practical_exercise_3, Methods 3, 2021, autumn semester

Alina Kereszt

29.09.2021

Exercises and objectives

The objectives of the exercises of this assignment are:
1) Download and organise the data and model and plot staircase responses based on fits of logistic functions
2) Fit multilevel models for response times
3) Fit multilevel models for count data

REMEMBER: In your report, make sure to include code that can reproduce the answers requested in the exercises below (**MAKE A KNITTED VERSION**)
REMEMBER: This assignment will be part of your final portfolio

Exercise 1

Go to https://osf.io/ecxsj/files/ and download the files associated with Experiment 2 (there should be 29). The data is associated with Experiment 2 of the article at the following DOI https://doi.org/10.1016/j.concog.2019.03.007

1) Put the data from all subjects into a single data frame

```
df <-
    list.files(pattern = "*.csv") %>%
    map_df(~read_csv(.))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
```

```
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
```

```
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
```

```
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
```

```
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
```

```
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
```

```
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
```

```
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
```

```
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
```

```
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
```

```
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
```

```
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
```

```
##
##
## -- Column specification --------------------------------------------------------
## cols(
##   trial.type = col_character(),
##   pas = col_double(),
##   trial = col_double(),
##   jitter.x = col_double(),
##   jitter.y = col_double(),
##   odd.digit = col_double(),
##   target.contrast = col_double(),
##   target.frames = col_double(),
##   cue = col_double(),
##   task = col_character(),
##   target.type = col_character(),
##   rt.subj = col_double(),
##   rt.obj = col_double(),
##   even.digit = col_double(),
##   seed = col_double(),
##   obj.resp = col_character(),
##   subject = col_character()
## )
```

2) Describe the data and construct extra variables from the existing variables
   i. add a variable to the data frame and call it *correct* (have it be a *logical* variable). Assign a 1 to each row where the subject indicated the correct answer and a 0 to each row where the subject indicated the incorrect answer (**Hint:** the variable *obj.resp* indicates whether the subject answered "even", *e* or "odd", *o*, and the variable *target_type* indicates what was actually presented.

```
df$correct <- ifelse((df$target.type == "odd" & df$obj.resp == "o") | (df$target.type == "even" & df$ob
df$correct <- as.factor(df$correct) #encoded as "factor" because encoding as "logical" changes values t
```

```
ii. describe what the following variables in the data frame contain, _trial.type_, _pas_, _trial_, _targ
  - _trial.type_: A staircase procedure was performed before the actual experiment procedure in order t
  - _pas_: The participant's experience on the PAS (Perceptual Awareness Scale), which categorizes how
  - _trial_: The number of the trial at which the data was collected. Lies along a continuum and should
  - _target.contrast_: The amount of contrast between the background and the stimulus. Lies along a con
  - _cue_: The number code for the cue. There are 32 discrete possible combinations that do not lie alo
  - _task_: The amount of numbers that can be shown (singles = 1 even + 1 odd; pairs = 2 even + 2 odd;
  - _target_type_: Whether the visual stimulus presented to the participant consisted of an even or an
  - _rt.subj_: The amount of time it took the participant to decide where their experience fell on the
  - _rt.obj_: The amount of time it took the participant to make the judgment as to whether they had se
  - _obj.resp_: The participant's response as to whether they had seen an even or an odd number. It sho
  - _subject_: The ID for the participant completing the experiment. Because the participants' behavior
  - _correct_: Whether the participant indicated the correct answer as to whether they had seen an even
```

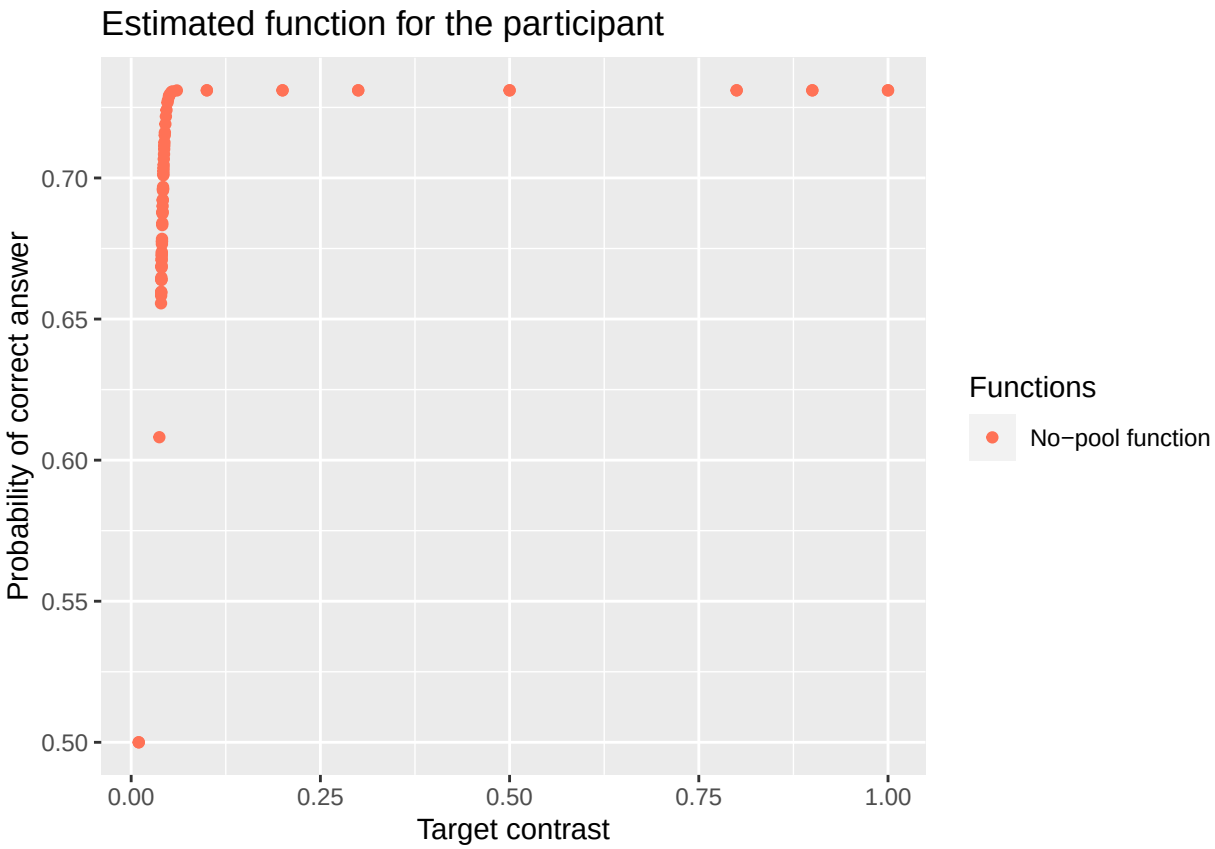
```
#Encode them appropriately
df$trial <- as.numeric(df$trial)
df$trial.type <- as.factor(df$trial.type)
df$pas <- as.factor(df$pas)
df$target.contrast <- as.numeric(df$target.contrast)
df$cue <- as.factor(df$cue)
df$task <- as.factor(df$task)
df$target.type <- as.factor(df$target.type)
df$rt.subj <- as.numeric(df$rt.subj)
df$rt.obj <- as.numeric(df$rt.obj)
```

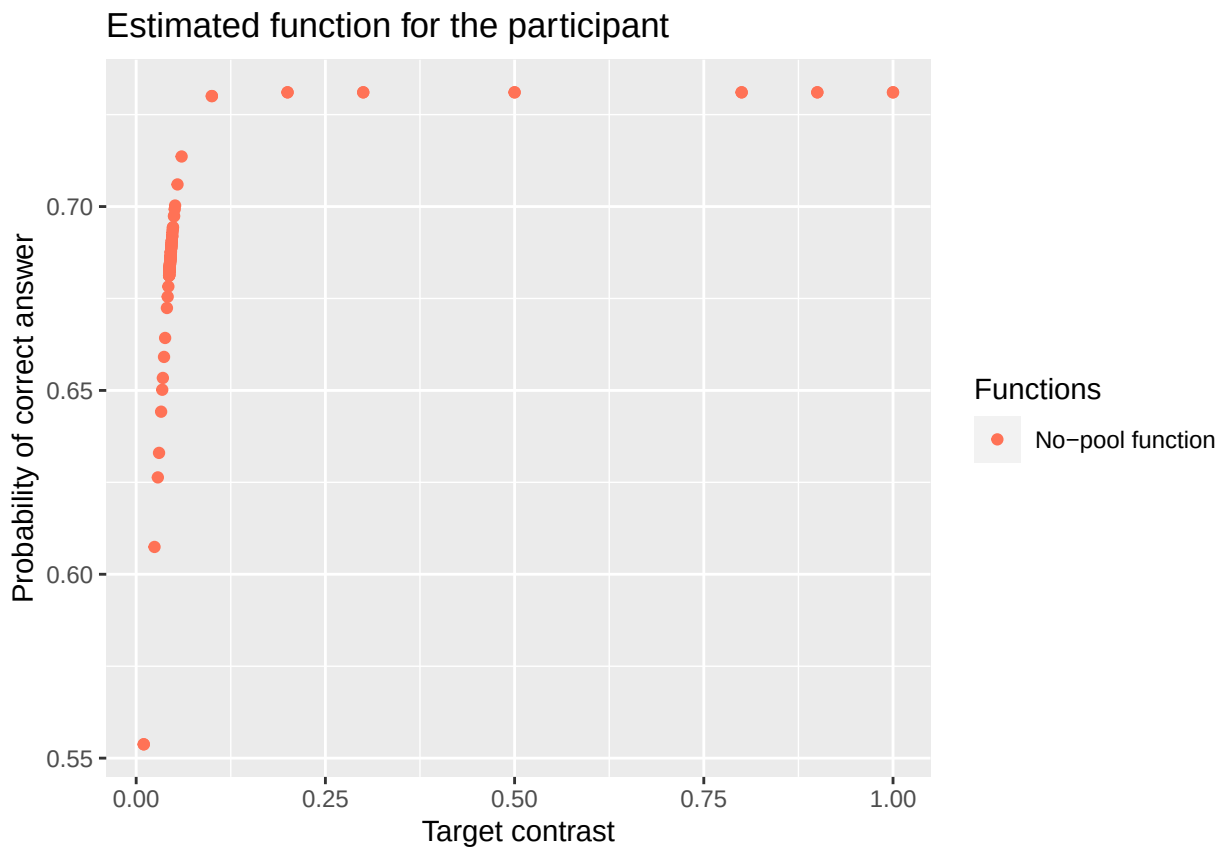
```r
df$obj.resp <- as.factor(df$obj.resp)

#Anonymize subjects (also facilitates later analysis if the subjects are represented only by numbers)
df$subject <- as.numeric(df$subject)
df$subject <- as.factor(df$subject)
```

iii. for the staircasing part __only__, create a plot for each subject where you plot the estimated fun

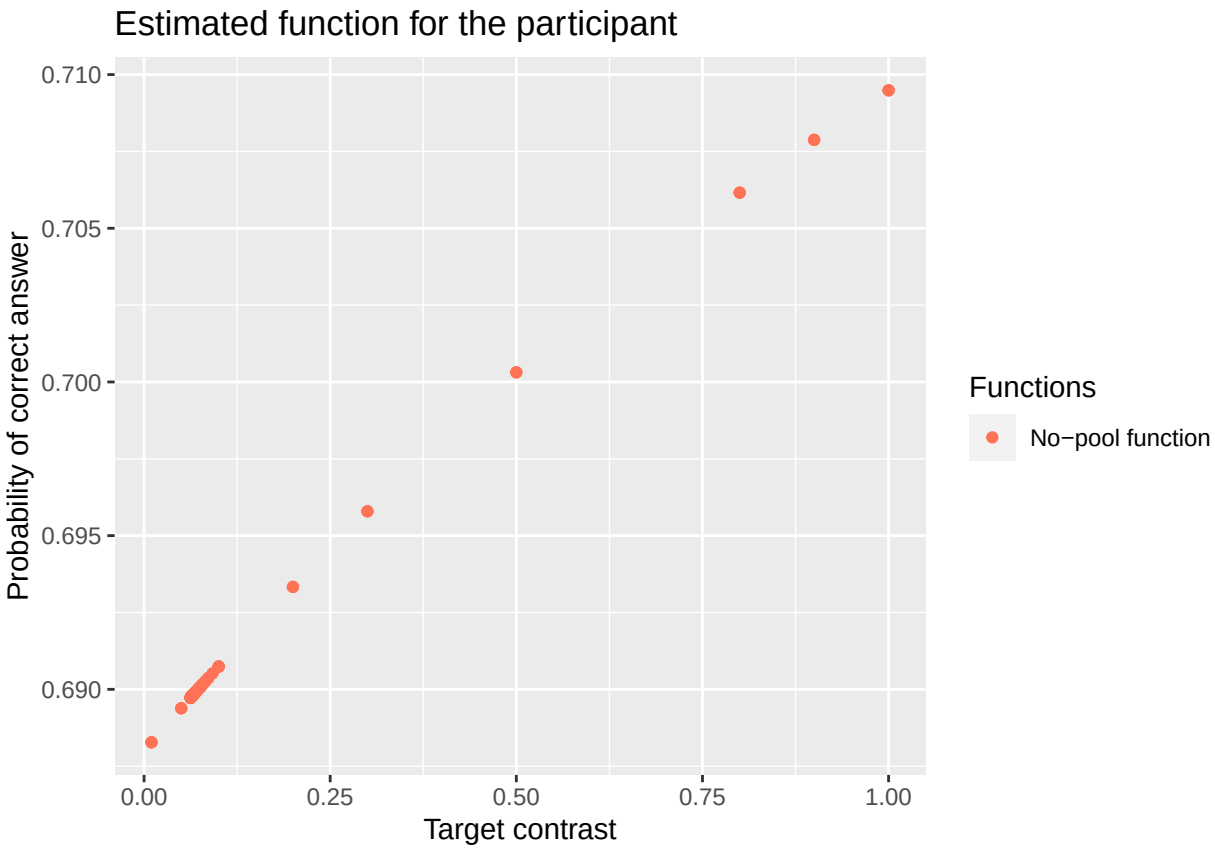
```r
staircase <- df %>%
  filter(trial.type == "staircase") #isolate staircasing data

for (i in 1:29) {
  loopdf <- staircase %>%
    filter(subject == i) #isolate each subject since we are not organizing the data hierarchically, but
  loopmodel <- glm(correct ~ target.contrast, data = loopdf, family = binomial) #run logistic regressio
  loopdf <- loopdf %>%
    mutate(inv = inv.logit(loopmodel$fitted.values)) #get the inverse logit of the fitted values
  plot <- ggplot(loopdf, aes(x = target.contrast, y = inv)) +
    geom_point(aes(color = "coral1")) +
    scale_color_manual(labels = c("No-pool function"), values = c("coral1")) +
    guides(color=guide_legend("Functions")) +
    labs(title = "Estimated function for the participant", x = "Target contrast", y = "Probability of c
  print(plot) #actually get the plot to show up
}
```

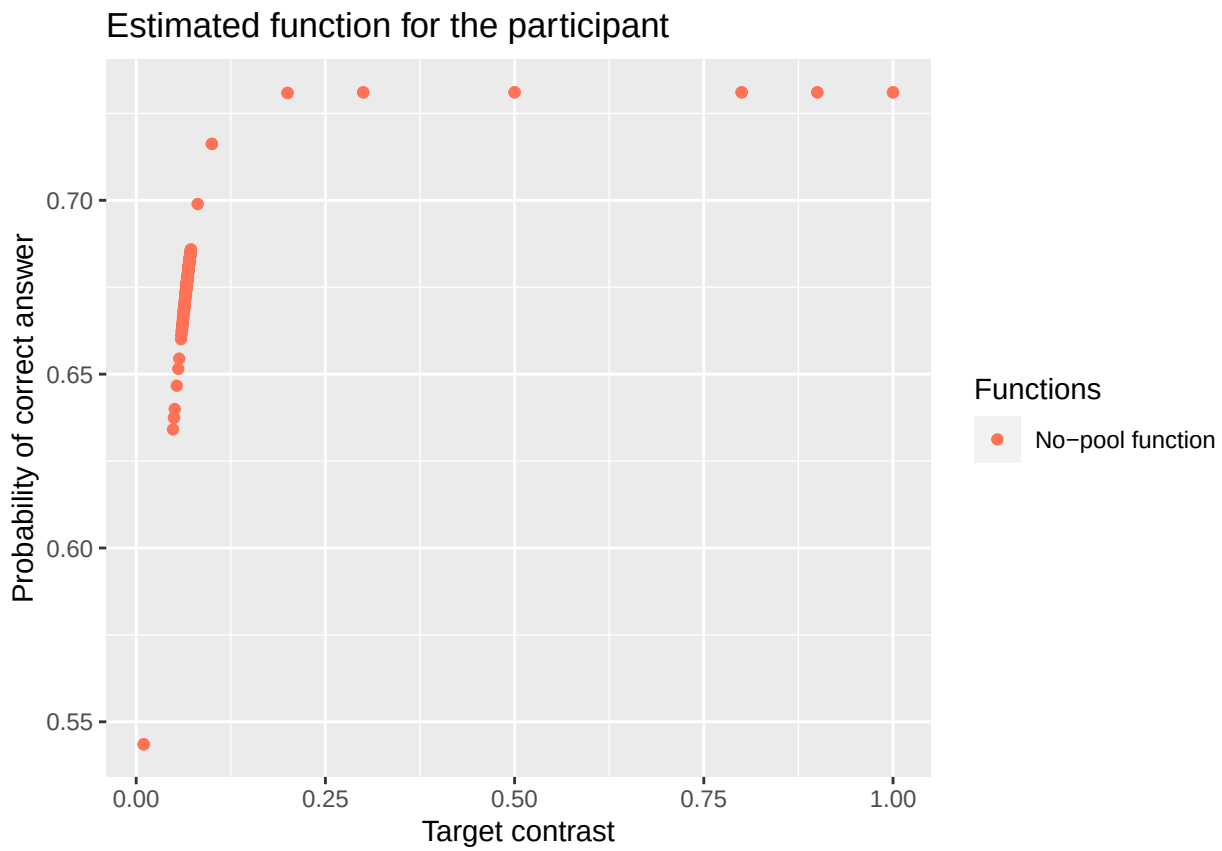
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```


Estimated function for the participant
Probability of correct answer
0.70
0.65
0.60
0.55
0.50
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

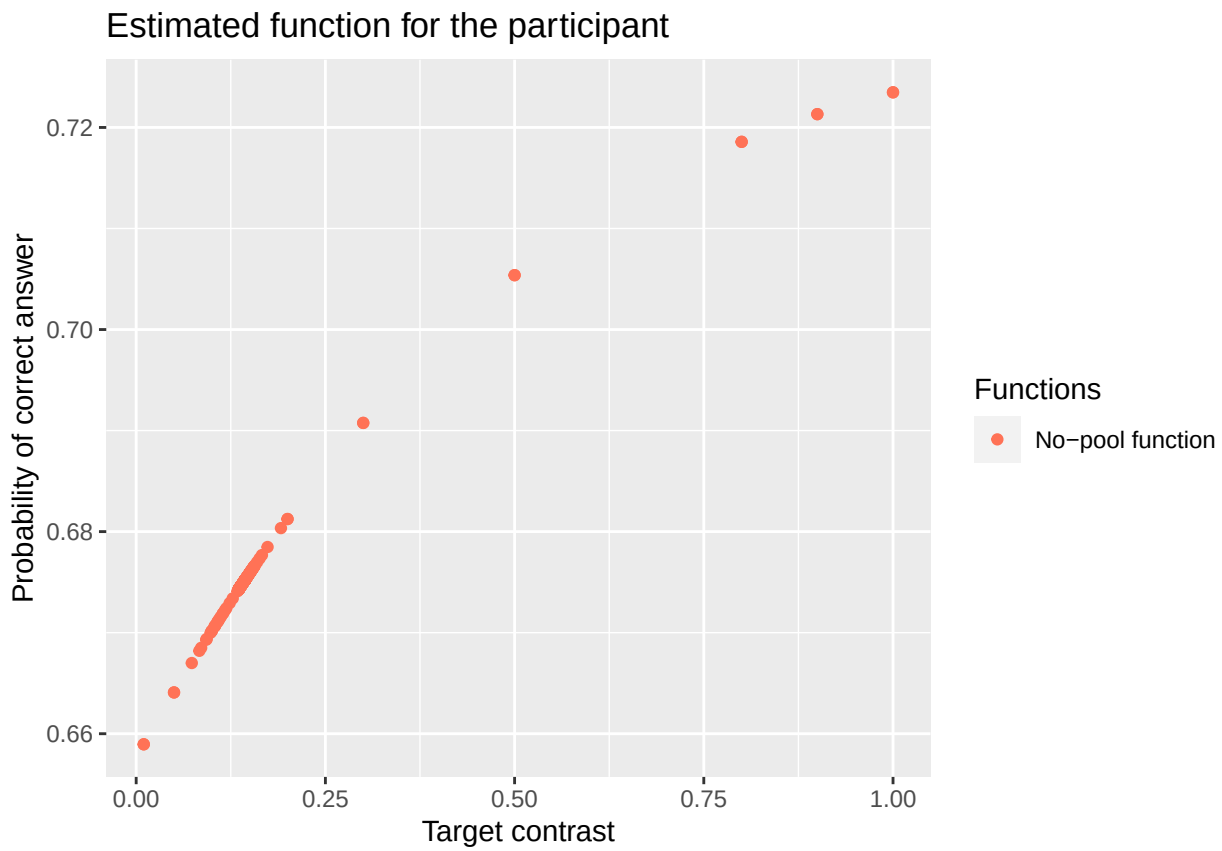



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

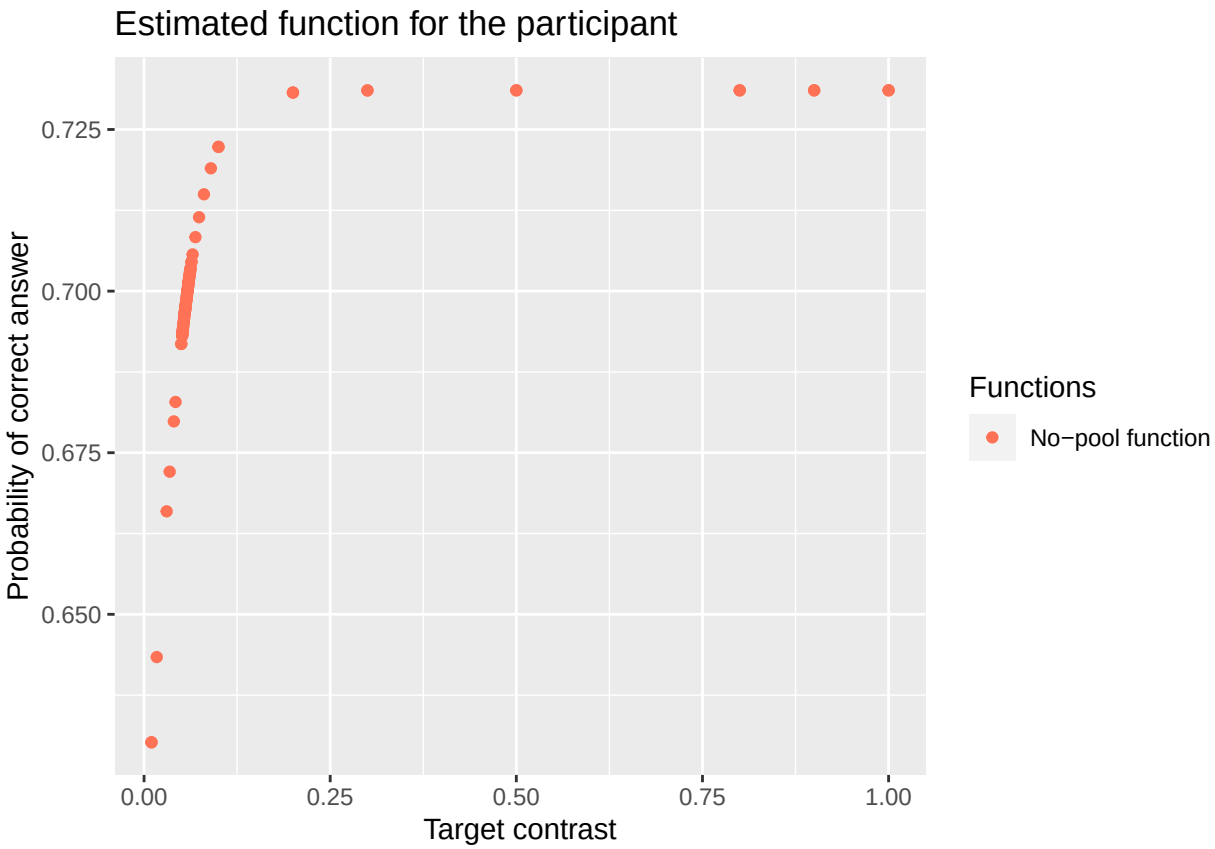

Estimated function for the participant
Probability of correct answer
0.710
0.705
0.700
0.695
0.690
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

