Lab assignment 3: Probabilistic models

Deadline: 24.1.

1 Introduction

The goal of this assignment is to develop a model of how (some) adjectives are interpreted and to optimize the model on experimental data.

The model is built as part of the Rational Speech Act Theory (RSA; not to be confused with the Representational similarity analysis that you learned in Lab Assignment 2). The RSA that we will study here is discussed in Goodman and Frank (2016) and Lecture 15/01 but we do not make use of the whole power of those cognitive models. The implementation you work on has been discussed in Qing and Franke (2014).[1] Even though you can probably finish the task without checking Qing and Franke (2014), I strongly recommend you to take a look at the paper, as it will make more sense of what you are programming on the theoretical/conceptual level (the paper is 6 pages long and the model is discussed on pages 1-3).

2 What will you need?

You will need this file, which also includes snippets of the R code and the description of the task. The R code is also condensed in a separate file in this zip folder. You can take that R file and directly build on that. Finally, you will need data, which are included as a csv file 'adjective-data.csv' in the same zip folder.

The file requires a few R packages. Obviously, you need to install them to make everything work. The most difficult one can be the package BayesianTools: this is a package for Bayesian inference. It is a small, lightweight package, currently maintained. However, it is developed just by one person and it is not much known in the community so it could be that not every issue with installation is ironed out. I will say a bit more about the package below.

3 What will you hand in?

I expect you to hand in a pdf file that answers question specified below (Task 1–5). Some of the questions ask for graphs. You can generate them in your R code and paste them in your pdf, or you can use programs that weave R code together with text and output pdf (like knitr, used in this description).

4 Model specification

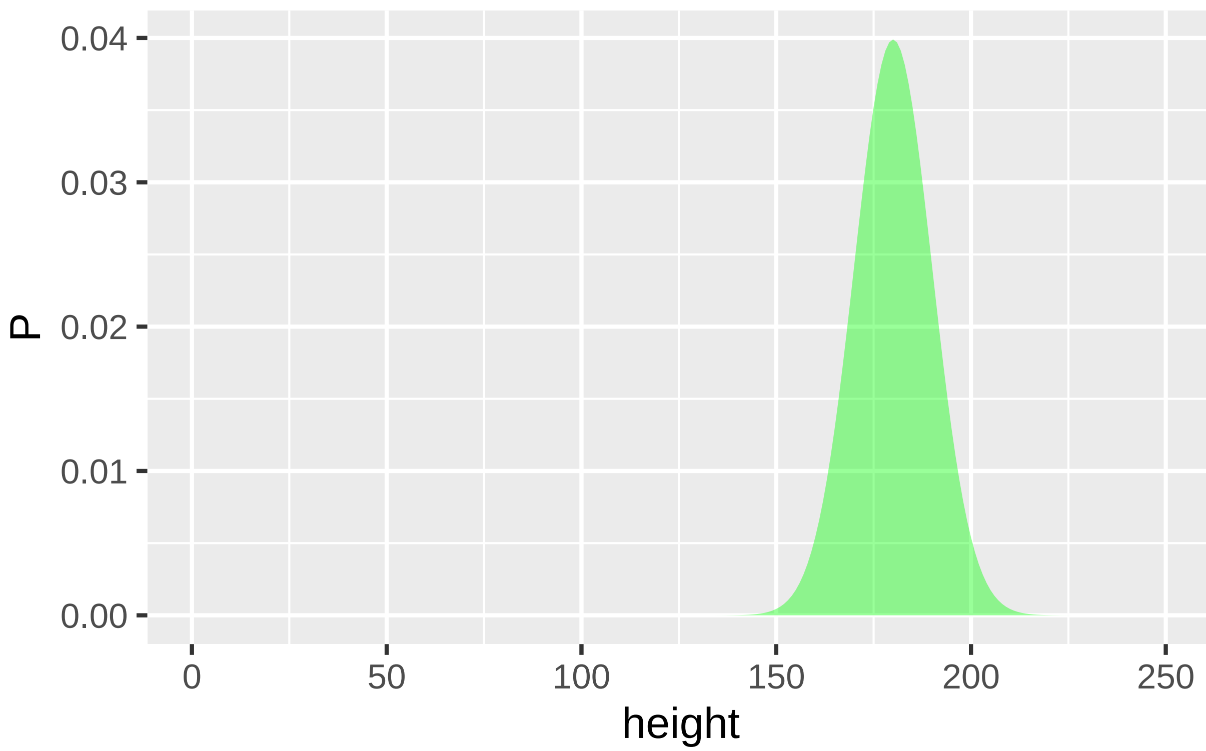
We will develop a model of the interpretation of adjectives. Basically, we are interested in the production of sentences like 'x is tall'. When would it make sense to say a sentence like that?

