Supplementary Code

This document contains an overview of the code used, following the ordering of the Results section. We start by loading the necessary R packages.

```
library(tidyverse)
```

```
## -- Attaching packages -----------------------------------------------------------------------------
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ---------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggthemes)
library(patchwork)
library(glue)
```

```
##
## Attaching package: 'glue'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(nlme)
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:glue':
##
##     collapse
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(mgcv)
```

```
## This is mgcv 1.8-36. For overview type 'help("mgcv-package")'.
```

```
library(latex2exp)
library(furrr)
```

```
## Loading required package: future
```

Next, we set the default theme for plotting with `ggplot2`:

```
theme_set(theme_bw())
theme_update(
  panel.border = element_blank(),
  axis.line = element_line(),
  panel.grid = element_blank(),
  strip.background = element_blank()
)
```

All the functions with R code are contained in the directory named `code/`. The line below sources all these functions so they are available in this document. We follow the convention that each function has its own file, so in order to find the code for a function named `foo()`, open the file `code/foo.r`.

```
walk(list.files("./code", full.names = TRUE), source)
```

Finally, we source the R script which creates simulated datasets which can be used to run all the code.

```
source(params$data_script_path)
```

If it does not already exist, we create a directory which will contain the generated figures.

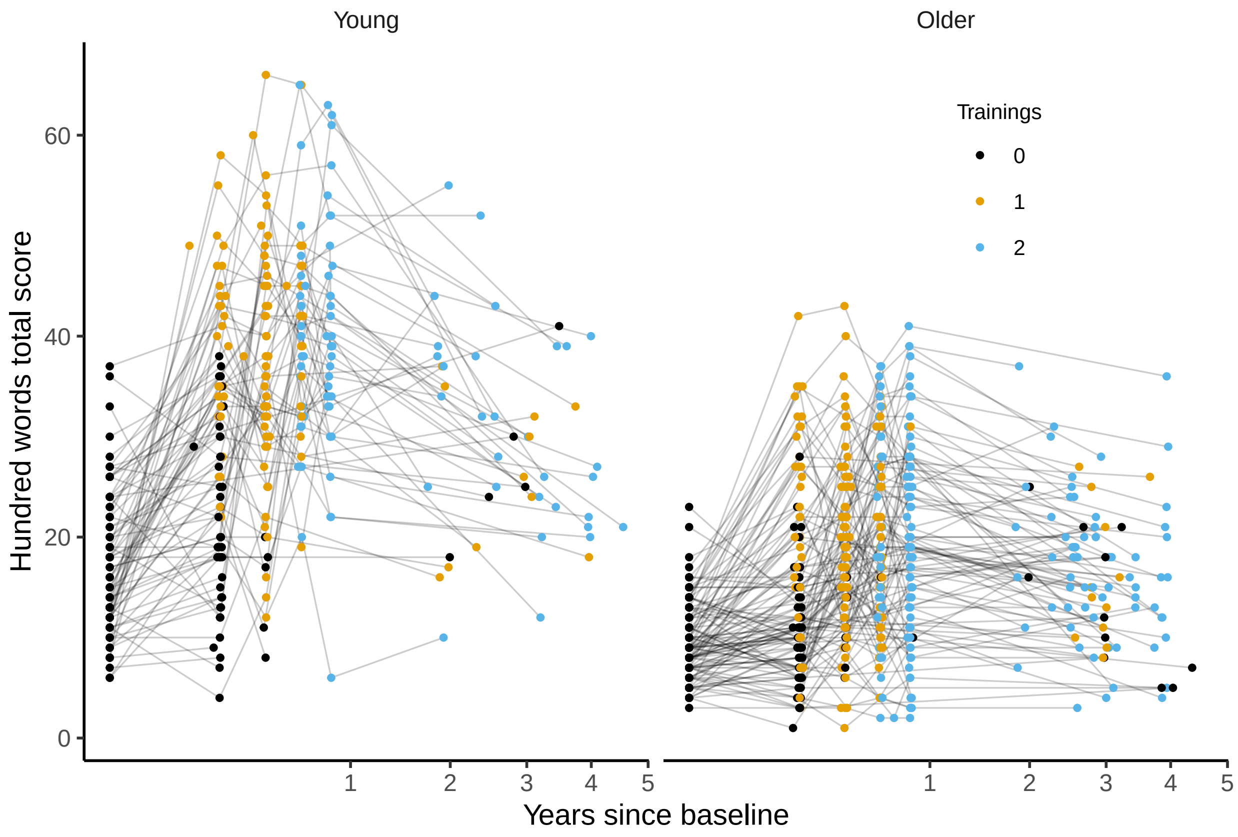
```
if(!dir.exists("figures")) dir.create("figures")
```

Section 2.1

This chapter presents code for the section titled "Is there an intervention effect on memory performance, and can effects be observed three years after the cessation of intervention?".

Figure 1 is created with a call to `create_figure_1()`.

```
create_figure_1(memdat_long)
```



```
ggsave("figures/hwt_score.png",
       width = 16, height = 10, units = "cm")
```

Modeling strategy

There are multiple effects at play simultaneously in these data:

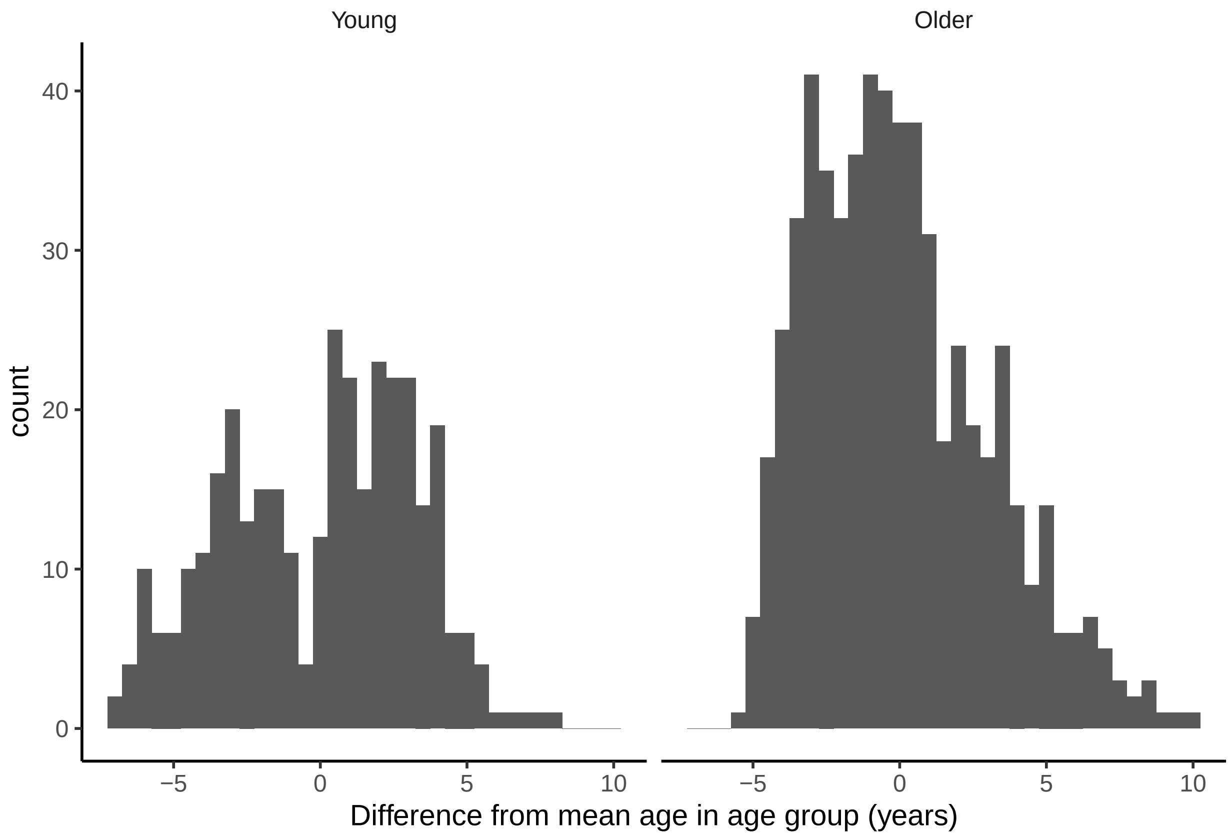
- Participants get up to five years older during the full study period, and we might thus suspect a change in their overall level due to *aging*.
- The effect of memory training leads to increased performance, and this *training effect* might eventually decay during rest.
- The pure *retest effect* of having taken the test previously leads to increased scores even when the participant's actual ability is constant.
- Participants belong to two different age groups.
- There might be other confounds, e.g. the participant's sex.

We will now describe the model components related to each of these effects.