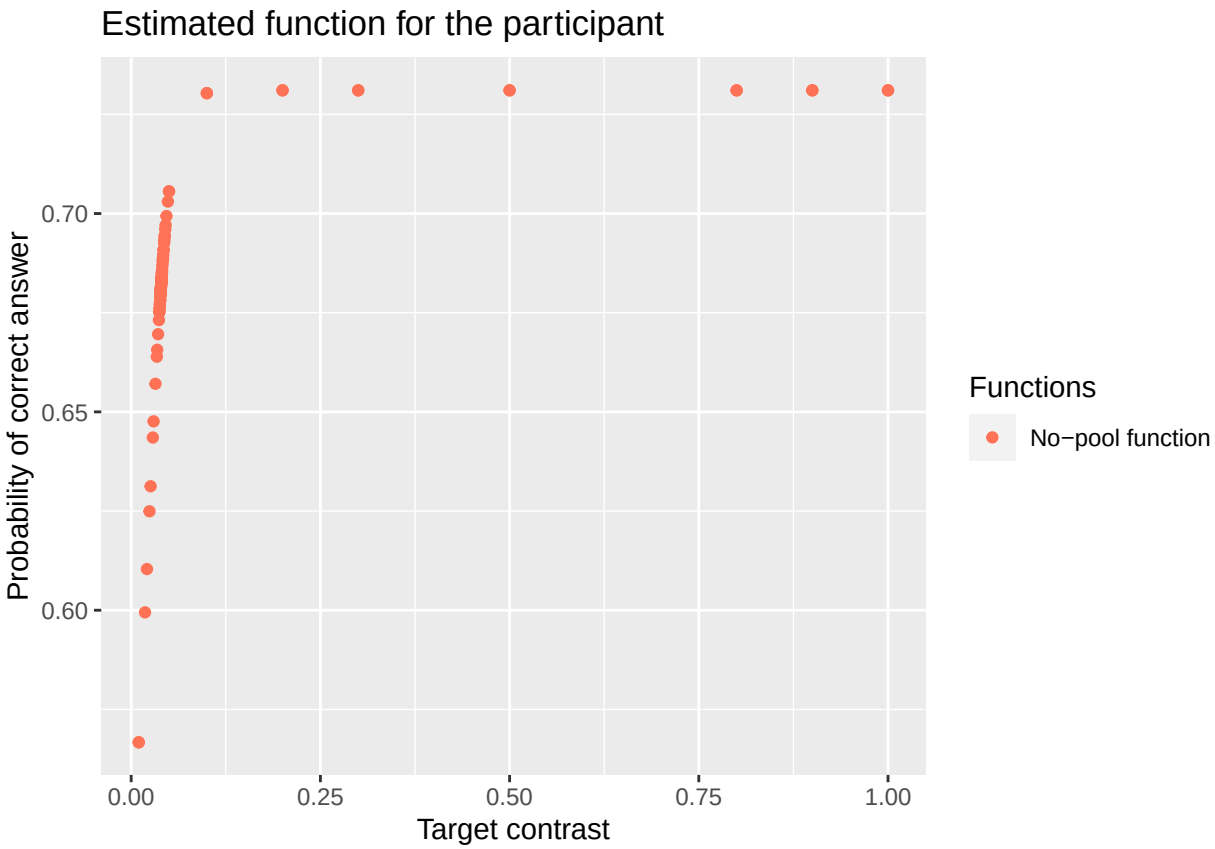


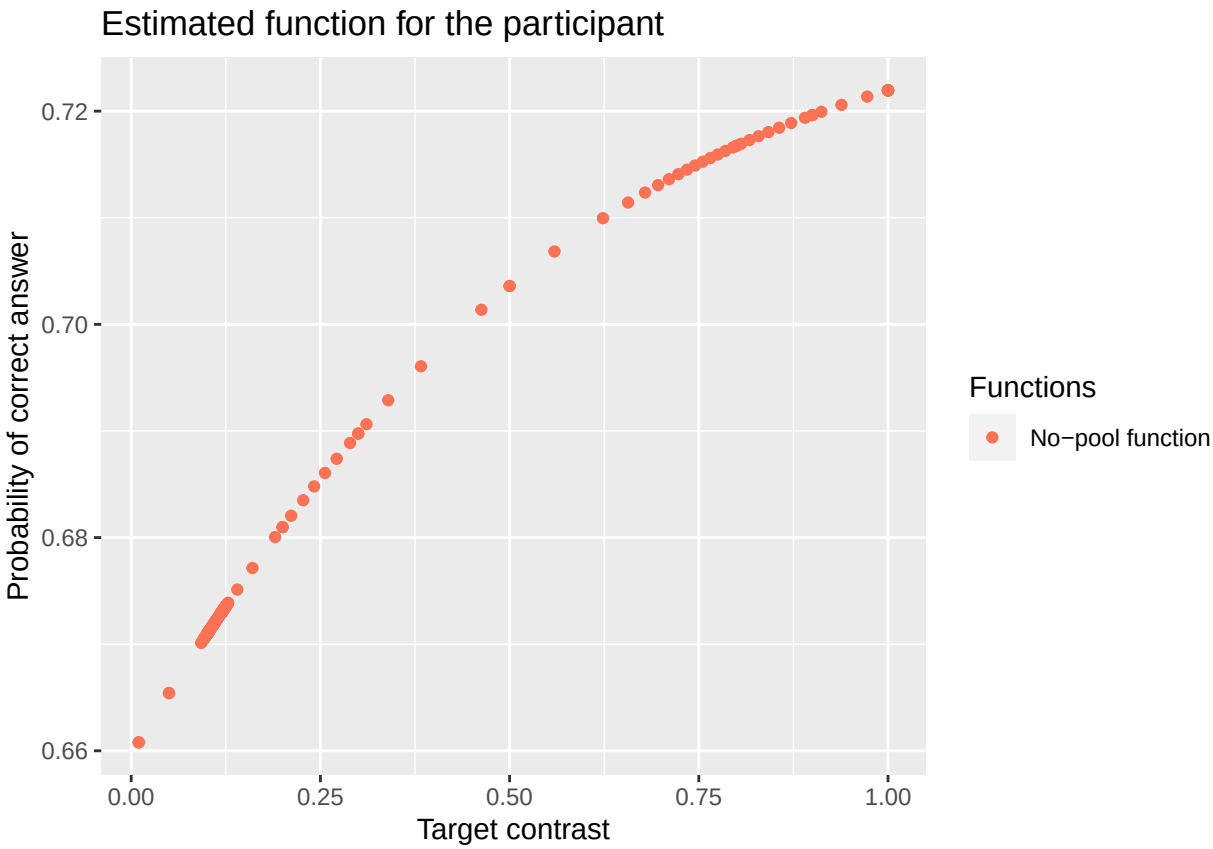

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

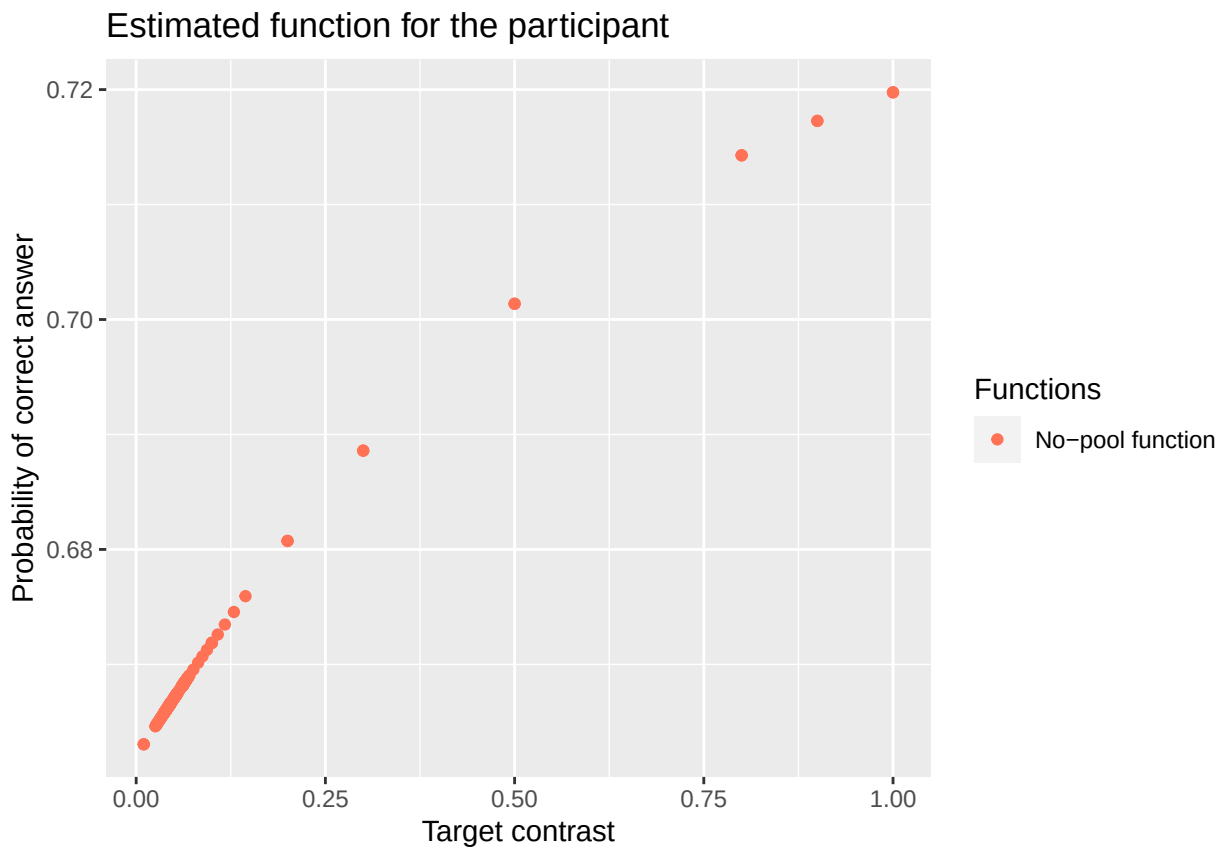


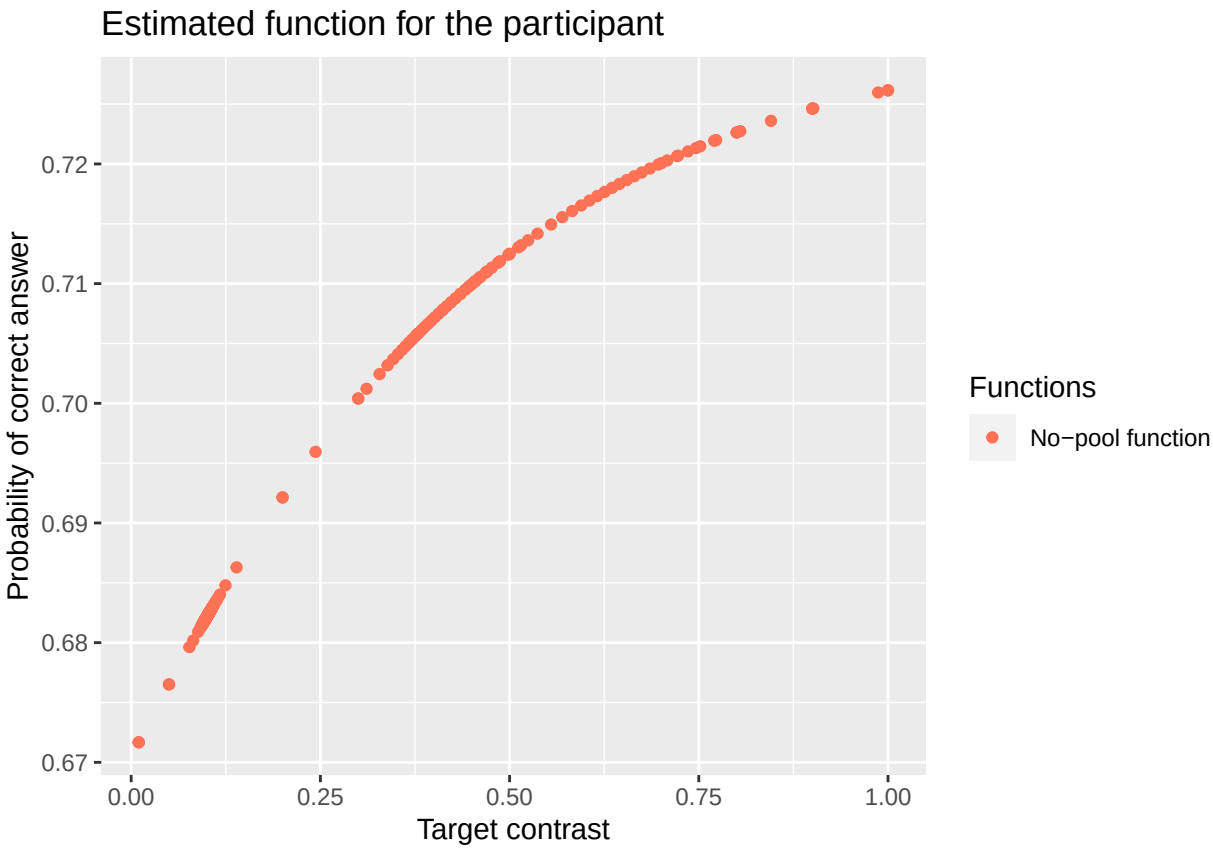

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

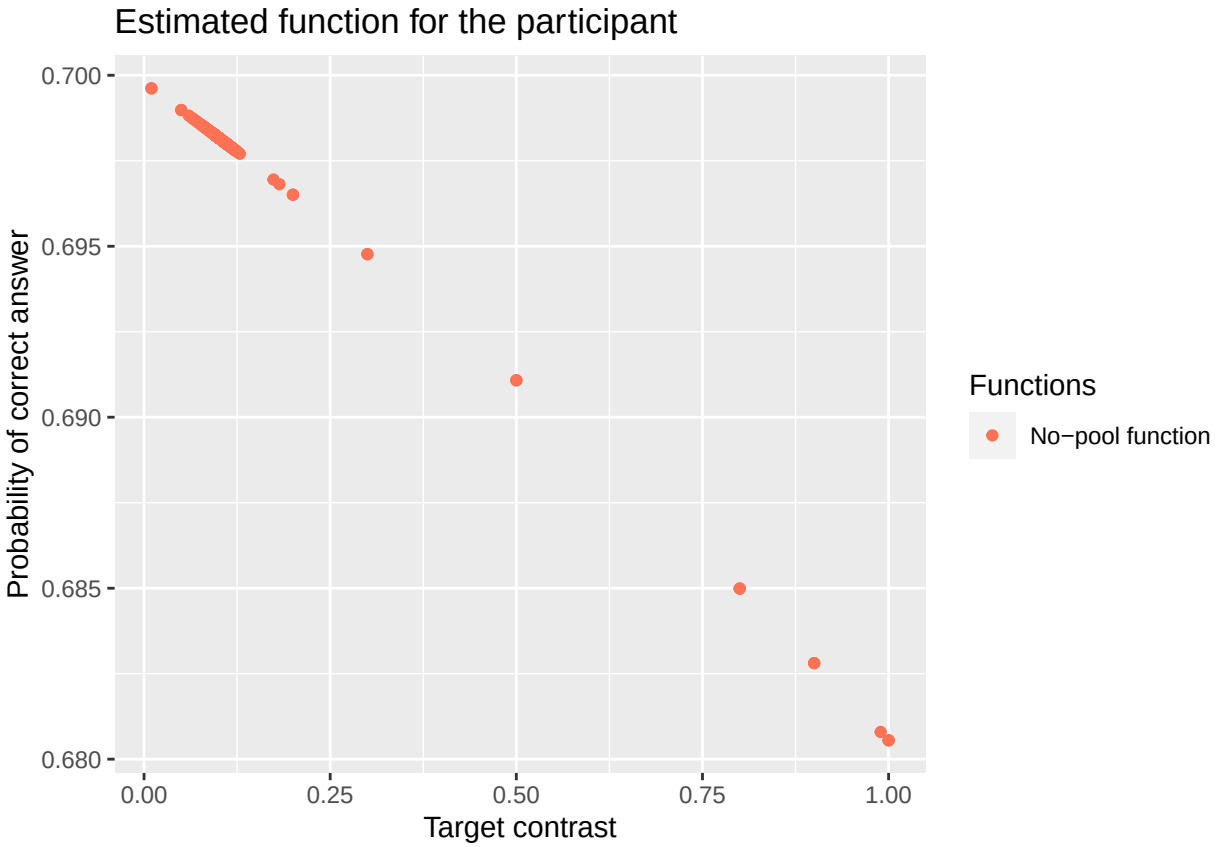

Estimated function for the participant
Probability of correct answer
0.725
0.700
0.675
0.650
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

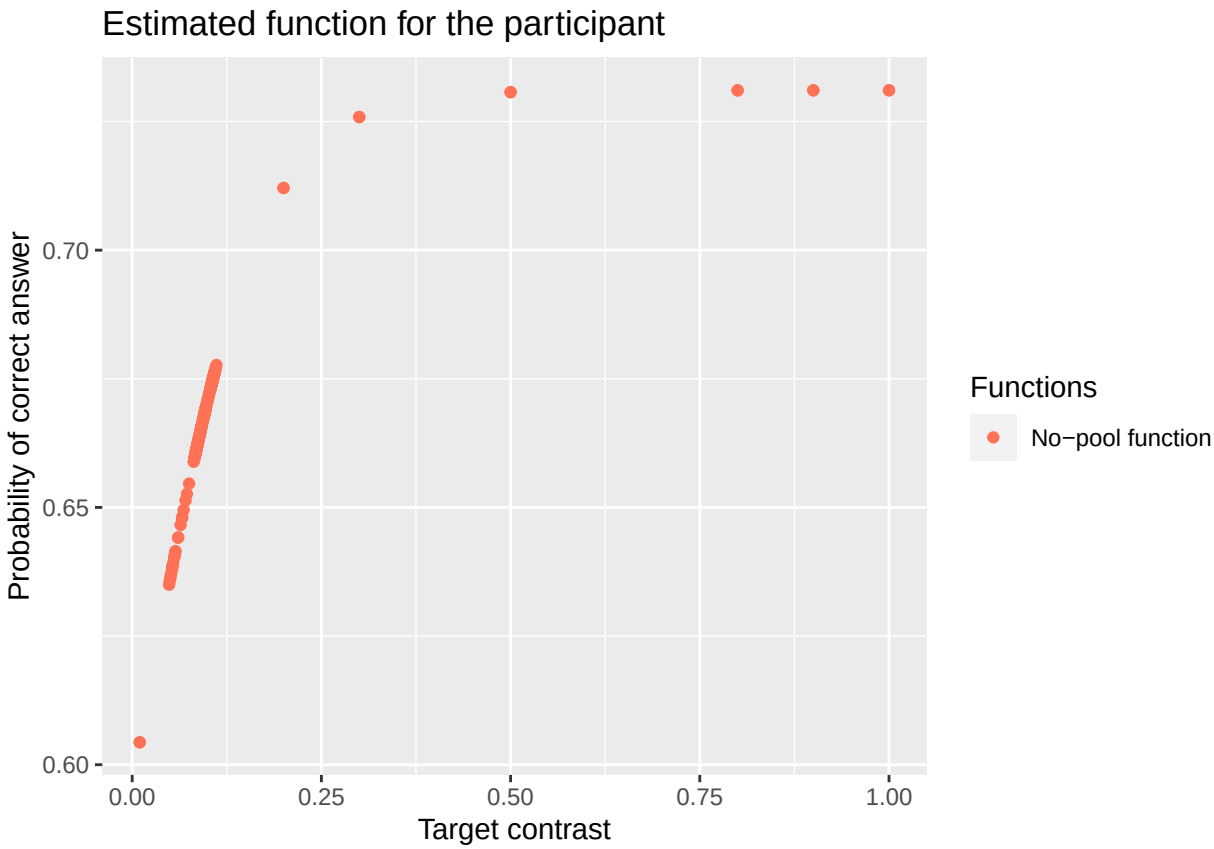

Estimated function for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

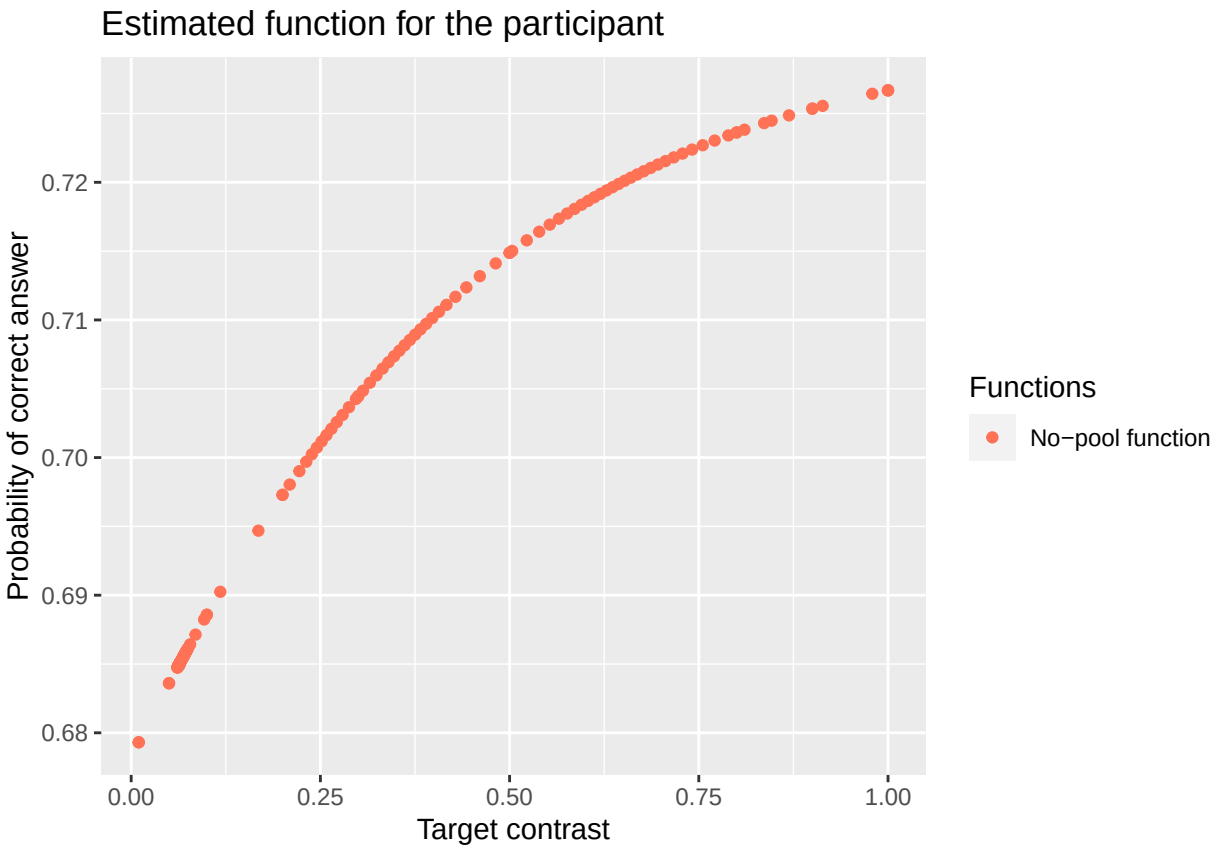

Estimated function for the participant
Probability of correct answer
0.72
0.70
0.68
0.66
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

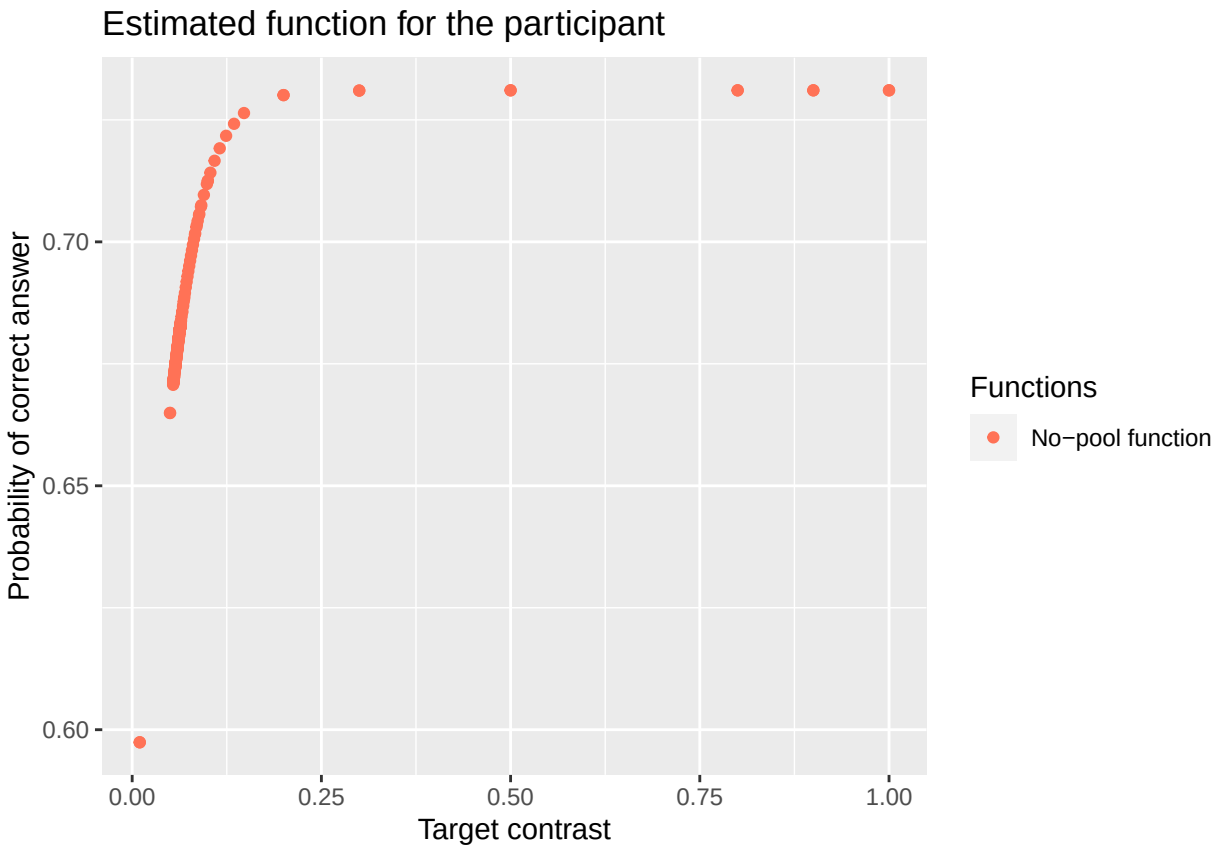

Estimated function for the participant
Probability of correct answer
0.72
0.70
0.68
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

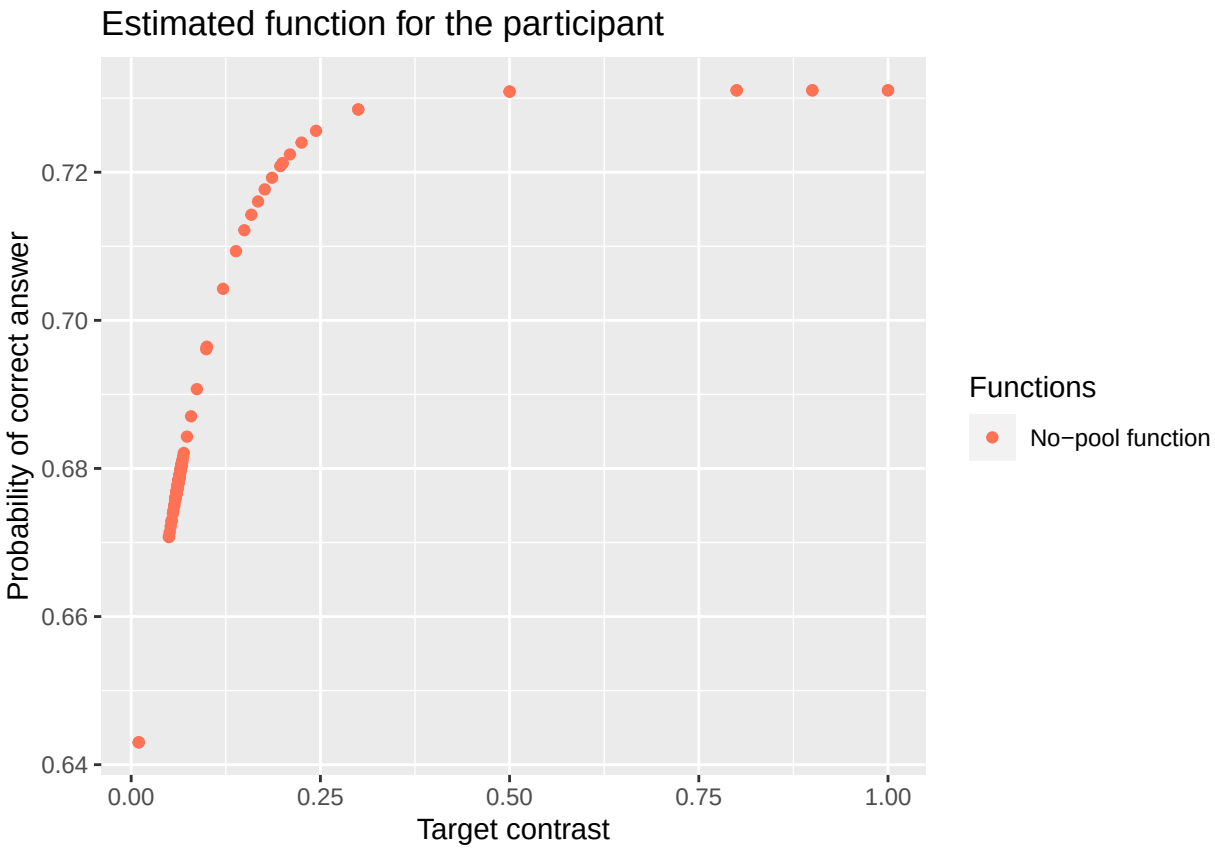

Estimated function for the participant
Probability of correct answer
0.72
0.71
0.70
0.69
0.68
0.67
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

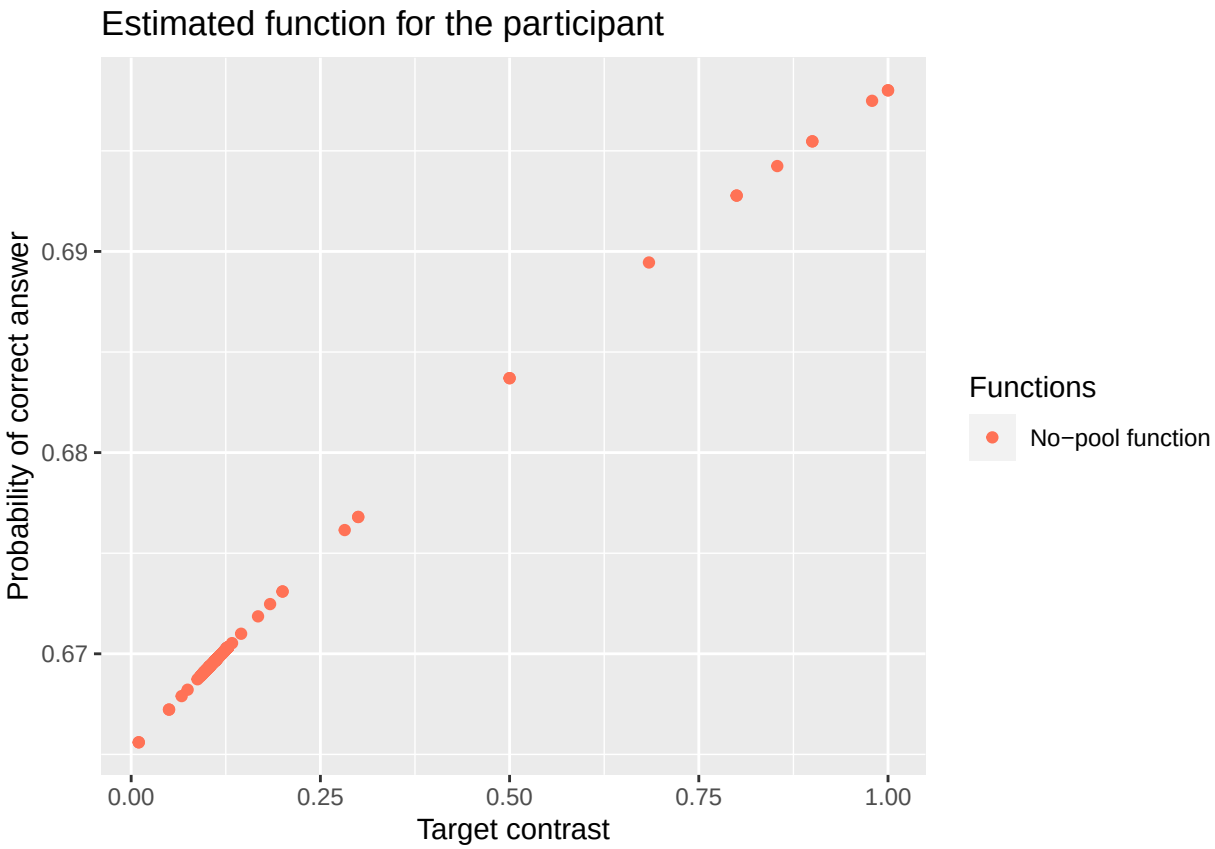

Estimated function for the participant
Probability of correct answer
0.700
0.695
0.690
0.685
0.680
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

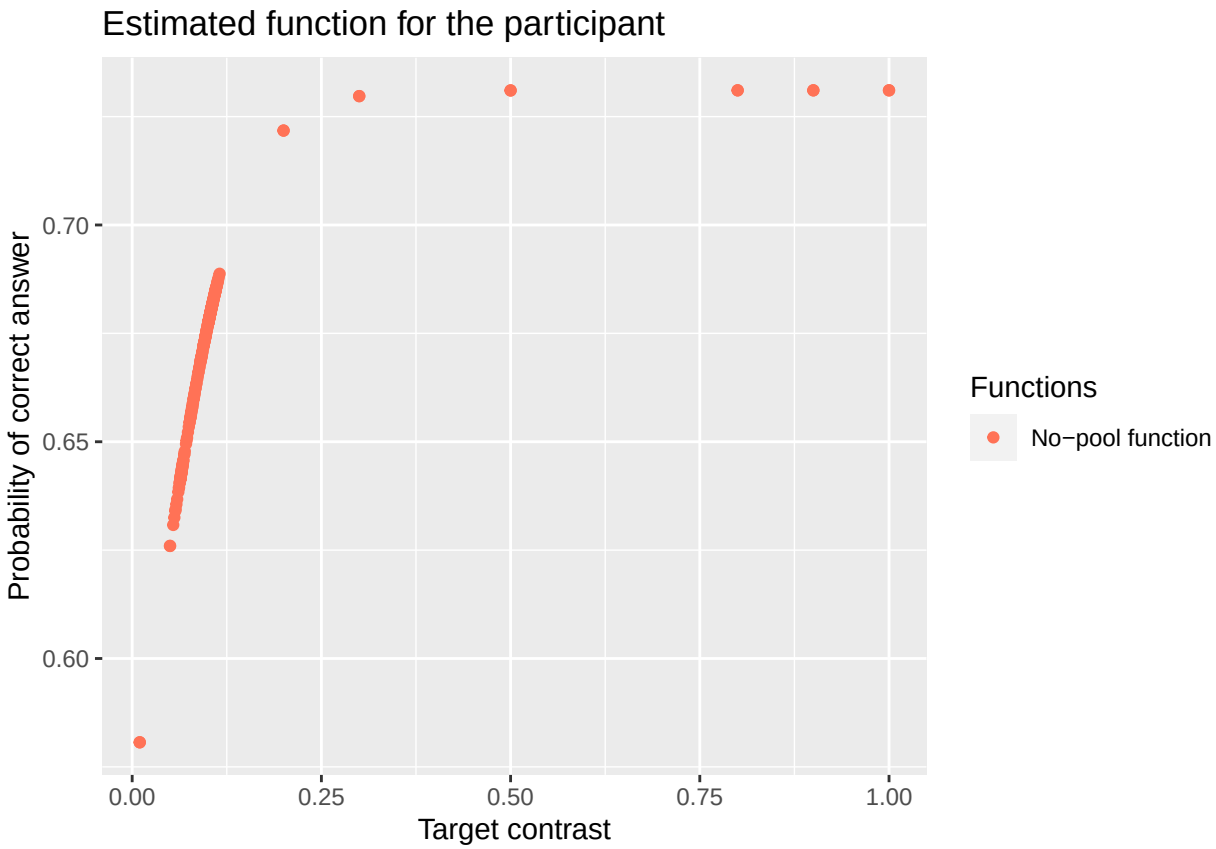

Estimated function for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

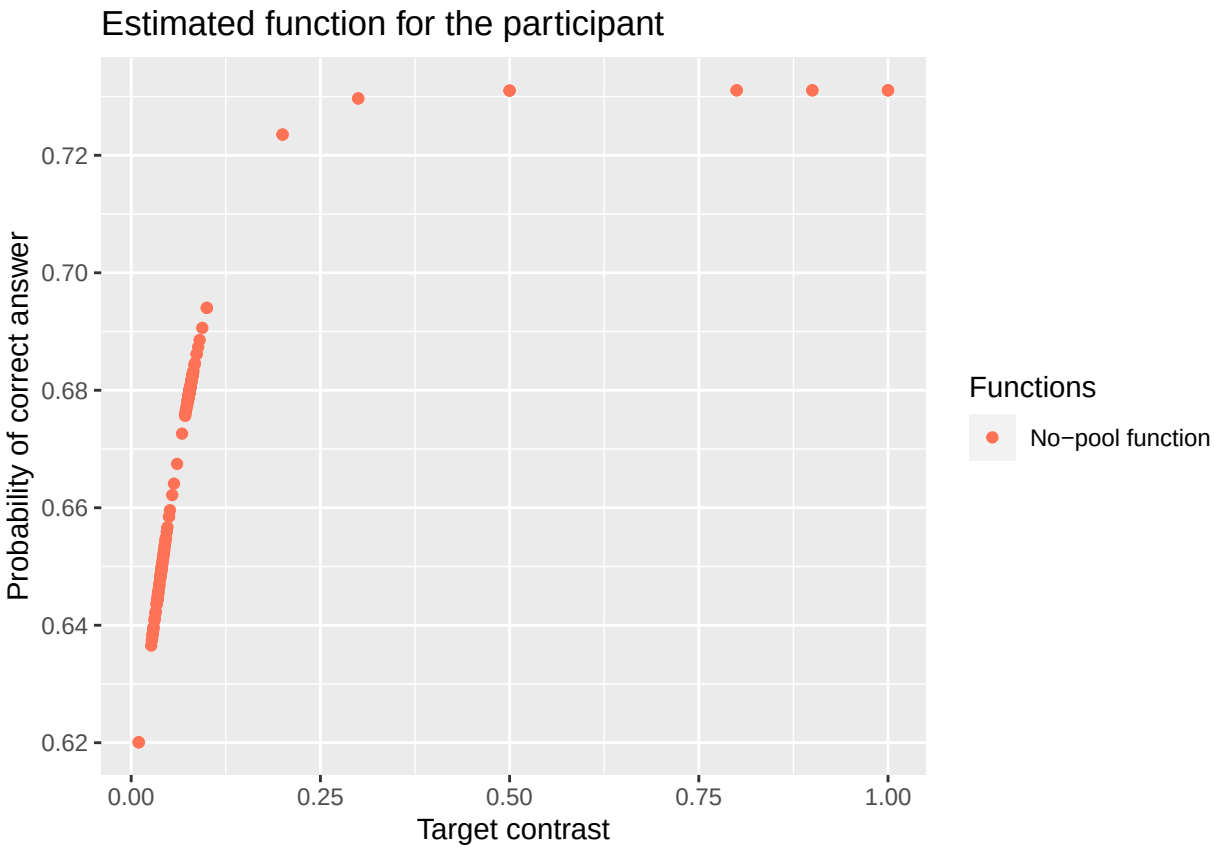

Estimated function for the participant
Probability of correct answer
0.72
0.71
0.70
0.69
0.68
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function