The common insight is that adjectives like 'tall' come with a threshold, Θ, which is a degree. Being tall is equivalent to being taller than the threshold.

Let's say we roughly know people's distribution of heights. Let's say, for the sake of the argument, that it is a normal distribution with mean 180 (cm) and st.d. 10. Let's plot it:

[1] https://pdfs.semanticscholar.org/7e72/6fe322f9cc4512f29ce14f8cd163bc52c794.pdf

```r
library(ggplot2)

# we will work with points 1 to 250 (cm)
scale.points <- c(1:250)

# we create a dataframe for plotting
example.height <- data.frame(x = scale.points)

# we use sapply, which is a vectorized function application; see help if
# you don't understand it

# we add y, which is just the probability density function described above
# (normal distribution)
example.height$y <- sapply(example.height$x, function(x) {
    dnorm(x, mean = 180, sd = 10)
})

# this starts the plot creation
g1 <- ggplot(example.height, aes(x = x, y = y))

# we make the plot more pretty: we specify it should fill in area and add
# labels
g1 <- g1 + geom_area(fill = "green", alpha = 0.4) + xlab("height") + ylab("P") +
    theme_gray(20)
```



What happens when a listener hears ‘x is tall’? He should update the prior belief by assuming that x’s height is above the threshold, i.e., he now knows that $d_A(x) > \Theta$, where A is in this case the adjective ‘tall’ and $d_A(x)$ is the height of x. The new belief of the listener, written as $\rho_0(d_A(x)|A;\Theta)$ is the conditional probability $P(d_A(x)|d_A(x) > \Theta)$, which updates the prior distribution as follows:

$$\rho_0(d_A(x)|A;\Theta) \quad = \quad \begin{cases} \frac{P(d_A(x))}{\int_{\Theta}^{max} P(d_A(x))dx} & d_A(x) \geq \Theta \\ 0 & d_A(x) < \Theta \end{cases}$$