Aging effect

In order to capture the aging effect, we defined a variable measuring the participant's deviation from the mean age in the age group, at each timepoint. The function `plot_age_dev_histogram()` plots histograms for this deviation in each group.

```
plot_age_dev_histogram(memdat_long)
```



Using this variable, we can capture the aging effect within each age group with the term

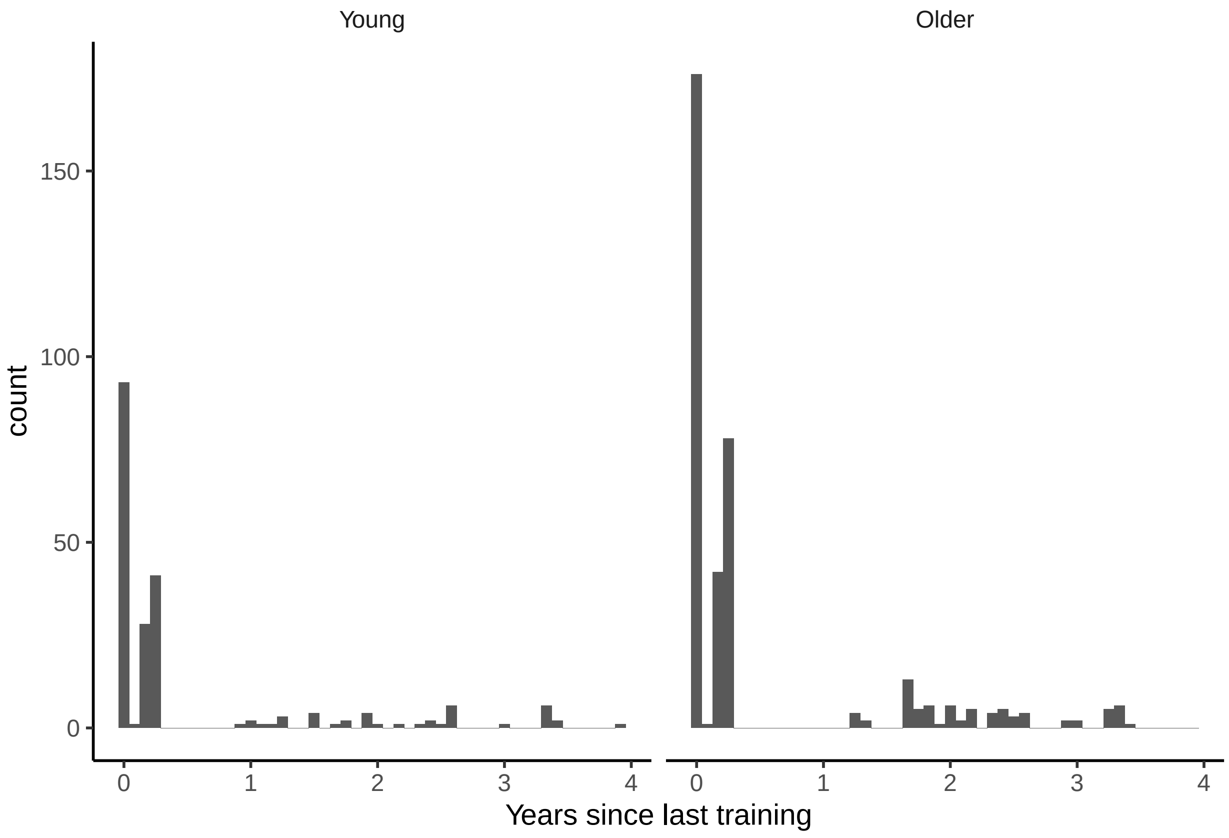
$$b_a = (\beta_{a1} d_1 + \beta_{a2} d_2)\Delta a,$$

where Δa is the participant's difference from her/his age group mean, d_1 is a dummy variable which equals 1 if the participant is in the "Young" age group and 0 otherwise, d_2 is a similarly defined dummy variable for belonging to the "Older" age group, β_{a1} is the aging effect in the "Young" group and β_{a2} is the aging effect in the "Older" group. This assumes that the aging effect is linear in each age group, but that assumption is realistic over a time interval of ten years. For practical purposes, this age deviation variable is very similar to the "time since baseline" variable, and as will be described later, using either of these leads to similar conclusions.

Training effect

Each participant completed 0, 1, or 2 training periods. We created a variable measuring the time since last training. A histogram of this variable is shown in Supplementary Figure 4, and produced with `create_figure_S4()`. The peak at zero corresponds to the timepoints immediately after a training session, and the long tail to the right corresponds to the follow-up timepoints.

```r
create_figure_S4(memdat_long)
```



```r
ggsave("figures/years_since_last_training.png",
       width = 16, height = 6, units = "cm")
```

We hypothesized that completing a training session might lead to an immediate change (most likely an increase) in the cognitive ability of interest, and furthermore that the effect of training might change (most likely decrease) as time since training increases. We modeled this with an exponential function of the form

$$b_t \exp\left[-l_t \Delta t\right],$$

where again d_1 and d_2 are dummy variables for the age groups and Δt is the time since last training. The term b_t is defined as

$$b_t = \left(\beta_{t11} d_1 + \beta_{t21} d_2\right) d_{t1} + \left(\beta_{t12} d_1 + \beta_{t22} d_2\right) d_{t2},$$

where d_1 and d_2 are dummy variables for age groups “Young” and “Older”, and d_{t1} and d_{t2} are dummy variables for first and second training. Next, the term l_t is

$$l_t = \lambda_1 d_1 + \lambda_2 d_2,$$

where d_1 and d_2 are defined as before, and hence λ_1 and λ_2 are the exponential decay parameters in the groups “Young” and “Older”, respectively.

The reason for writing the model in this two-step form is that the mathematical formulation then gets very similar to the R syntax.

Putting it all together, for a participant in group “Young” the contribution is

$$\left(\beta_{t11} d_{t1} + \beta_{t12} d_{t2}\right) \exp\left(-\lambda_1 \Delta t\right)$$

and for a participant in the “Older” group it is

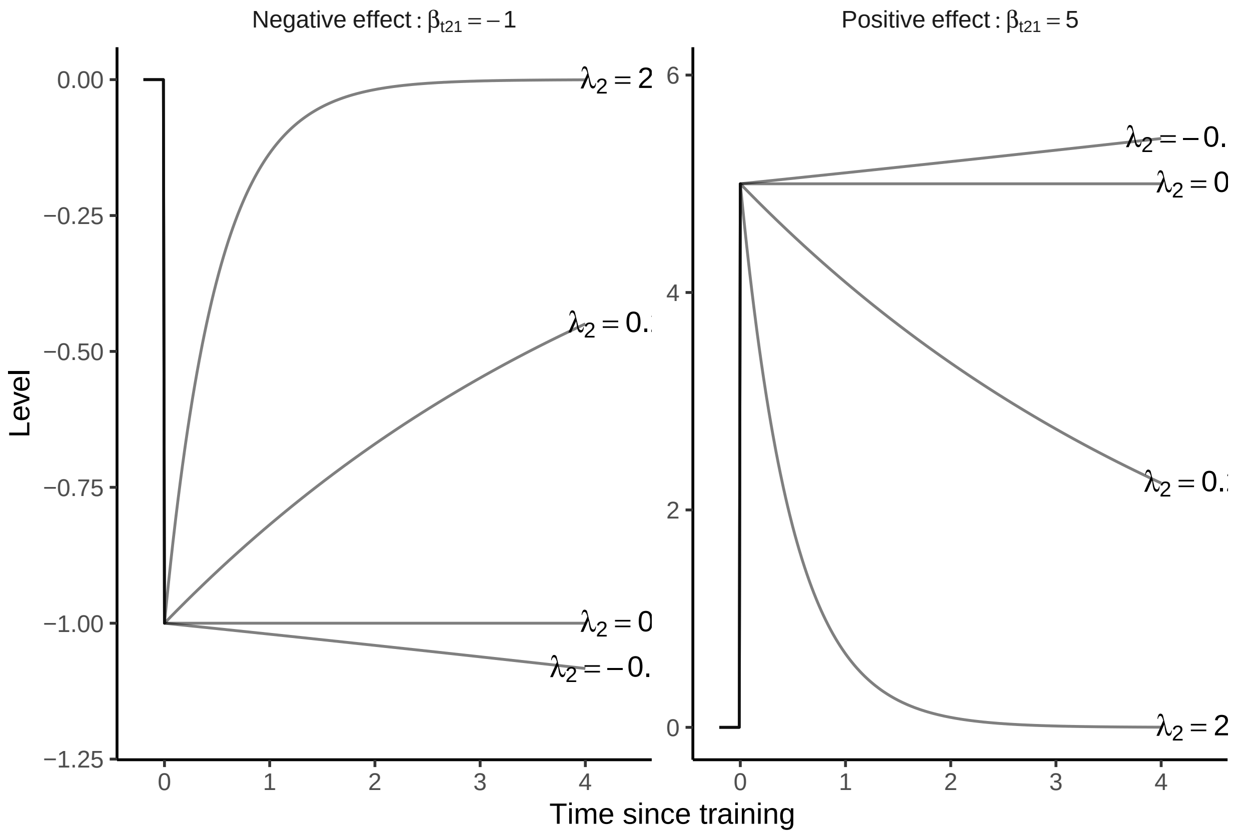
$$\left(\beta_{t21} d_{t1} + \beta_{t22} d_{t2}\right) \exp\left(-\lambda_2 \Delta t\right).$$

For ease of presentation, we will focus on the “Older” group, keeping in mind that the effects will be estimated for each age group. The coefficient β_{t21} captures the initial effect of the first training ($d_{t1} = 1, d_{t2} = 0$), since when $\Delta t = 0$ and $d_{t2} = 0$, then

$$\left(\beta_{t21} d_{t1} + \beta_{t22} \times 0\right) \exp\left(-\lambda_2 \times 0\right) = \beta_{t21} \times 1 = \beta_{t21}.$$