Estimated function for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

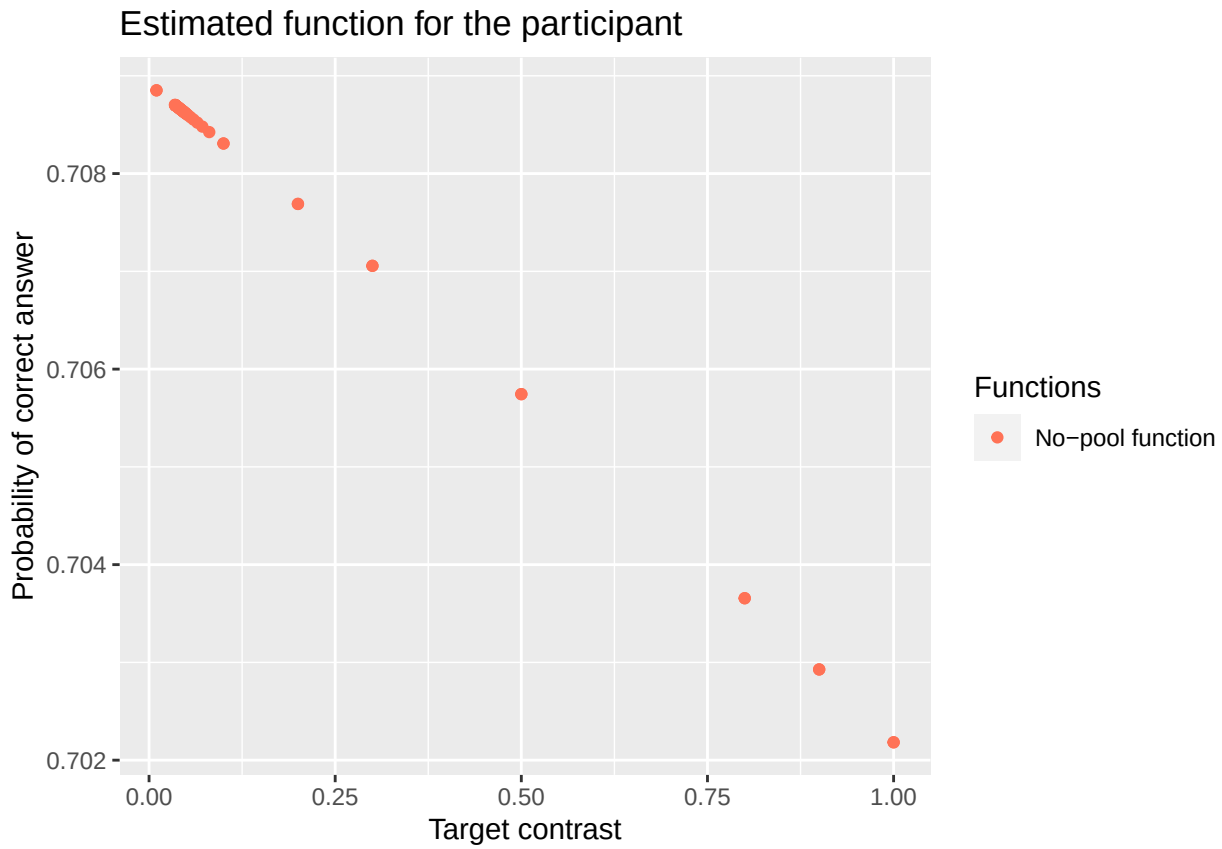

Estimated function for the participant
Probability of correct answer
0.72
0.70
0.68
0.66
0.64
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function


Estimated function for the participant
Probability of correct answer
0.69
0.68
0.67
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function

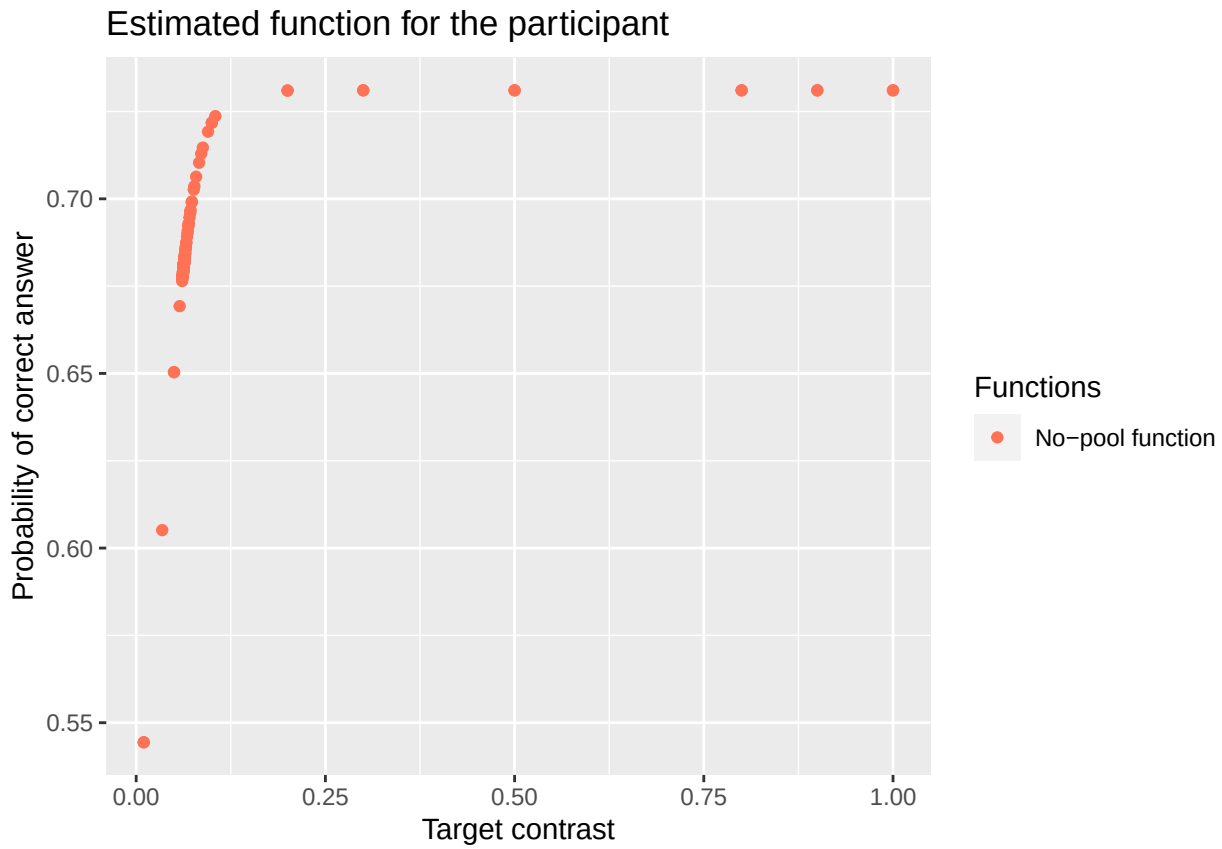

Estimated function for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function