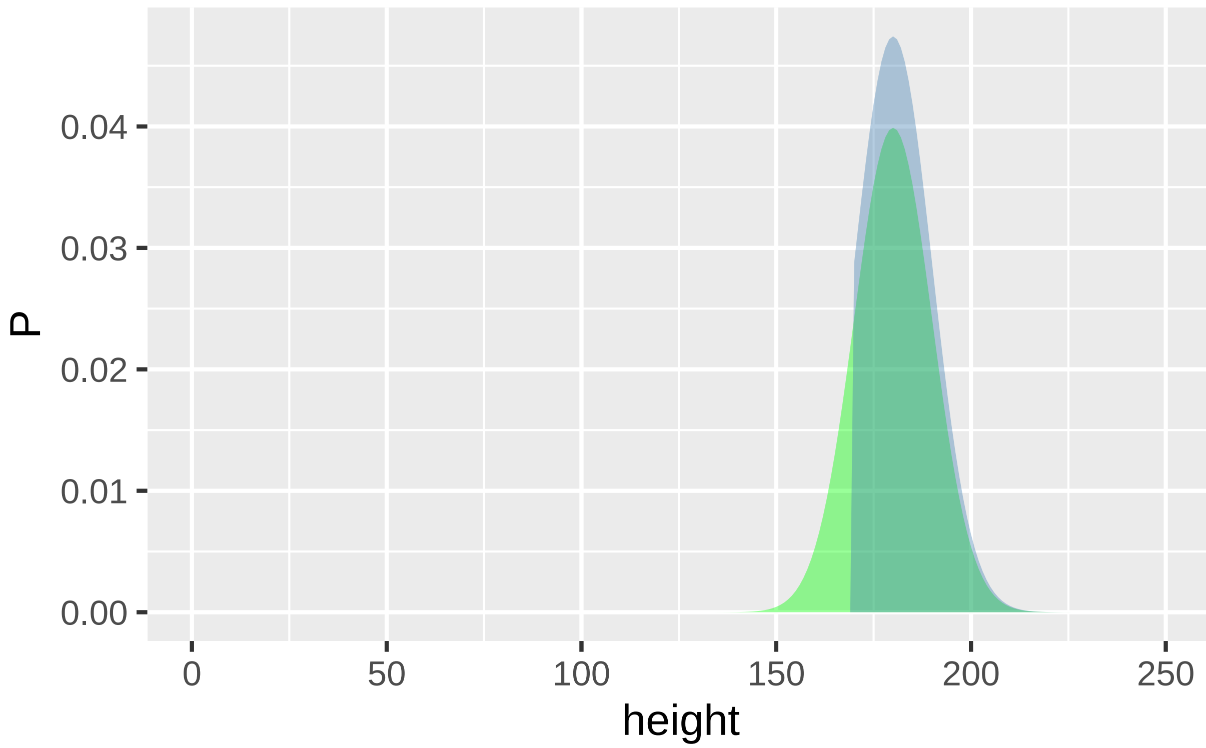
We will now build this function in R. Let's call the function literal.listener. The function is provided below:

```
literal.listener <- function(x, threshold, densityf, cumulativef) {

    ifelse(x >= threshold, densityf(x)/(1 - cumulativef(threshold)), 0)

}
```

Let us break this function down. It takes four arguments: the height x of an individual, threshold (Θ), and two functions: probability density function and cumulative distribution function. The first one is needed to calculate $P(d_A(x))$, the second one to calculate $\int_{\Theta}^{max} P(d_A(x))dx$. Take a look at the function, make sure you understand it.

Let's see how this function updates listener's beliefs, assuming that threshold is 170 (cm). Note what we fill in as densityf and cumulativef below: we work with the same normal distribution as for our listener's prior belief about heights, $N(180, 10)$.

```
threshold <- 170

example.height$updated <- sapply(example.height$x, function(x) {
    literal.listener(x = x, threshold = threshold, densityf = function(x) {
        dnorm(x, 180, 10)
    }, cumulativef = function(x) {
        pnorm(x, 180, 10)
    })
})

# this starts the plot creation
g1 <- ggplot(example.height, aes(x = x, y = y))
g1 <- g1 + geom_area(fill = "green", alpha = 0.4)

# we add the result of updated belief
g1 <- g1 + geom_area(aes(y = updated), fill = "steelblue", alpha = 0.4)
g1 <- g1 + xlab("height") + ylab("P") + theme_gray(20)
```



What you can see in the graph is that the updated belief (in blue) cuts off probability at 170 and adds weight to degrees above 170, tracing the original distribution.

Of course, this is a simplistic example. We assumed that the threshold is 170. But normally, when we talk, how are we to know what the threshold is?

Qing and Franke take the value of Θ to arise through communicative efficiency. Details are in the paper. Let me just put in the formula for Expected Success (ES), which describes the expected chance of success to convey the height of someone by adopting a particular threshold ($\rho_0(d_A(x)|\emptyset;\Theta)$ is the probability of tallness of x after the speaker said nothing; it is equivalent to the prior distribution):

$$\begin{aligned} ES(\Theta) &= \sum_{d_A(x)<\Theta} P(d_A(x)) \cdot \rho_0(d_A(x)|\emptyset;\Theta) \quad + \\ & \sum_{d_A(x)\geq\Theta} P(d_A(x)) \cdot \rho_0(d_A(x)|A;\Theta) \end{aligned}$$

This function is implemented below:

```
expected.success <- function(threshold, scale.points, densityf, cumulativef) {


    sum(sapply(scale.points[1]:max(1, scale.points[which(scale.points == threshold) -
        1]), function(x) {
        densityf(x) * densityf(x)
    })) + sum(sapply(scale.points[which(scale.points == threshold)]:scale.points[length(scale.points)],
        function(x) {
            densityf(x) * literal.listener(x, threshold, densityf, cumulativef)
        }))

}
```

The function takes four arguments: the threshold, the scale points (because we sum discretized degress; for example, for tallness, we sum degrees per centimeter up to 250 cm) and densityf and cumulativef, which are passed onto literal.listener. Take a look at the function, make sure you understand it.