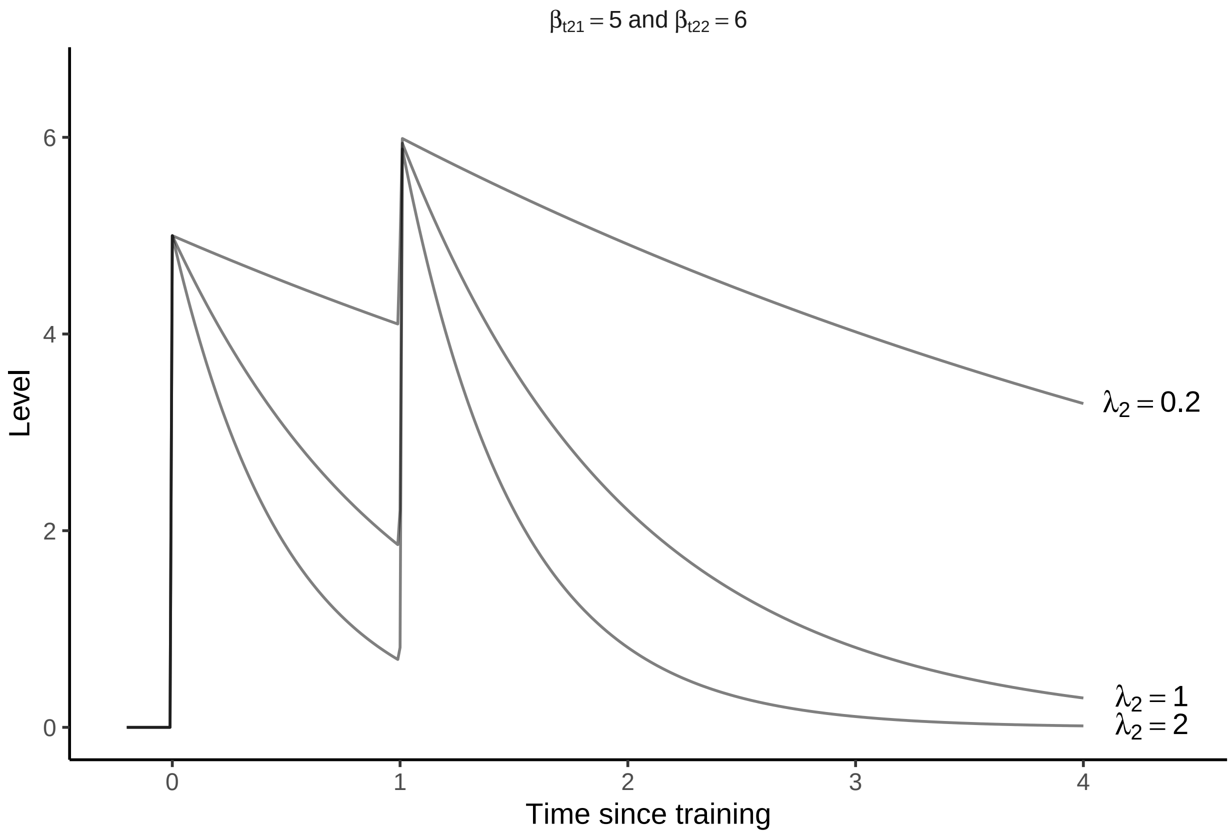
The coefficient λ_2 describes how the effect of training prevails as time since the training increases. We illustrate this for a few scenarios in the plot below, focusing on the effect of the first training. Note that this are purely hypothetical values intended to aid in interpreting model parameters. On the y-axis, 0 denotes the baseline level, and we assume that a training has been completed at time 0. Note that negative β_{t21} (top plot) means that the training effect is negative, and that negative λ_2 means that the effect gets further and further away from the baseline level: while this is unrealistic, such effects are not a priori ruled out by the model. Focusing on positive β_{t21} (bottom two plots), a large value of λ_2 means that the training effect disappears fast, while a λ_2 value close to zero indicates that the training effect is retained for a long period.

```r
illustrate_training_effect_parameters(trainings = 1L)
```



We can also visualize the effect of two trainings. The figure below shows this for $\beta_{t21} = 5$ and $\beta_{t22} = 6$, and with training sessions completed both at time 0 and at time 1.

```
illustrate_training_effect_parameters(trainings = 2L)
```



Retest effect

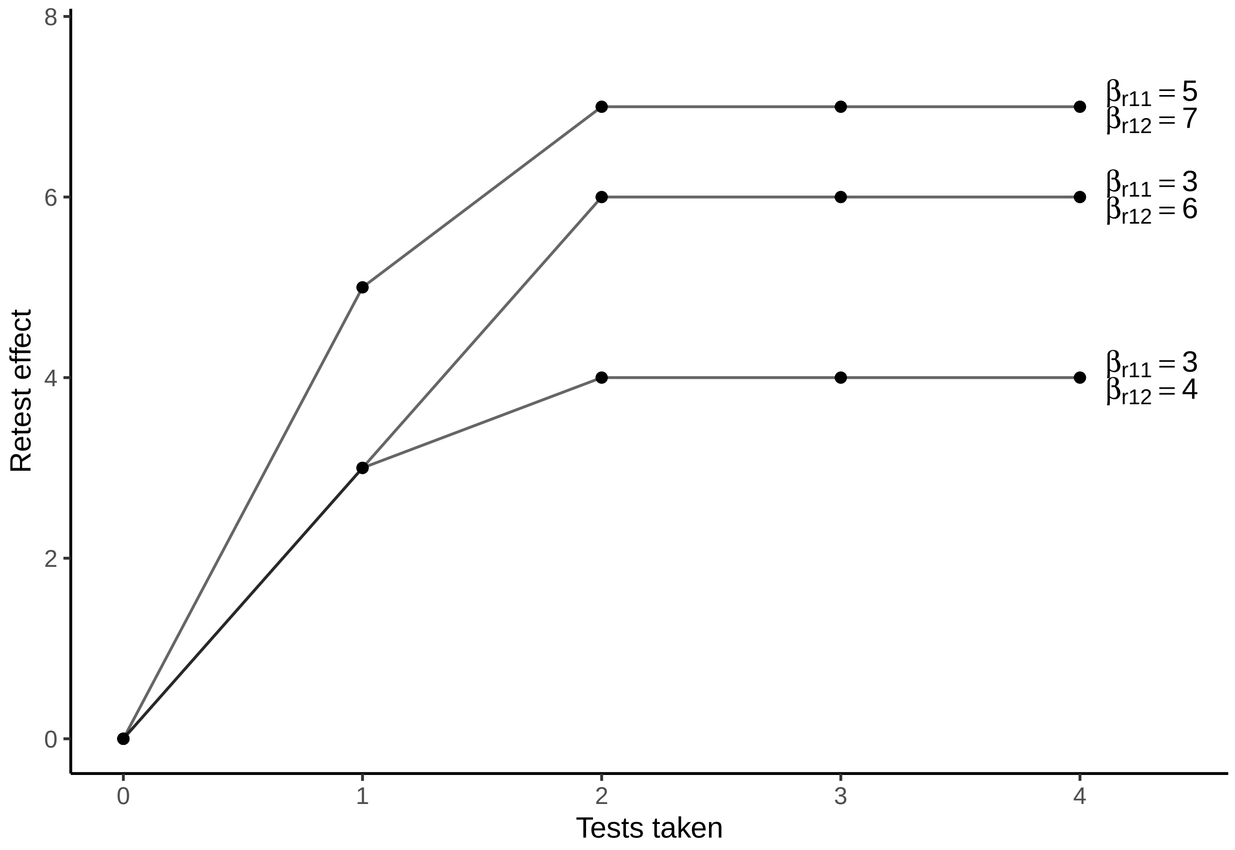
We assumed that the retest effect levels off after the first few times the participant has taken the test. We do this by defining the model term

$$b_r = (\beta_{r11} d_1 + \beta_{r21} d_2)\, d_{r=1} + (\beta_{r12} d_1 + \beta_{r22} d_2)\, d_{r\geq 2}$$

where as before d_1 and d_2 are dummy variables for age groups "Young" and "Older". Furthermore, $d_{r=1}$ is a dummy variable for the event that the participant has taken the test exactly once before, and $d_{r\geq 2}$ is a dummy variable for the event that the participant has taken the test two or more times before. β_{r11} is the retest effect for the first test in age group "Young", β_{r21} is the retest effect for the first test in age group "Older", β_{r12} is the retest effect for two or more tests in age group "Young", and β_{r22} is the retest effect for two or more tests in age group "Older".

This is a piecewise linear model, as illustrated for some combinations of β_{r11} and β_{r21} in the plot below. We also tried a model in which the effect of two or more retests was split into the effect of exactly two retests and the effect of three or more retests, but this model gave quite similar estimates for two or three retests, so we keep the maximum complexity as described here.

```
illustrate_retest_effect(knots = 2L)
```

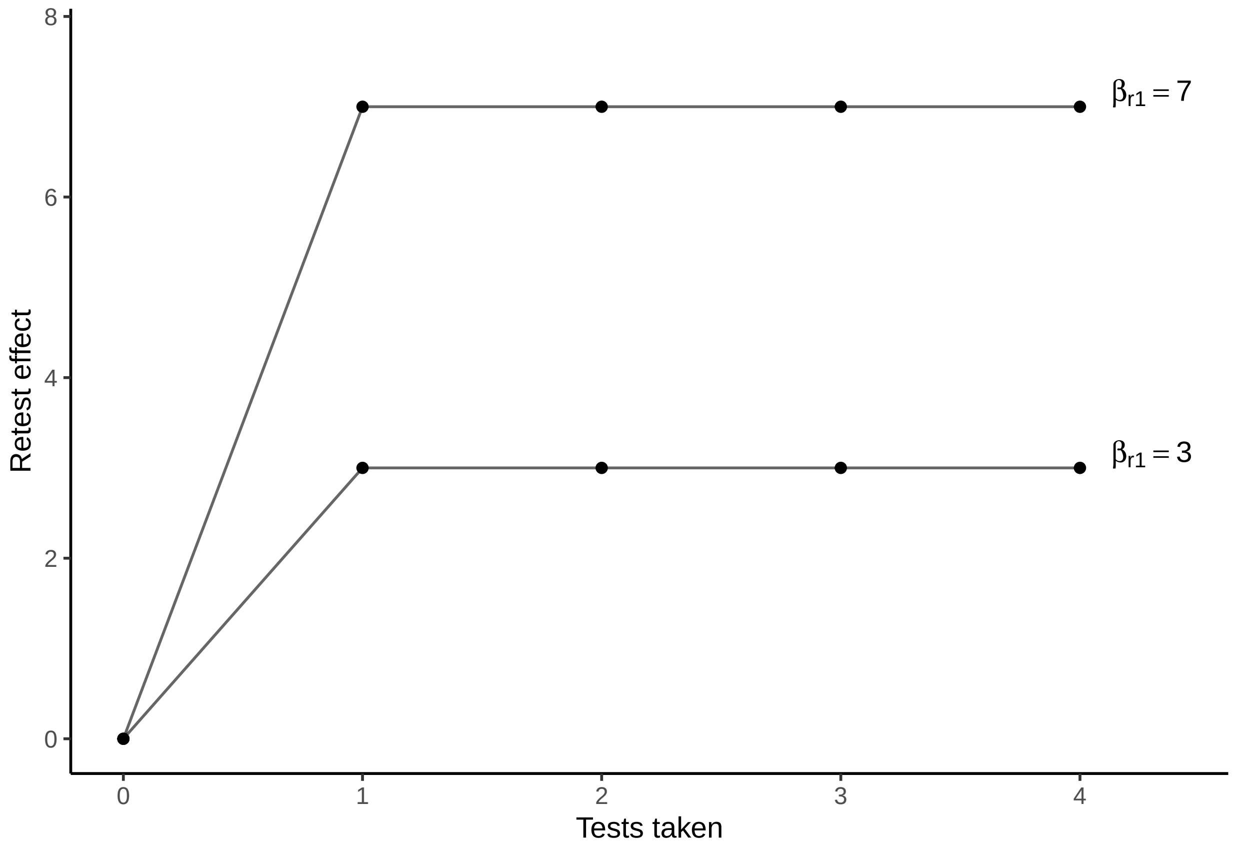


Actually, after model comparisons described below we ended up using a model with a retest effect on the form