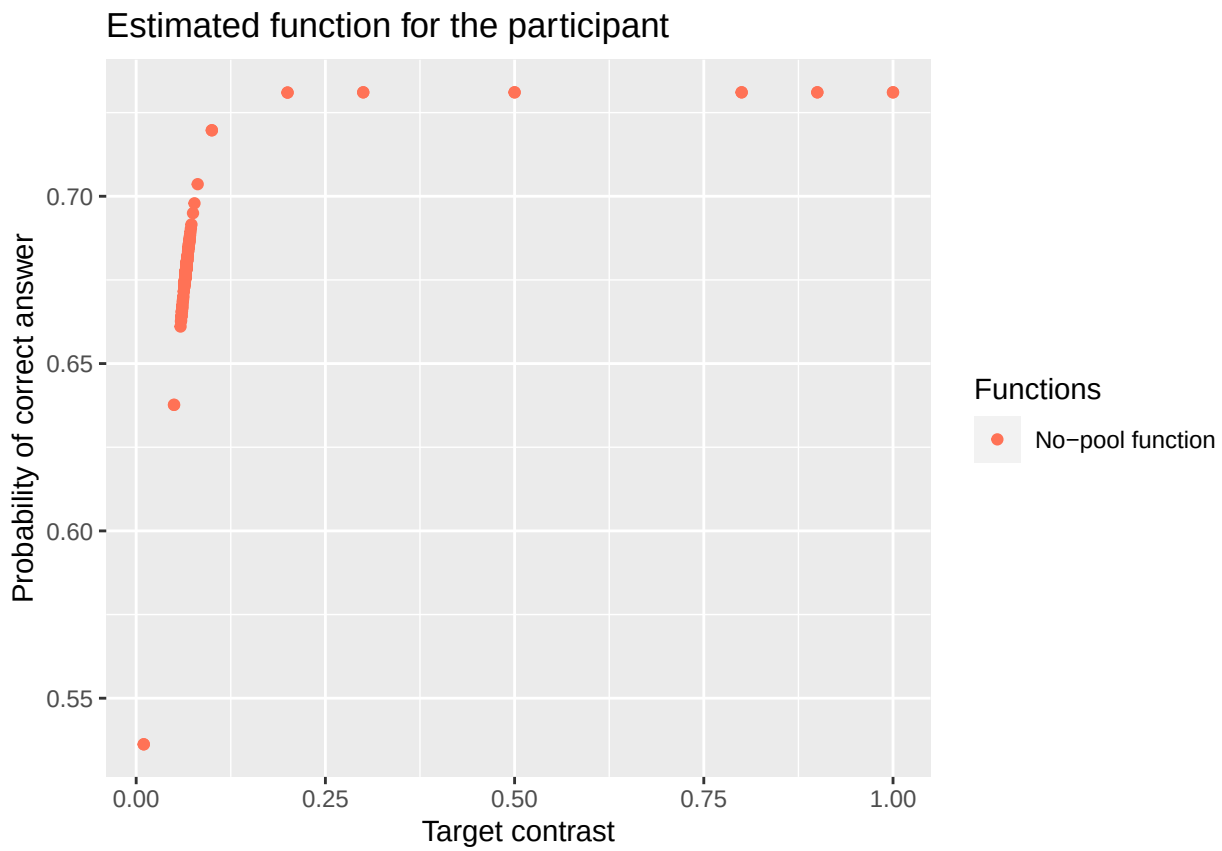

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



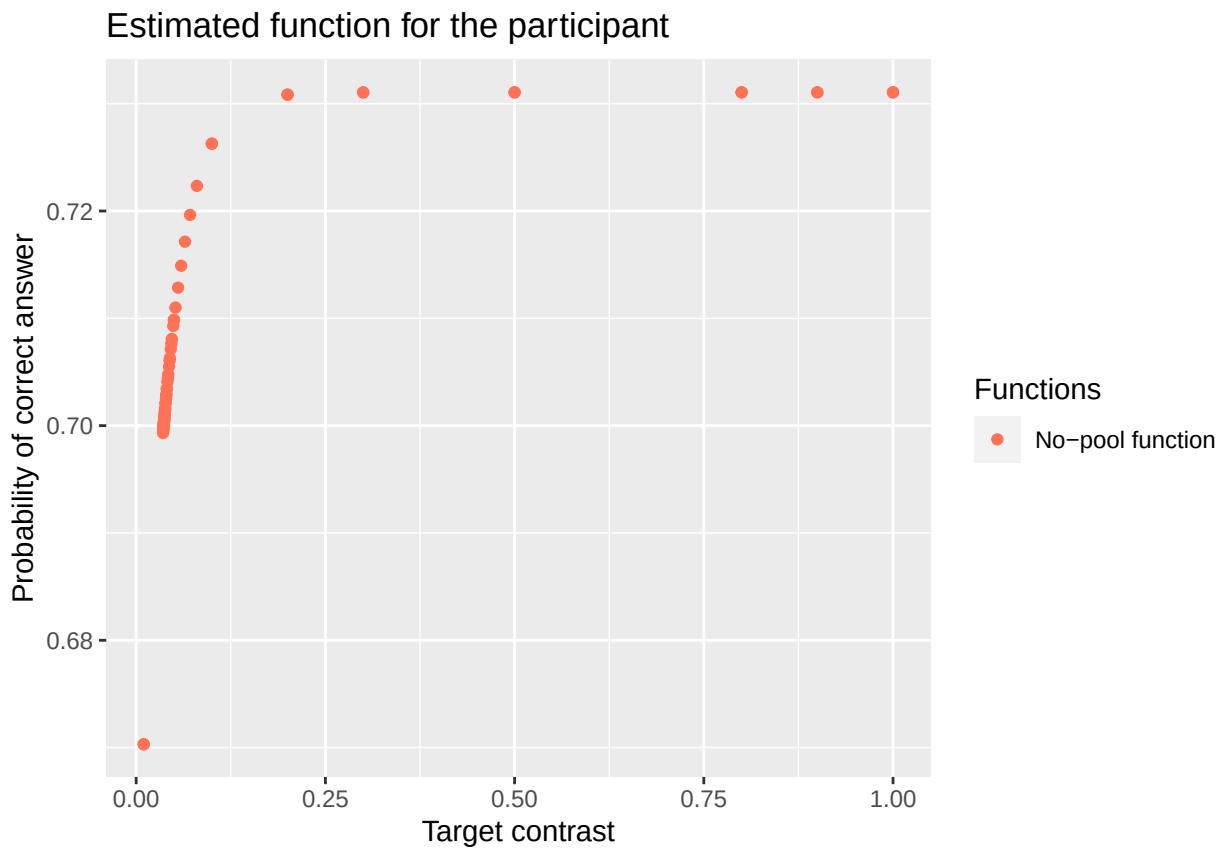

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



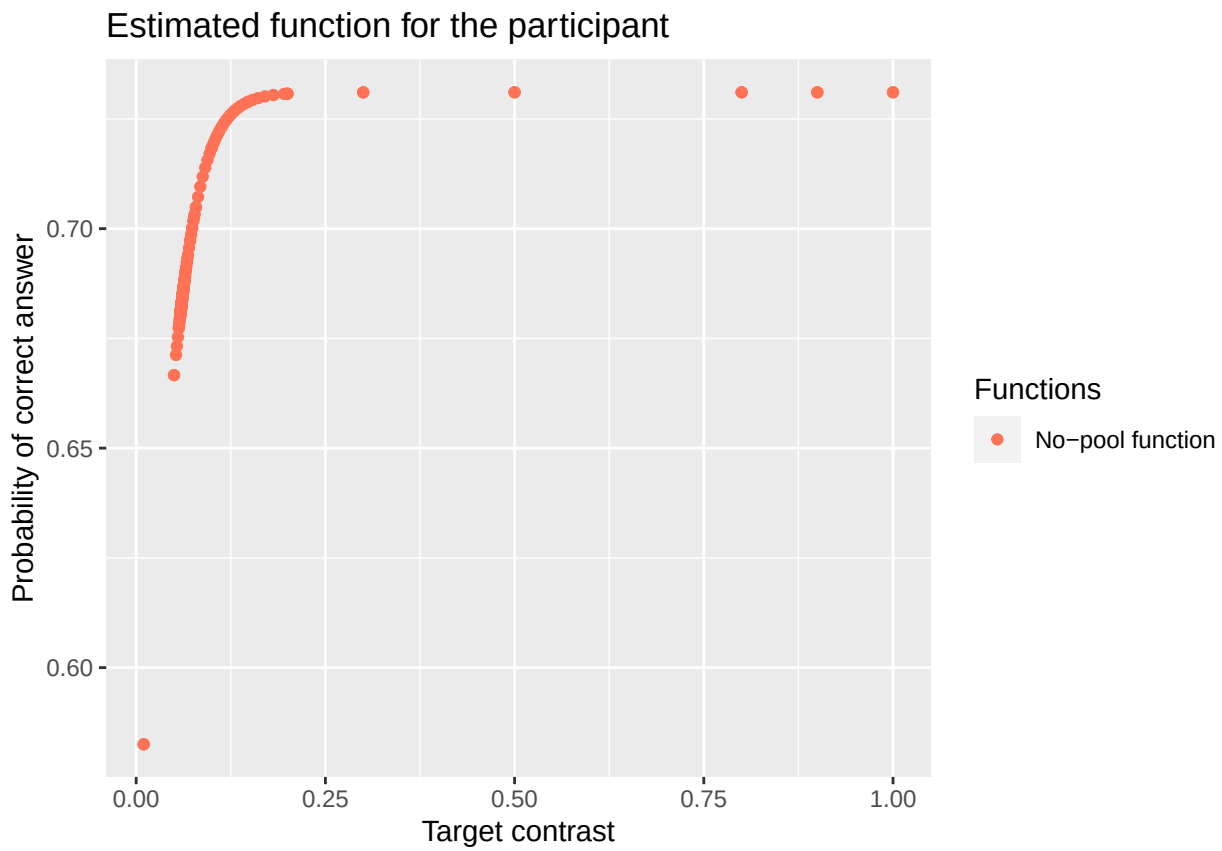

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



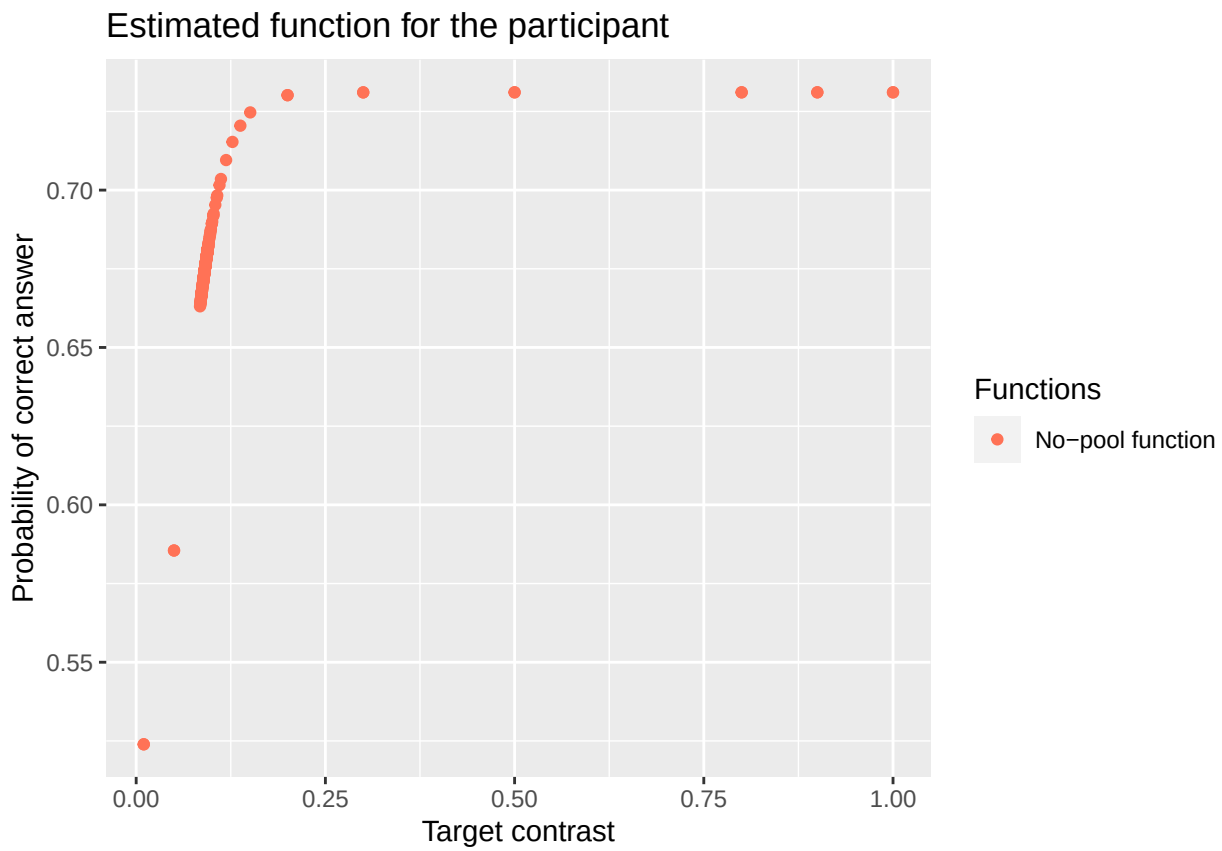

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



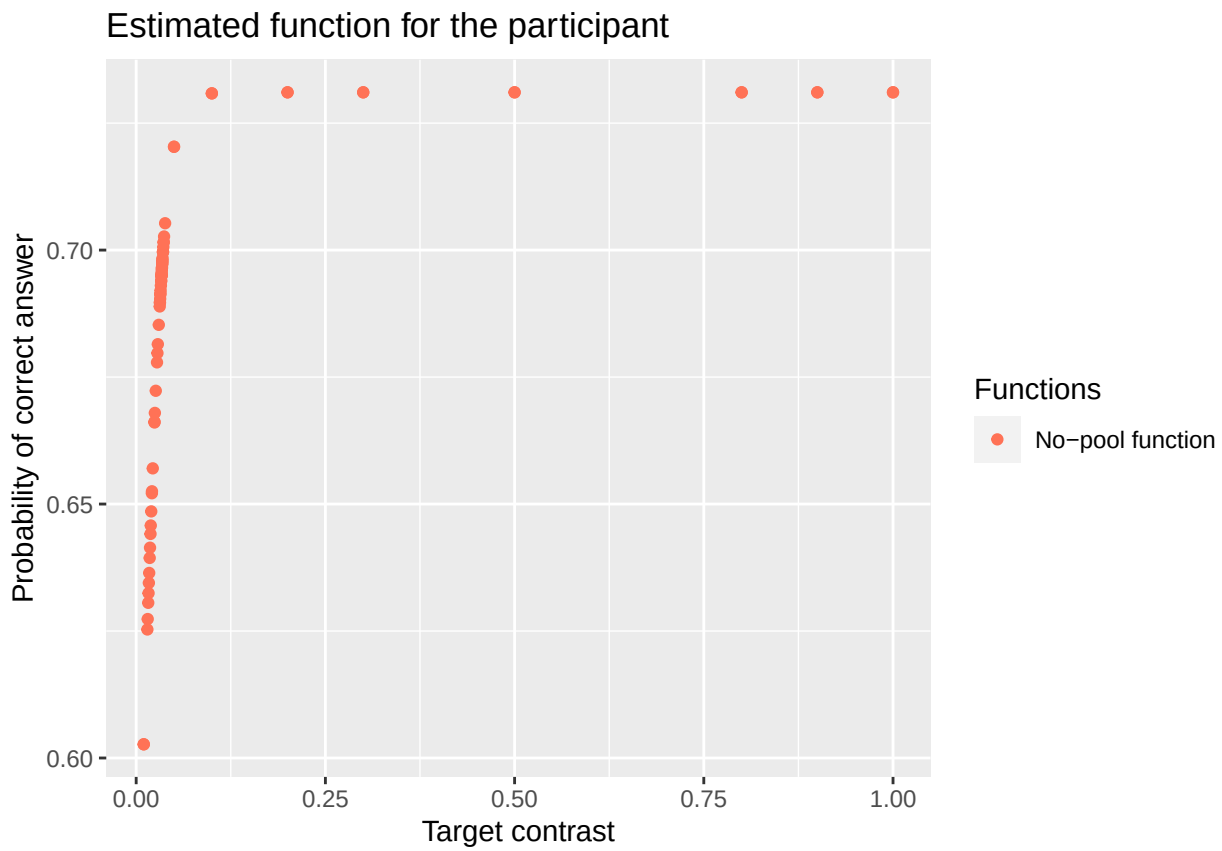

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



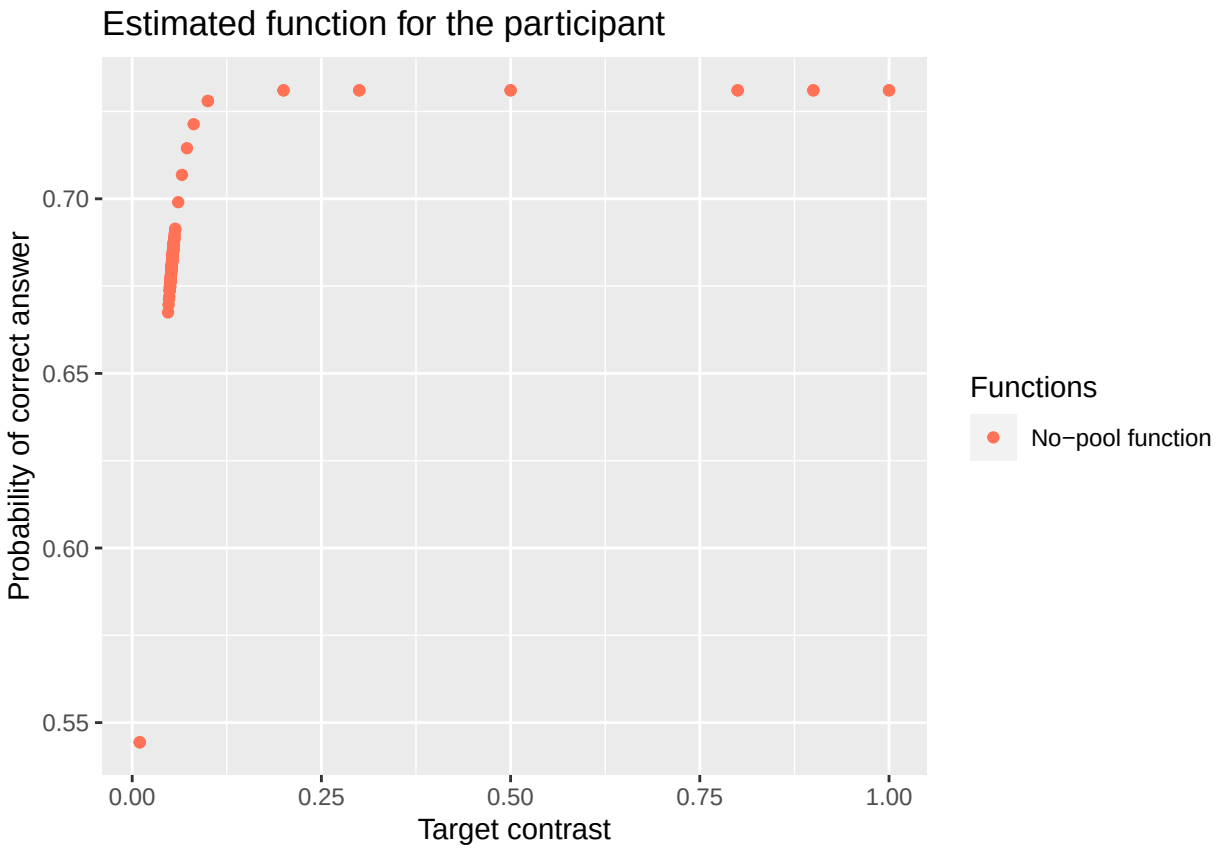

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

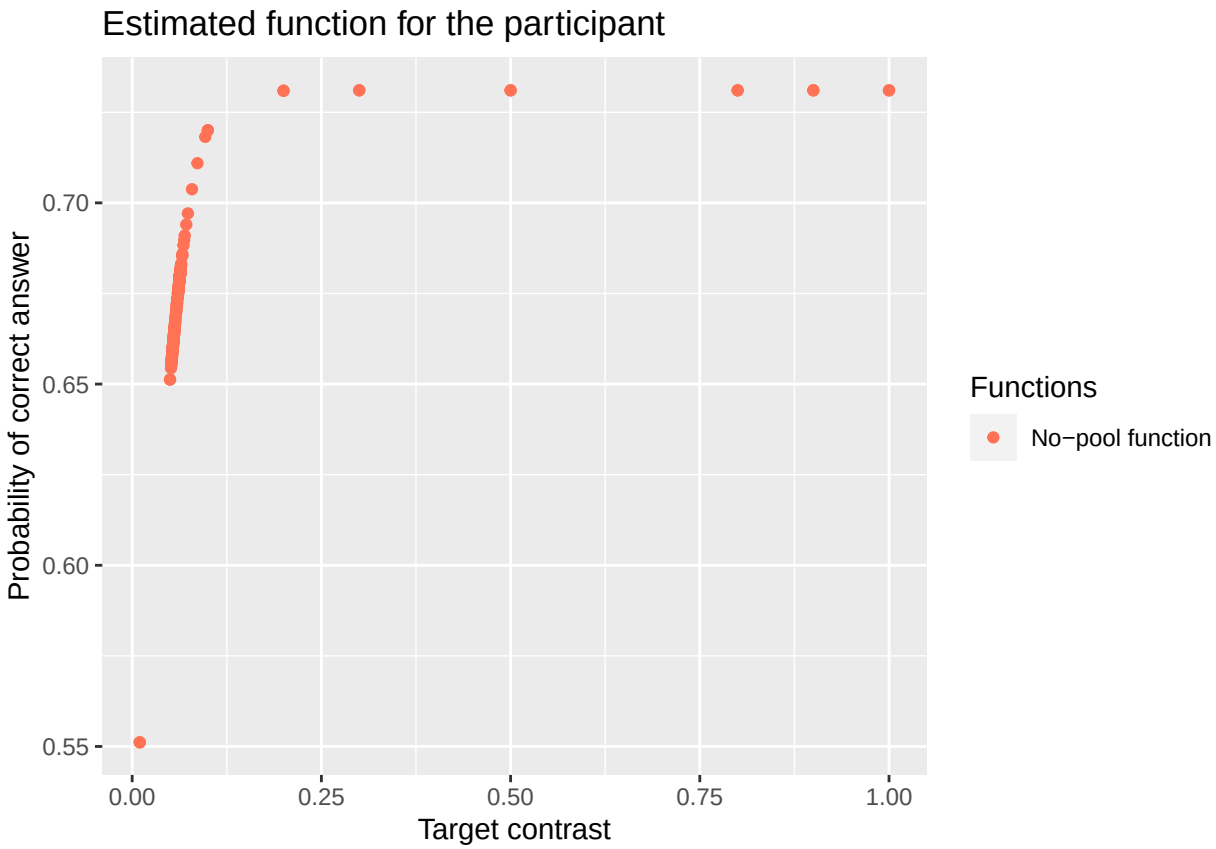


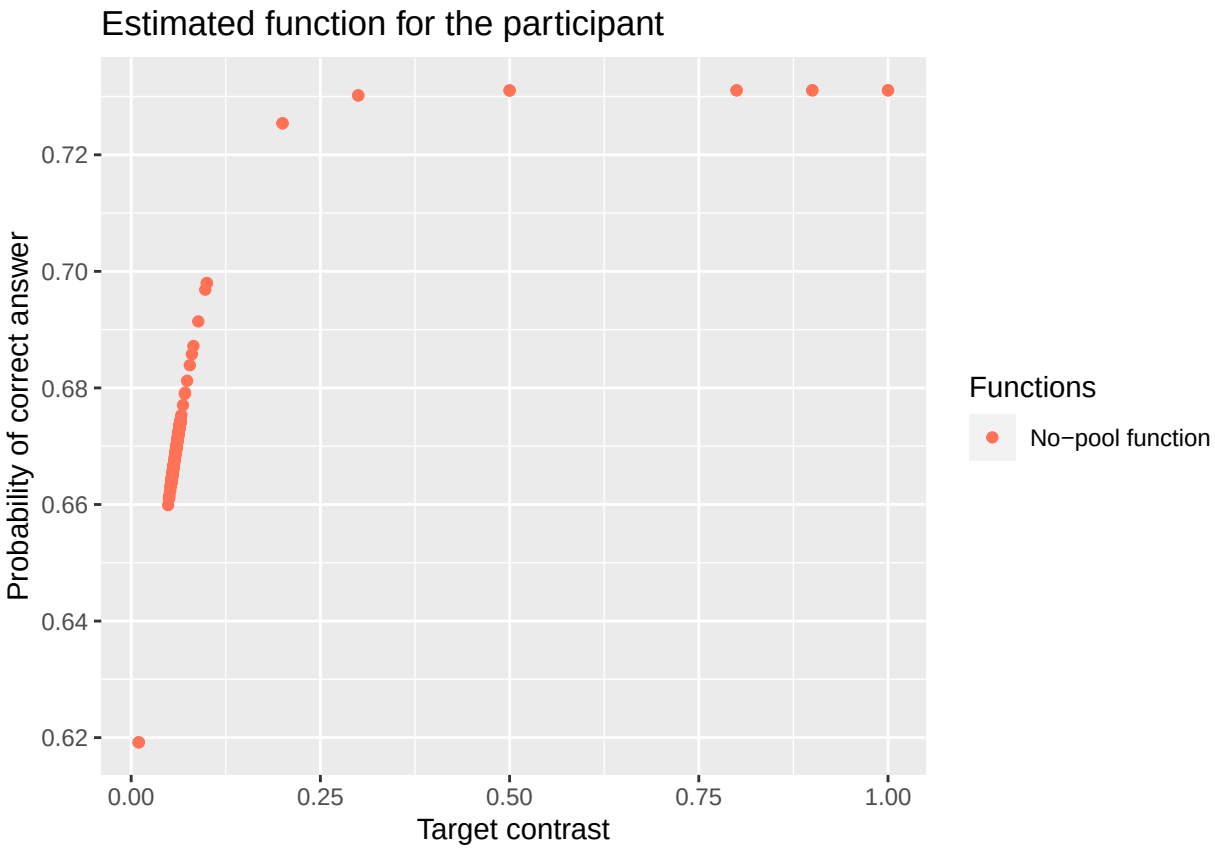

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

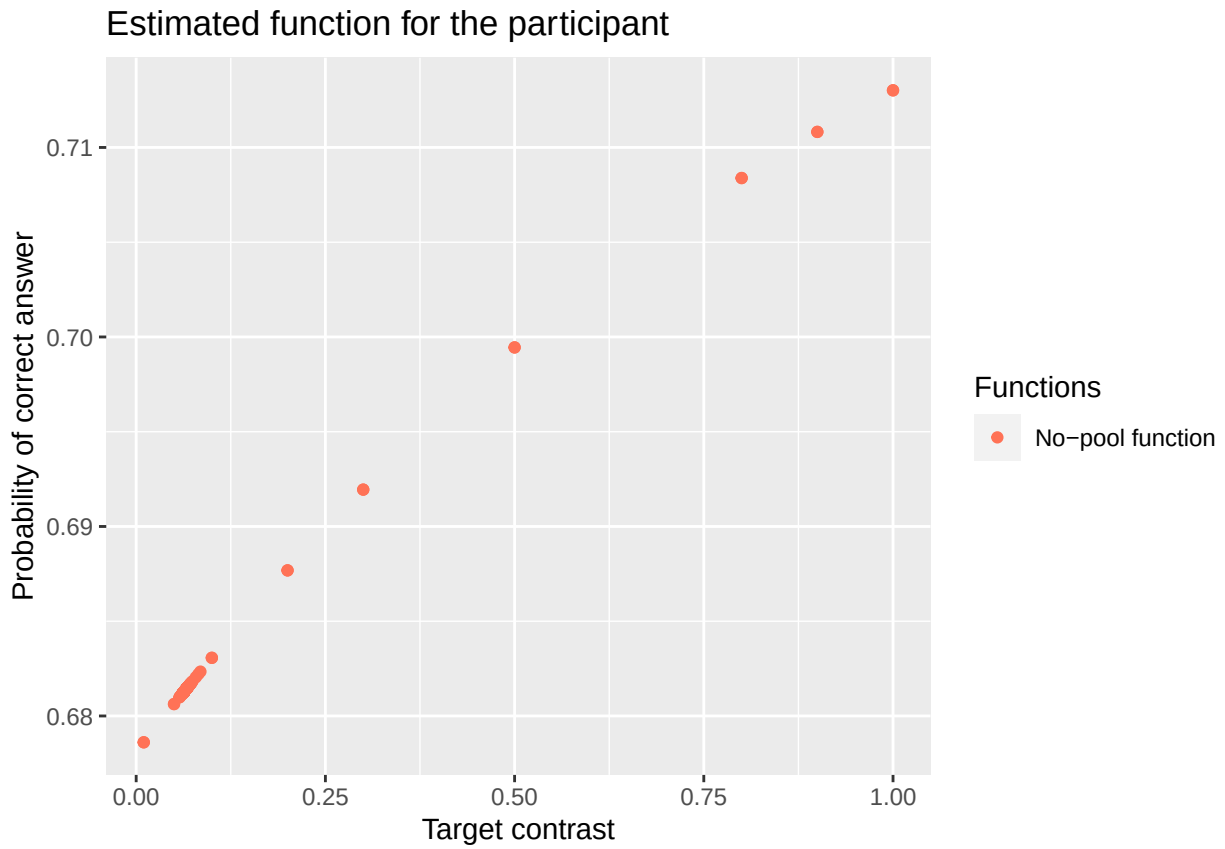



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```


Estimated function for the participant
Probability of correct answer
0.70
0.65
0.60
0.55
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function


Estimated function for the participant
Probability of correct answer
0.70
0.65
0.60
0.55
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function


Estimated function for the participant
Probability of correct answer
0.72
0.70
0.68
0.66
0.64
0.62
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
No−pool function



There does not appear to be sufficient data to plot the logistic functions: data points are sparse and the function is highly variable from participant to participant. iv. on top of those plots, add the estimated functions (on the *target.contrast* range from 0-1) for each subject based on a partial pooling model (use `glmer` from the package `lme4`) where unique intercepts and slopes for *target.contrast* are modelled for each *subject*

```
model_1 <- lme4::glmer(correct ~ target.contrast + (1 | subject), family = binomial, data = staircase)
summary(model_1)
```

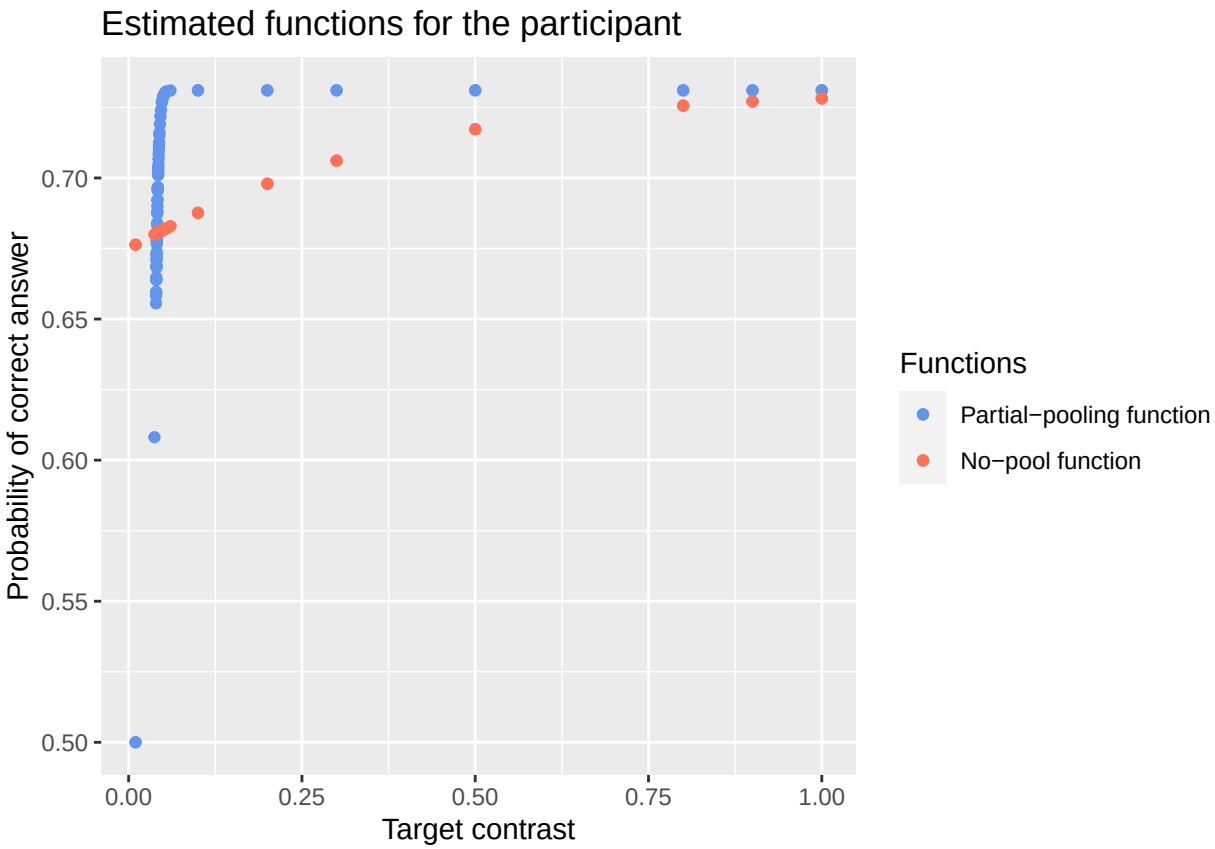
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ target.contrast + (1 | subject)
##    Data: staircase
##
##      AIC      BIC   logLik deviance df.resid
##   5999.0   6018.9  -2996.5   5993.0     5600
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.6646  0.1464  0.5603  0.5902  0.7214
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.03959  0.199
## Number of obs: 5603, groups:  subject, 29
##
## Fixed effects:
```

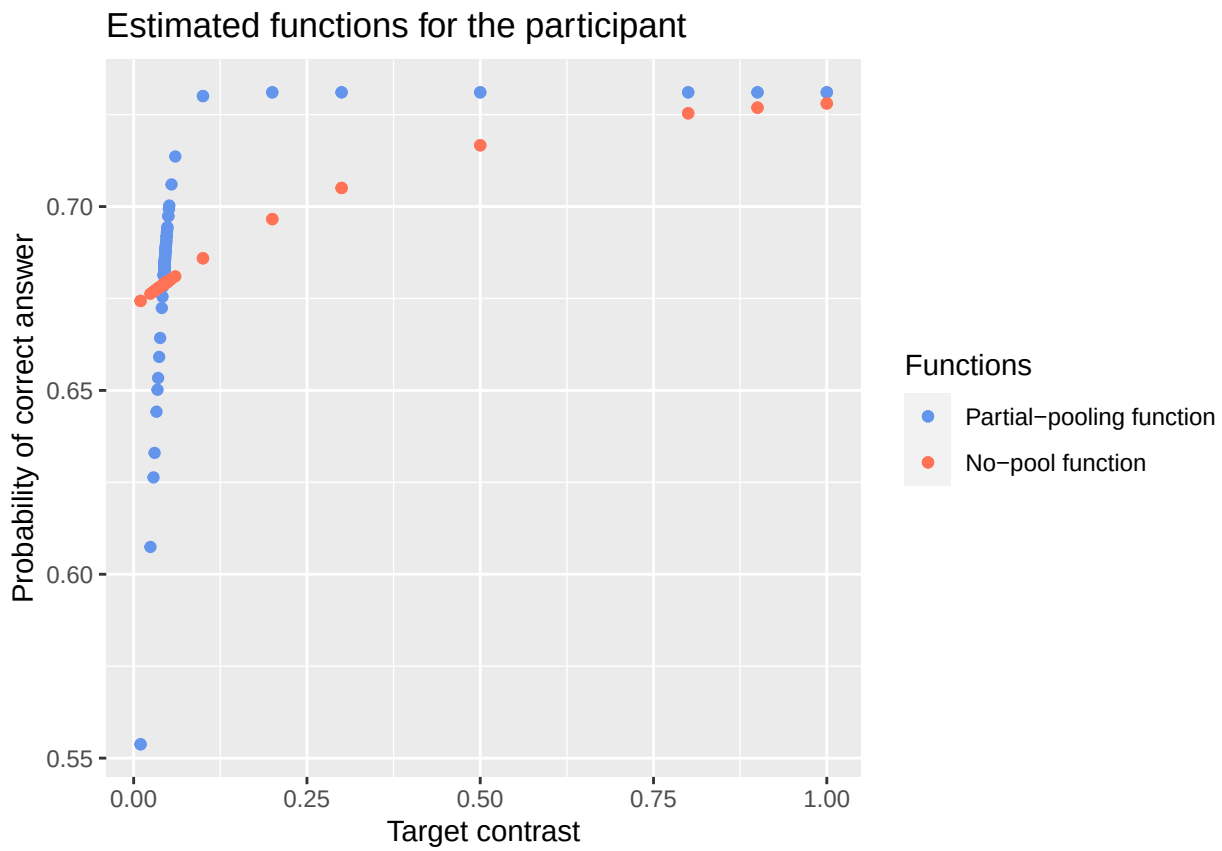
```
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.88538    0.06121   14.46  < 2e-16 ***
## target.contrast  3.20846    0.40207    7.98 1.47e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## trgt.cntrst -0.538
```

```r
staircase <- staircase %>%
  mutate(pooled_inv = inv.logit(fitted.values(model_1)))