Three other functions need to be implemented in the model of adjective production. These are: (i) utility of threshold U; (ii) probability of using threshold; (iii) and given some degree d, how likely the speaker will use the adjective for d. These are defined as (where c and λ are free paramaters called *coverage parameter* and *lambda*, respectively):

$$U(\Theta; c) = ES(\Theta) + c \cdot \int_{[\Theta, max]} P(d_A(x))$$

$$P(\Theta; \lambda; c) = \frac{e^{\lambda \cdot U(\Theta; c)}}{\sum_t e^{\lambda \cdot U(t; c)}}$$

$$\sigma(A|d; \lambda; c) = \sum_{\Theta \leq d} P(\Theta; \lambda; c)$$

Task 1: Given the distribution of height as described above (normal distribution, mean=180, st.d.=10), summarize the probability of using a threshold and how likely the speaker will use the adjective *tall*. For the summary, report at least two things. First, show two graphs: (i) what the probability of using threshold looks like on the scale $1 - 250$cm; (ii) the graph of the function σ on the same scale. Second, you should state which degree point has the highest probability of being used as a threshold, and on which degree point it is most likely the speaker will use the adjective *tall*. If the two values differ or are the same, say in a few words why you think this is so. For the task, use free parameters $\lambda = 50$ and $c = 0$.

Obviously, for this task, you will have to implement the three functions shown above. You should make use of the following code as your starting point, fill in the body of the functions.

```
utility <- function(threshold, scale.points, coverage.parameter, densityf,
    cumulativef) {
    ...
}


probability.threshold <- function(threshold, scale.points, lambda, coverage.parameter,
    densityf, cumulativef) {

    ...

}


use.adjective <- function(degree, scale.points, lambda, coverage.parameter,
    densityf, cumulativef) {

    ...

}
```

Notes: again, it might help if you check the paper of Qing and Franke to understand what is going on. Second, use.adjective can be slow if you repeatedly calculate the denominator in probability.threshold. Since this needs to be calculated only once, it is better to calculate the denominator directly in the body of the function use.adjective and plug that value in and not repeat calculation afterwards (this optimization trick is called memoization). Finally, in U, you can either use *max* value from scale.points (e.g., 250 cm), or, if you are going to work with predefined cumulative distribution function (e.g., cdf for normal distribution) it might be easier to just use the maximum value as present in that predefined function (i.e., infinity in case of normal distribution). The latter is easier to implement. You can also implement U using sum operation instead of cdf (but that's more work). In the two other functions, you need scale.points, because you will be summing up over them.