$$b_r = (\beta_{r1} d_1 + \beta_{r2} d_2)\, d_{r \geq 1},$$

where β_{r1} is the effect of having taken the tests once or more before in the "Young" group, β_{r2} is the same for the "Older" group, and $d_{r \geq 1}$ is a dummy variable for the event that the participant has taken the test once or more before. This simplified model is illustrated in the plot below.

```
illustrate_retest_effect(knots = 1L)
```



Other covariates

The final term of the model accounts for possible offset effects of sex and for differing intercepts between each age group.

$$b_0 = \beta_{01} d_1 + \beta_{02} d_2 + \beta_m d_m,$$

where d_1 and d_2 are the dummy variables for age group as used before, β_{01} is the intercept in age group “Young”, β_{02} is the intercept in the age group “Older”, d_m is a dummy variable for being male, and β_m is the difference between males and females.

Model comparison

Putting it all together, the model described in the previous section becomes

$$E(y) = b_0 + b_a + b_t \exp\left(-l_t \Delta t\right) + b_r,$$

where $E(y)$ denotes the expected value of y given a person’s value of the predictor variables. The noise terms consist of a systematic part (random effects), which captures the fact that repeated measures of the same individual are correlated, and a residual part which captures each participant’s random deviations around her/his own curve. Using `nlme`, the full model can be fitted in R with the following code:

```r
fit_mod_full <- function(memdat_long){
  nlme(
    # hwt_z is Z-transformed HundredWords_Total
    model = hwt_z ~ b0 + ba +
      bt * exp(-lambda * time_since_training) + br,
    # take dataset containing only participants with more than one measurement
    data = memdat_long,
    fixed = list(
      # define an intercept per age group and a Male-vs-Female difference
      b0 ~ 0 + SexMale + age_group,
      # aging effect, estimate separately per age group
      ba ~ 0 + age_group:age_dev,
      # immediate training effect of the first and then second training
      bt ~ 0 + age_group:trainings_dummy1 +
        age_group:trainings_dummy2,
      # exponential decay of training effect, per age group
      lambda ~ 0 + age_group,
      # retest effect per age group and for 1 and for >=2
      br ~ 0 + age_group:retests_dummy1 +
        age_group:retests_dummy2om
    ),
    # random intercept
    random = b0 ~ 1,
    # define the grouping variable corresponding to participant id
    groups = ~ CrossProject_ID,
    start = c(
      b0 = c(0, -.9, -.1), ba = c(0, 0), bt = rep(1, 4),
      lambda = c(0, 0), br = rep(.2, 4)
    )
  )
}
```

This model may be unnecessarily complex. To this end, we tested a number of simplifications to the model terms. See the function `run_candidate_models()` for all details.

NOTE: When running this function on simulated data, some models might fail to converge, and a message will be printed. However, unless all models fail, the code below will still work.

```
# Run all candidate models
models <- run_candidate_models(memdat_long)
```

```
## All models ran successfully
```

We compared the models in terms of AIC and BIC. Such comparison of fixed effects is justified because all models were estimated with maximum likelihood.

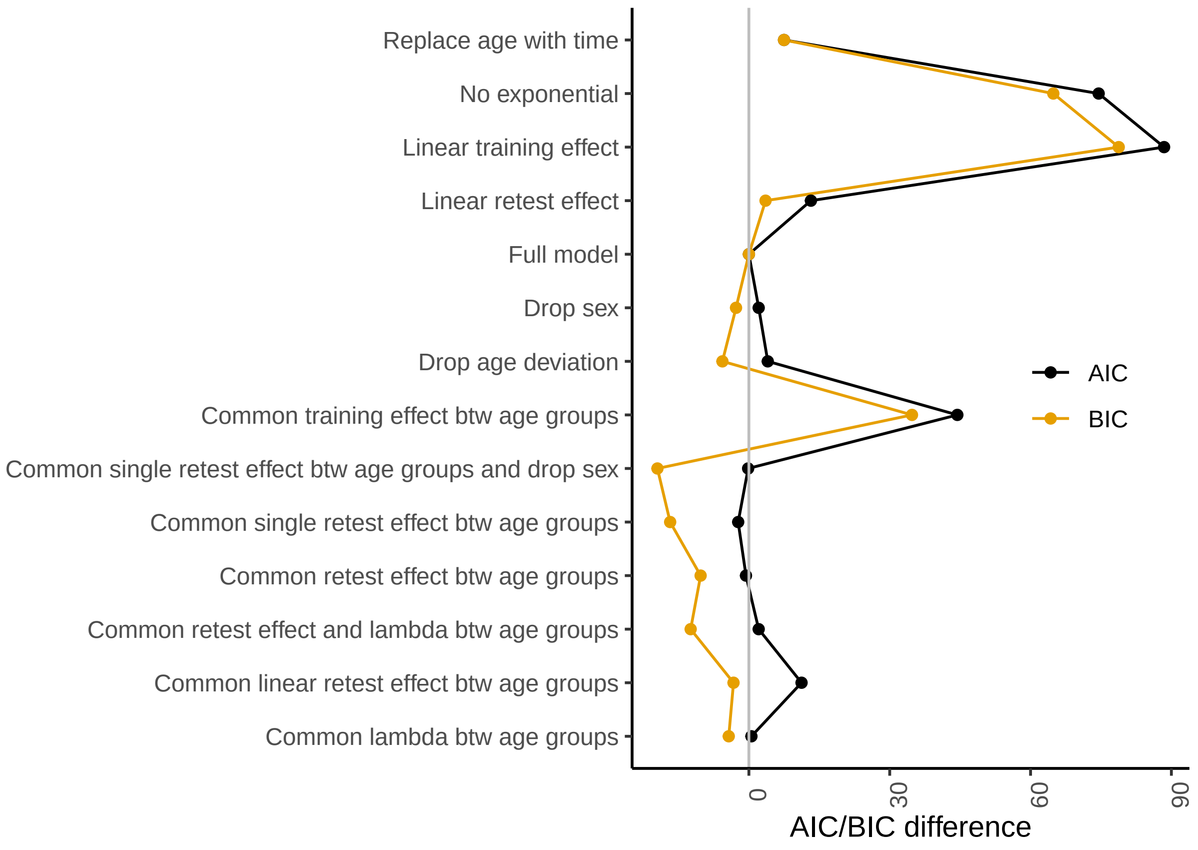
```
response <- "hwt_z"
comp_df <- compare_models(models)
knitr::kable(comp_df, row.names = FALSE, digits = 0)
```

Model	df	AIC	BIC	logLik
Common lambda btw age groups	16	1693	1771	-831
Common linear retest effect btw age groups	14	1704	1772	-838
Common retest effect and lambda btw age groups	14	1695	1763	-833
Common retest effect btw age groups	15	1692	1765	-831
Common single retest effect btw age groups	14	1691	1758	-831
Common single retest effect btw age groups and drop sex	13	1693	1755	-833
Common training effect btw age groups	15	1737	1810	-854

Model	df	AIC	BIC	logLik
Drop age deviation	15	1697	1769	-833
Drop sex	16	1695	1772	-831
Full model	17	1693	1775	-829
Linear retest effect	15	1706	1778	-838
Linear training effect	15	1781	1854	-876
No exponential	15	1767	1840	-869
Replace age with time	17	1700	1782	-833

A comparison of all the models tested is shown in the figure below. The full model is defined as the reference level, and the difference to the full model in terms of the Akaike information criterion (AIC) and Bayesian information criterion (BIC) are plotted. These criteria use slightly different assumptions, but when comparing two or more models, the one with the *lowest* value is likely to generalize best to predict the response value on new data. That is, lower AIC/BIC indicates that the model strikes the right balance between being sufficiently complex to capture the real effects and being simple enough to avoid overfitting.

```
plot_model_comparison(comp_df)
```



A definition of all the models referred to in the plot is given in the table below.

