Computational Cognitive Science 2019-2020

Tutorial 2: Parameter Estimation

Changelog:

- **2019.10.11**: Typos, notation/argument names.
- **2019.10.03**: Initial release for 2019.

Files in this tutorial's materials:

```
manifest.txt
data.csv
tutorial2.pdf
tutorial2.Rmd
tutorial02.Rproj
```

General Information

The exercises in this tutorial are designed so that you can do them by directly editing the R-markdown (.Rmd) file. If you want to execute code in R-markdown you need to put your code in a R-cell like so:

```
2+3
```

```
## [1] 5
```

You can run cells in R-markdown using the little green "play" arrow in the cell, the "run" button at the top of the editor window, or keyboard shortcuts visible from the run button's menu. Running a cell will show its output directly below the cell. Alternatively, you can generate pdf or html documents by executing `Knit` (in the toolbar in R-studio), which is a great way to communicate your results.

You can also copy your code into a script file and execute those or simply perform one-off calculations in the R console. In this case, be sure you're running R commands from the **console** window, rather than the **terminal** window.

Package: GGplot2 for Data Visualization

This tutorial will use ggplot2, a powerful library for data visualization. It doesn't come with base R, so we'll have to install it – although if you're using a laboratory computer it may already be there.

This command will load the library if it's available:

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

If that command gave a warning message, you need to install the package. To do that, run the command below, then try `require(ggplot2)` again.

```
install.packages("ggplot2", dependencies = TRUE)
```

GGplot2 follows a specific approach to data visualization known as the 'grammar of graphics.' We will use examples from the library here without going in-depth into the overall principles, but you may find it helpful to refer to the package's documentation.