Help: here are a few tests that your implementation of the three functions should pass. If you do not pass these tests (they should give you TRUE), most likely you did something wrong. Right now, we get Errors because the functions are not implemented yet.

```
# probability.threshold is a probability, so if you sum up all values it
# generates, the result should be 1
round(sum(sapply(1:10, function(x) {
    probability.threshold(x, 1:10, 50, 0, function(x) {
        dnorm(x, 5, 1)
    }, function(x) {
        pnorm(x, 5, 1)
    })
}))) == 1
```

```
## Error in probability.threshold(x, 1:10, 50, 0, function(x) {: '...' used in an incorrect
context
```

```
# for narrow normal distribution, prob. threshold should be max just one
# value above the average
which(sapply(1:10, function(x) {
    probability.threshold(x, 1:10, 50, 0, function(x) {
        dnorm(x, 5, 1)
    }, function(x) {
        pnorm(x, 5, 1)
    })
}) == max(sapply(1:10, function(x) {
    probability.threshold(x, 1:10, 50, 0, function(x) {
        dnorm(x, 5, 1)
    }, function(x) {
        pnorm(x, 5, 1)
    })
}))) == 6
```

```
## Error in probability.threshold(x, 1:10, 50, 0, function(x) {: '...' used in an incorrect
context
```

```
# use.adjective should be very unlikely on values 5 and smaller and very
# likely afterwards
round(sapply(1:10, function(x) {
    use.adjective(x, 1:10, 50, 0, function(x) {
        dnorm(x, 5, 1)
    }, function(x) {
        pnorm(x, 5, 1)
    })
})[5], 3) == 0.005
```

```
## Error in use.adjective(x, 1:10, 50, 0, function(x) {: '...' used in an incorrect context
```

```
round(sapply(1:10, function(x) {
    use.adjective(x, 1:10, 50, 0, function(x) {
        dnorm(x, 5, 1)
    }, function(x) {
        pnorm(x, 5, 1)
    })
})[6], 3) == 1
```

```
## Error in use.adjective(x, 1:10, 50, 0, function(x) {: '...' used in an incorrect context
```

Task 2: Explore expected.success and use.adjective for various prior distribution functions. For this task, assume that coverage.parameter c is at 0 and lambda is at 50.

Above, we looked at heights, which are distributed normally. Let's consider two other cases. For each of those, calculate expected.success and use.adjective (the latter is implemented in Task 1). Then, plot the results, in the same way the results of function application are plotted in previous graphs. Furthermore, report which degree has the max value for expected.success in these two cases.

- IQ is claimed to be normally distributed, with mean 100 and roughly 95% of cases falling between 70-130 range. (Check normal distribution on wikipedia if you don't know it.) Specify the normal distribution and generate figures for expected.success and use.adjective using this distribution and report which degree has the highes expected.success. We assume that the relevant adjective in this case is 'smart' and we are interested in when it would make sense to express 'x is smart'. You should also adapt scale.points (there is no reason these points should go above 150).

- Waiting times have a gamma distribution. Specifics depend on situations and context, but it might be reasonable to say that waiting for buses is a gamma distribution with mean 2 and variance 2 (in minutes). (Check gamma distribution on wikipedia if you don't know it.) Specify the gamma distribution and generate figures for expected.success and use.adjective for this case (we assume that the relevant adjective is 'late', i.e., when would it make sense to say 'the bus is late'?) and report which degree has the highes expected.success. Again, you should adapt scale.points (it is enough if the scale goes up to 30).

5 Data

We apply the model to data from an experiment reported in Solt & Gotzner, Semantics and Linguistic Theory 22 (also in the zipped folder). In particular, we will look at experiment 1, section 3.1. The idea behind the experiment is that participants have to choose entities that they would classify using a given adjective. The grid of entities they are presented with represents the comparison class. The entities in this grid are distributed in various ways w.r.t. the property described by the adjective.

The experimental results are available in 'adjective-data.csv' (courtesy of the authors). For discussion of the experiment, you can also check Qing and Franke (who also replicate the experiment; their discussion is succint and clear).

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

There are four prior distributions for the adjectives. These are the labels and their description in Qing and Franke:

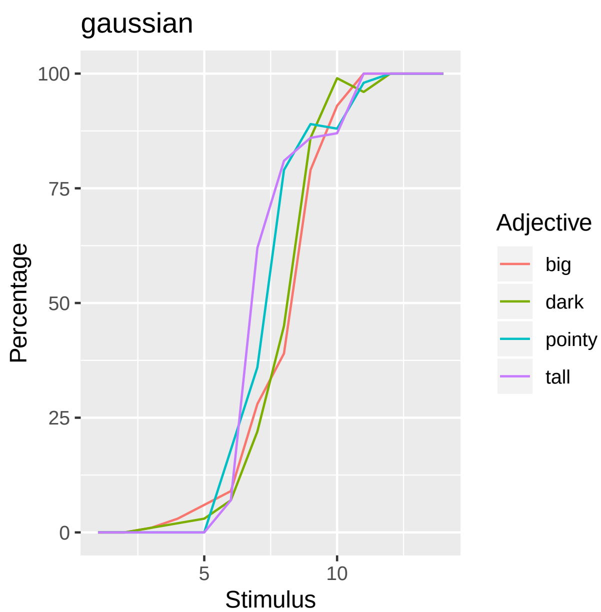
```
gaussian.dist <- c(1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0, 0, 0)
left.skew.dist <- c(2, 5, 6, 6, 5, 4, 3, 2, 1, 1, 1, 0, 0, 0)
moved.dist <- c(0, 0, 0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1)
right.skew.dist <- c(1, 1, 1, 2, 3, 4, 5, 6, 6, 5, 2, 0, 0, 0)
```