Model	Modification of full model
Linear training effect	Replace the dummy variables for one or two trainings completed with a linear regression slope for number of trainings completed. Thus, b_t becomes $b_t = (\beta_{t1} d_1 + \beta_{t2} d_2) n_t$, where n_t is the number of trainings completed and β_{t1} and β_{t2} are the slopes in each age group.
No exponential	Remove the whole exponential term $\exp(-l_t \Delta t)$ from the training effect b_t.
Common training effect btw age groups	Assume the training effect is equal between age groups, i.e., $b_t = \beta_{t1} d_{t1} + \beta_{t2} d_{t2}$, where β_{t1} is the effect of the first training and β_{t2} is the effect of the second training.
Linear retest effect	Replace dummy variables for retest effect with a linear regression slope for number of tests taken, i.e., $b_r = (\beta_{r1} d_1 + \beta_{t2} d_r) n_r$, where n_r is the number of tests taken, and β_{r1} and β_{r2} are the slopes in each age group.
Common linear retest effect btw age groups	Replace dummy variables for retest effect with a linear regression slope for number of tests taken and use a single slope common for both age groups.
Replace age with time	Use time since baseline rather than age deviation from age group mean, i.e., $b_a = (\beta_{a1} d_1 + \beta_{a2} d_2) \Delta t_b$, where Δt_b is time since baseline.
Drop sex	Remove offset effect for Sex, i.e., $b_0 = \beta_{01} d_1 + \beta_{02} d_2$.
Drop age deviation	Remove the aging effect b_a.
Common lambda btw age groups	Assume the exponential decay parameter is identical for both age groups, i.e., $l_t = \lambda$.
Common retest effect and lambda btw age groups	Combination of modifications applied by models "Common retest effect btw age groups" and "Common lambda btw age groups".
Full model	No modification.
Common single retest effect btw age groups and drop sex	Combination of modifications applied by models "Common single retest effect btw age groups" and "Drop sex".
Common retest effect btw age groups	Retest effect identical between age groups, i.e., $b_r = \beta_{r1} d_{r=1} + \beta_{r2} d_{r \geq 2}$, where β_{r1} is the effect of the first test and β_{r2} is the effect of two or more tests.
Common single retest effect btw age groups	Retest effect only for *one or more* test and common across age groups, and sex not included. That is, $b_r = \beta_r d_{r \geq 1}$ where β_r is the effect of having taken the test once or more before.

Model parameters

We define the chosen model as `mod` and show its parameter estimates:

```r
mod <- models[["Common single retest effect btw age groups"]]
tabulate_memory_parameters(mod, memdat)
```

Age group	Variable	Estimate (95% CI)	p-value
Young	Intercept	18.5 (16.7, 20.4)	0.176

Age group	Variable	Estimate (95% CI)	p-value
Young	Age slope	-0.08 (-0.52, 0.36)	0.718
Young	1st training	17.6 (15.7, 19.5)	2.2e-61
Young	2nd training	20.6 (18.3, 22.8)	1.4e-59
Young	Exp. decay	4.41 (2.87, 5.96)	3.7e-08
Older	Intercept	9.69 (8.12, 11.3)	4.8e-33
Older	Age slope	-0.54 (-0.90, -0.17)	0.005
Older	1st training	7.23 (5.76, 8.70)	1.4e-20
Older	2nd training	8.11 (6.42, 9.80)	8.7e-20
Older	Exp. decay	1.86 (0.17, 3.55)	0.032
Common effect	Male - female offset	-1.85 (-3.64, -0.07)	0.043
Common effect	>=1 retest	2.80 (1.50, 4.09)	2.8e-05

Temporal extent of the training effect

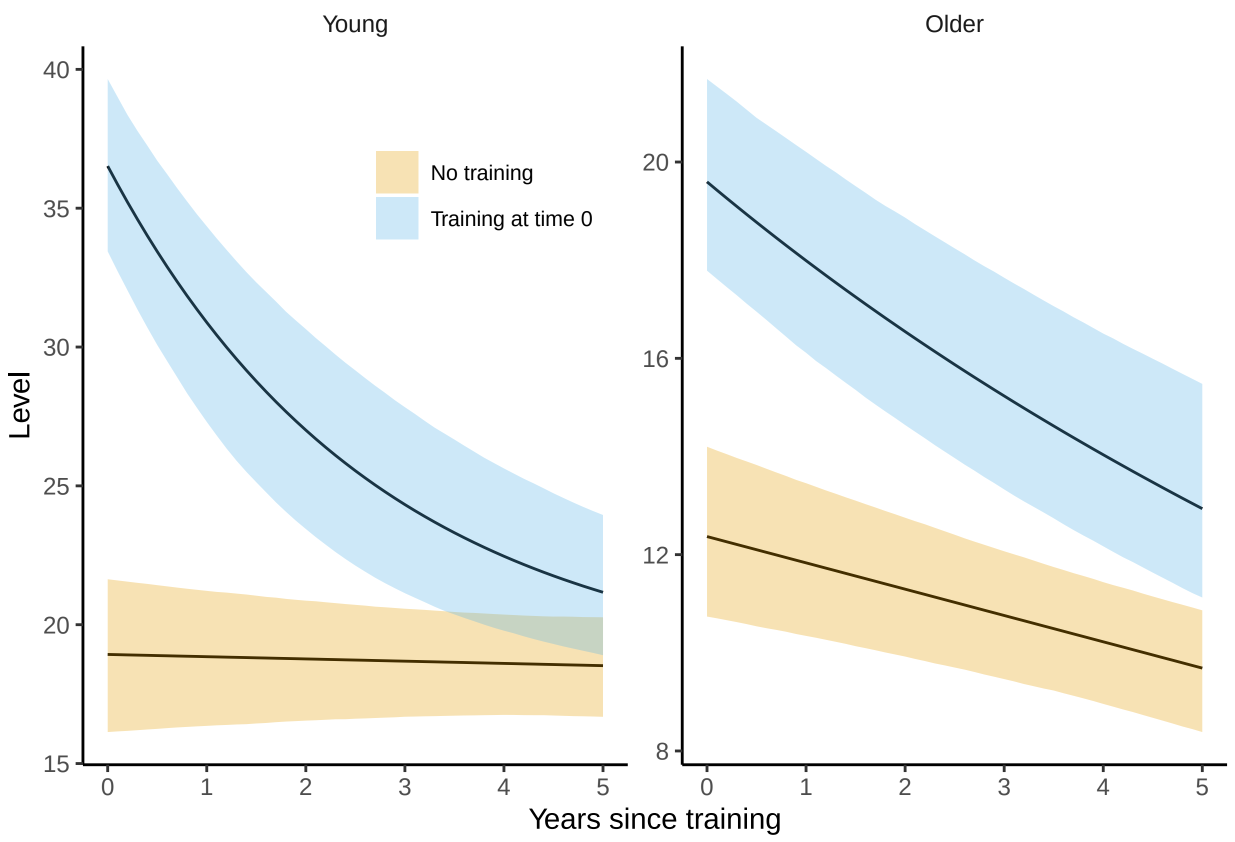
In order to show the estimated extent of the training effect, we first need to compute bootstrap confidence intervals. This is achieved with the following code. Note that in order to be reliable, the number of bootstrap samples should be at least in the thousands.

```r
grids <- boot_create_grid()
nresamp <- params$bootstrap_samples # Number of bootstrap samples. Was 10000 in actual analyses.

if(params$rerun_bootstrap){
  # Bootstrapped response values on grid
  plan(params$parallel_strategy) # to run in parallel, set parallel_strategy to "multisession"
  bootgrid <- future_map_dfr(seq_len(nresamp), function(i){
    # Fit updated models on bootstrapped datasets
    response <- "hwt_z"
    upmod <- update(mod, data = boot_sampfun(mod, memdat_long, "hwt_z"))
    grids$grid %>%
      mutate(
        fit0 = boot_predfun(upmod, grids$grid0),
        fit = boot_predfun(upmod, grids$grid)
      )
    }, .options = furrr_options(seed = 123L), .progress = TRUE
  )
  plan("default")
  if(!dir.exists("data")) dir.create("data")
  saveRDS(bootgrid, "data/bootgrid.rds")
} else {
  bootgrid <- readRDS("data/bootgrid.rds")
}
```

The plots below show the estimated temporal extent of the training in each age group. It is assumed that a single training session has been completed at time zero, and the development of the expected level is plotted with the solid black lines with green shaded areas representing 95 % confidence intervals. The solid black lines at the bottom represent the reference level with no training, with red shaded areas corresponding to 95 % confidence intervals. The plots here are shown for Females, but Males would have identical curves except for a vertical shift along the y-axis.

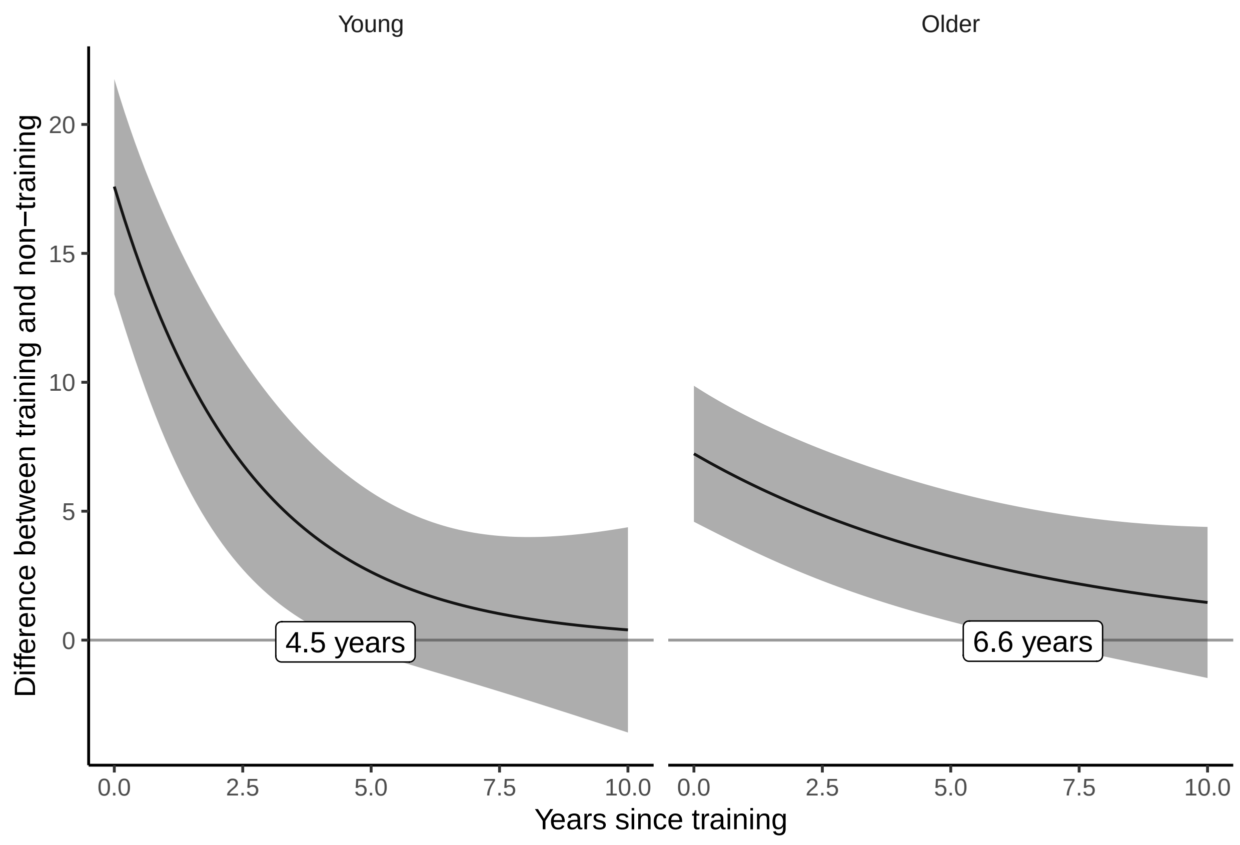
```r
create_figure_2(bootgrid, memdat, mod, grids)
```



```
ggsave("figures/temporal_training_extent.png",
       width = 16, height = 10, units = "cm")
```