for (i in 1:29) {
  #same as in previous exercise
  loopdf <- staircase %>%
    filter(subject == i)
  loopmodel <- glm(correct ~ target.contrast, data = loopdf, family = binomial)
  loopdf <- loopdf %>%
    mutate(inv = inv.logit(loopmodel$fitted.values))
  #now for overlaying the pooled function
  plot <- ggplot(loopdf, aes(x = target.contrast, y = inv)) +
    geom_point(aes(y = inv, color = "coral1")) +
    geom_point(aes(y = pooled_inv, color = "cornflowerblue")) +
    scale_color_manual(labels = c("Partial-pooling function", "No-pool function"), values = c("cornflowe
    guides(color=guide_legend("Functions")) +
    labs(title = "Estimated functions for the participant", x = "Target contrast", y = "Probability of
  print(plot) #actually get the plot to show up
}
```

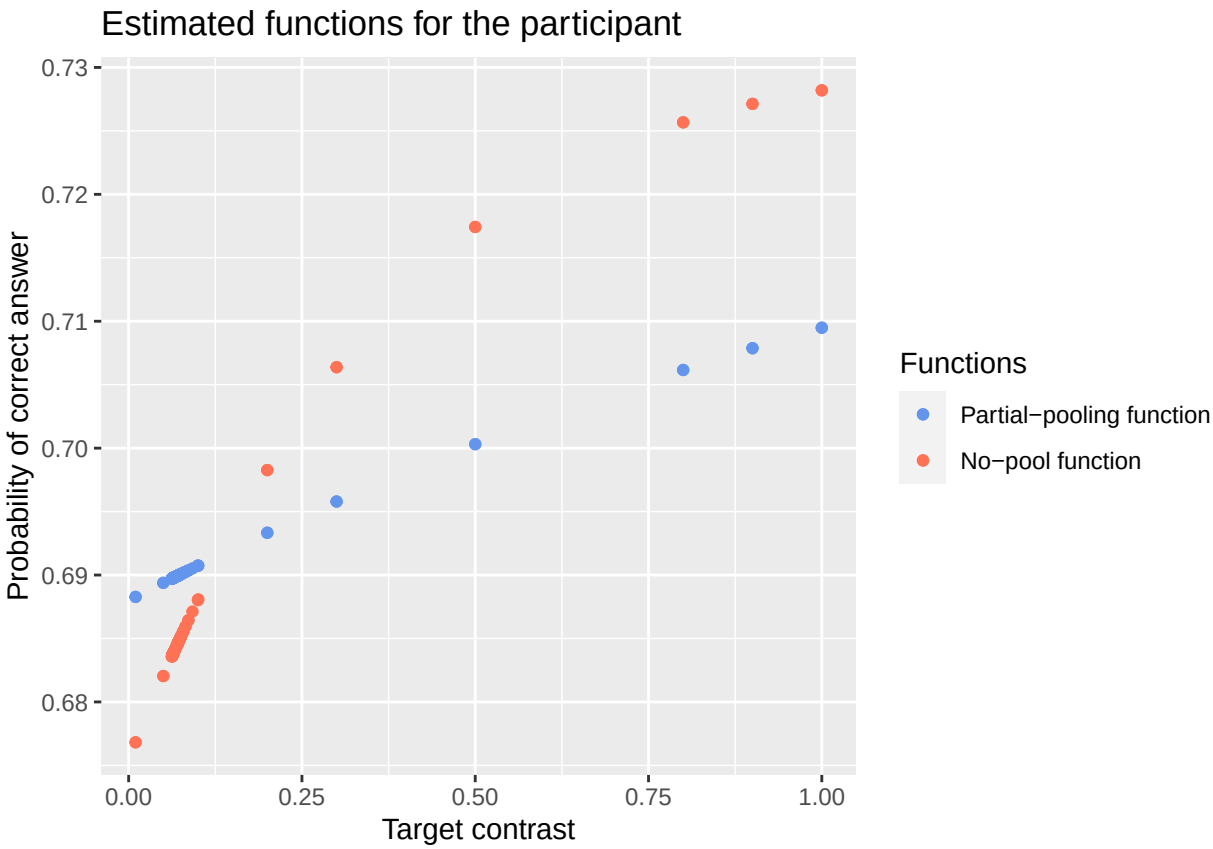
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

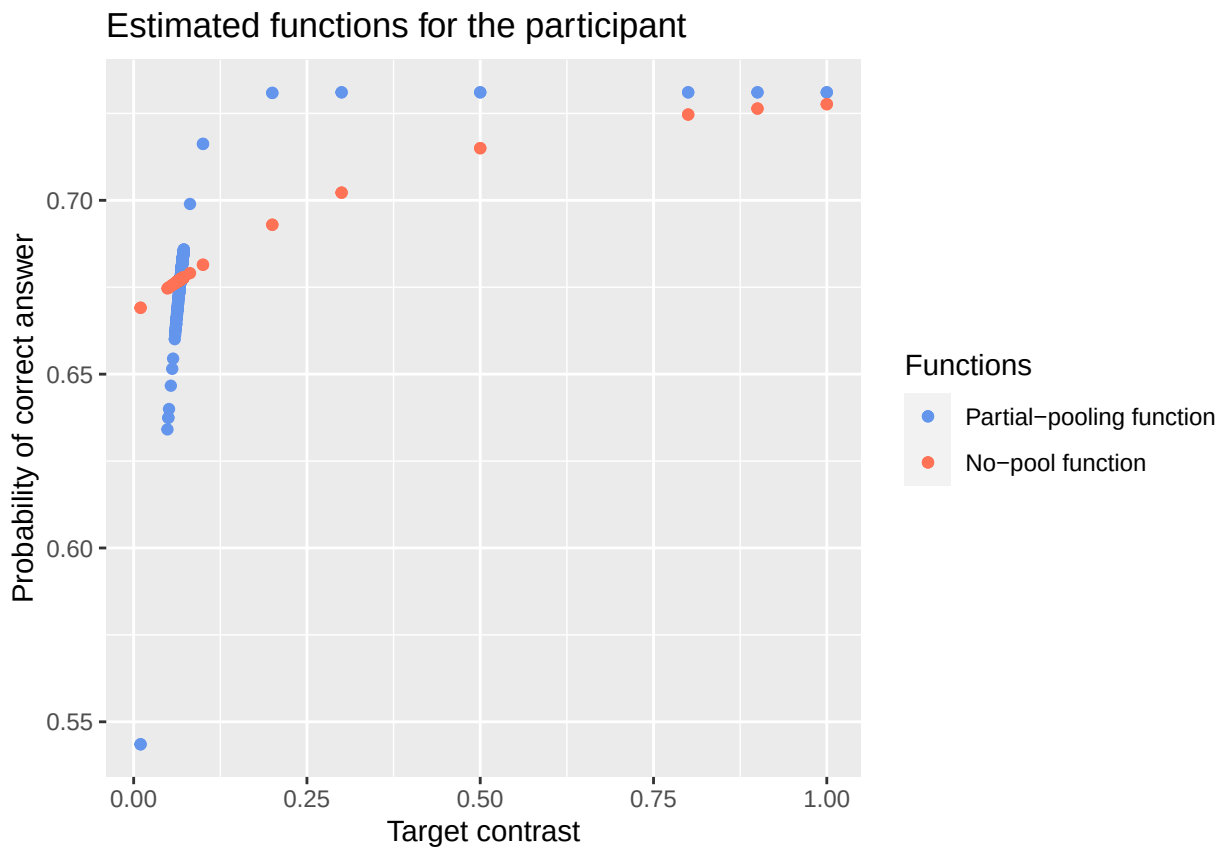
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```


Estimated functions for the participant
Probability of correct answer
0.70
0.65
0.60
0.55
0.50
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

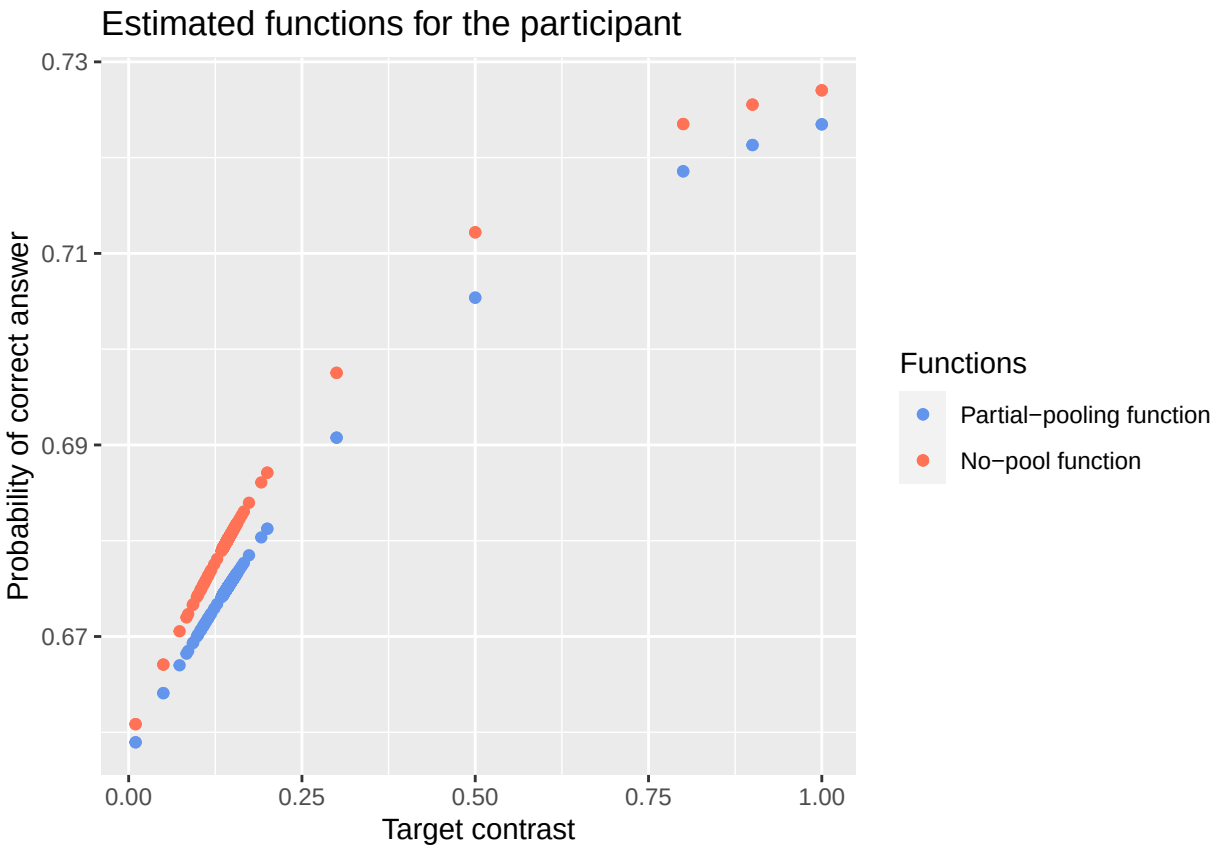



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

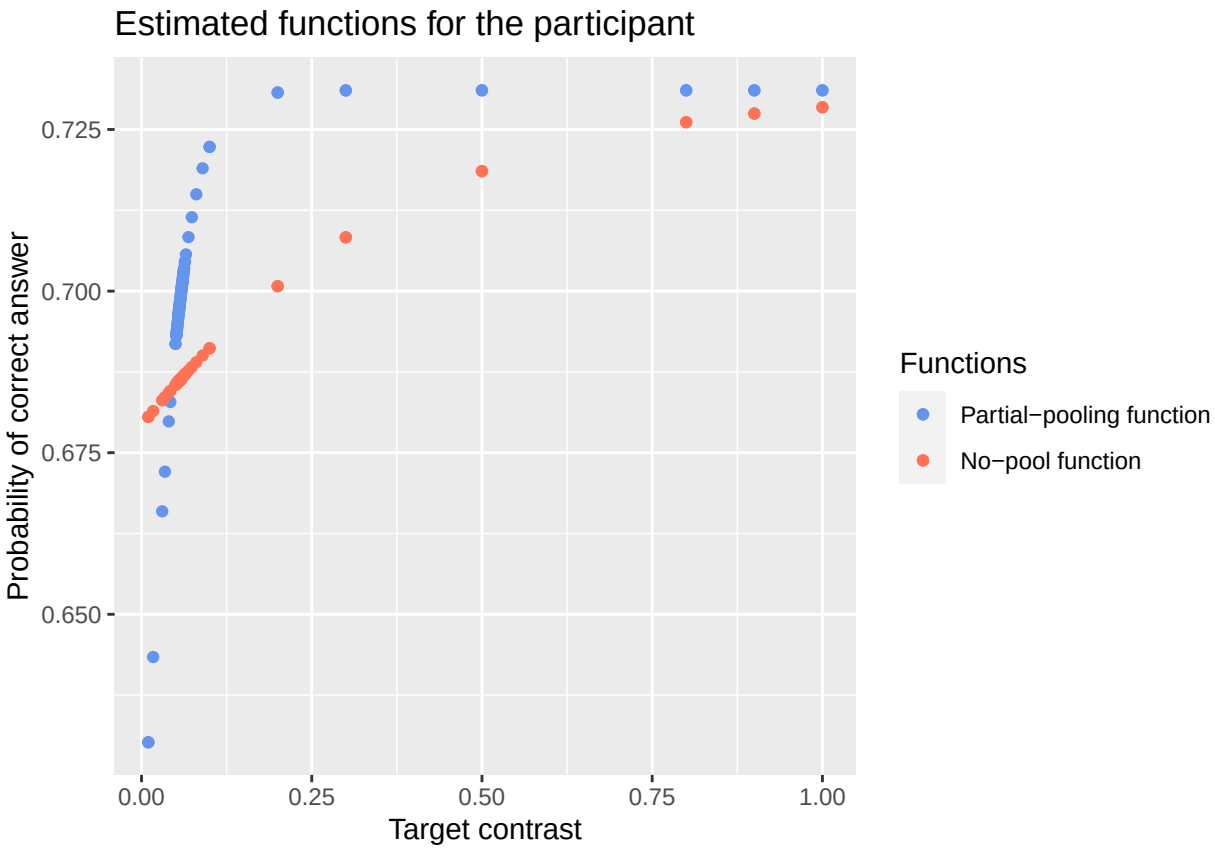

Estimated functions for the participant
Probability of correct answer
0.73
0.72
0.71
0.70
0.69
0.68
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

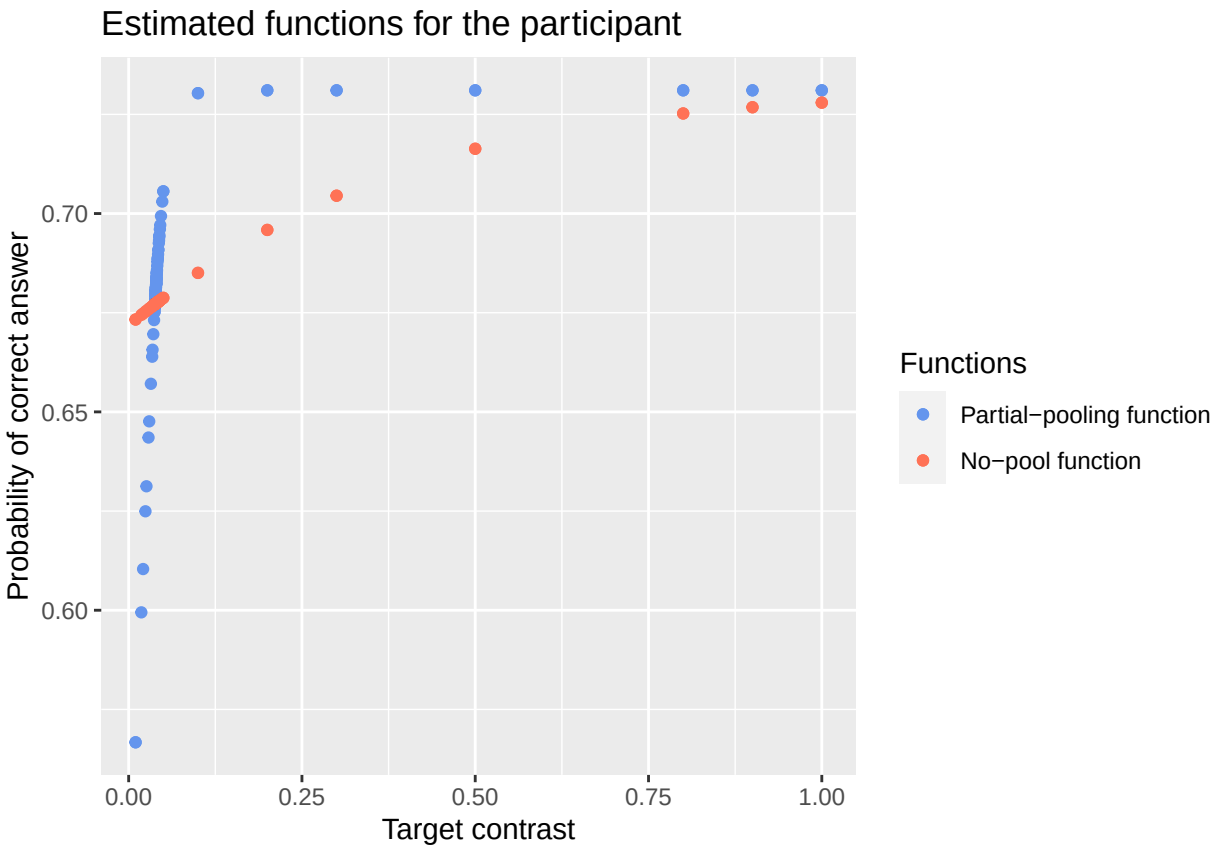


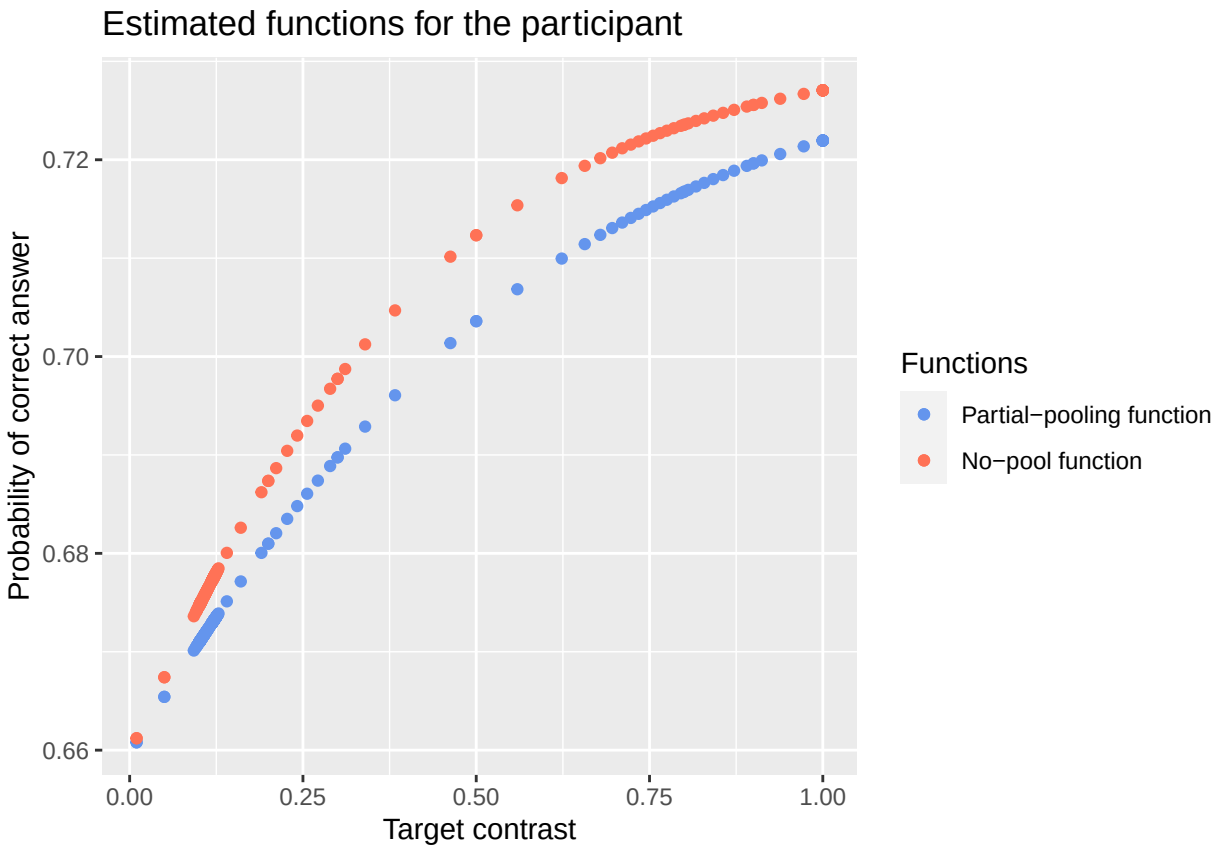

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

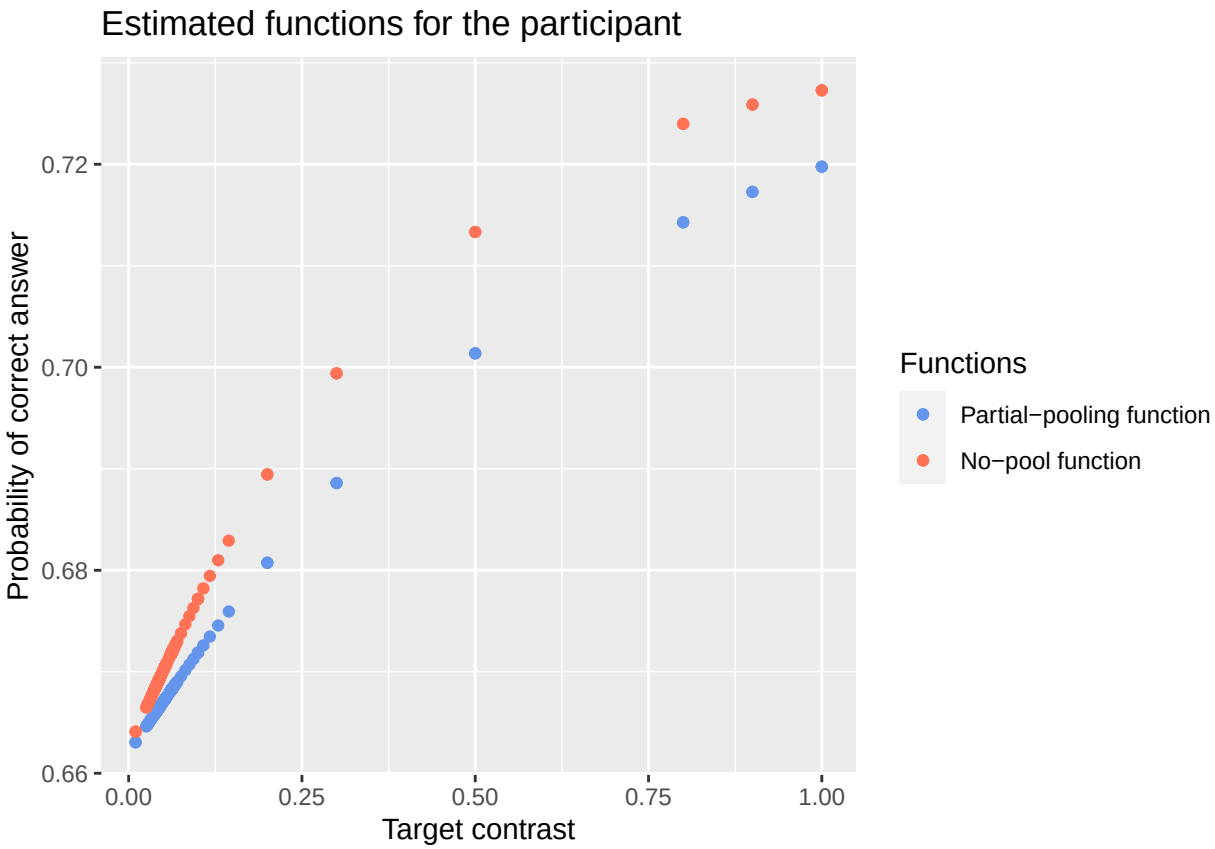


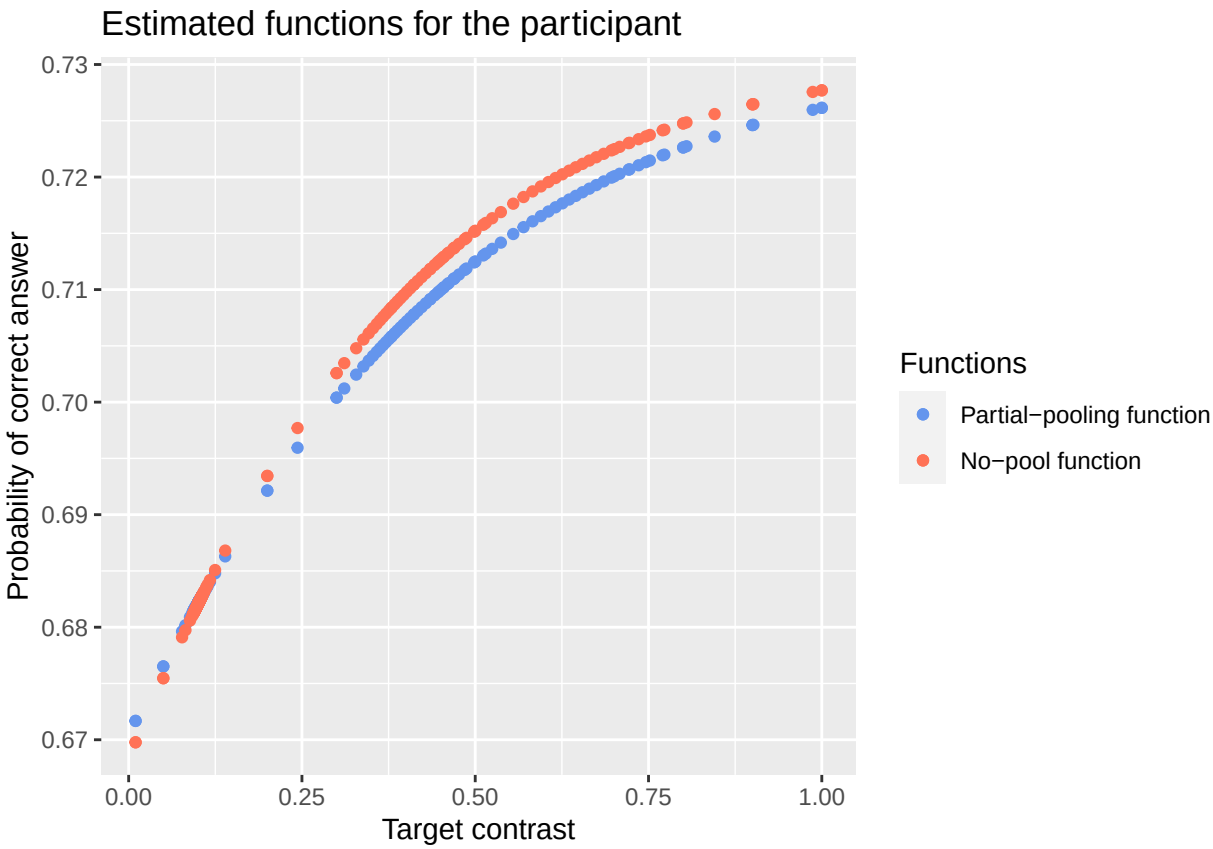

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

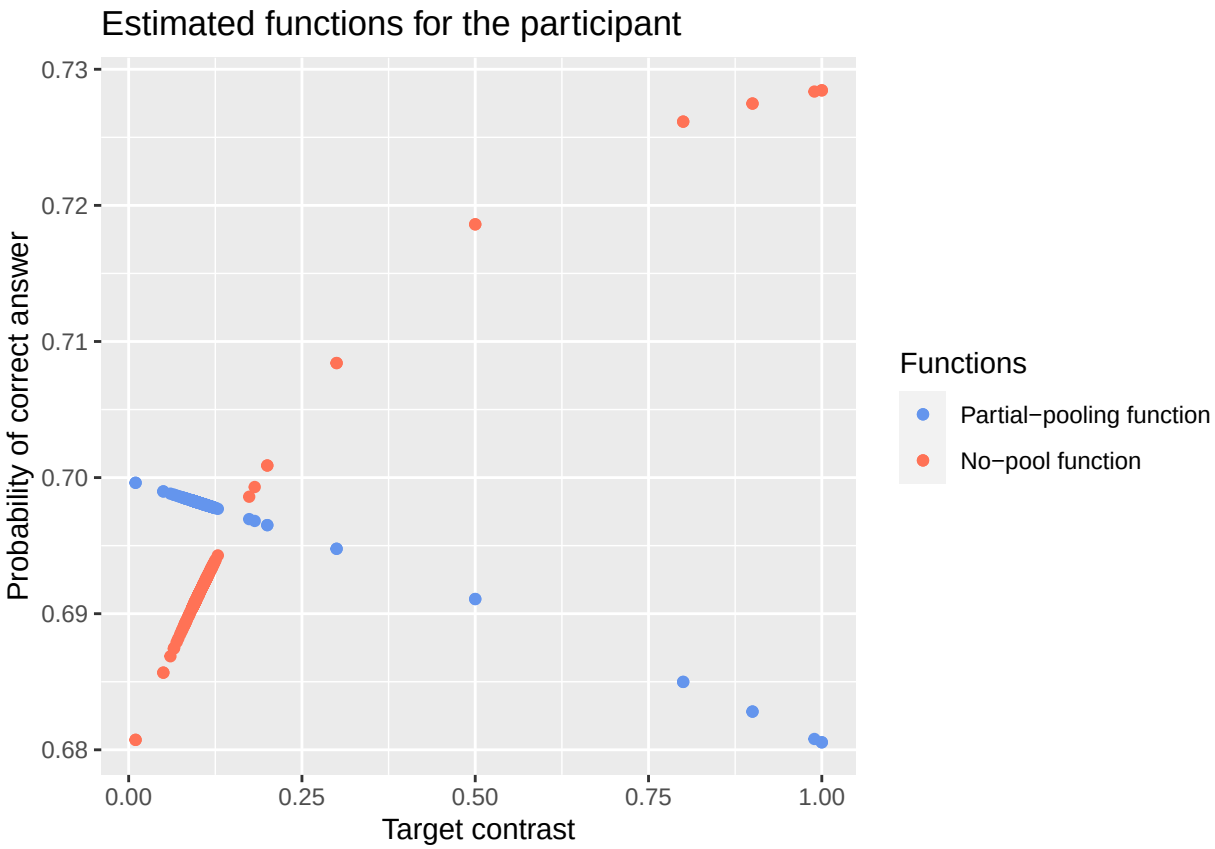

Estimated functions for the participant
Probability of correct answer
0.725
0.700
0.675
0.650
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

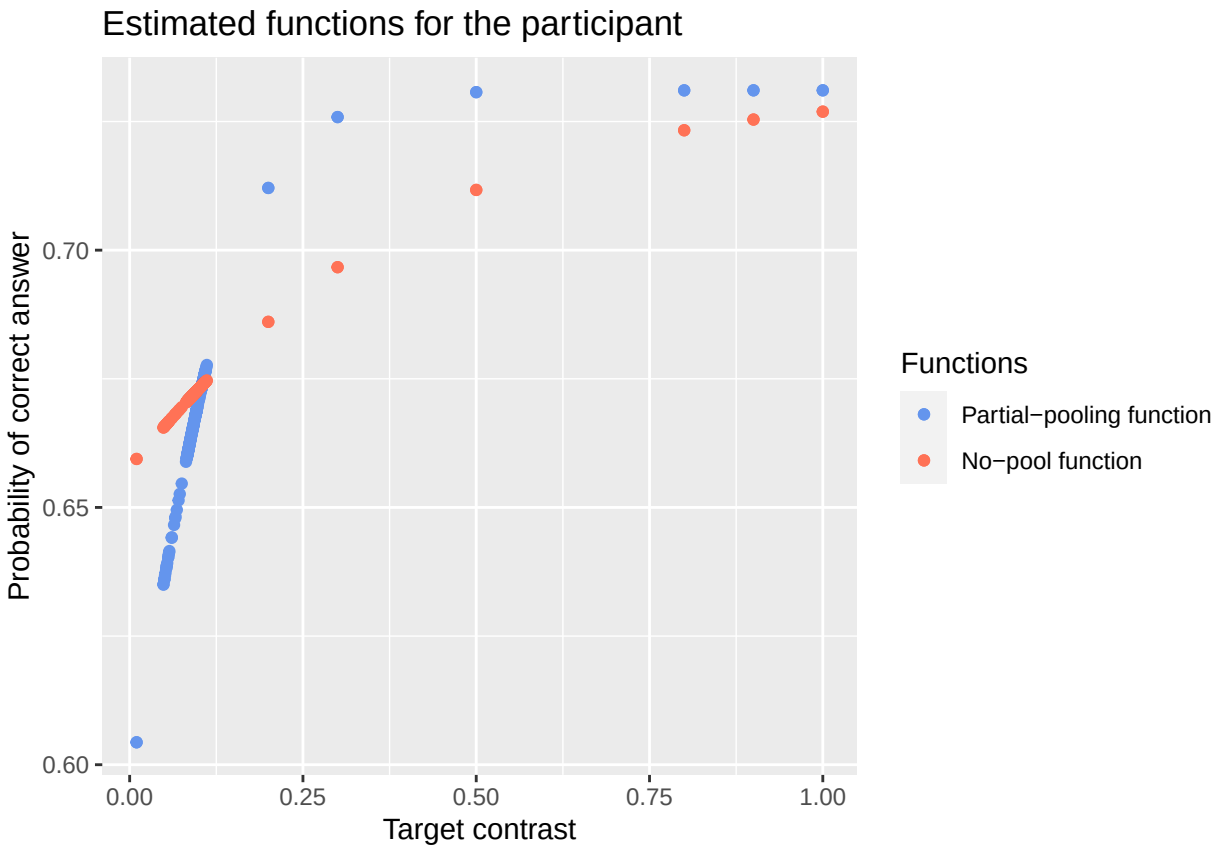

Estimated functions for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

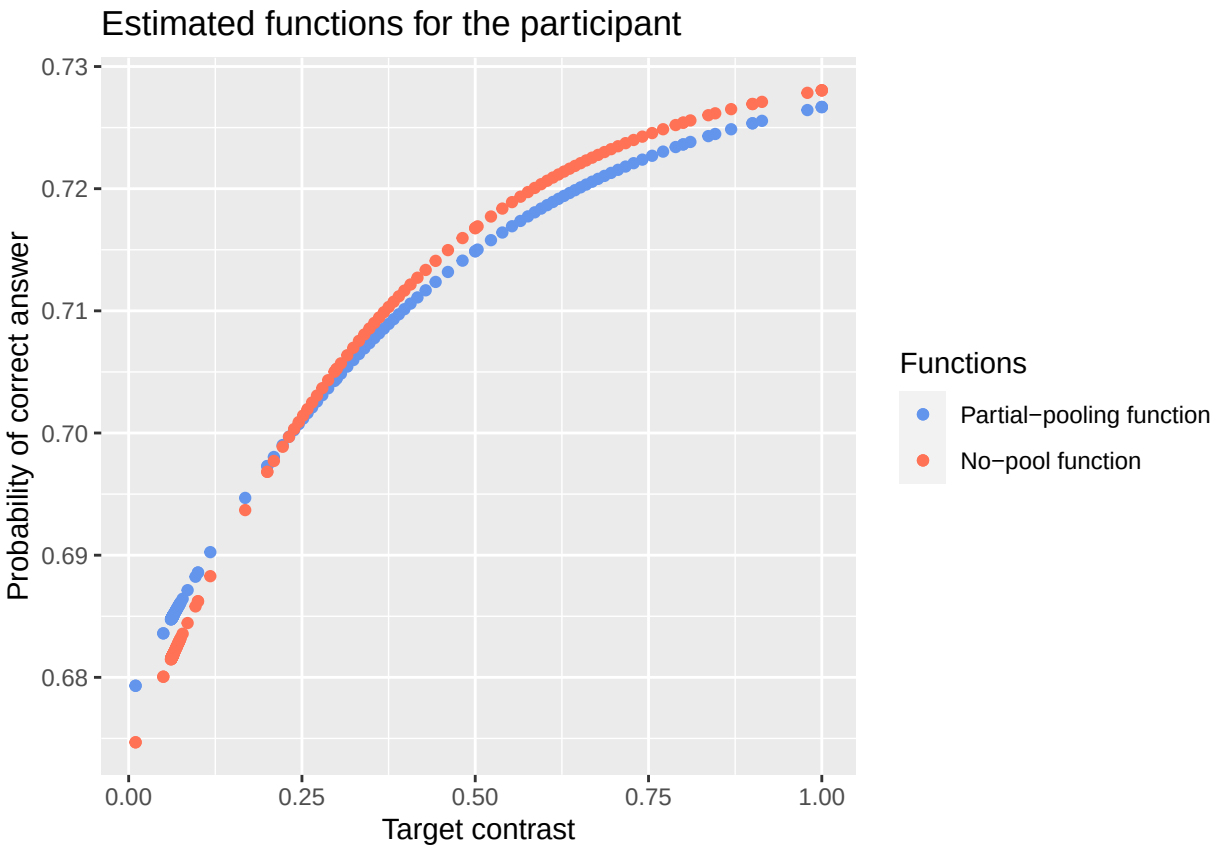

Estimated functions for the participant
Probability of correct answer
0.72
0.70
0.68
0.66
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

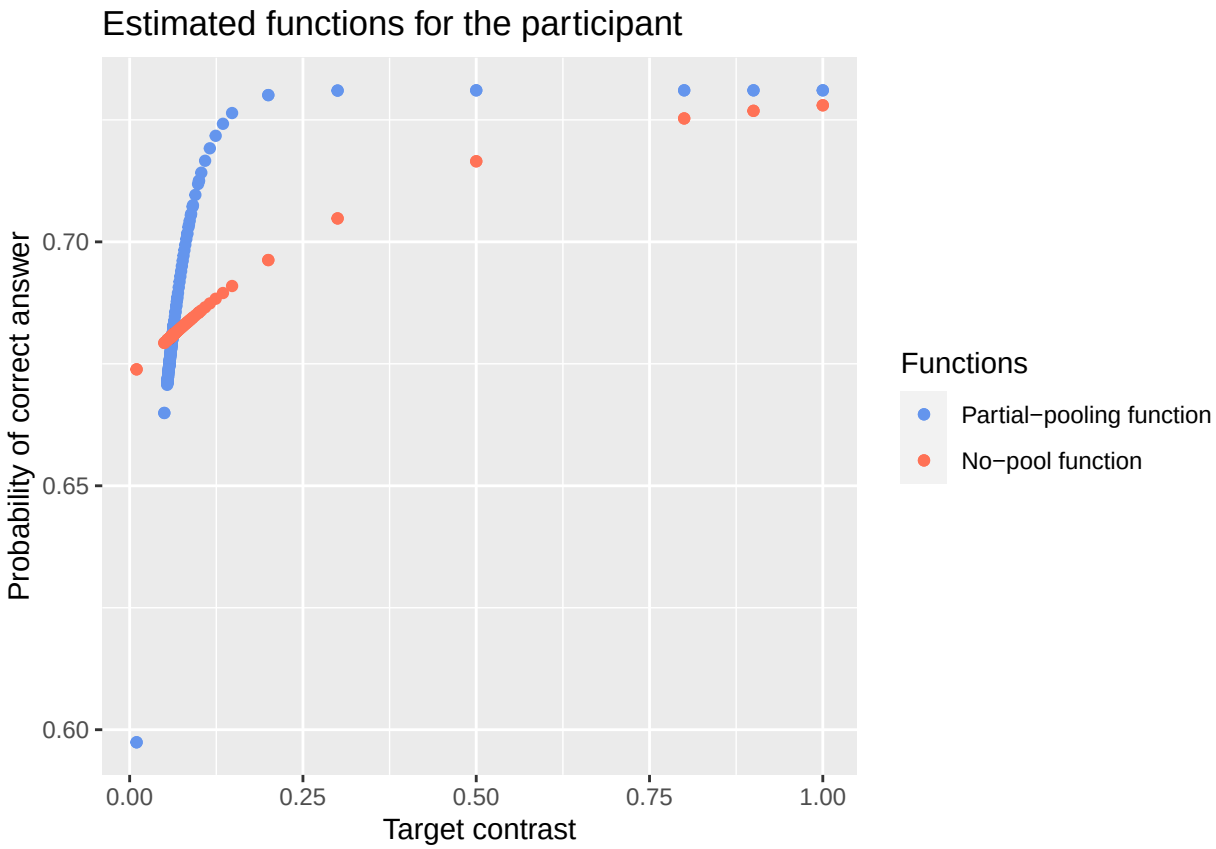

Estimated functions for the participant
Probability of correct answer
0.72
0.70
0.68
0.66
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

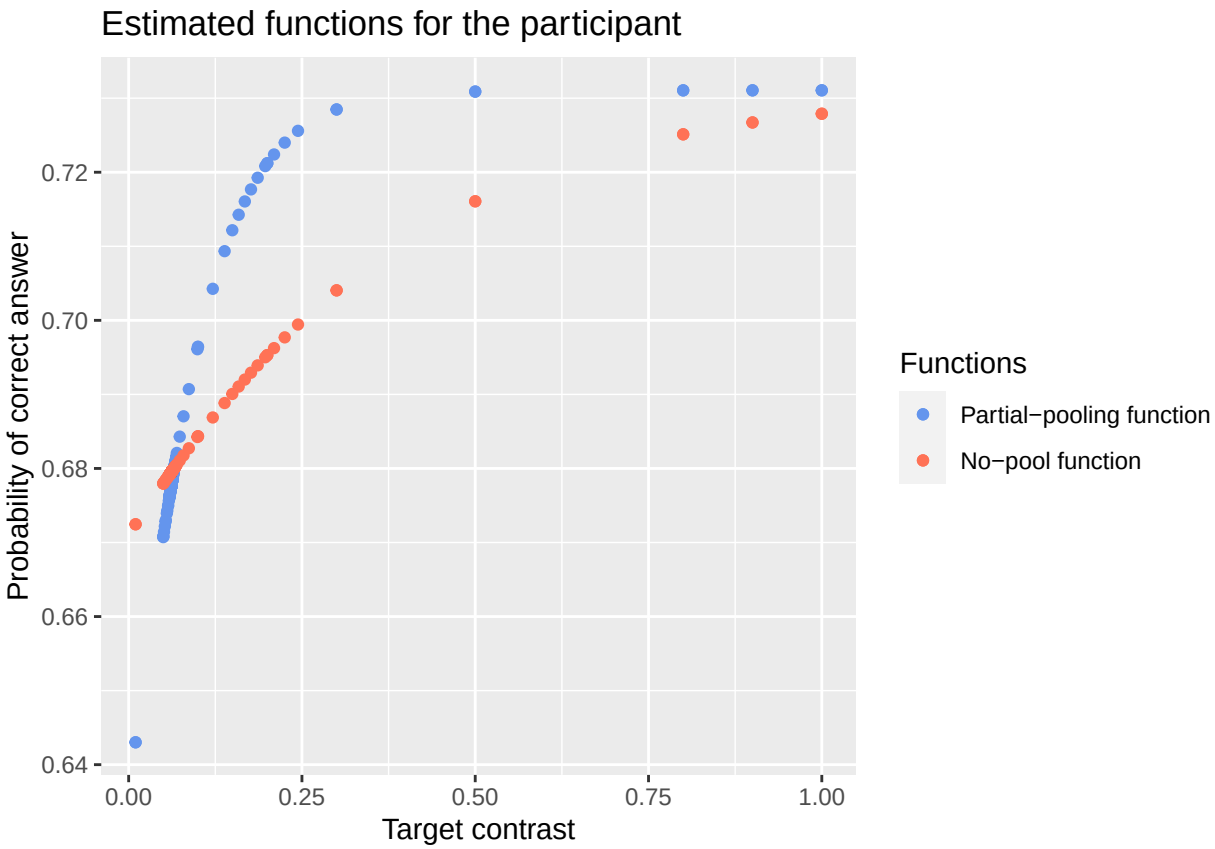

Estimated functions for the participant
Probability of correct answer
0.73
0.72
0.71
0.70
0.69
0.68
0.67
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

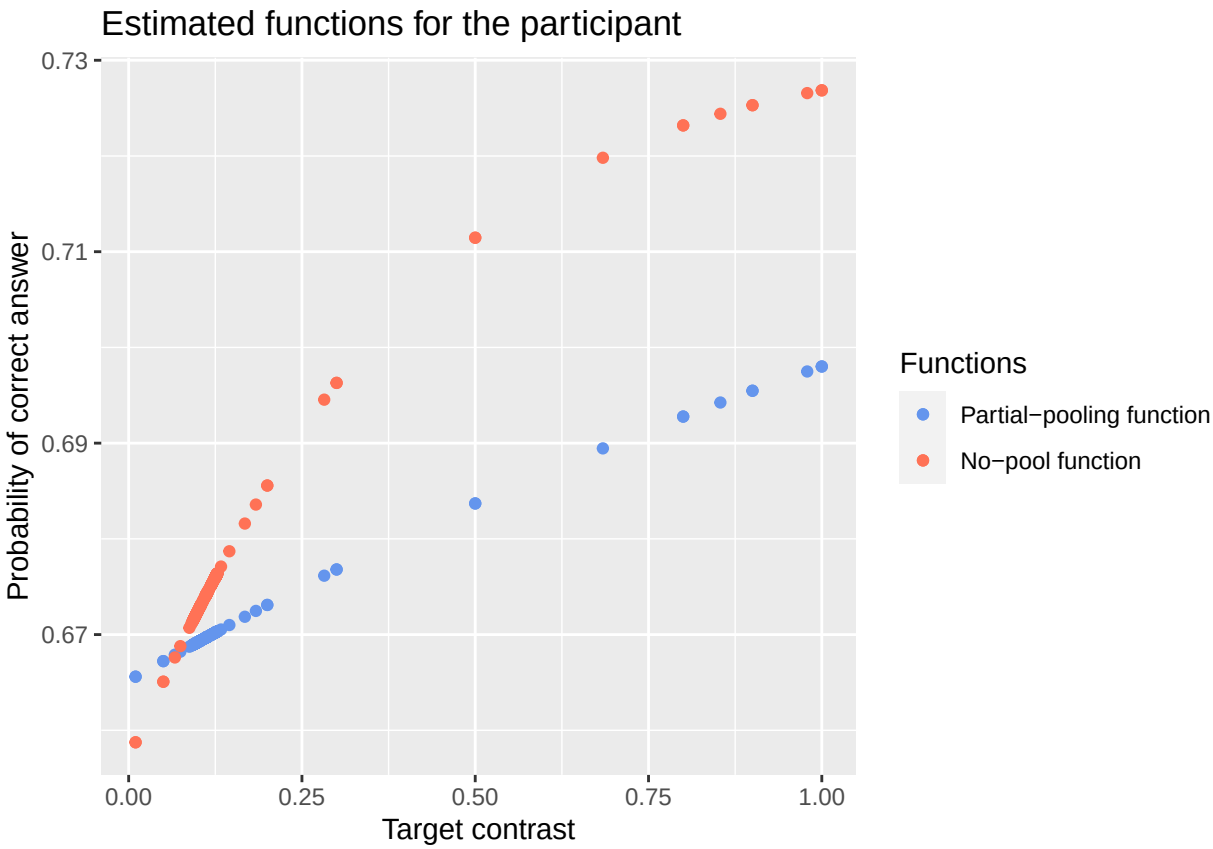

Estimated functions for the participant
Probability of correct answer
0.73
0.72
0.71
0.70
0.69
0.68
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

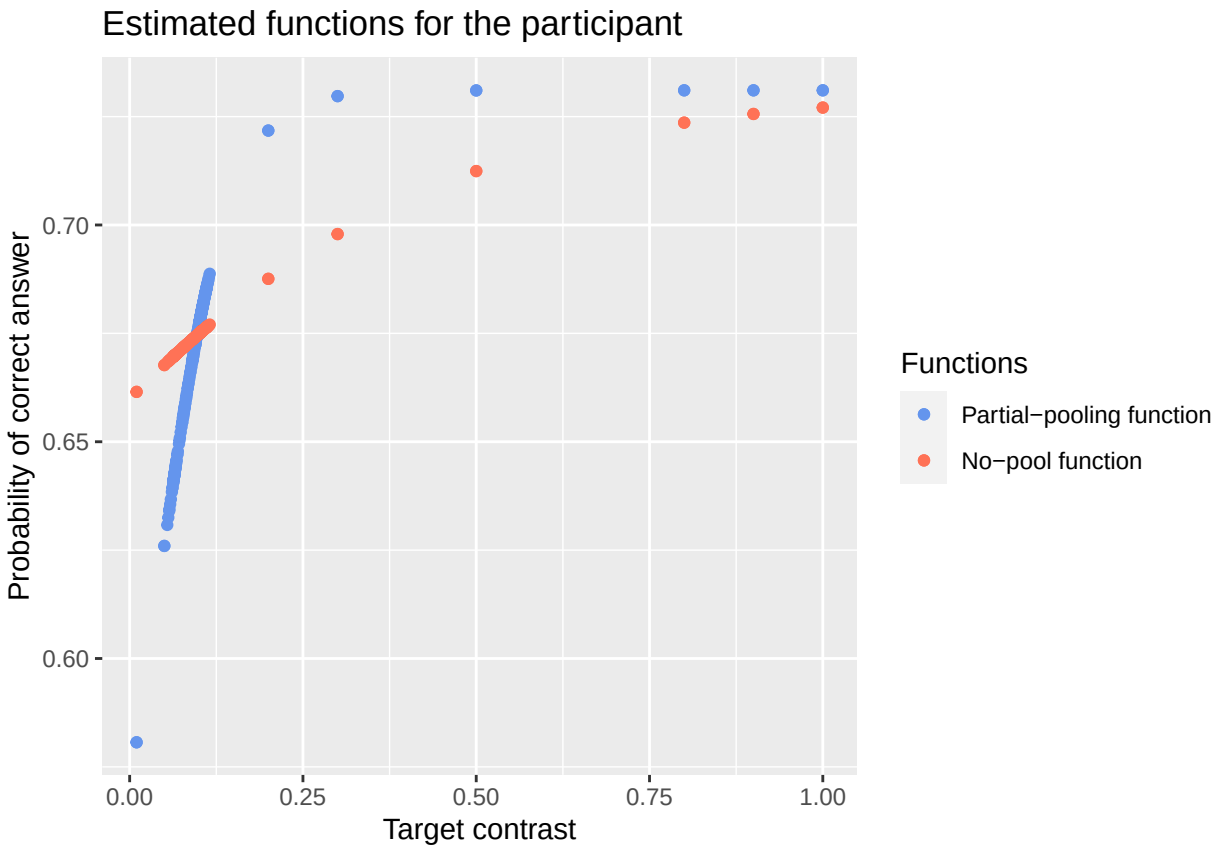

Estimated functions for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

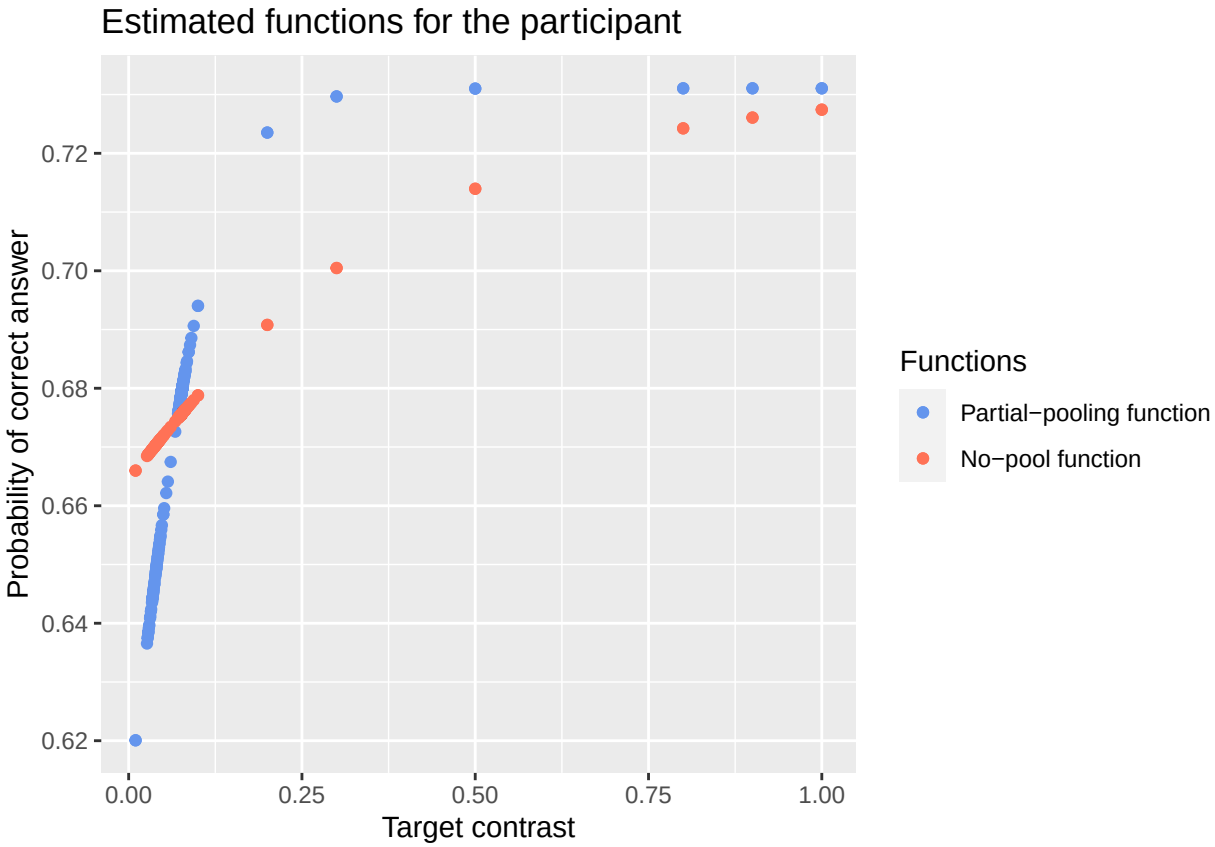

Estimated functions for the participant
Probability of correct answer
0.73
0.72
0.71
0.70
0.69
0.68
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function


Estimated functions for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

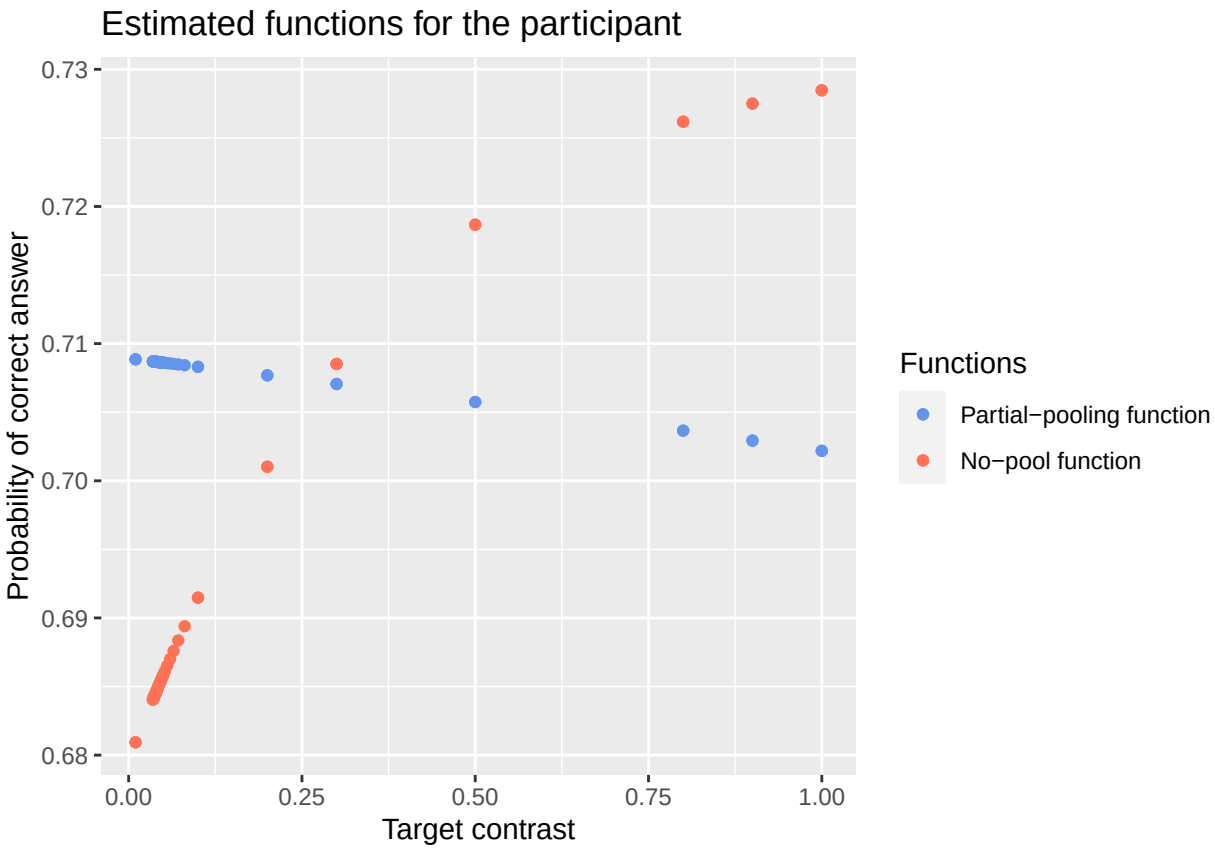

Estimated functions for the participant
Probability of correct answer
0.72
0.70
0.68
0.66
0.64
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function


Estimated functions for the participant
Probability of correct answer
0.73
0.71
0.69
0.67
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function

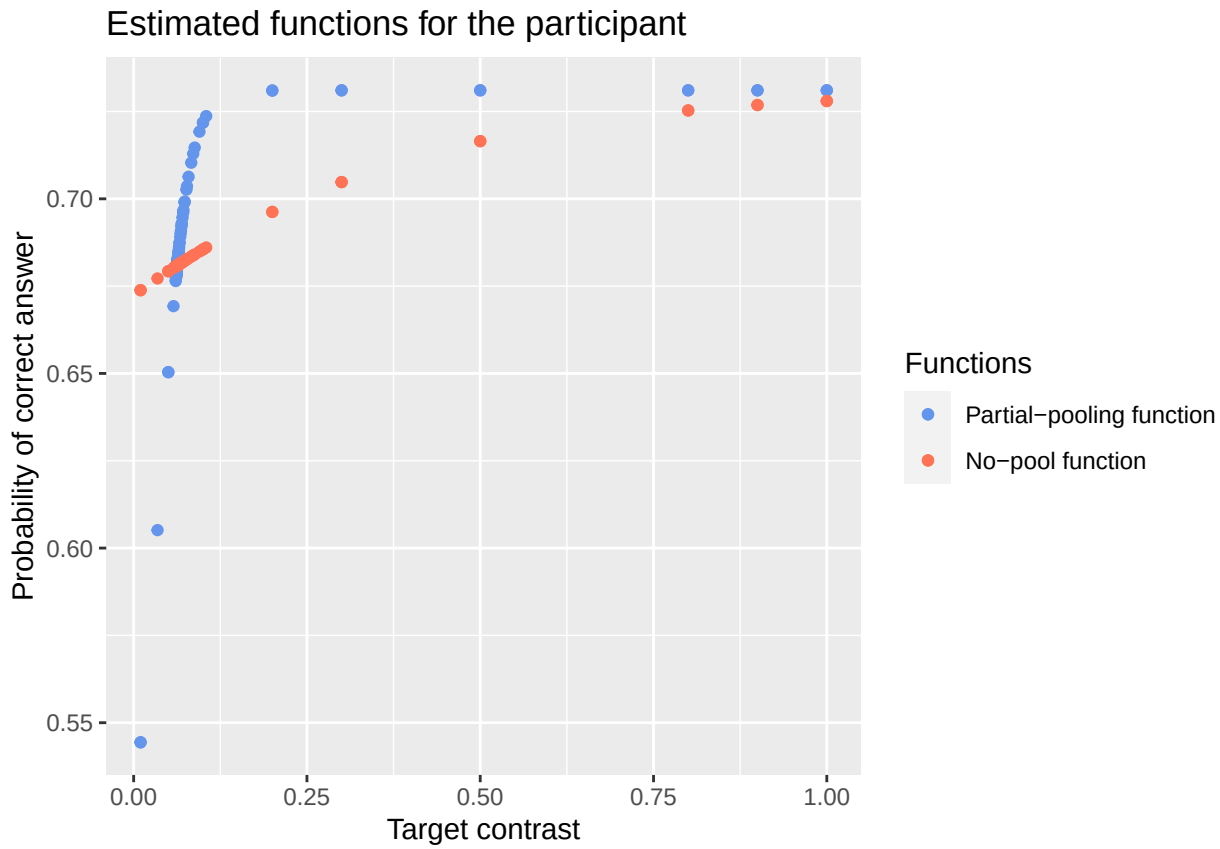

Estimated functions for the participant
Probability of correct answer
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function



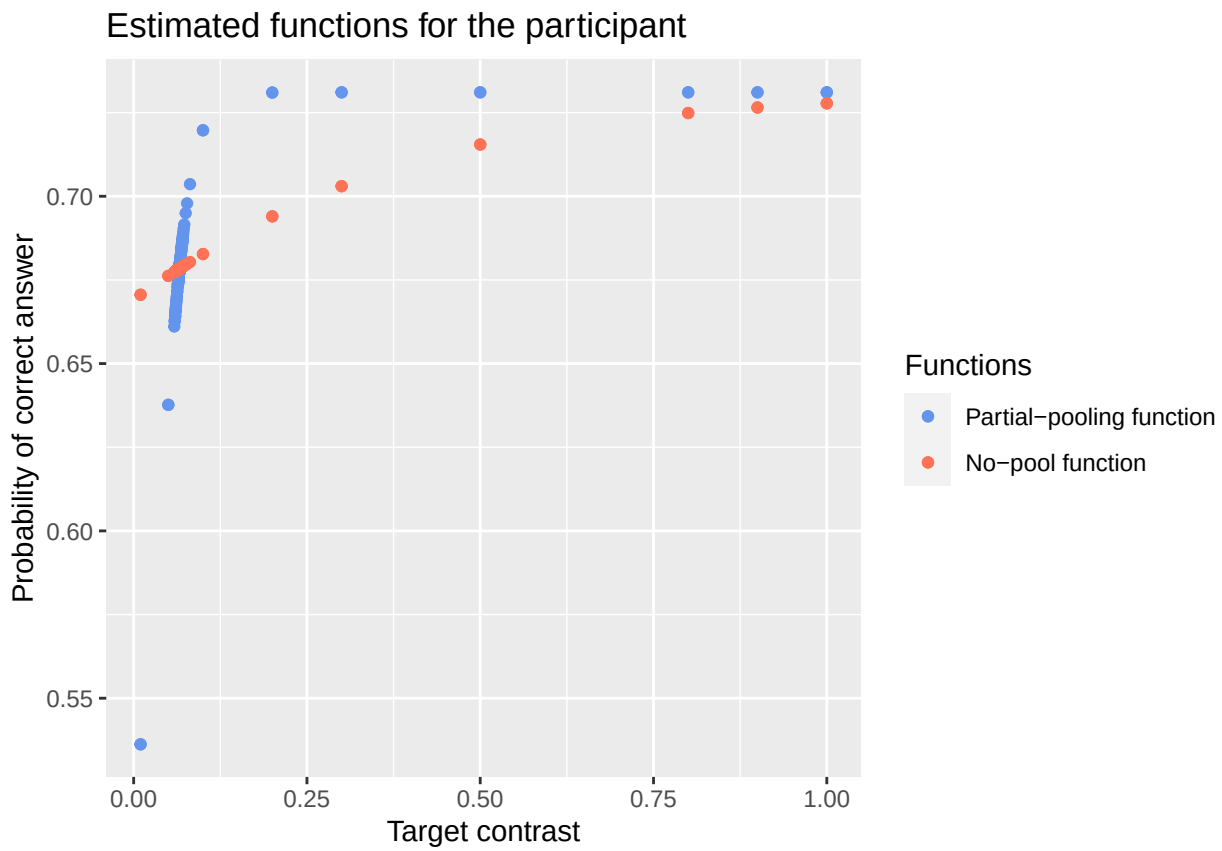

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



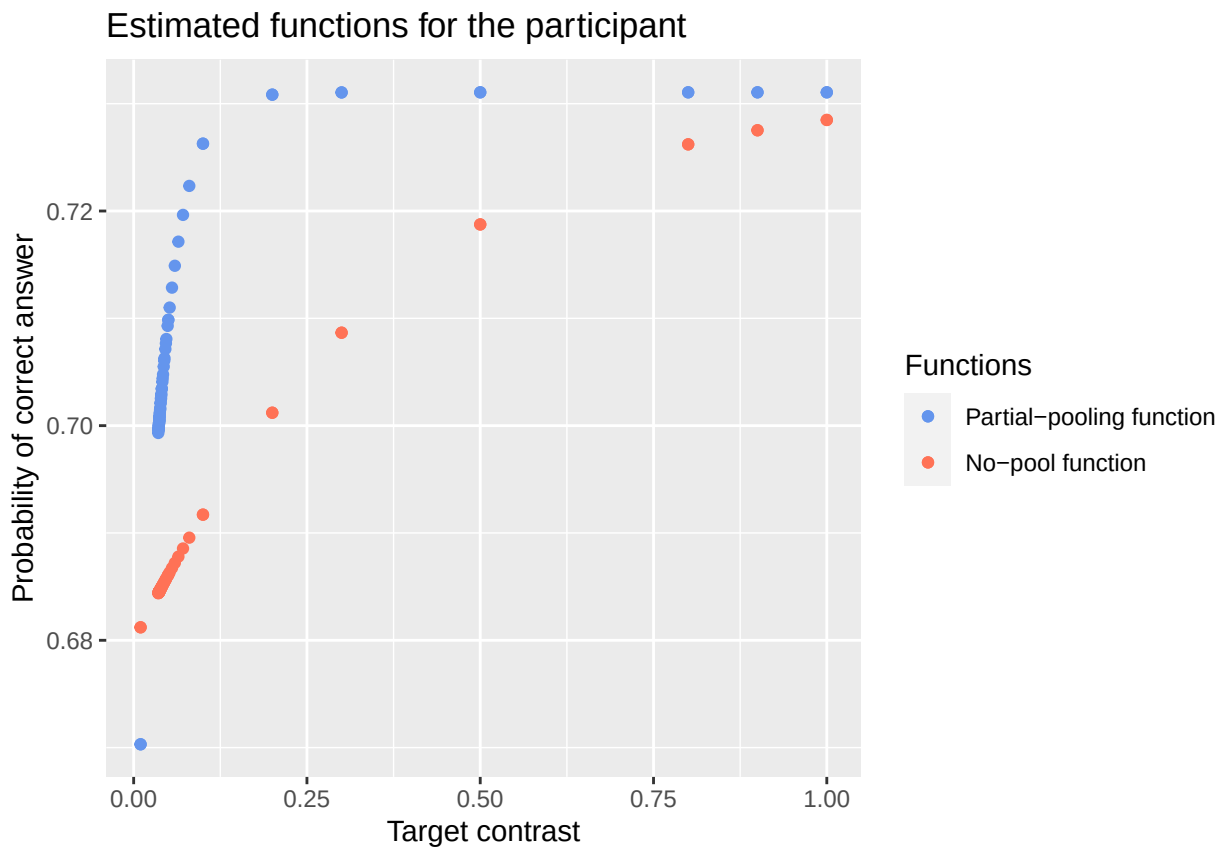

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



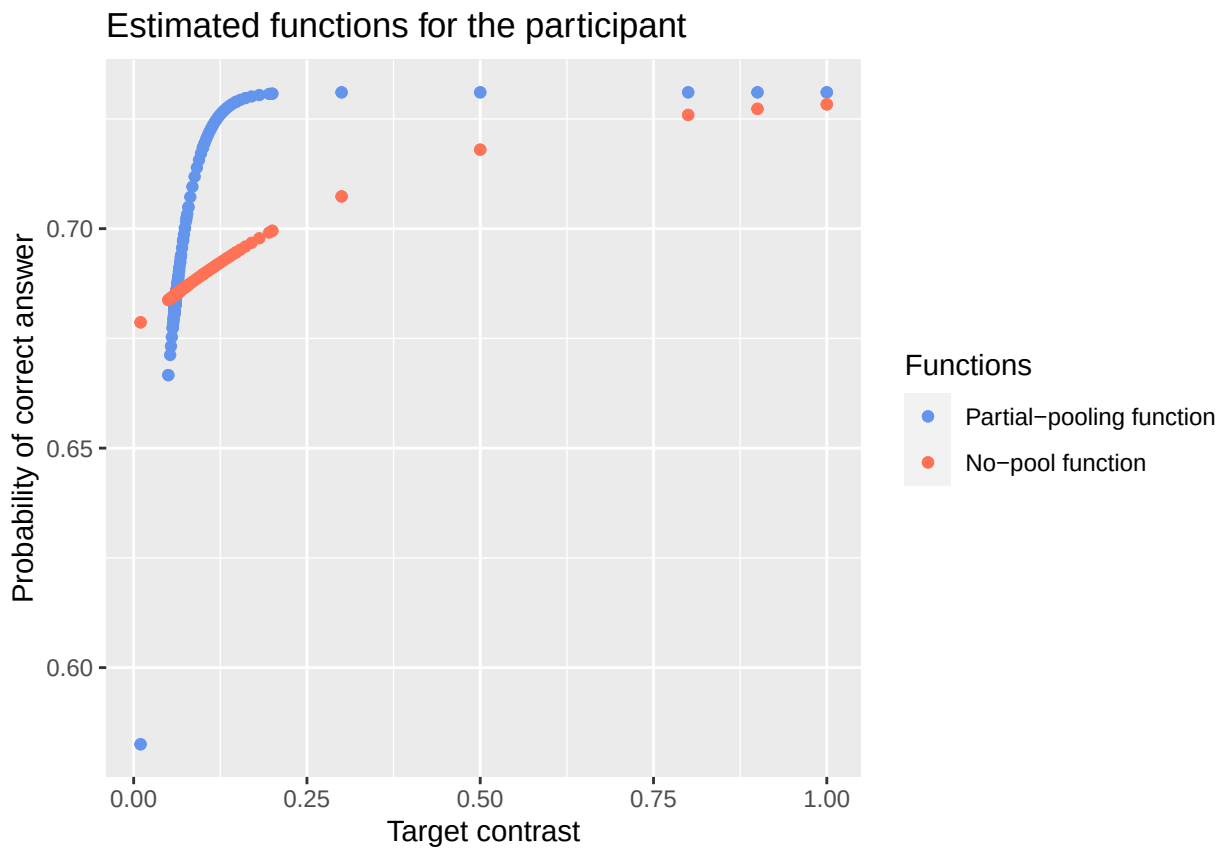

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



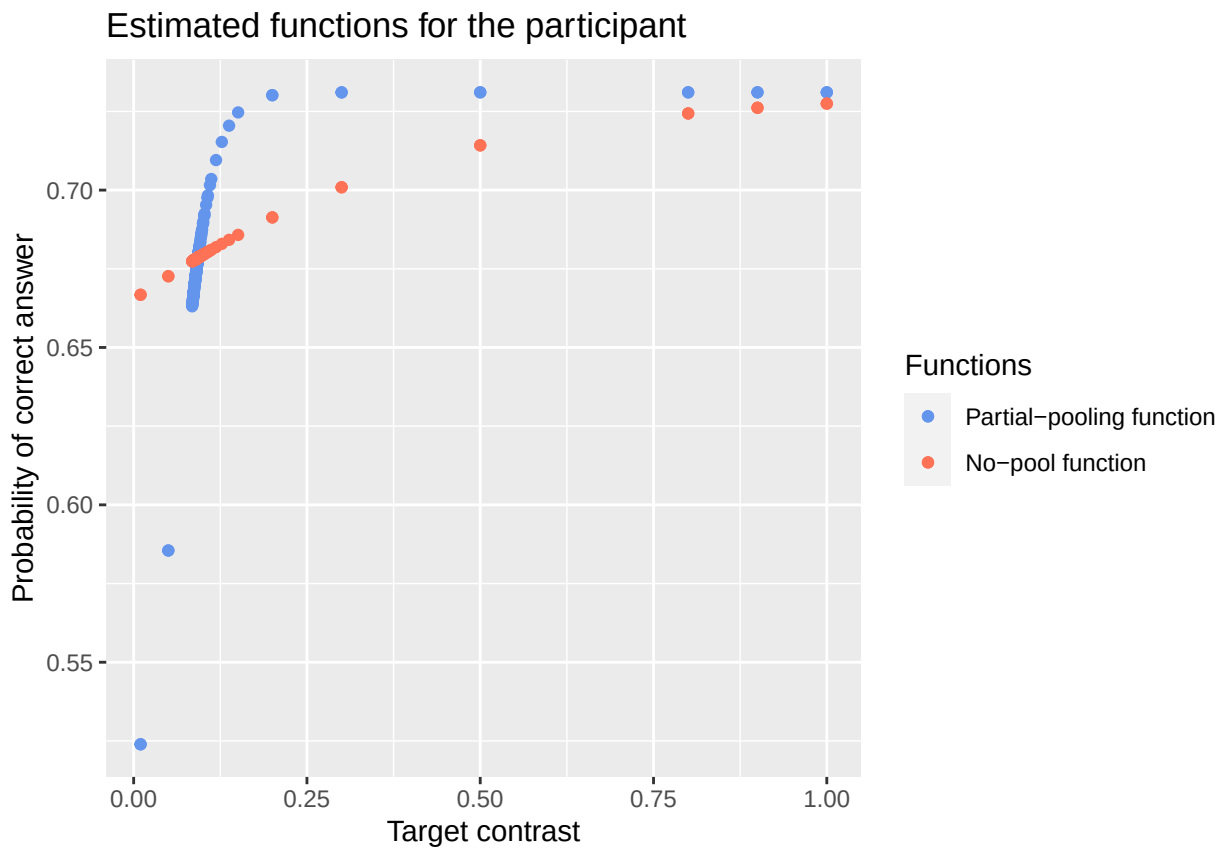

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



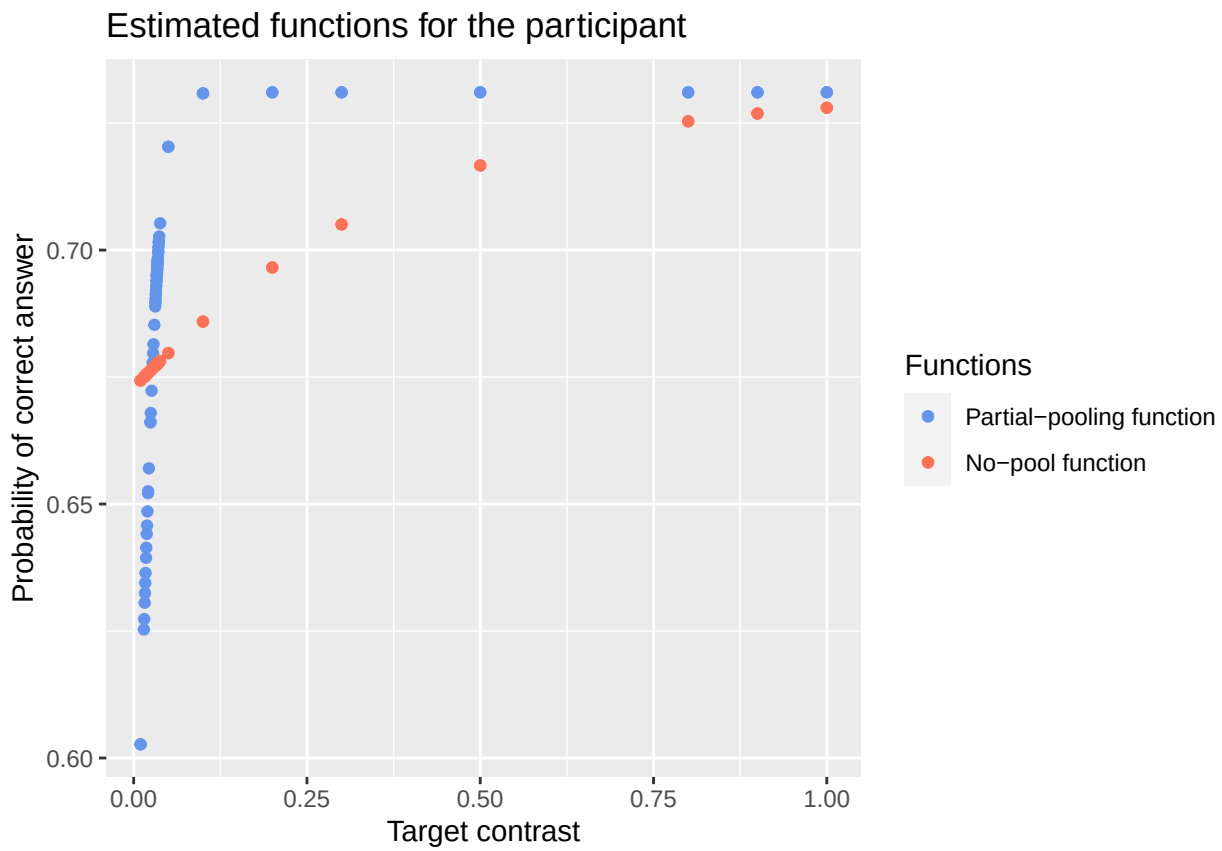

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



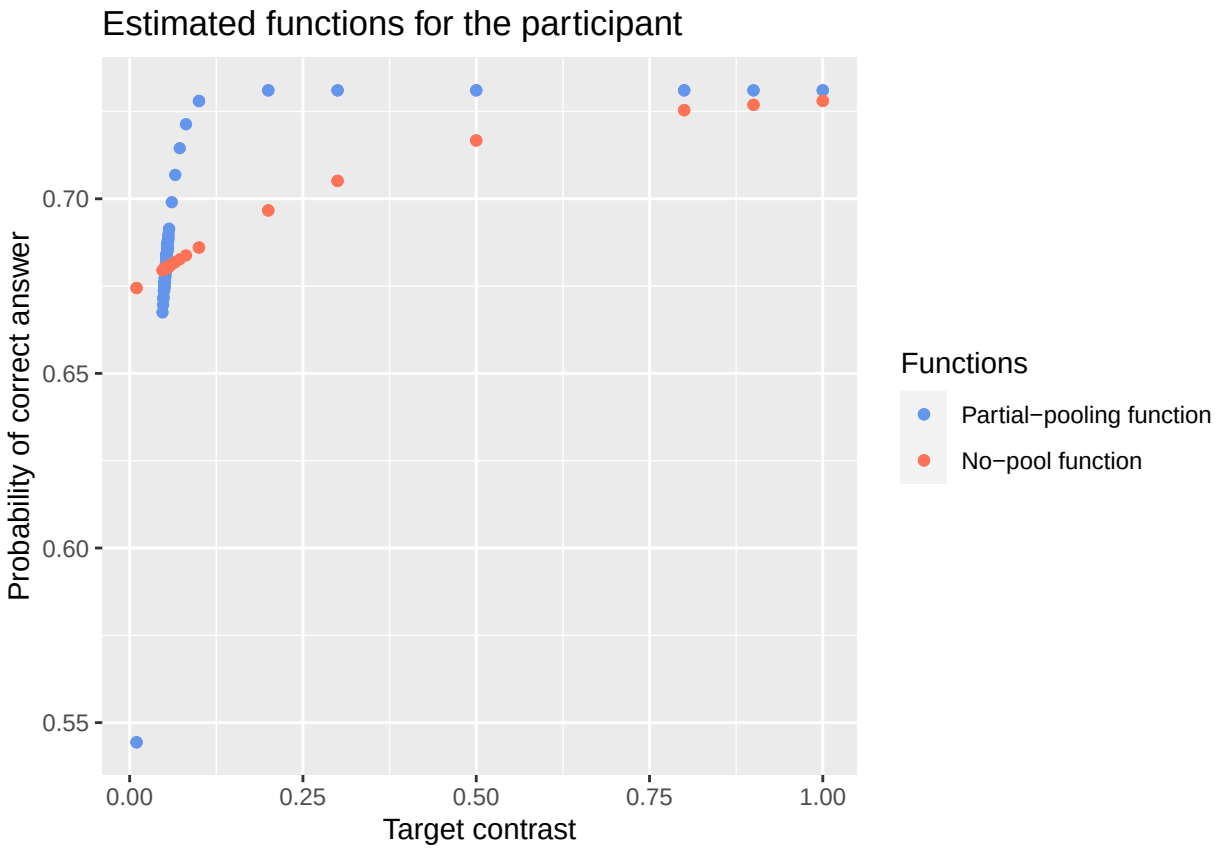

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

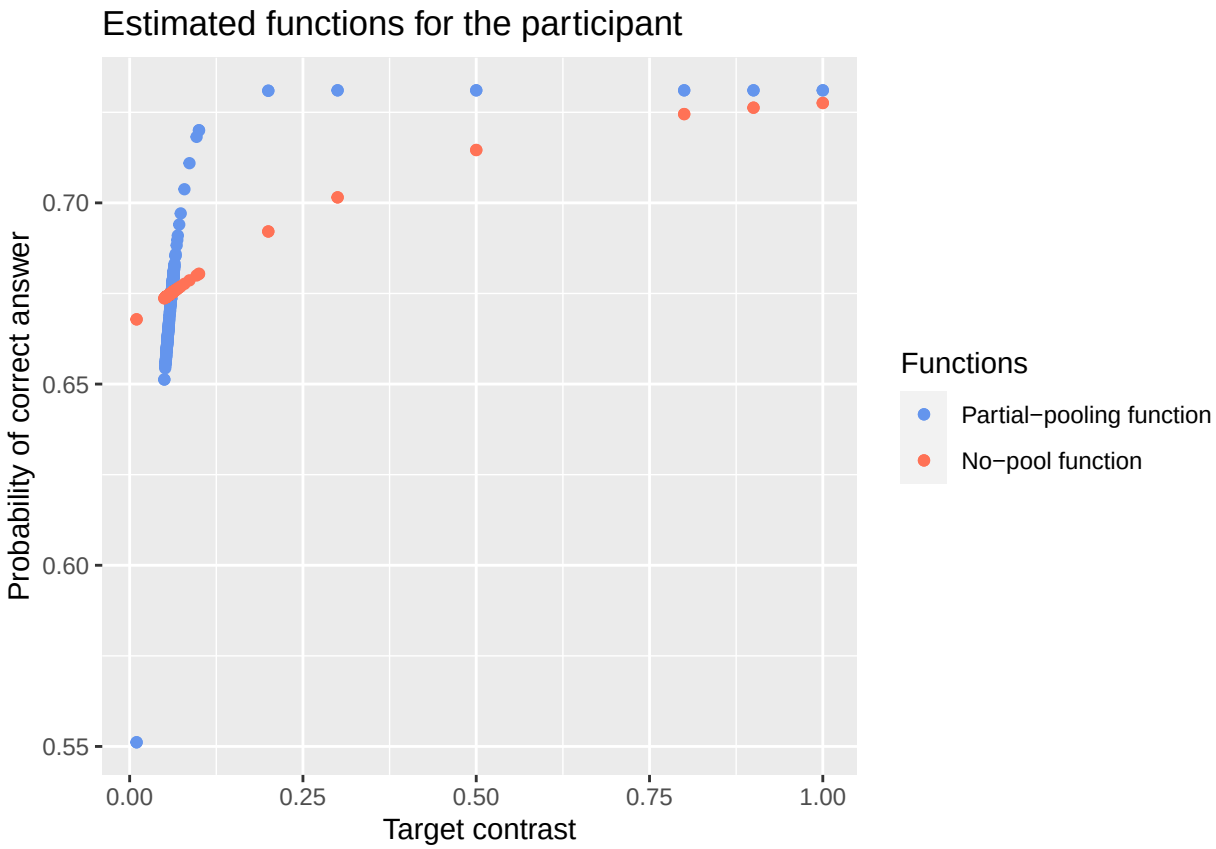


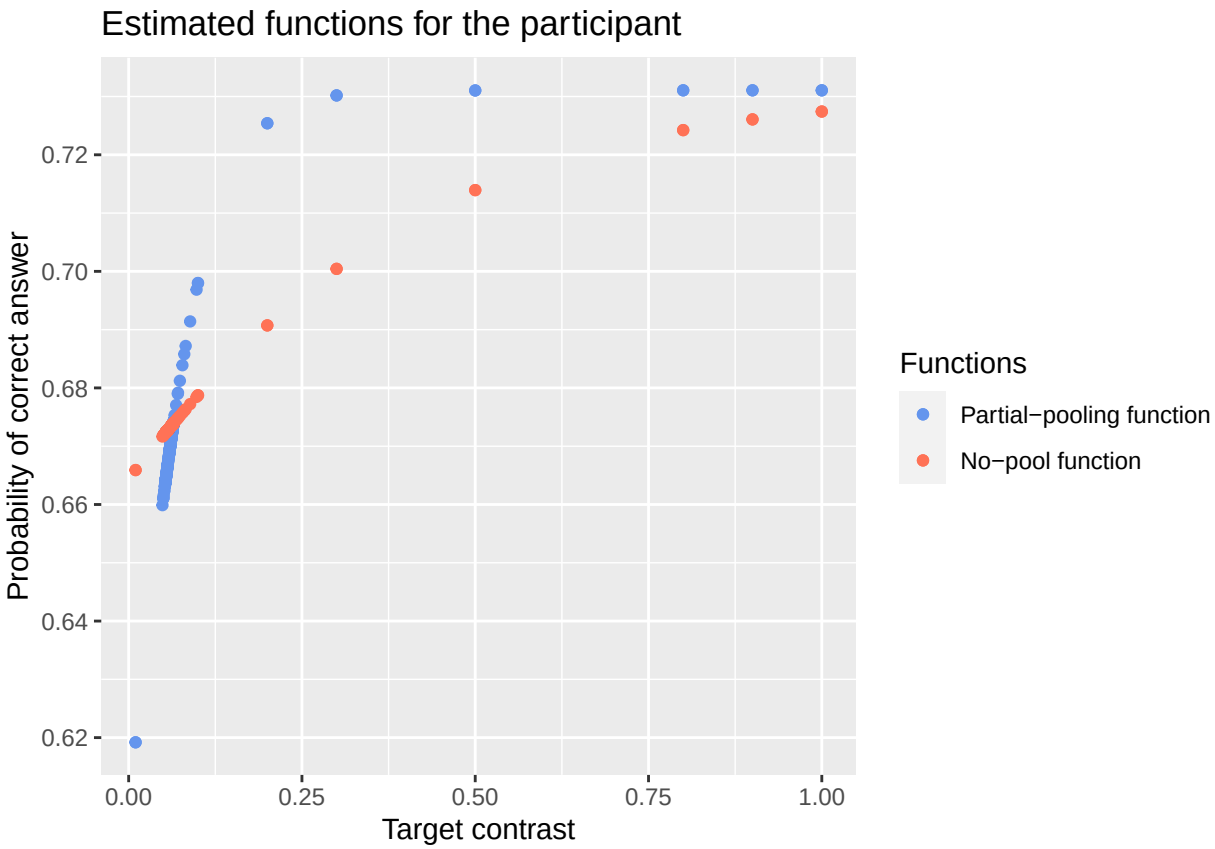

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

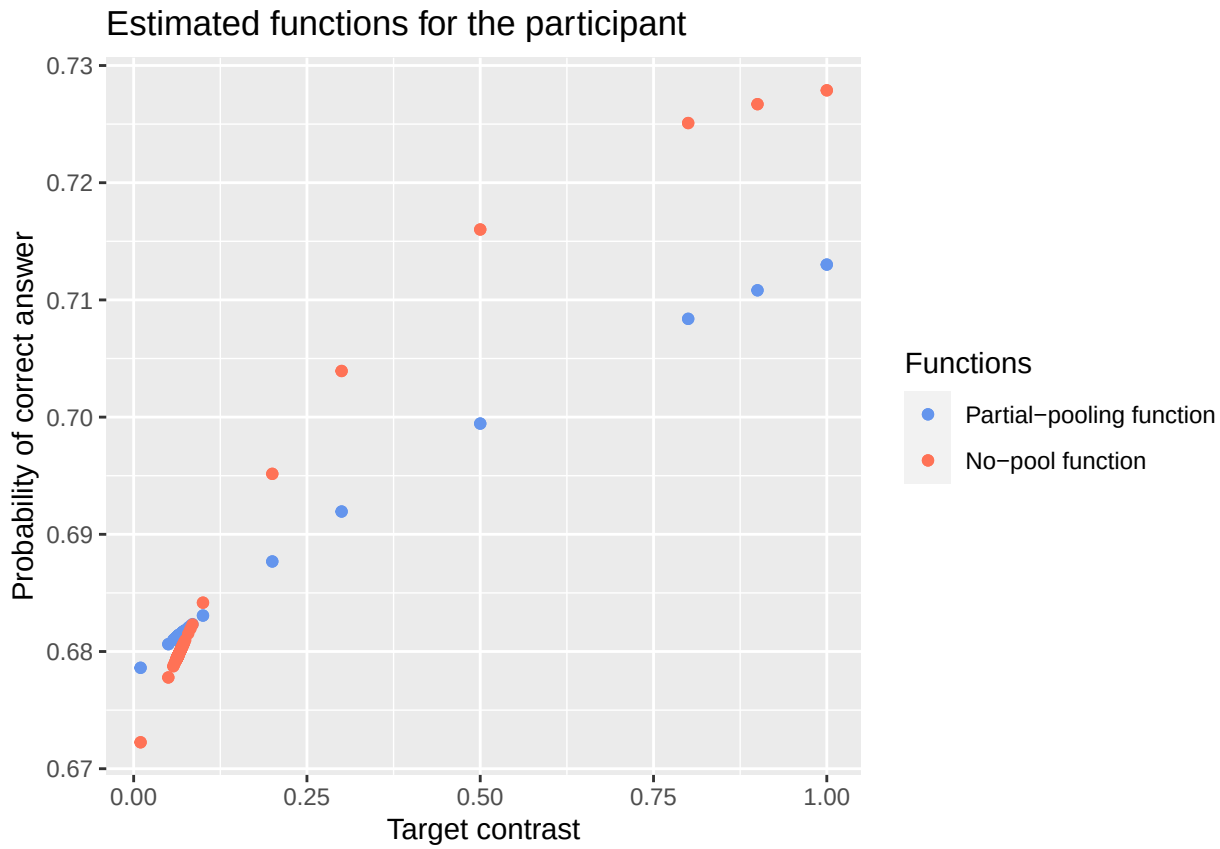



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```


Estimated functions for the participant
Probability of correct answer
0.70
0.65
0.60
0.55
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function


Estimated functions for the participant
Probability of correct answer
0.70
0.65
0.60
0.55
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function


Estimated functions for the participant
Probability of correct answer
0.72
0.70
0.68
0.66
0.64
0.62
0.00
0.25
0.50
0.75
1.00
Target contrast
Functions
Partial−pooling function
No−pool function



v. in your own words, describe how the partial pooling model allows for a better fit for each subject When the data is analyzed by taking each subject separately, only a small portion of the data is used (none of the other participants' data is included, despite them having taken part in the same experiment under the same conditions). When using a hierarchical model, all of the data is used in order to calculate the estimated function, with the different subjects simply recognized as having *slightly* different curves that nevertheless resemble the overall curve, thereby allowing more general tendencies to be represented. Essentially, the hierarchical model recognizes both the differences and the similarities in the sample of participants.

Exercise 2

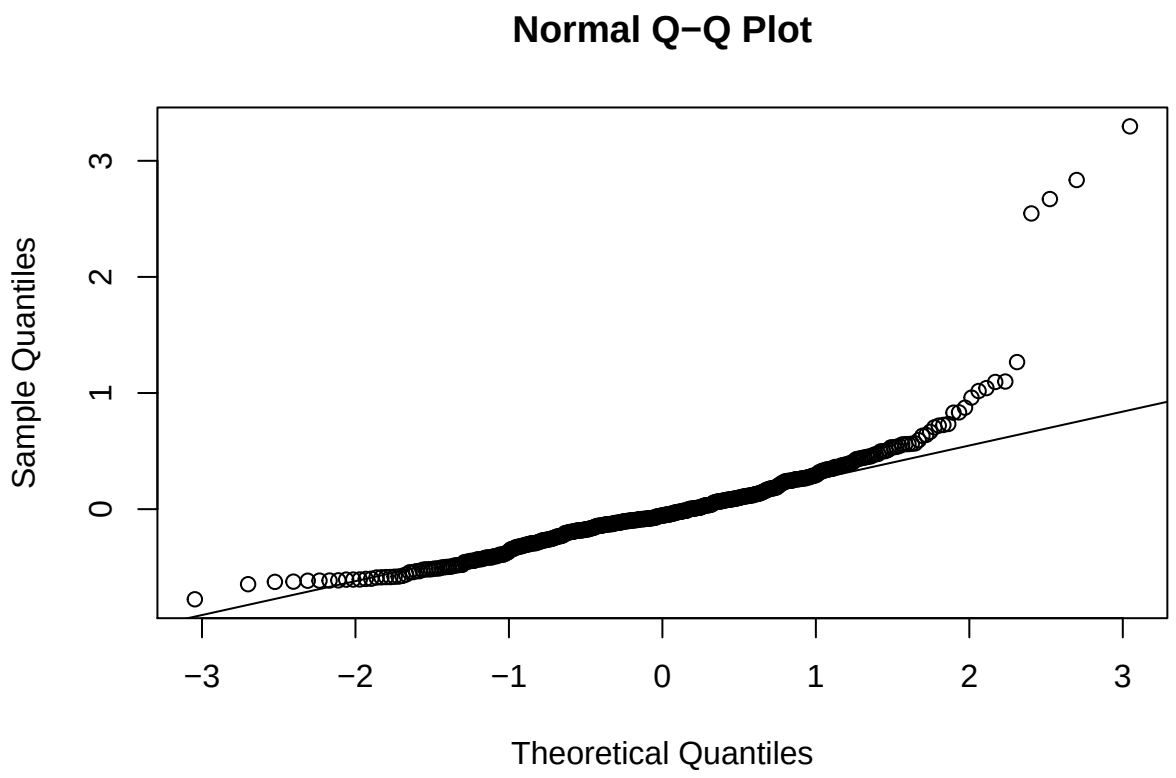
Now we **only** look at the *experiment* trials (*trial.type*)