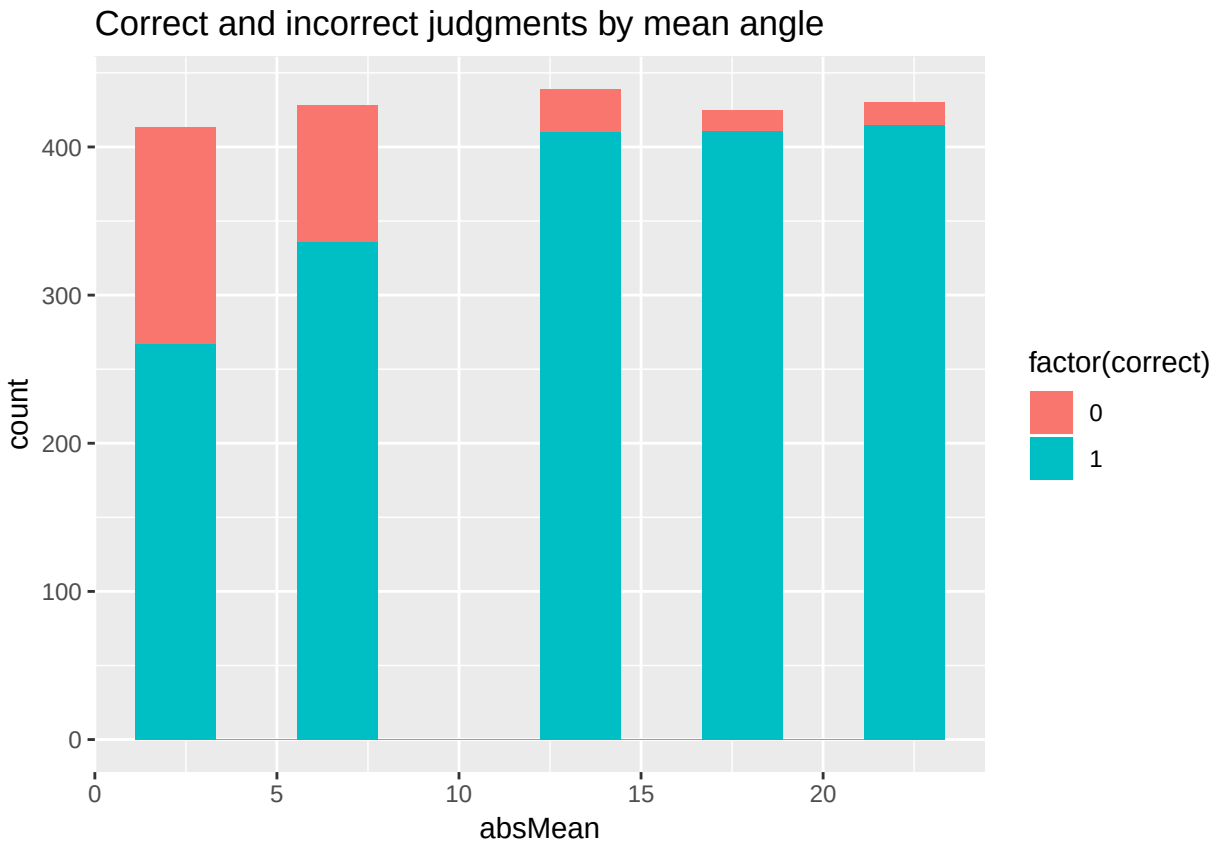
Fitting Reaction Times

We will continue working with our replication of the Smith and Vickers (1988) data. To simplify our task somewhat we will only look at data for the mean angle of 22.5 degrees. We will also ignore if the participants' responses were correct or not. This is a debatable decision, especially if one wants to provide a general model of the cognitive process underlying the task. However, it should be less of an issue for this particular condition, as most responses for this angle should be correct.

```
data <- read.csv("data.csv", header = TRUE)
```

To verify our hunch that most of the 22.5 angle judgments were correct, we can visualize the distribution of correct and incorrect judgments for each angle. The code below produces a histogram of judgments and steps through some key elements of data visualization with ggplot2 along the way.

```
ggplot(data) + # the core 'ggplot' function takes the data as input
  aes( # 'aesthetics' mapping: which aspects of the data will we plot?
    x=absMean, # the values of the 'absMean' column will be plotted on the x axis
    fill=factor(correct)) + # the 'correct' column will set the color fill of the bars
      # ^ the 'factor' function converts from numbers (0,1) to strings ("0","1")
  geom_histogram( # 'geometry' layer: what kind of plot is this?
    bins=10) + # specify a sensible number of bins for this histogram
  ggtitle("Correct and incorrect judgments by mean angle") # add title
```



***Exercise*: Use exact counts to confirm that most of the responses in the mean angle 22.5 condition were correct. One nice way to get counts in R is to use the `table()` command.**

***Exercise*: The plot above clearly shows a relationship between accuracy, in terms of number of correct responses, and average angle of the stimuli. Can you think of other factors in the data which might have some relation to mean accuracy? If so, try to make a plot with these factors to check.**

We now store all rows for which the mean angle was 22.5 degrees (both positive and negative angles). This is the data set we will be working with in this tutorial.

```
data.22 <- data[data$absMean == 22.5,]
```

In this tutorial we will focus exclusively on the decision times (or reaction times [RTs]). Before doing any more advanced analysis it is a good idea to look at the data.

***Exercise*: Plot the distribution of decision times. You can modify the plotting code above.**

***Exercise*: If our model of RTs predicts a distribution - what are some important qualities that this distribution should have?**