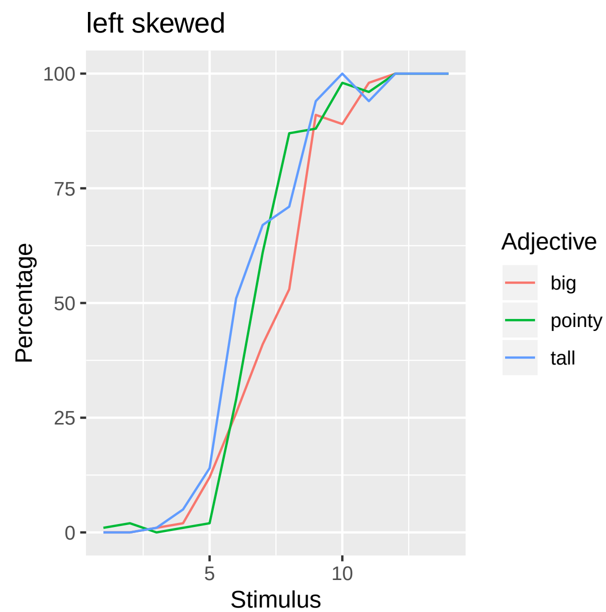
We will approximate them as follows:

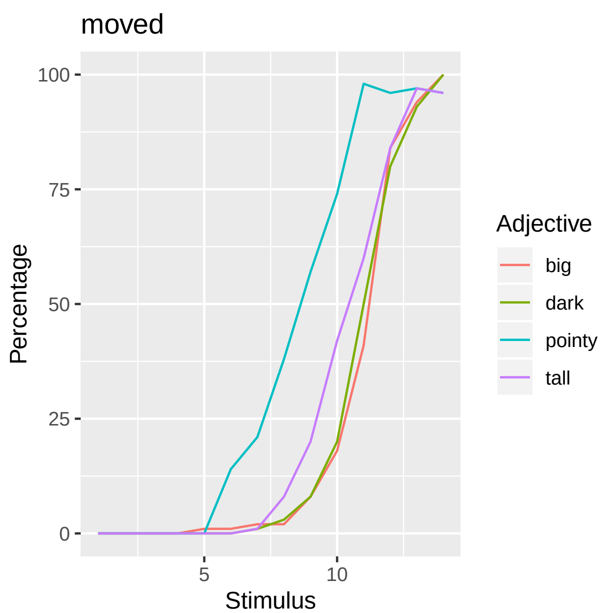
- gaussian.dist: normal distribution with 6 as mean and st.d. 2
- left skew: gamma distribution with shape 4 and scale 1.5
- moved dist: normal distribution with 9 as mean and st.d. 2
- right skew: this could be modeled as a combination of two normal distributions or, after transformation, as beta distribution; since it adds other complications, we will ignore this one in the rest of this exercise

Here is a graphical summary of the relevant results. On the x-axis, we see a stimulus type (i.e., stimulus type 1 for tall is very short, and it increases in length until reaching the maximal height at 14). On the y-axis, we see how big percentage of stimuli of that type were chosen as reprenting the relevant adjective.

```r
data.gaus <- data.adjective[data.adjective$Distribution == "gaussian", ]
data.left <- data.adjective[data.adjective$Distribution == "left", ]
data.moved <- data.adjective[data.adjective$Distribution == "moved", ]

library(ggplot2)
library(gridExtra)
p.g <- ggplot(data.gaus, aes(x = Stimulus, y = 100 * percentage, colour = Adjective)) +
    geom_line() + ylab("Percentage") + ggtitle("gaussian")
p.l <- ggplot(data.left, aes(x = Stimulus, y = 100 * percentage, colour = Adjective)) +
    geom_line() + ylab("Percentage") + ggtitle("left skewed")
p.m <- ggplot(data.moved, aes(x = Stimulus, y = 100 * percentage, colour = Adjective)) +
    geom_line() + ylab("Percentage") + ggtitle("moved")
```