```
exp <- df %>%
  filter(trial.type == "experiment")
```

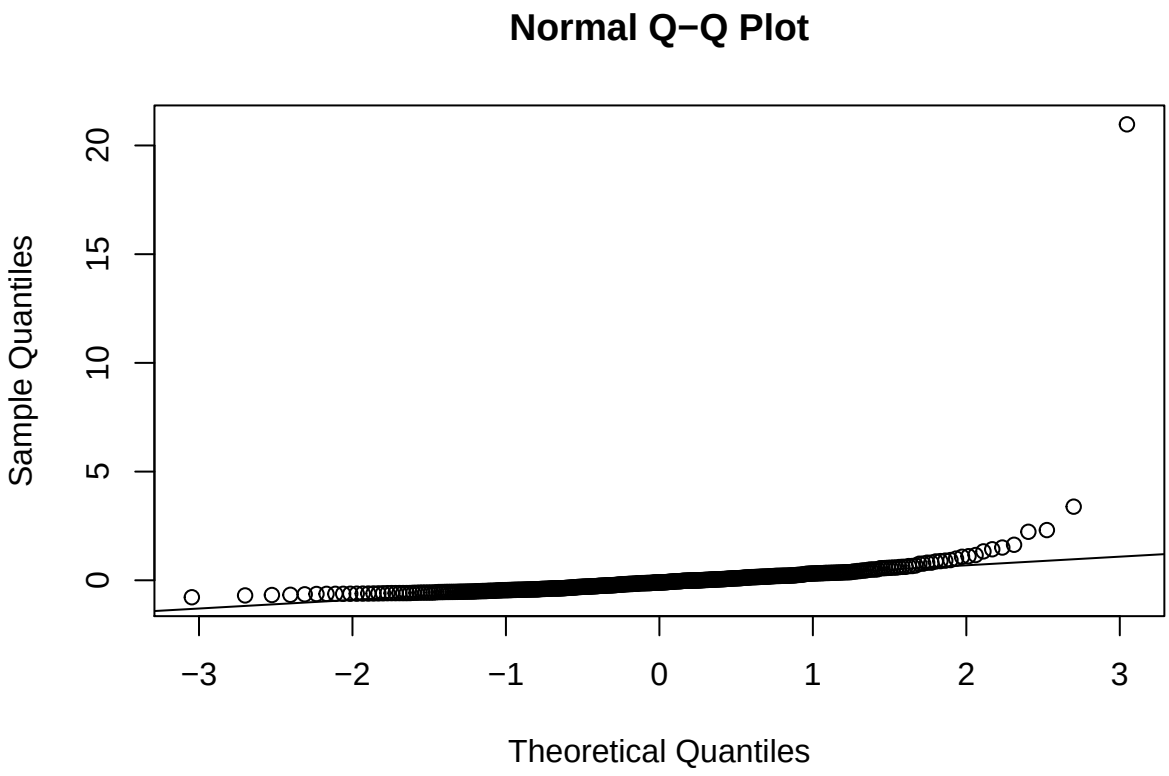
1) Pick four subjects and plot their Quantile-Quantile (Q-Q) plots for the residuals of their objective response times (*rt.obj*) based on a model where only intercept is modeled