Next, in the plot below we show the estimated difference between the two curves in the previous plot, with 95 % confidence intervals.

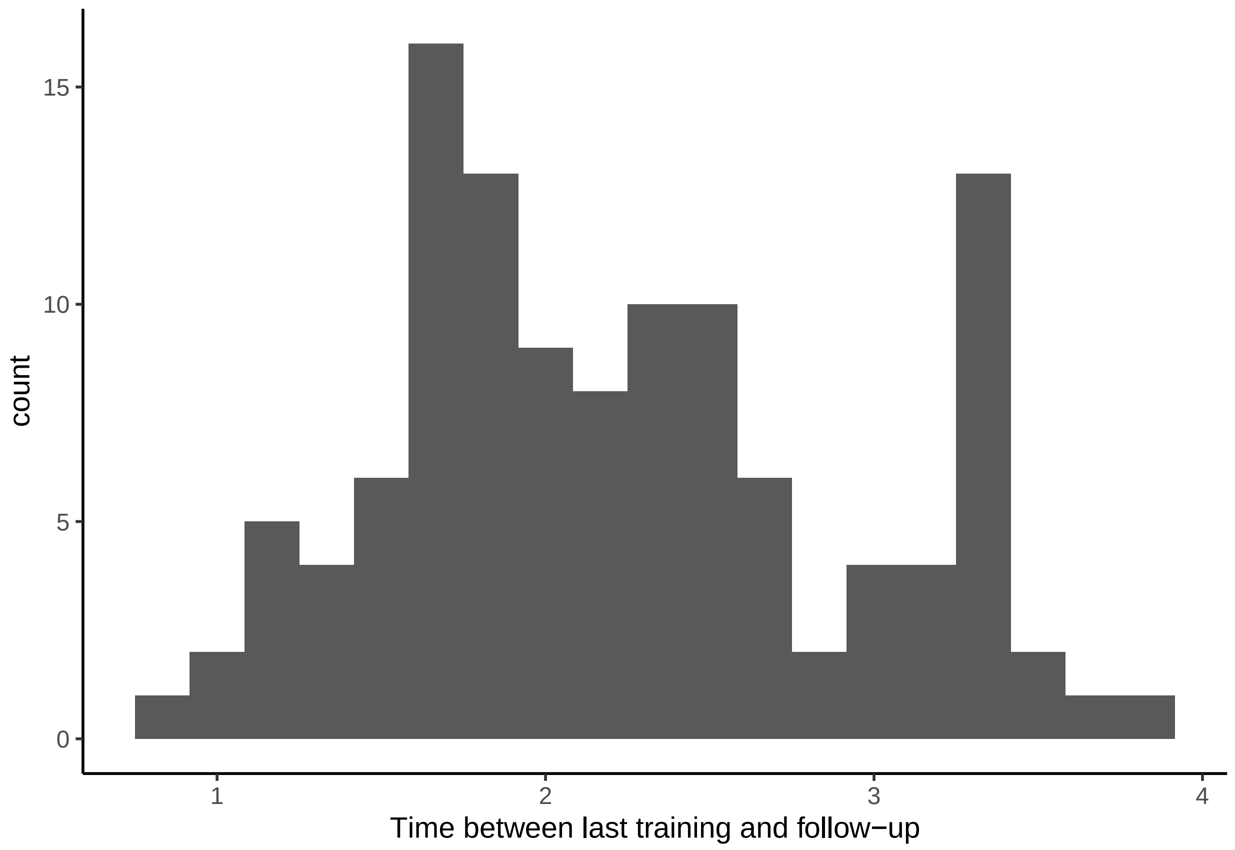
```
create_figure_S1(bootgrid, memdat, mod, grids)
```



```
ggsave("figures/temporal_training_extent_10yrs.png",
       width = 16, height = 10, units = "cm")
```

We finally create Supplementary Figure 3.

```
create_figure_S3(memdat_long)
```

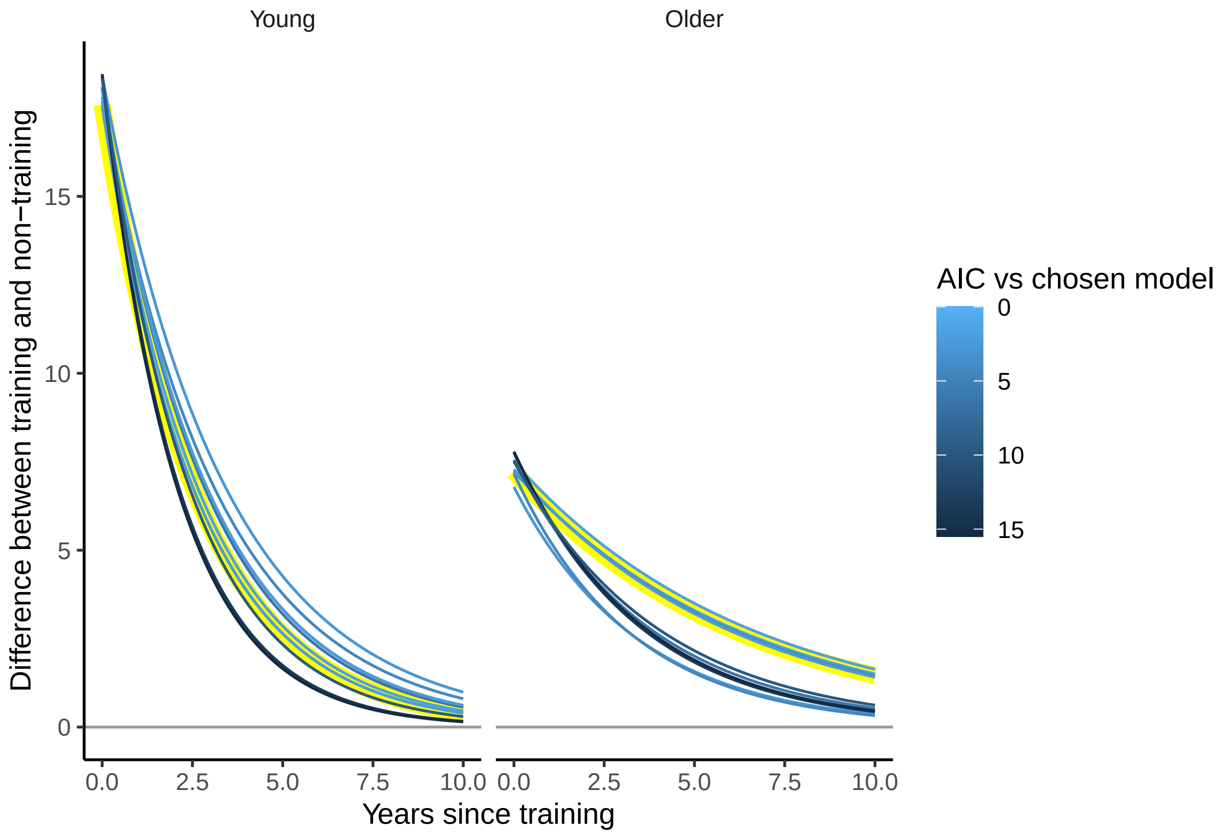


```
ggsave("figures/last_training_follow_up_intervals.png",
       width = 8, height = 6, units = "cm")
```