Task 3: You will now check on a subset of the data how our model's predictions correlate with observations (more specifically: what is Pearson's correlation coefficient, r, between model's predictions and actual observations). You will also study what the role of prior belief is. You can use the R function *cor* in this task.

For the first check, let us consider predictions and observations with respect to the adjective 'big' in its three distributions. You should check the value of r between the data and the model three times, one per distribution. Furthermore, check the value of r with respect to three different parameters of the model (9 comparisons in total):

- lambda=40, coverage.parameter=0.1
- lambda=40, coverage.parameter=-0.1
- lambda=40, coverage.parameter=0

What coverage parameter gives us the best and the worst linear correlation? On which distribution do we get the best results?

After you are done with answering this, investigate what the role of prior distribution is. You can check this by hand on a small sample. Pick the coverage parameter that worked best in the previous subquestion. Then, select two distributions (e.g., left skew and moved) but in the model, flip their prior belief distributions (so, for example, gamma distribution is used on 'moved distribution' and normal distribution is used on 'left skewed distribution'). Report what prior distribution you used and report r. Answer these questions: Do we see that r is affected? Does the model suffer in having worse linear correlation with respect to the observed data? If you answered 'yes' to the two question, say in a few words why this is the case.

6 Bayesian modeling

Finally, we are ready to do Bayesian modeling on the dataset. We will use the package BayesianTools for this.

```r
library(BayesianTools)
```

You will probably have to install this first. Uncomment and use:

```r
# install.packages('BayesianTools')
```

I saw that having an older version of R than 3.5 could cause problems with the package. If you experience any, try to update your R software.

If you are going to work on Bayesian modeling more extensively, I recommend that you use one of the popular and commonly used languages: STAN or JAGS. We do not do so here because these are languages independent of R and learning them would require much more energy and time. The advantage of BayesianTools is that the package is fully embedded in R, so using it is relatively easy, and furthermore, it can be combined with any function written in R. We will leverage this last point in this assignment.

We are going to build a model that will learn posterior distribution of the two free parameters, coverage.parameter and lambda. In a Bayesian model, we have to specify two things:

- prior distribution (NB. This is now the prior distribution for the parameters we want to study, i.e., coverage.parameter and lambda, this distribution is not related to prior distributions in the interpretation of adjectives!)
- likelihood

Let's try a simple example. Let's assume that we will want to find the posterior distribution of two parameters that should predict percentage of acceptance of data.gaus.big in each stimulus type (1-14). Let's call the two parameters $p1$ and $p2$. We will assume the following priors for $p1$ and $p2$:

- $p1 \sim Unif(0, 0.1)$, that is $p1$ is generated by Uniform distribution with the smallest possible value 0 and the largest possible value 0.1
- $p2 \sim Unif(0.1, 1)$, that is $p2$ is generated by Uniform distribution with the smallest possible value 0.1 and the largest possible value 1

We will assume that the likelihood is:

- data.gaus.big$percentage $\sim N(p1 + p2, 0.1)$

In words, we assume that percentage in data.gaus.big is generated by a normal distribution with standard deviation 0.1 and mean the sum of the two parameters, $p1$ and $p2$. We will now write everything in the model and see what the posterior distribution of the two parameters is.

