process models.

Question: Does the random walk model account for your experience in the experiment? How would you improve it?

Solution/notes:

Reasonable people might disagree on this point. There are certainly some patterns in the data that the random walk model as given doesn't anticipate, but perhaps there are simple ways to extend it. It might be worth discussing the importance of individual differences, and whether it would be fruitful to extend a random-walk model to accommodate them.

References

Farrell, Simon, and Stephan Lewandowsky. 2018. *Computational Modeling of Cognition and Behavior*. Cambridge University Press.

Luce, Duncan, R. 1963. *Response Times Their Role in Inferring Elementary Mental Organization*. Oxford University Press.

Smith, Philip L., and Douglas Vickers. 1988. "The Accumulator Model of Two-Choice Discrimination." *Journal of Mathematical Psychology* 32 (2). Elsevier: 135–68. https://doi.org/10.1016/0022-2496(88)90043-0.