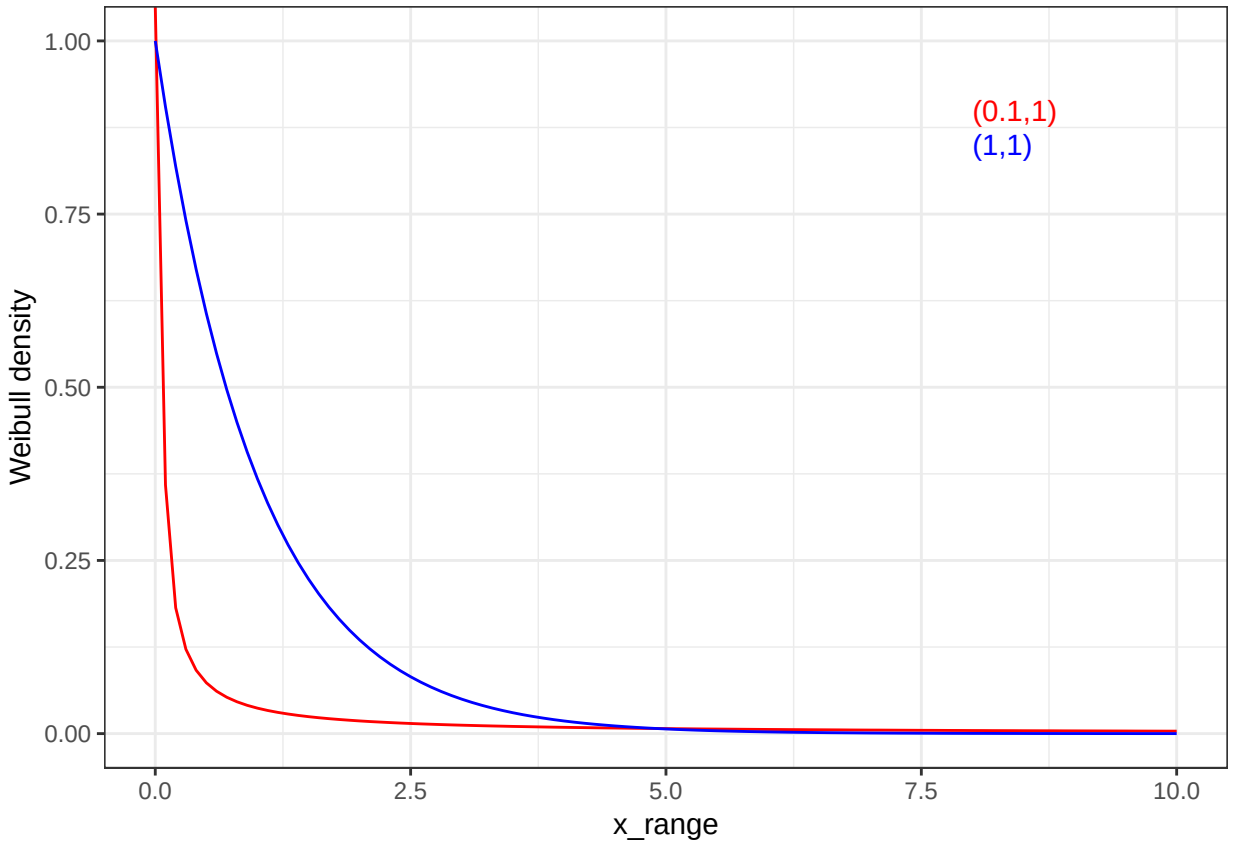
We discussed the random walk model in tutorial 1, which allowed us to account for both the choice (left or right) and the decision times in speeded-decision tasks. In this tutorial we will use a different approach and model the decision times with a Weibull distribution.

The Weibull distribution has a few advantages over the previous random walk model. First, the random walk model assumes discrete time steps - and therefore cannot be directly applied to our data. Second, using a standard statistical distribution allows us to conveniently and directly compute quantities like moments and likelihoods. Finally, a Weibull represents the distribution of the minimum of several Weibull distributions and as such one could argue can account for the speeded-decisions between the two options in our experiment.

Let's get an intuition for what the Weibull distribution looks like. The density function of the Weibull, as well as a random number generators and quantiles are available in R via `dweibull()`, `rweibull()` and `qweibull()`. As you can see from the documentation of these functions, two parameters determine the Weibull density - its scale and shape. We can visualize the effect of varying these parameters over a range of inputs.

```
x_range <- seq(0,10, by=0.1)

ggplot() +
  aes(x = x_range) + # set same x range for all lines
  geom_line(aes(y = dweibull(x_range, shape=0.1, scale=1)),
            color="red") + # plot *red* line for weibull with shape & scale (0.1, 1)
  annotate("text", x=8, hjust=0,
           y=.9, col="red", label="(0.1,1)") + # red annotation for (0.1, 1)
  geom_line(aes(y = dweibull(x_range, shape=1, scale=1)),
            color="blue") + # plot *blue* line for weibull with shape & scale (1, 1)
  annotate("text", x=8, hjust=0,
           y=.85, col="blue", label="(1,1)") +  # lower, blue annotation for (1, 1)
  ylab("Weibull density") + # label y axis
  theme_bw() # optional: give plot a nicer theme
```



***Exercise*: Extend the plot above by adding the density of the Weibull distribution for the following shape and scale parameters: [(shape = 1.5, scale = 1), (1.5, 3), (5,3), (9, 3)], using `dweibull(range, shape, scale)`. How do the parameters influence the shape of the distribution?**

Fitting Decision Times with the Weibull Distribution

We will augment the Weibull distribution by an additional parameter - a shift parameter. The shift parameter shifts the entire distribution, changing its mean but leaving the overall shape (after the point it is shifted to) unchanged. The shift parameter has been used in previous research and it allows to represent an initial interval that might correspond to the time required to encode the stimulus.