Robustness check

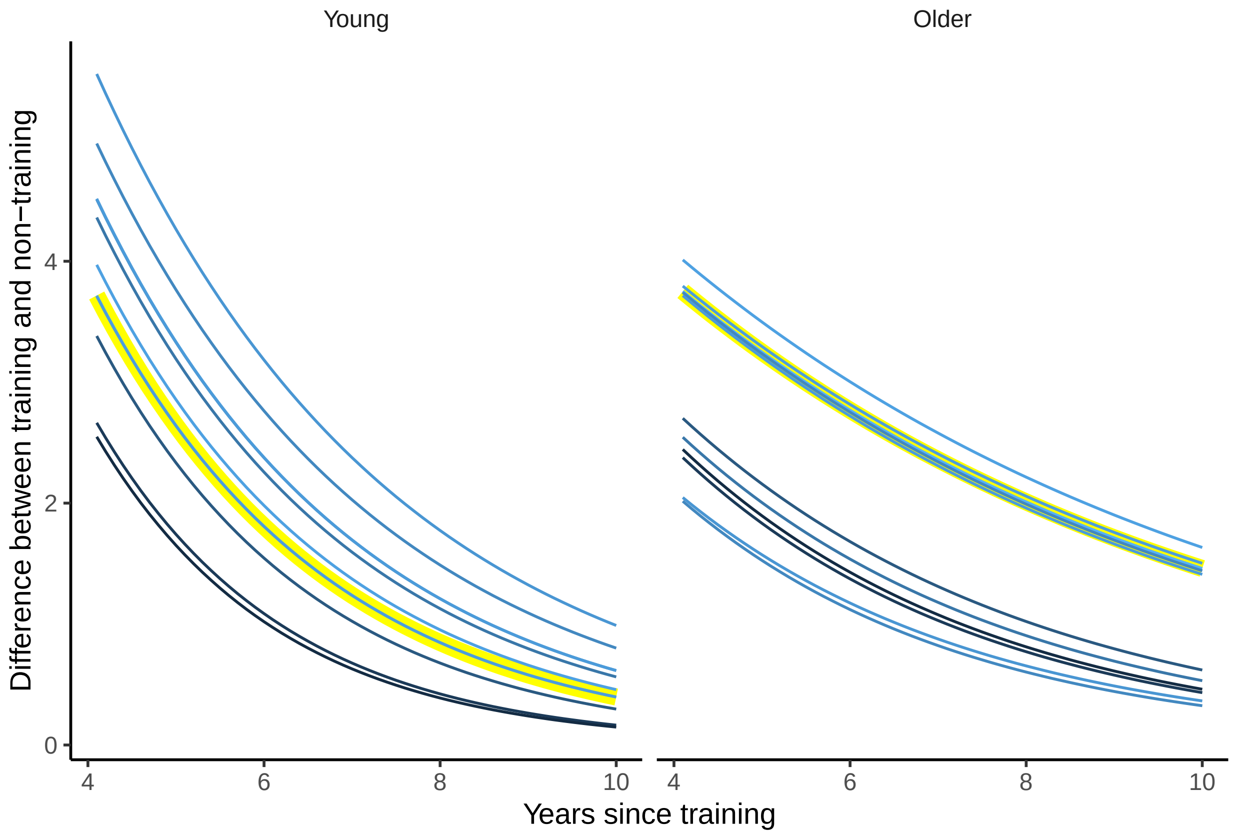
As a means of checking that the results are not very sensible to small variations in model specification, we computed the same difference between training and non-training shown in the last plot for all models described in the section on model comparison. The plot below shows the estimates for all of these models except for the three models which were far worse than the rest in terms of AIC/BIC. The color scale shows the model's AIC compared to the full model used in the model comparisons.

```
plots <- plot_robustness_check(models, grids, memdat, comp_df)
plots$plot1
```



In the above plot, it is hard to see that most models overlap the chosen model. We can highlight this by zooming in on a particular time range, as shown below for times between 4 and 5 years after training. We take this to support that the estimated temporal extent of the training effect does not depend sensitively on the particular choices made in the model selection.

```
plots$plot2
```

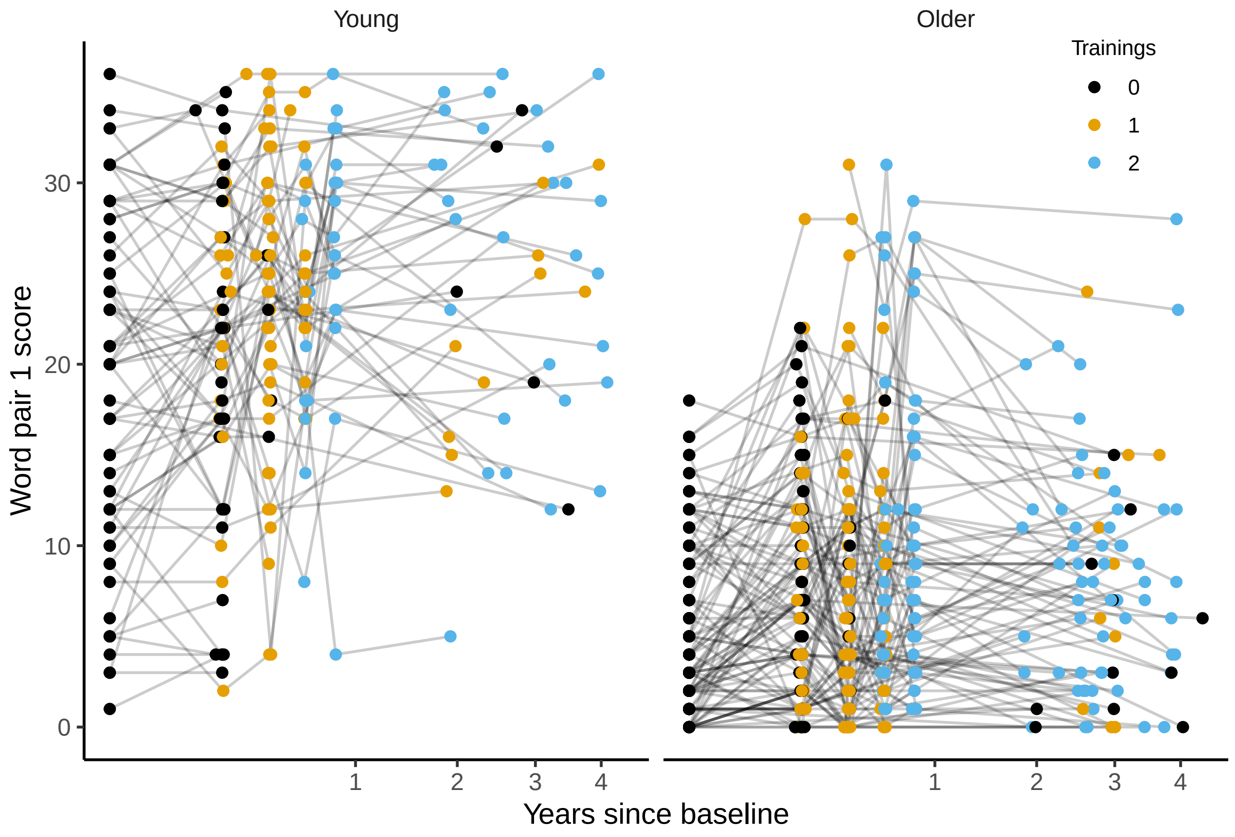


Section 2.2

This section contains code for reproducing the results in the section titled “Is there a transfer effect on a non-trained memory task, and if so, does it relate to the 100-words test performance, and does it persist after cessation of intervention?”.

The function `create_figure_3` reproduces Figure 3.

```
create_figure_3(memdat_wp_long)
```



```r
ggsave("figures/transfer_task.png",
  width = 16, height = 10, units = "cm")
```

The same model as for 100-words was estimated, but now with WordPair1 as outcome.

```r
mod <- fit_mod_common_single_retests(memdat_wp_long, response = "wp1_z")
```

We next show the estimated model parameters.

```r
tabulate_memory_parameters(mod, memdat_wp, response = "WordPair_1")
```

Age group	Variable	Estimate (95% CI)	p-value
Young	Intercept	19.5 (17.8, 21.2)	1.6e-13
Young	Age slope	-0.13 (-0.54, 0.29)	0.547
Young	1st training	1.89 (0.35, 3.42)	0.017
Young	2nd training	3.94 (1.79, 6.09)	3.8e-04
Young	Exp. decay	-0.30 (-2.87, 2.27)	0.818
Older	Intercept	6.22 (4.78, 7.66)	4.3e-18
Older	Age slope	-0.58 (-0.92, -0.25)	8.0e-04
Older	1st training	0.37 (-0.86, 1.60)	0.560
Older	2nd training	2.36 (0.78, 3.95)	0.004
Older	Exp. decay	1.66 (-3.24, 6.56)	0.510
Common effect	Male - female offset	-1.94 (-3.63, -0.26)	0.025
Common effect	>=1 retest	1.88 (0.84, 2.93)	4.9e-04