```
# Isolate subjects
sub001 <- exp %>% filter(subject == "1")
sub005 <- exp %>% filter(subject == "5")
sub016 <- exp %>% filter(subject == "16")
sub020 <- exp %>% filter(subject == "20")

# Model only the intercept of their objective response times (basically the mean)
int001 <- lm(rt.obj ~ 1, data = sub001)
int005 <- lm(rt.obj ~ 1, data = sub005)
int016 <- lm(rt.obj ~ 1, data = sub016)
```

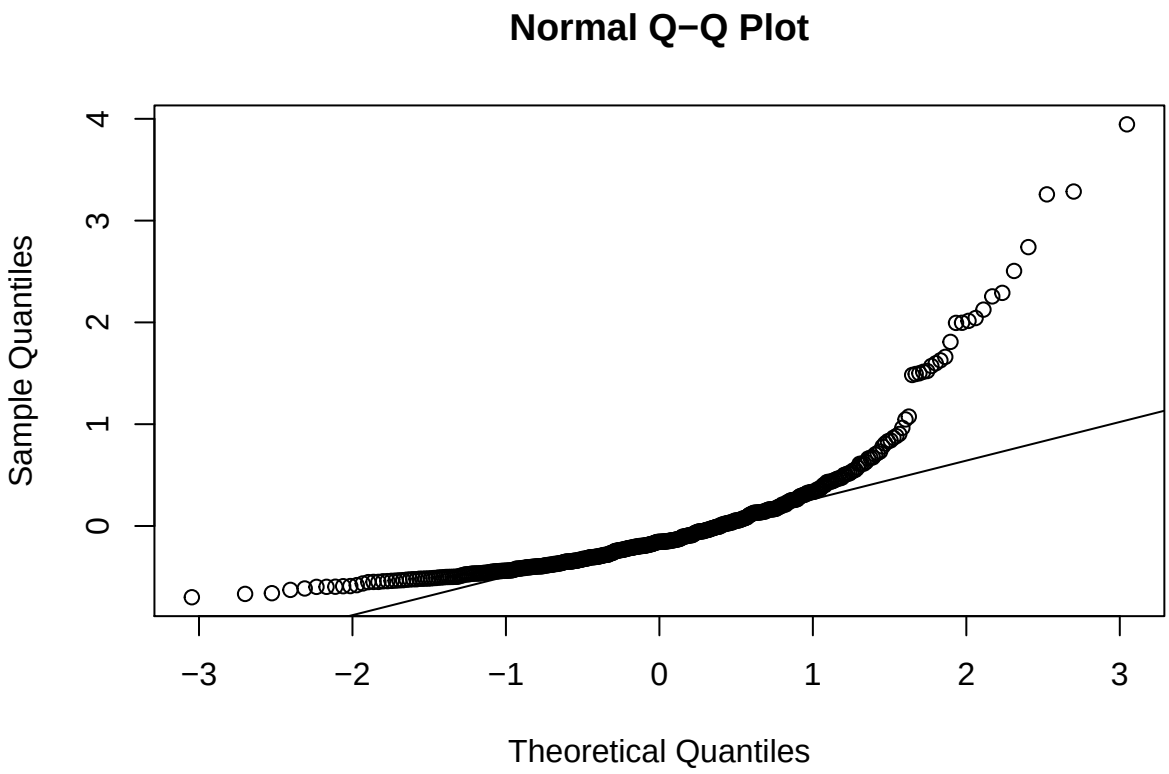
```r
int020 <- lm(rt.obj ~ 1, data = sub020)

# Plot residuals with Q-Q plots
qqnorm(residuals(int001))
qqline(residuals(int001))
```



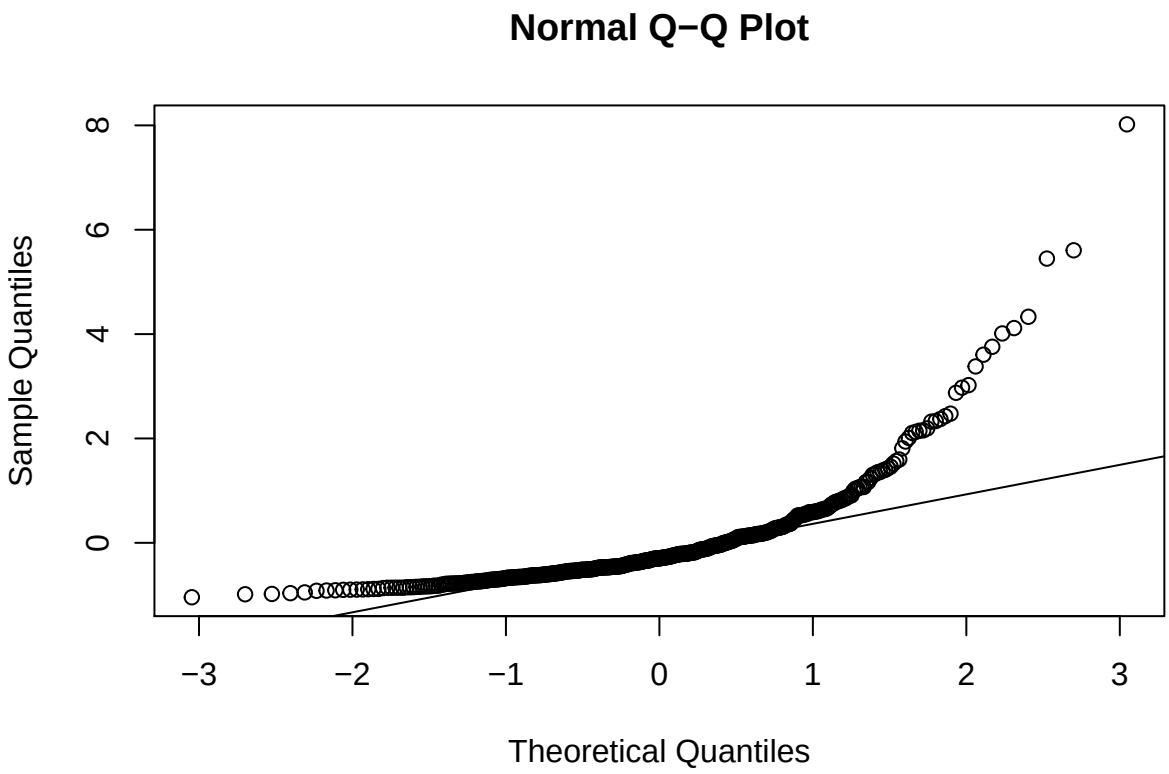

```r
qqnorm(residuals(int005))
qqline(residuals(int005))
```




```
qqnorm(residuals(int016))
qqline(residuals(int016))
```

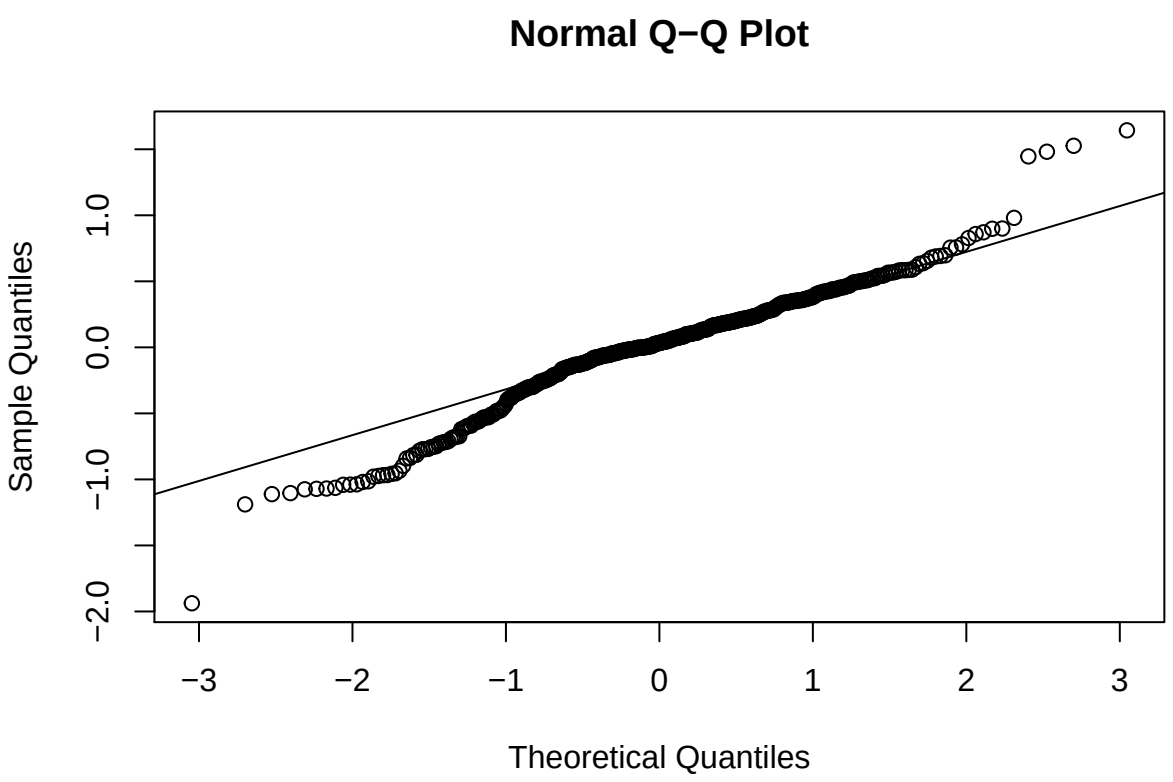



```
qqnorm(residuals(int020))
qqline(residuals(int020))
```

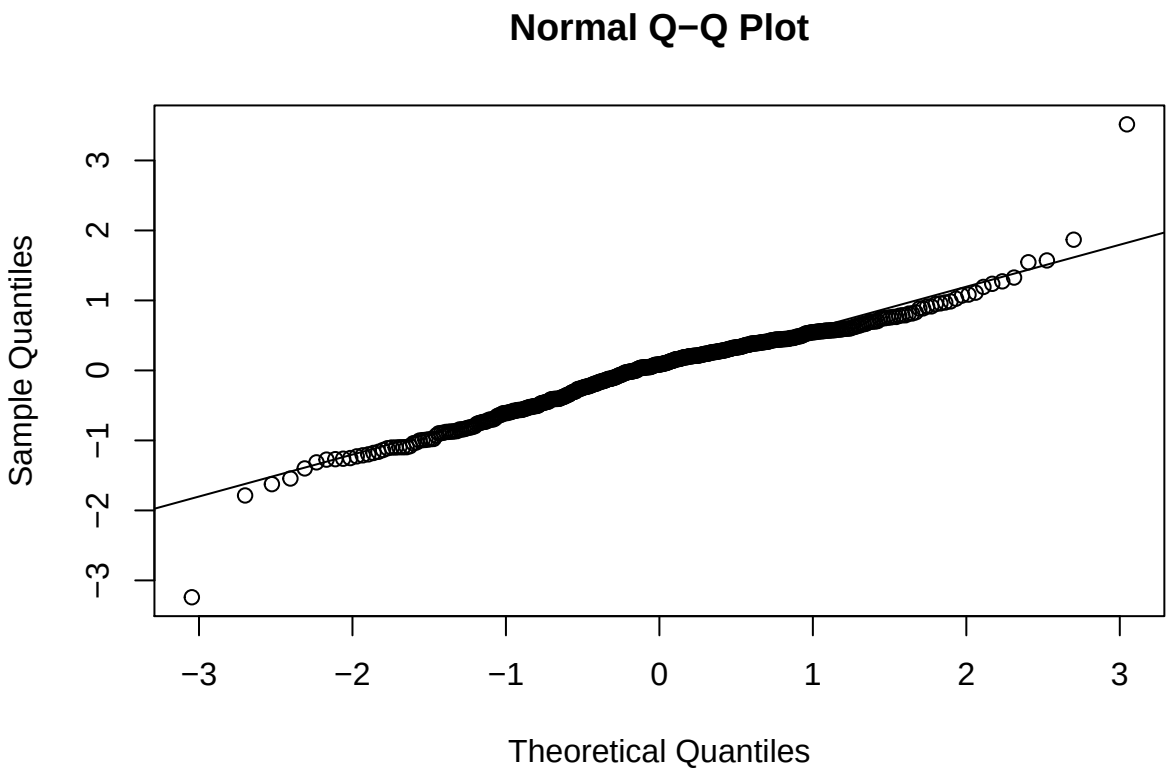


i. comment on these The residuals are clearly not normally distributed (they do not approximate the line), and they are heavily skewed. ii. does a log-transformation of the response time data improve the Q-Q-plots?

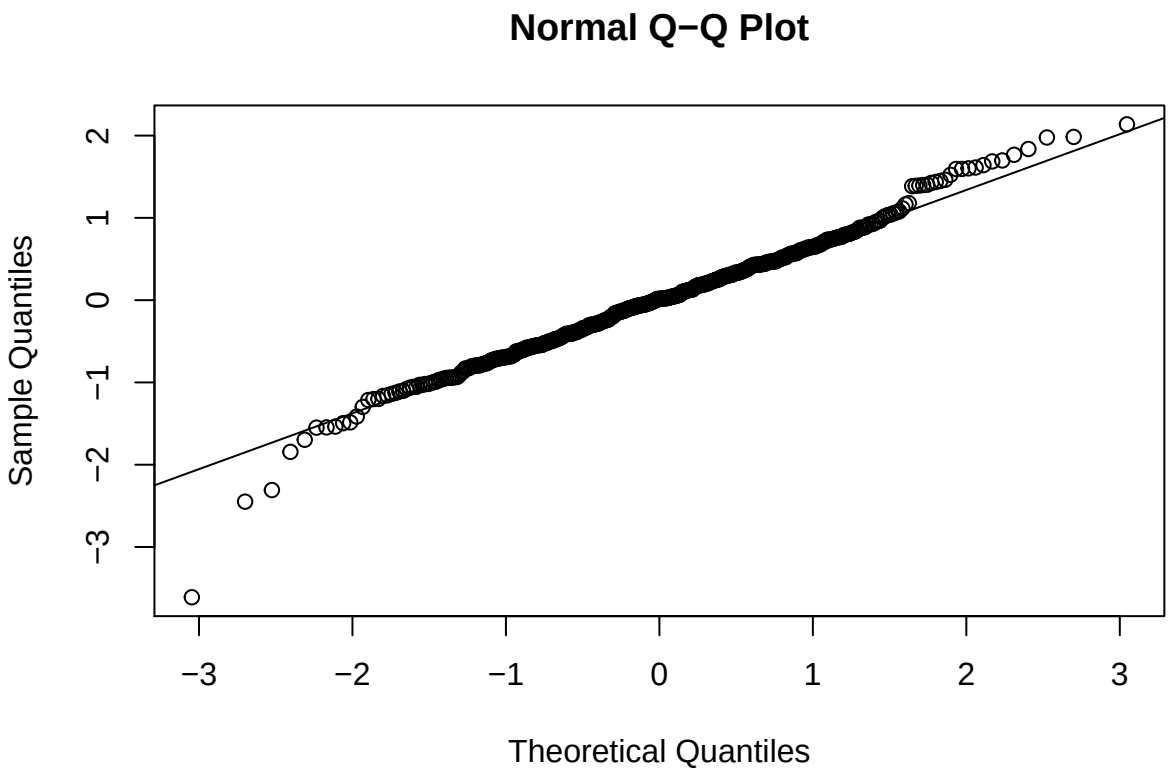
```
#Log-transform variables
sub001$log.rt.obj <- log(sub001$rt.obj)
sub005$log.rt.obj <- log(sub005$rt.obj)
sub016$log.rt.obj <- log(sub016$rt.obj)
sub020$log.rt.obj <- log(sub020$rt.obj)