With this additional parameter the mean of the shifted Weibull is:

```
weibull.mean <- function(shape, scale, shift=0) {
  scale * gamma(1+1/shape) + shift
}
```

We will start our parameter estimation by fitting the mean of the shifted Weibull to the decision times in our data set using the root-mean-squared error (RMSE) that you saw in the lectures. We will use the default optimization procedure in R `optim()`. Also note that we are using the parameter order `[shape, scale, shift]` and not `[shift, scale, shape]` as in Farrell and Lewandowsky (2018).

```
wb.rmse <- function(rtData, par) {
  if (par[1] <= 0 | par[2] <0){
    # for negative shape/scale parameters the Weibull is undefined
    # so we return a high error
    10000000
```

```
  }
  else {
    sqrt((weibull.mean(par[1], par[2], par[3]) - mean(rtData))^2)
  }
}
```

***Exercise*: Fit the Weibull using the `optim` function for the RMSE function above on the decision time (`RT`) data (`data.22`). The `optim` function requires you to pass starting values to the optimization algorithm, set these to (255, 255, 0).**

***Exercise*: What are the parameters resulting from the optimization?**

***Exercise*: What is the corresponding mean of the Weibull for those parameters? What is the error for the optimized parameters?**

***Exercise*: How well does the mean match the experiment data? Plot the fitted density and the experiment data (you might want to exclude the more extreme RTs of the participants for this plot). Is this a satisfying fit? Do you have any intuitions as to how it might be improved?**

One way to improve our approach is to take into account the shape of the distribution. Often, this is done by not only fitting the mean, but fitting several quantiles of the data.

```
q_p <- c(.1,.3,.5,.7,.9)

wb.qrmse <- function(rtData, par, qp) {
  if(any(par<=0)){
    return(10000000)
  }
  q_pred <- qweibull(qp, shape=par[1], scale=par[2]) + par[3]
  sqrt(mean((q_pred-quantile(rtData, qp))^2))
}
```

***Exercise*: Fit the shifted Weibull to the decision-time data. Use the RMSE of the 5 quantiles `q_p` instead of the mean for optimization. Again, use `(225, 255, 0)` as starting values for the optimization.**

***Exercise*: What parameters did the optimization find? Compare them to the parameters we found using only the mean for the RMSE.**

***Exercise*: What is the RMSE for the model trained on the quantiles? Compare it to the RMSE for the model trained only on the mean.**

***Exercise*: Plot the results of the optimization and the participants' decision times - how well does the mean match the experiment data?**

***Exercise*: A nice way to display the accuracy of model predictions is to plot the true values against the predictions. This is usually called a Q-Q plot. Plot the Q-Q plot for the quantiles for our data (`quantile()`) against the quantiles of the fitted Weibull (`qweibull()`). Additionally, plot a line, $f(x) = x$ on top of the data (try `geom_abline()` in ggplot2). What does this line correspond to?**

***Exercise*: By default `optim` uses the Nelder-Mead optimization procedure, that you saw in the lecture. The optimization procedure requires us to provide starting guesses for the parameters. Evaluate the results of the optimization (for the quantile optimization) for a range of starting values. For scale and shape generate 250 random uniform starting values between 0.1 and 500 (`runif()`). For the shift generate 250 random uniform values between 0 and 10. Loop through all starting values and calculate the RMSE error of the found parameters and store the errors in an array. Then plot and summarize (`summary()`) the optimized errors - how much do they vary?**

With three free parameters there are many equivalent solutions for the task and the optimization procedures we have used can lead to different results for different starting values. As you have seen, to avoid sub-optimal parameter estimates, it is a good idea to run optimization procedures for many starting values.

When we fit the mean of the Weibull to the data we saw that achieving a low error does not guarantee that our model parameters capture the underlying data. In general, it is a good idea to check how model predictions look and how closely they match the actual data, as well as subgroups in our data.