Here is the prior:

```
prior <- createUniformPrior(lower = c(0, 0.1), upper = c(0.1, 1), best = NULL)
```

Now, we are going to implement the likelihood (actually, BayesianTools uses log-likelihood, i.e., likelihood transformed by log). It is specified here (the first line just loads a subset of data that we will use):

```
data.gaus.big <- subset(data.gaus, Adjective == "big")

likelihood <- function(param1) {

    collect <- 0

    for (i in 1:14) {
        collect <- collect + dnorm(data.gaus.big$percentage[i], mean = param1[1] +
            param1[2], sd = 0.1, log = TRUE)

    }

    return(collect)
}
```

We put everything in setup, specify the number of iterations and run sampling:

```
bayesianSetup <- createBayesianSetup(likelihood = likelihood, prior = prior)

iter = 10000
settings = list(iterations = iter, message = FALSE)
out <- runMCMC(bayesianSetup = bayesianSetup, settings = settings)
```

Sampling collects 10,000 draws from our posterior distribution on the two parameters, $p1$ and $p2$. The samples are stored in the object out. We can explore it using the function summary:

```
summary(out)

## # # # # # # # # # # # # # # # # # # # # # # # # #
## ## MCMC chain summary ##
## # # # # # # # # # # # # # # # # # # # # # # # # #
##
## # MCMC sampler:  DEzs
## # Nr. Chains:  3
## # Iterations per chain:  3334
## # Rejection rate:  0.682
## # Effective sample size:  1102
## # Runtime:  0.872  sec.
##
## # Parameters
##         psf   MAP  2.5% median 97.5%
## par 1 1.004 0.056 0.001  0.049 0.097
## par 2 1.007 0.414 0.346  0.420 0.501
##
## ## DIC:  228.142
## ## Convergence
##  Gelman Rubin multivariate psrf:
##
## ## Correlations
```

```
##        par 1  par 2
## par 1  1.000 -0.662
## par 2 -0.662  1.000
```

To read this, check MAP (maximum a posteriori estimation), which gives us point estimates – the mode of the distributions of the parameters. What we see is that $p1$ is kept low (since it cannot rise above 0.1, anyway) and the second parameter is around 0.4. This makes sense since most values are extreme (either 0 or 1) so having a mean of normal distribution around 0.5 minimizes the distance from those extreme values.

Task 4: You will now implement the parameter estimation for our model using the data from data.gaus.big (that is, we check only one adjective and only one distribution of that adjective).

I specify the prior distribution for you. The prior distribution for coverage.parameter is Uniform distribution Unif(-1,0) (that is, any value between -1 and 0 including -1 and 0 is equally likely, no other values are possible). The prior distribution for lambda is Unif(1, 50). This is how this is encoded:

```
prior <- createUniformPrior(lower = c(-1, 1), upper = c(1, 50), best = NULL)
```

What you need to do is to specify likelihood. You should draw on the previous example. However, unlike in the previous example, the likelihood should be:

- data.gaus.big$percentage $\sim N(Model(coverage.parameter, lambda), 0.1)$

Where $Model$ is our model from Section 3. (In particular, you should make use of the function use.adjective in the new likelihood. Roughly, instead of adding up param1 and param2, you need to do something else:)

```
likelihood <- function(param1) {

    collect <- 0

    for (i in 1:14) {
        collect <- collect + dnorm(data.gaus.big$percentage[i], mean = ...,
            sd = 0.1, log = TRUE)

    }

    return(collect)
}
```

After you are done, you can run this model by uncommenting and specifying:

```
# bayesianSetup <- createBayesianSetup(likelihood = likelihood, prior =
# prior)

# iter = 10000

# settings = list(iterations = iter, message = FALSE)

# out <- runMCMC(bayesianSetup = bayesianSetup, settings = settings)
```

Task 5: Explore the distribution of the parameters in out. What do we see? What values have been found for lambda and coverage.parameter? Report summary statistics on lambda and coverage.parameter (MAP, median, 2.5percent to 97.5percent interval). Discuss briefly the summaries (i.e., are values in out widespread or very narrow? If so, why do you think this is the case and is it good/bad?)

References

Goodman, Noah D, and Michael C Frank. 2016. Pragmatic language interpretation as probabilistic inference. *Trends in cognitive sciences* 20:818–829.

Qing, Ciyang, and Michael Franke. 2014. Gradable adjectives, vagueness, and optimal language use: A speaker-oriented model. In *Semantics and linguistic theory*, volume 24, 23–41.