# Model intercepts
log.int001 <- lm(log.rt.obj ~ 1, data = sub001)
log.int005 <- lm(log.rt.obj ~ 1, data = sub005)
log.int016 <- lm(log.rt.obj ~ 1, data = sub016)
log.int020 <- lm(log.rt.obj ~ 1, data = sub020)

# Plot residuals with Q-Q plots
qqnorm(residuals(log.int001))
qqline(residuals(log.int001))
```



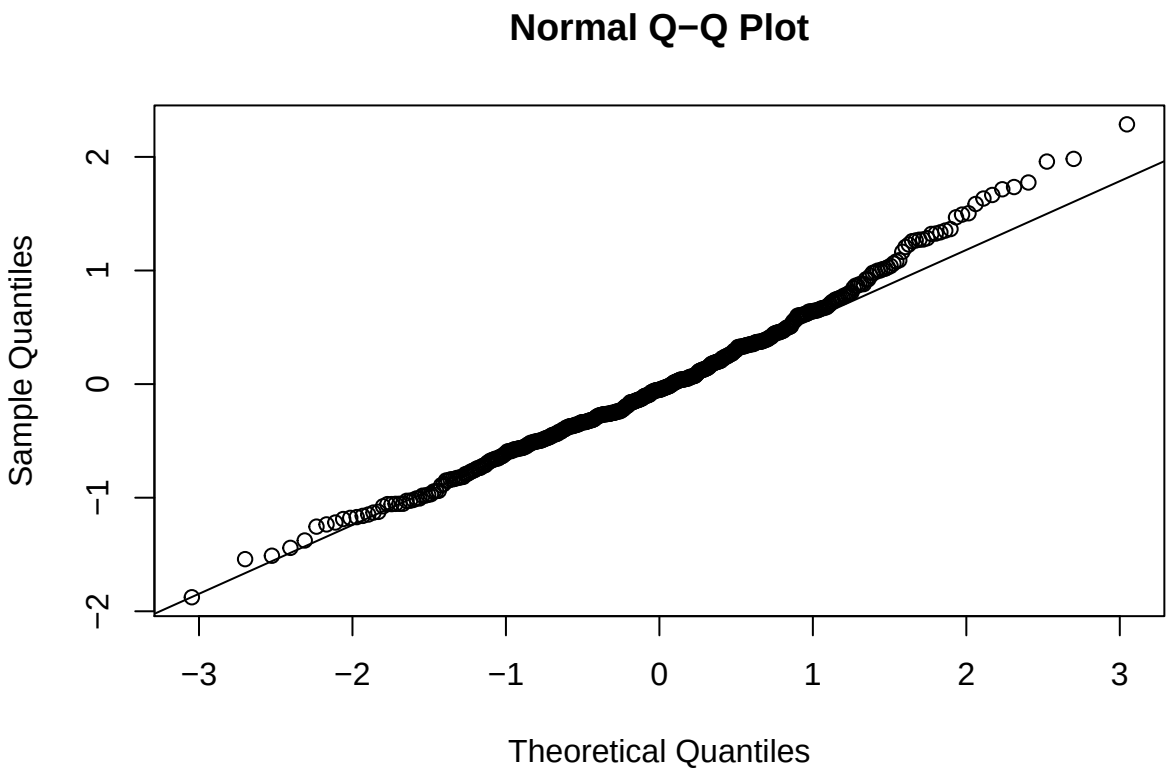

```
qqnorm(residuals(log.int005))
qqline(residuals(log.int005))
```




```
qqnorm(residuals(log.int016))
qqline(residuals(log.int016))
```




```
qqnorm(residuals(log.int020))
qqline(residuals(log.int020))
```



The residuals appear significantly more normal after log-transformation, being skewed to a much smaller extent. 2) Now do a partial pooling model modeling objective response times as dependent on *task*. (set `REML=FALSE` in your `lmer`-specification)

```
#Why REML = FALSE?
#It's generally good to use REML, if it is available, when you are interested in the magnitude of the r

# Build models
model_2.1 <- lmerTest::lmer(rt.obj ~ task + (1 | subject), data = exp, REML = FALSE)
model_2.2 <- lmerTest::lmer(rt.obj ~ task + (1 + pas | subject), data = exp, REML = FALSE)
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning: Model failed to converge with 1 negative eigenvalue: -8.3e+01
```

```
# Compare models
AIC(model_2.1, model_2.2)
```

```
##           df      AIC
## model_2.1  5 61940.21
## model_2.2 14 61939.76
```

```
# Residual variance:
c(var(resid(model_2.1)),
  var(resid(model_2.2))
)
```

```
## [1] 8.158282 8.125095
```

```
# Residual standard deviation
c(summary(model_2.1)$sigma,
  summary(model_2.2)$sigma
)
```

```
## [1] 2.858998 2.854746
```

i. which would you include among your random effects and why? (support your choices with relevant measu
Subject has to be included as a random effect as it represents sub-groups that significantly influenc
ii. explain in your own words what your chosen models says about response times between the different ta

```
summary(model_2.1)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ task + (1 | subject)
##    Data: exp
##
##      AIC      BIC   logLik deviance df.resid
##  61940.2  61977.4 -30965.1  61930.2    12523
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
##  -0.630  -0.155  -0.072   0.051 101.443
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 0.1147   0.3386
##  Residual             8.1739   2.8590
## Number of obs: 12528, groups:  subject, 29
##
## Fixed effects:
##                   Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)     1.120e+00  7.689e-02  4.775e+01  14.568  < 2e-16 ***
## taskquadruplet -1.532e-01  6.257e-02  1.250e+04  -2.449  0.01433 *
## tasksingles    -1.915e-01  6.257e-02  1.250e+04  -3.061  0.00221 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr
## taskqudrplt -0.407
## tasksingles -0.407  0.500
```

The model shows that reaction times are highest for pairs tasks, with somewhat lower reaction times for

3) Now add *pas* and its interaction with *task* to the fixed effects

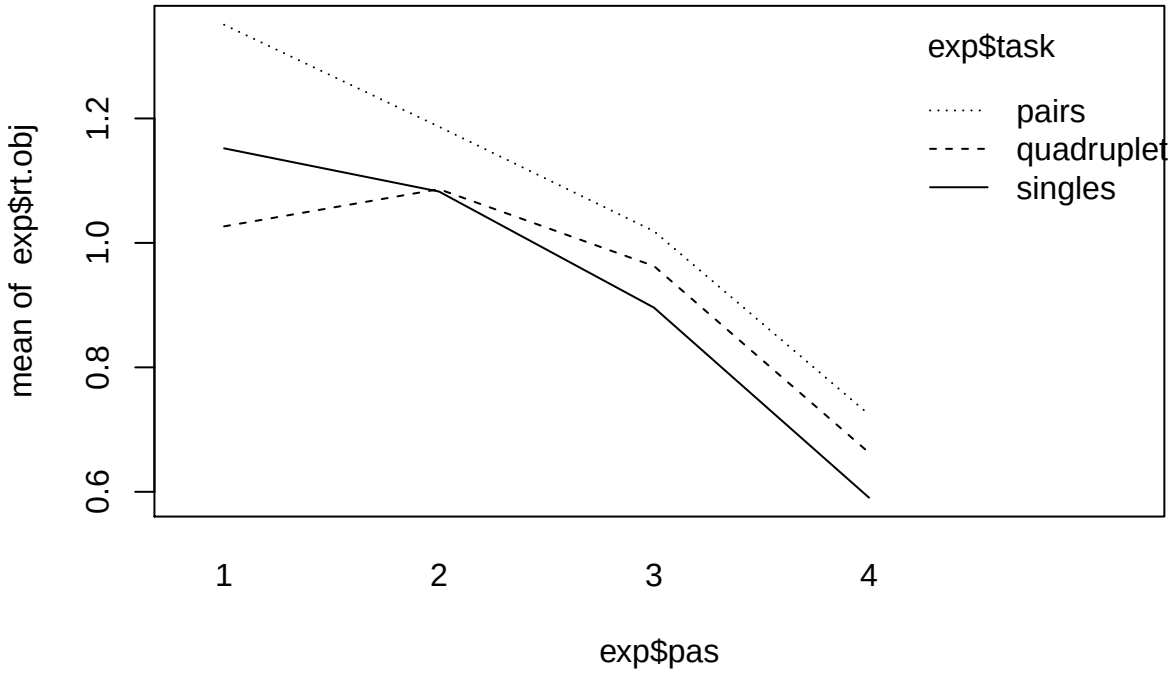
```
#When you want to model both main effects and their interactions, use an asterisk (*) between the effec
model_2.3 <- lmerTest::lmer(rt.obj ~ pas * task + (1 | subject), data = exp, REML = FALSE)
summary(model_2.3)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ pas * task + (1 | subject)
##    Data: exp
##
```

```
##      AIC      BIC   logLik deviance df.resid
##  61917.4  62021.5 -30944.7  61889.4    12514
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
##  -0.667  -0.152  -0.064   0.047 101.556
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 0.09749  0.3122
##  Residual             8.14982  2.8548
## Number of obs: 12528, groups:  subject, 29
##
## Fixed effects:
##                        Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)          1.335e+00  9.786e-02  1.569e+02  13.644  < 2e-16 ***
## pas2                -1.394e-01  1.150e-01  1.228e+04  -1.213  0.22521
## pas3                -3.359e-01  1.314e-01  1.184e+04  -2.557  0.01058 *
## pas4                -5.795e-01  1.341e-01  9.034e+03  -4.321 1.57e-05 ***
## taskquadruplet      -3.229e-01  1.068e-01  1.251e+04  -3.023  0.00251 **
## tasksingles         -2.156e-01  1.155e-01  1.252e+04  -1.866  0.06205 .
## pas2:taskquadruplet  2.273e-01  1.582e-01  1.252e+04   1.437  0.15067
## pas3:taskquadruplet  2.624e-01  1.822e-01  1.252e+04   1.440  0.14979
## pas4:taskquadruplet  2.583e-01  1.797e-01  1.251e+04   1.437  0.15068
## pas2:tasksingles     1.251e-01  1.661e-01  1.253e+04   0.753  0.45131
## pas3:tasksingles     9.453e-02  1.852e-01  1.252e+04   0.510  0.60972
## pas4:tasksingles     7.097e-02  1.746e-01  1.252e+04   0.406  0.68444
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas2   pas3   pas4   tskqdr tsksng ps2:tskq ps3:tskq
## pas2        -0.553
## pas3        -0.499  0.425
## pas4        -0.504  0.414  0.395
## taskqudrplt -0.562  0.478  0.420  0.410
## tasksingles -0.517  0.441  0.386  0.373  0.474
## ps2:tskqdrp  0.378 -0.697 -0.282 -0.276 -0.677 -0.319
## ps3:tskqdrp  0.328 -0.280 -0.681 -0.238 -0.587 -0.279  0.397
## ps4:tskqdrp  0.333 -0.284 -0.249 -0.658 -0.595 -0.282  0.402    0.348
## ps2:tsksngl  0.368 -0.672 -0.279 -0.270 -0.330 -0.697  0.481    0.194
## ps3:tsksngl  0.327 -0.276 -0.677 -0.242 -0.296 -0.625  0.199    0.484
## ps4:tsksngl  0.343 -0.290 -0.255 -0.681 -0.314 -0.662  0.211    0.185
##             ps4:tskq ps2:tsks ps3:tsks
## pas2
## pas3
## pas4
## taskqudrplt
## tasksingles
## ps2:tskqdrp
## ps3:tskqdrp
## ps4:tskqdrp
## ps2:tsksngl  0.195
## ps3:tsksngl  0.176    0.437
```

```
## ps4:tsksngl  0.507    0.460    0.413
```

```
interaction.plot(exp$pas, exp$task, exp$rt.obj) #plot the interaction to help me understand it better
```



The more complex model doesn't appear to be better than the more simple one: only some of the fixed effects and none of the interaction effects are similar. i. how many types of group intercepts (random effects) can you add without ending up with convergence issues or singular fits?

```
model_2.4 <- lmerTest::lmer(rt.obj ~ task + (1 | subject) + (1 | cue) + (1 | trial) + (1 | target.type)
model_2.5 <- lmerTest::lmer(rt.obj ~ task + (1 | subject) + (1 | cue) + (1 | trial) + (1 | target.type)
```

```
## boundary (singular) fit: see ?isSingular
```

We succeeded at adding 5 group intercepts before convergence issues happened.
ii. create a model by adding random intercepts (without modelling slopes) that results in a singular fi

```
# I take the model with convergence issues from the previous exercise
print(VarCorr(model_2.5), comp = 'Variance')
```

```
##  Groups      Name        Variance
##  trial       (Intercept) 0.0020319
##  cue         (Intercept) 0.0957207
##  subject     (Intercept) 0.1000015
##  pas         (Intercept) 0.0347612
##  task        (Intercept) 0.0000000
##  target.type (Intercept) 0.0025884
##  Residual                8.1138210
```

iii. in your own words - how could you explain why your model would result in a singular fit?
When a group intercept is modeled, it is because the grouping variable accounts for some variance in th

This same can be done for all other group effects modeled. However, at some point, the combinations of

Exercise 3

1) Initialise a new data frame, `data.count`. *count* should indicate the number of times they categorized their experience as *pas* 1-4 for each *task*. I.e. the data frame would have for subject 1: for task:singles, pas1 was used # times, pas2 was used # times, pas3 was used # times and pas4 was used # times. You would then do the same for task:pairs and task:quadruplet

```
data.count <- df %>%
  group_by(subject, task, pas) %>%
  summarise('count' = n()) #create grouping data frame
```

```
## `summarise()` has grouped output by 'subject', 'task'. You can override using the `.groups` argument
```

```
data.count$subject <- as.factor(data.count$subject)
data.count$task <- as.factor(data.count$task)
data.count$pas <- as.factor(data.count$pas)
data.count$count <- as.integer(data.count$count) #ensure correct data format
```

2) Now fit a multilevel model that models a unique "slope" for *pas* for each *subject* with the interaction between *pas* and *task* and their main effects being modelled

```
model_3.2 <- glmer(count ~ pas + task + pas:task + (1 + pas | subject),
                   data = data.count,
                   family = 'poisson',
                   control = glmerControl(optimizer = "bobyqa"))
summary(model_3.2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas + task + pas:task + (1 + pas | subject)
##    Data: data.count
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   3148.4   3232.7  -1552.2   3104.4      318
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.3871 -0.7853 -0.0469  0.7550  6.5438
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 0.3324   0.5765
##          pas2        0.3803   0.6167   -0.75
##          pas3        1.1960   1.0936   -0.84  0.63
##          pas4        2.3736   1.5407   -0.86  0.42  0.72
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          4.03570    0.10976  36.770  < 2e-16 ***
## pas2                -0.02378    0.11963  -0.199 0.842456
## pas3                -0.51365    0.20717  -2.479 0.013164 *
```

```
## pas4                -0.77292    0.29075  -2.658 0.007851 **
## taskquadruplet       0.11490    0.03127   3.674 0.000239 ***
## tasksingles         -0.23095    0.03418  -6.756 1.42e-11 ***
## pas2:taskquadruplet -0.11376    0.04605  -2.470 0.013508 *
## pas3:taskquadruplet -0.20902    0.05287  -3.954 7.69e-05 ***
## pas4:taskquadruplet -0.21500    0.05230  -4.111 3.94e-05 ***
## pas2:tasksingles     0.19536    0.04830   4.045 5.23e-05 ***
## pas3:tasksingles     0.24299    0.05369   4.526 6.02e-06 ***
## pas4:tasksingles     0.56346    0.05101  11.045  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas2   pas3   pas4   tskqdr tsksng ps2:tskq ps3:tskq
## pas2        -0.742
## pas3        -0.829  0.613
## pas4        -0.847  0.412  0.703
## taskqudrplt -0.151  0.138  0.080  0.057
## tasksingles -0.138  0.126  0.073  0.052  0.484
## ps2:tskqdrp  0.102 -0.198 -0.054 -0.039 -0.679 -0.328
## ps3:tskqdrp  0.089 -0.082 -0.125 -0.034 -0.592 -0.286  0.402
## ps4:tskqdrp  0.090 -0.083 -0.048 -0.093 -0.598 -0.289  0.406    0.354
## ps2:tsksngl  0.098 -0.188 -0.052 -0.037 -0.342 -0.708  0.490    0.203
## ps3:tsksngl  0.088 -0.080 -0.124 -0.033 -0.308 -0.637  0.209    0.486
## ps4:tsksngl  0.092 -0.085 -0.049 -0.091 -0.324 -0.670  0.220    0.192
##             ps4:tskq ps2:tsks ps3:tsks
## pas2
## pas3
## pas4
## taskqudrplt
## tasksingles
## ps2:tskqdrp
## ps3:tskqdrp
## ps4:tskqdrp
## ps2:tsksngl  0.205
## ps3:tsksngl  0.184    0.451
## ps4:tsksngl  0.507    0.474    0.427
```

i. which family should be used?
A Poisson-distribution should be used as we are investigating count data for specific units of measureme
HOWEVER:

```
mean(data.count$count)
```

```
## [1] 53.32647
```

```
var(data.count$count)
```

```
## [1] 1214.002
```

The Poisson-distribution assumes that the mean and the variance of the count variable is equal. If the v
ii. why is a slope for _pas_ not really being modeled?
pas is encoded as factor, i.e. a categorical variable, meaning that it is assumed not to lie along a
iii. if you get a convergence error, try another algorithm (the default is the _Nelder_Mead_) - try (_b
iv. when you have a converging fit - fit a model with only the main effects of _pas_ and _task_. Compare

```
model_3.2b <- glmer(count ~ pas + task + (1 + pas | subject),
                    data = data.count,
                    family = 'poisson',
                    control = glmerControl(optimizer = "bobyqa"))
summary(model_3.2b)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas + task + (1 + pas | subject)
##    Data: data.count
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   3398.5   3459.8  -1683.3   3366.5      324
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.5885 -0.9001 -0.0477  0.8253  6.5100
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 0.3325   0.5766
##          pas2        0.3805   0.6169   -0.75
##          pas3        1.1895   1.0906   -0.84  0.63
##          pas4        2.4221   1.5563   -0.86  0.42  0.73
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.004594   0.108722  36.833   <2e-16 ***
## pas2           -0.006531   0.116615  -0.056   0.9553
## pas3           -0.509923   0.204448  -2.494   0.0126 *
## pas4           -0.663848   0.291955  -2.274   0.0230 *
## taskquadruplet  0.003294   0.018188   0.181   0.8563
## tasksingles     0.004307   0.018177   0.237   0.8127
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas2   pas3   pas4   tskqdr
## pas2        -0.742
## pas3        -0.832  0.623
## pas4        -0.850  0.412  0.723
## taskqudrplt -0.084  0.000  0.001 -0.002
## tasksingles -0.084  0.000  0.000  0.000  0.501
```

v. indicate which of the two models, you would choose and why

```
# AIC comparison:
AIC(model_3.2, model_3.2b)
```

```
##            df      AIC
## model_3.2  22 3148.441
## model_3.2b 16 3398.549
```

```r
# Residual variance (using formula):
c(var(residuals(model_3.2)),
  var(residuals(model_3.2b))
)
```

```
## [1] 2.055365 2.828064
```

```r
# Residual standard deviation (using formula):
c(sd(residuals(model_3.2)),
  sd(residuals(model_3.2b))
)
```

```
## [1] 1.433654 1.681685
```

When checking AIC scores, residual variance and standard deviation, the model _with_ interactions appea
vi. based on your chosen model - write a short report on what this says about the distribution of ratin

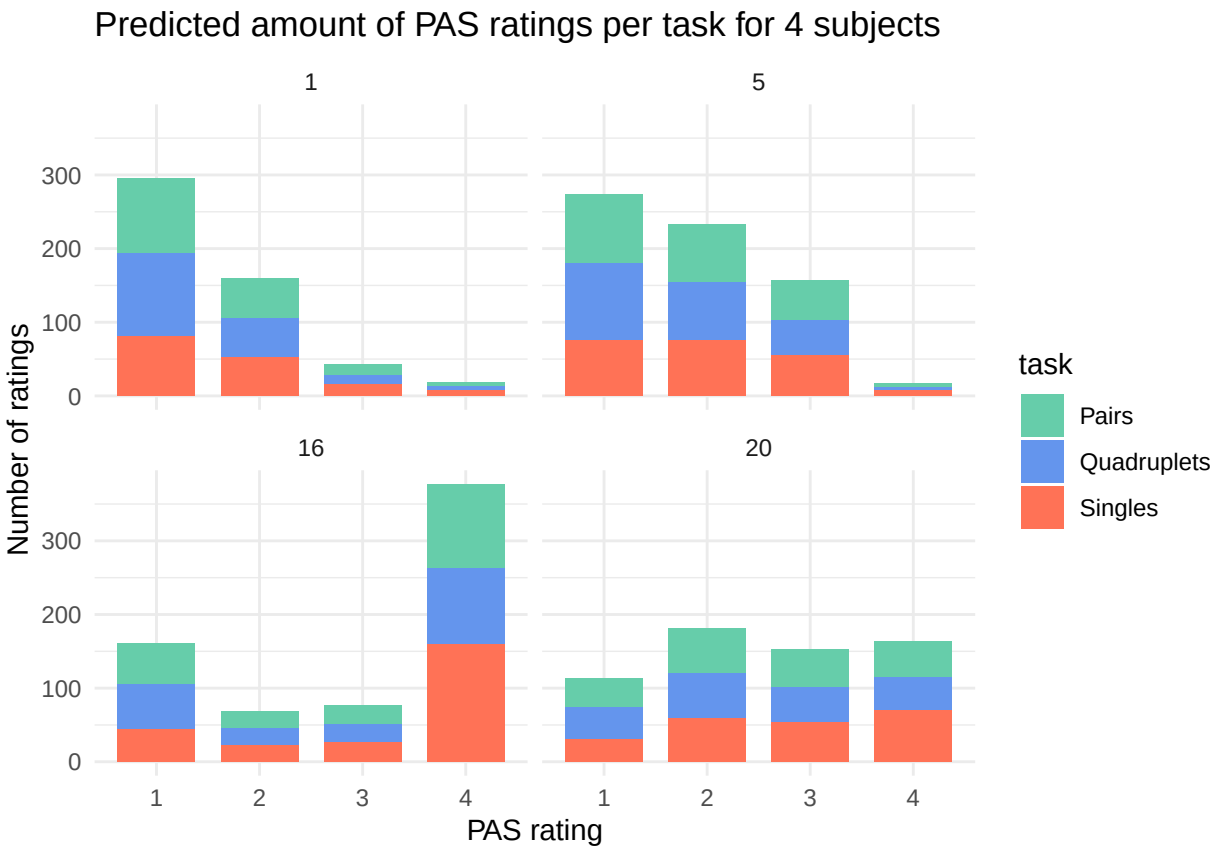
```r
#When you want to interpret the output of a Poisson regression, you have to take the exp() of output. I
normal <- fixef(model_3.2)
exp <- exp(fixef(model_3.2))
poisson <- as.data.frame(cbind(normal, exp))

#If the value is negative, it implies a decrease, and the exponentiated value has to be subtracted from
poisson$corrected <- ifelse((poisson$normal < 0), 1 - poisson$exp, poisson$exp)
poisson$direction <- ifelse((poisson$normal < 0), "decrease", "increase")

poisson
```

```
##                           normal        exp   corrected direction
## (Intercept)          4.03570123 56.5825836 56.58258362  increase
## pas2                -0.02377598  0.9765044  0.02349556  decrease
## pas3                -0.51364811  0.5983089  0.40169110  decrease
## pas4                -0.77292326  0.4616615  0.53833846  decrease
## taskquadruplet       0.11490066  1.1217620  1.12176199  increase
## tasksingles         -0.23094567  0.7937826  0.20621741  decrease
## pas2:taskquadruplet -0.11375513  0.8924765  0.10752353  decrease
## pas3:taskquadruplet -0.20901514  0.8113830  0.18861705  decrease
## pas4:taskquadruplet -0.21500491  0.8065375  0.19346252  decrease
## pas2:tasksingles     0.19536301  1.2157522  1.21575224  increase
## pas3:tasksingles     0.24299389  1.2750608  1.27506084  increase
## pas4:tasksingles     0.56346179  1.7567435  1.75674346  increase
```

As a baseline, i.e. for when the PAS rating is "1" and the task is "pairs", there are ~56.58 instances.
An increase in probability for so rating a trial is shown for all interactions moving along the PAS sca
All main and interaction effects are significant, p < .05, with the exception of moving from PAS rating
vii. include a plot that shows the estimated amount of ratings for four subjects of your choosing.

```r
# When making predictions from a regression model, we can make either in-sample or out-of-sample predic
# When making in-sample predictions, we basically obtain the predicted values for the y axis based on t
# Since we are trying to estimate ratings for subjects already in the dataset, we make in-sample predic

#Extract the fitted values
data.count$fitted <- fitted(model_3.2)

#Select subjects
df_3.2 <- data.count %>%
  filter(subject == '1' | subject == '5' | subject == '16' | subject == '20')
```

```
ggplot(data = df_3.2, aes(x = pas, y = fitted, fill = task)) +
  geom_col(width = 0.75) +
  facet_wrap(~subject) +
  labs(title = "Predicted amount of PAS ratings per task for 4 subjects", x = "PAS rating", y = "Number
  scale_fill_manual(labels = c("Pairs", "Quadruplets", "Singles"), values = c("aquamarine3", "cornflowe
  guides(color = guide_legend("Task")) +
  theme_minimal()
```



3) Finally, fit a multilevel model that models *correct* as dependent on *task* with a unique intercept for each *subject*

```
model_3.3a <- glmer(correct ~ task + (1 | subject), family = 'binomial', data = df)
summary(model_3.3a)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + (1 | subject)
##    Data: df
##
##      AIC      BIC   logLik deviance df.resid
##  19927.2  19958.4  -9959.6  19919.2    18127
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7426 -1.0976  0.5098  0.6101  0.9111
##
```

```
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1775   0.4214
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.10071    0.08386  13.125  < 2e-16 ***
## taskquadruplet -0.09825    0.04190  -2.345    0.019 *
## tasksingles     0.18542    0.04337   4.276 1.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr
## taskqudrplt -0.256
## tasksingles -0.247  0.495
```

i. does _task_ explain performance?
ii. add _pas_ as a main effect on top of _task_ - what are the consequences of that?

```
model_3.3b <- glmer(correct ~ task + pas + (1 | subject), family = 'binomial', data = df)
summary(model_3.3b)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + pas + (1 | subject)
##    Data: df
##
##      AIC      BIC   logLik deviance df.resid
##  17424.9  17479.5  -8705.5  17410.9    18124
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.4872 -0.6225  0.3240  0.5767  1.6144
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1979   0.4449
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     0.08530    0.09143   0.933    0.351
## taskquadruplet -0.03055    0.04497  -0.679    0.497
## tasksingles    -0.01059    0.04687  -0.226    0.821
## pas2            0.95477    0.04419  21.604   <2e-16 ***
## pas3            1.97709    0.06219  31.793   <2e-16 ***
## pas4            3.12732    0.08626  36.255   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng pas2   pas3
```

```
## taskqudrplt -0.259
## tasksingles -0.225  0.489
## pas2        -0.212  0.021 -0.040
## pas3        -0.165  0.030 -0.045  0.355
## pas4        -0.123  0.016 -0.080  0.257  0.236
```

iii. now fit a multilevel model that models _correct_ as dependent on _pas_ with a unique intercept for

```
model_3.3c <- glmer(correct ~ pas + (1 | subject), family = 'binomial', data = df)
summary(model_3.3c)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas + (1 | subject)
##    Data: df
##
##      AIC      BIC   logLik deviance df.resid
##  17421.4  17460.4  -8705.7  17411.4    18126
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.5665 -0.6243  0.3244  0.5754  1.6017
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1981   0.4451
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.07044    0.08773   0.803    0.422
## pas2         0.95575    0.04410  21.671   <2e-16 ***
## pas3         1.97892    0.06201  31.914   <2e-16 ***
## pas4         3.12940    0.08579  36.476   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) pas2   pas3
## pas2 -0.223
## pas3 -0.173  0.352
## pas4 -0.136  0.253  0.231
```

iv. finally, fit a model that models the interaction between _task_ and _pas_ and their main effects

```
model_3.3d <- glmer(correct ~ task * pas + (1 | subject), family = 'binomial', data = df)
summary(model_3.3d)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task * pas + (1 | subject)
##    Data: df
##
##      AIC      BIC   logLik deviance df.resid
```

```
##  17431.0  17532.4  -8702.5  17405.0    18118
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.9329 -0.6276  0.3186  0.5750  1.6411
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1987   0.4458
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.083724   0.095677   0.875    0.382
## taskquadruplet       0.006494   0.064352   0.101    0.920
## tasksingles         -0.058974   0.070459  -0.837    0.403
## pas2                 0.963211   0.073725  13.065   <2e-16 ***
## pas3                 1.999940   0.102926  19.431   <2e-16 ***
## pas4                 3.049626   0.145543  20.953   <2e-16 ***
## taskquadruplet:pas2 -0.049301   0.100765  -0.489    0.625
## tasksingles:pas2     0.040480   0.105998   0.382    0.703
## taskquadruplet:pas3 -0.134128   0.142258  -0.943    0.346
## tasksingles:pas3     0.079972   0.145297   0.550    0.582
## taskquadruplet:pas4 -0.095815   0.199830  -0.479    0.632
## tasksingles:pas4     0.296561   0.196610   1.508    0.131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng pas2   pas3   pas4   tskq:2 tsks:2 tskq:3
## taskqudrplt -0.357
## tasksingles -0.323  0.481
## pas2        -0.329  0.464  0.420
## pas3        -0.242  0.332  0.298  0.322
## pas4        -0.175  0.235  0.206  0.224  0.177
## tskqdrplt:2  0.227 -0.639 -0.306 -0.704 -0.211 -0.149
## tsksngls:p2  0.220 -0.321 -0.665 -0.676 -0.209 -0.146  0.487
## tskqdrplt:3  0.161 -0.451 -0.217 -0.211 -0.695 -0.105  0.288  0.145
## tsksngls:p3  0.157 -0.232 -0.484 -0.204 -0.681 -0.104  0.147  0.323  0.490
## tskqdrplt:4  0.114 -0.321 -0.153 -0.147 -0.105 -0.698  0.205  0.102  0.144
## tsksngls:p4  0.114 -0.171 -0.357 -0.147 -0.103 -0.706  0.109  0.237  0.077
##             tsks:3 tskq:4
## taskqudrplt
## tasksingles
## pas2
## pas3
## pas4
## tskqdrplt:2
## tsksngls:p2
## tskqdrplt:3
## tsksngls:p3
## tskqdrplt:4  0.074
## tsksngls:p4  0.172  0.515
```

v. describe in your words which model is the best in explaining the variance in accuracy

```
#Residual variance:
c(var(residuals(model_3.3a)),
  var(residuals(model_3.3b)),
  var(residuals(model_3.3c)),
  var(residuals(model_3.3d))
)
```

```
## [1] 1.0704429 0.9386723 0.9386966 0.9383736
```

```
#AIC
anova(model_3.3a, model_3.3b, model_3.3c, model_3.3d, test = 'LR')
```

```
## Data: df
## Models:
## model_3.3a: correct ~ task + (1 | subject)
## model_3.3c: correct ~ pas + (1 | subject)
## model_3.3b: correct ~ task + pas + (1 | subject)
## model_3.3d: correct ~ task * pas + (1 | subject)
##            npar   AIC   BIC  logLik deviance     Chisq Df Pr(>Chisq)
## model_3.3a    4 19927 19958 -9959.6    19919
## model_3.3c    5 17421 17460 -8705.7    17411 2507.8224  1     <2e-16 ***
## model_3.3b    7 17425 17480 -8705.5    17411    0.4759  2     0.7883
## model_3.3d   13 17431 17532 -8702.5    17405    5.9376  6     0.4302
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

model_3.3c appears to be the most suitable model. It has the lowest residual variance and AIC, and the addition of multiple main effects or of interaction effects did not significantly improve the model's performance. PAS seems to account for most of the variance in the dataset.