Fitting Individual Participants

Let's check how well our model captures individual participants. To make our task a bit easier we will exclude participants who made fewer than 20 decisions.

```
t22 <- table(data.22$sessionId) #gives us counts for each id
t22 <- t22[t22 >= 20]

t22ids <- unique(names(t22))
```

Now we plot the QQ plots for these 5 participants.

```
# make new data frame with data only from these individuals
data.t22 <- data.22[data.22$sessionId %in% t22ids, ]
```

```
# run this cell once you've defined the model quantiles
qm = YOUR MODEL QUANTILES OBJECT HERE

# remove factor levels for excluded participants
data.t22$sessionId <- data.t22$sessionId[ , drop=TRUE]
# get RT quantiles for each individual
qe.t22 <- with(data.t22, tapply(RT, sessionId, quantile, probs=q_p))
# turn into data frame
data.qe.t22 <- data.frame(participant = rep(names(qe.t22), each=length(q_p)),
                          p_n = rep(paste0("P", 1:length(t22), ", N=", t22),
                                    each=length(q_p)), # label N for each subject
                          quantile = rep(q_p, length(qe.t22)),
                          # N.B. YOUR QUANTILES END UP ON THE LINE BELOW
                          model_q = rep(qm, length(qe.t22)),
                          rt = unlist(qe.t22, use.names = FALSE))

ggplot(data.qe.t22) +
  aes(x=rt, y=model_q, col=participant) +
  geom_point() +
  geom_abline(aes(slope=1, intercept=0)) +
  expand_limits(x=max(data.qe.t22$model_q), # this is a tweak to make sure
                y=min(data.qe.t22$rt)) + # the correct data range is plotted
  theme_bw() +
  guides(col="none") + # hide legend for color
  facet_wrap(p_n ~ .) + # separate plots for each individual, indicating N trials
```

```
  ggtitle("QQ plots for 5 individual participants vs model quantiles")
```

As you can see, our model does not capture these participants very well. This is not surprising, since participants' decision times differ and it is challenging to account for this variability if we only fit a single model. Therefore, researchers have attempted to characterize individual differences in their data by modelling each participant individually and then describing the estimated parameters for these individual models. In addition to fitting individual participants, we will now use maximum likelihood to estimate our parameters.

***Exercise*: What is the goal of maximum likelihood estimation? On a high level, how does it differ from our approach up to this point? What are some reasons for and against using MLE?**

```
wb.lldev <- function(par, data)  {
  # We are deviating from F&W's practice of assigning a very high (but finite)
  # NLL when parameters are invalid or the data will have probability zero.
  # You may want to consider some of the pros and cons of their approach.
  if (any (par<=0) || any(data - par[3] < 0)) {
    return (Inf)
  }
  ll <- dweibull(data - par[3], shape=par[1], scale=par[2])
  # Sometimes people will compute the "deviance", or twice the NLL. This term is
  # relevant in some model selection settings, but the difference isn't so
  # important here, and the MLE is unchanged by this transformation.
  -2*sum(log(ll))
}
```

***Question*: Why do we return the negative log-likelihood? Why do we subtract the shift parameter from the data?**

We will fit each participant using maximum likelihood and store the optimized parameter values in the `ppRTsParameters` data frame.

```
fit.wb.lldev <- function(p_id, df=data.t22) {
  d <- df[df$sessionId == p_id,]$RT # get RT data for participant
  d <- d[d < 1000] # remove outliers
  optP <- optim(c(100,1000,.1), fn = function(x) wb.lldev(x, d)) # run optimization
  return(optP[["par"]]) # return extracted parameters
}

opt <- t(sapply(t22ids, fit.wb.lldev))
colnames(opt) <- c("shapes", "scales", "shifts")

ppRTsParameters <- data.frame(id=rownames(opt), opt, row.names=NULL)
```

***Exercise*: Summarize the participant parameters. How much do the found parameters differ? Is there anything peculiar about the fitted parameters? If so, what might be going on?**

***Exercise*: Plot the fitted parameters and the real data.**

***Question*: Setting aside idiosyncrasies of the current case, what are advantages of fitting models for each participant individually? Are there any disadvantages? What would be the ideal way to account for individual differences?**

References

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