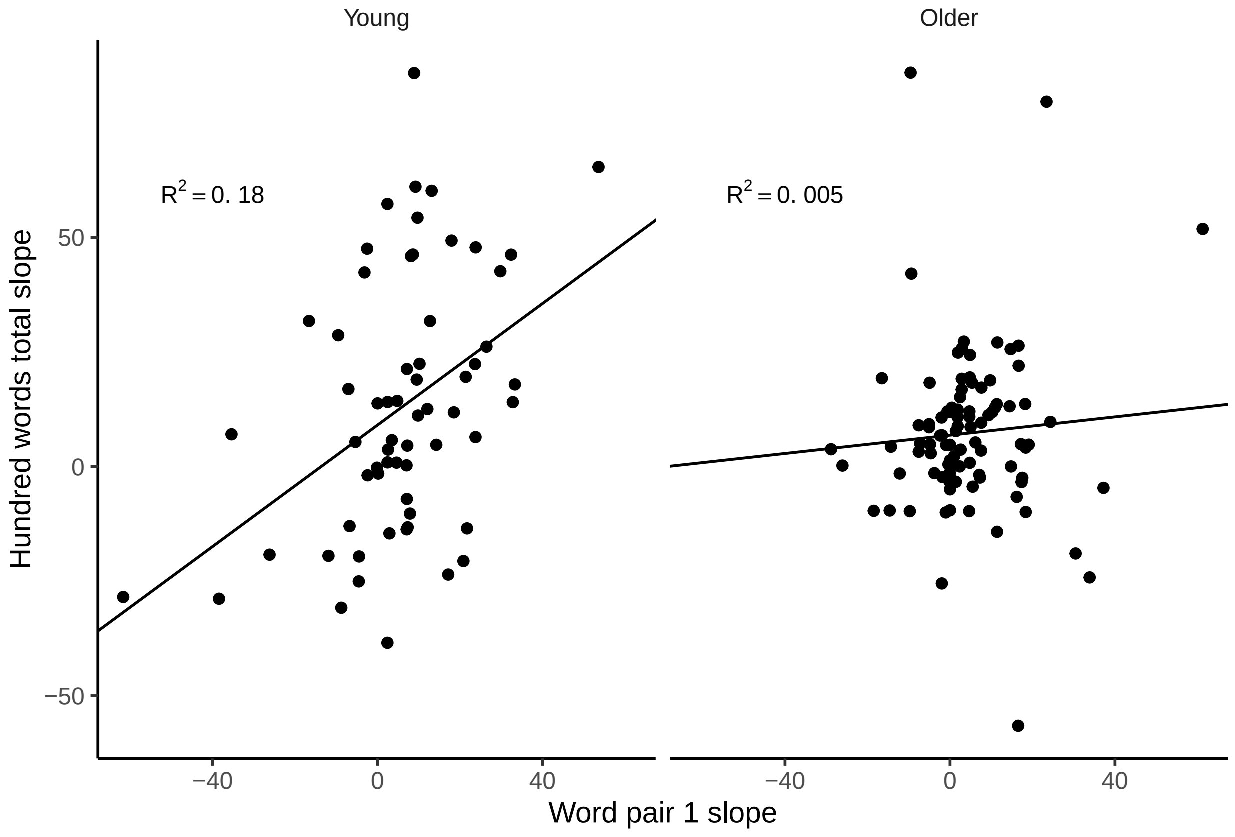
Correlated change in WP1 and 100-words

We next investigated the correlated change in WP1 and 100-words during the training period. We first fit individual regression models for 100-words and WP1 as a function of time.

```
slopedat <- fit_individual_regression_models(
  corrdat,
  formula1 = WordPair_1 ~ time,
  formula2 = HundredWords_Total ~ time + retests_dummy1om,
  mod1_name = "wp1_slope",
  mod2_name = "hwt_slope"
  )
```

Based on the results we create Figure 4.

```
create_figure_4(slopedat)
```



```
ggsave("figures/transfer_learning_slopes.png",
       width = 16, height = 10, units = "cm")
```

Next, the slope correlations were estimated separately in each age group. The table below shows correlation estimates with 95% confidence intervals.

```
compute_correlation_estimates(slopedat,
                              slope1 = "wp1_slope",
                              slope2 = "hwt_slope")
```

Age group	What	Correlation
Young	Slope correlation	0.42 (0.19, 0.61)

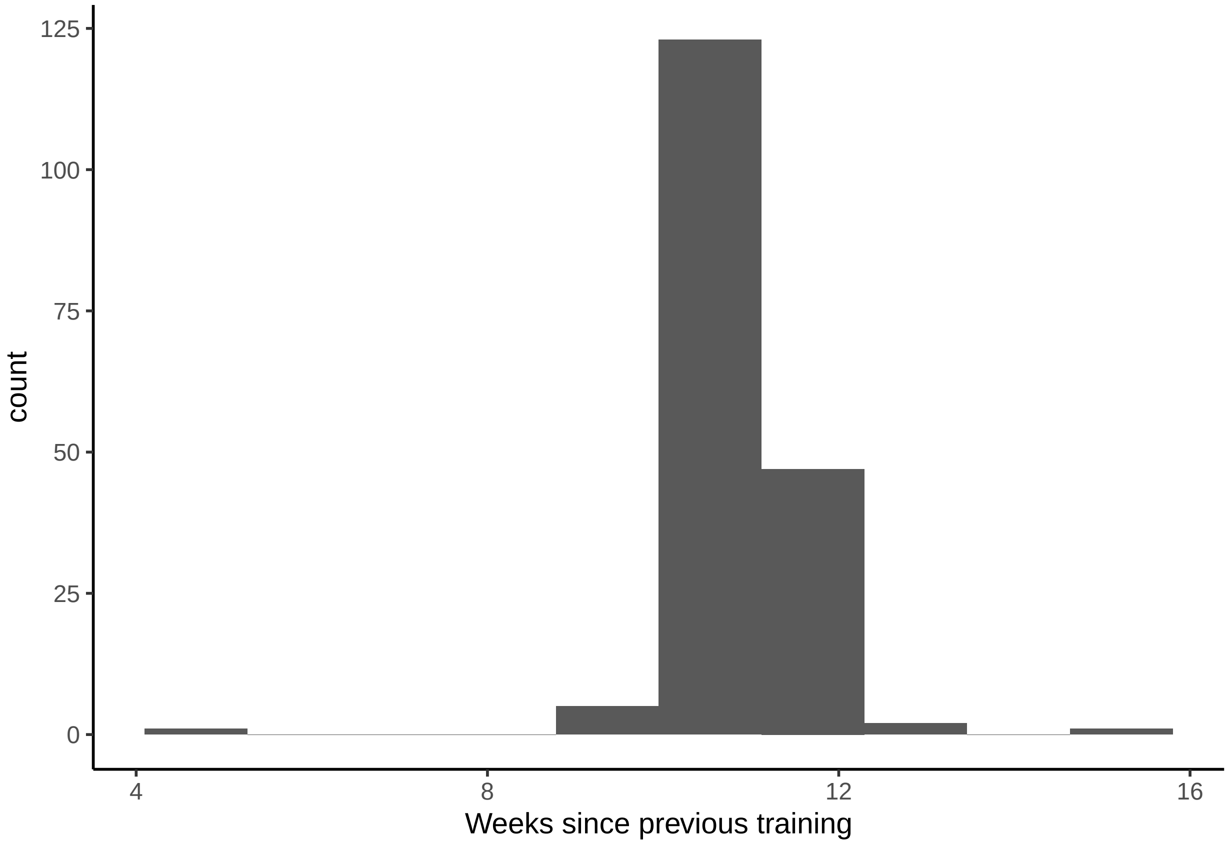
Age group	What	Correlation
Older	Slope correlation	0.07 (-0.13, 0.27)

Section 2.3

This section contains code used to produce the results in the section titled “Does the training intervention lead to an increase or maintenance of hippocampal volume relative to that observed with no training, and are such effects observed years after intervention?”.

We start by creating Supplementary Figure 2.

```
create_figure_S2(hippodat_long)
```



```
ggsave("figures/follow_up_intervals.png",
       width = 8, height = 6, units = "cm")
```

The following linear mixed model was fit to estimate the effect of training on hippocampal volume.

```
mod <- fit_hippocampus_model(hippodat_long)
```

We tabulate the model estimates.

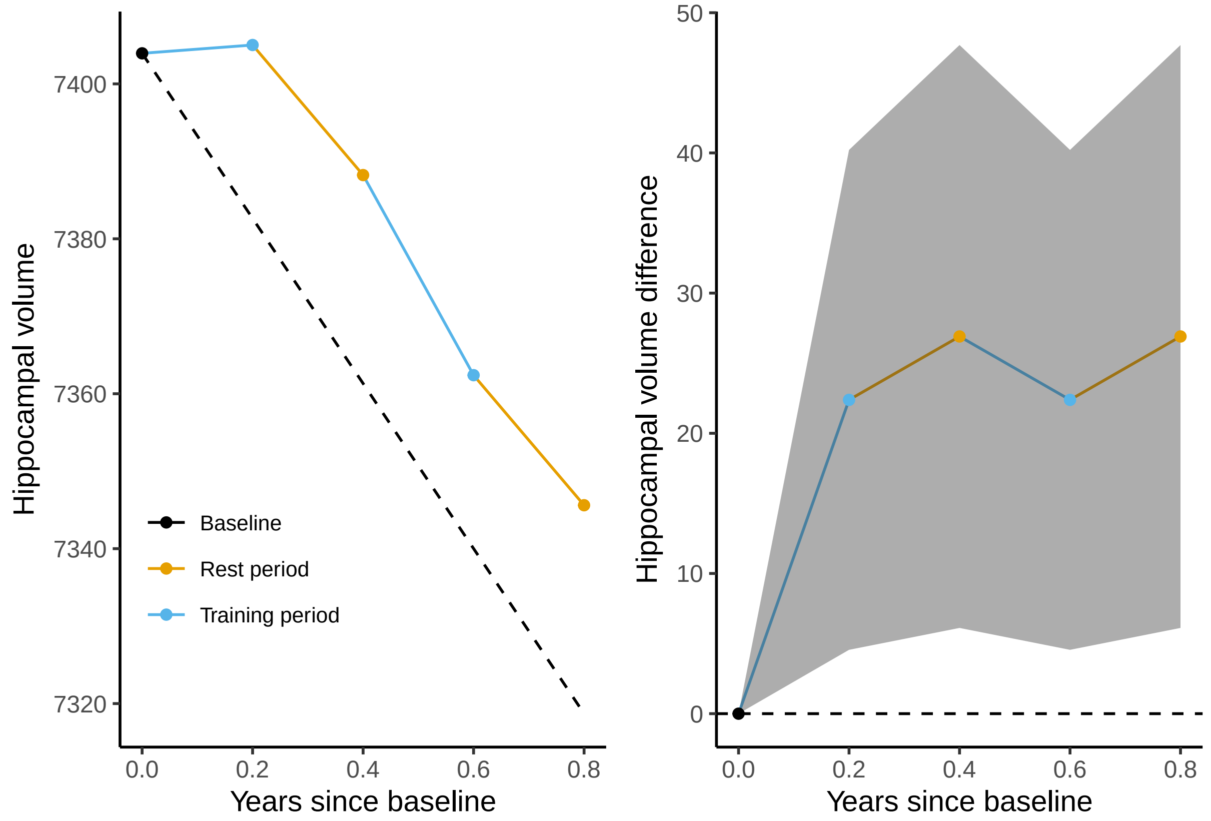
```
tabulate_hippocampus_model_results(mod$lme)
```

Parameter	Description	Estimate (95% CI)	p-value
Xage_groupYoung	Young intercept	8476 (8347, 8605)	3.1e-247

Parameter	Description	Estimate (95% CI)	p-value
Xage_groupOlder	Older intercept	7404 (7281, 7527)	5.5e-237
Xicv_z	ICV (1 standard deviation)	321 (232, 410)	1.5e-11
XSexMale	Male-female diff	-59.3 (-243.2, 125)	0.528
Xage_groupYoung:time	Young slope	-32.6 (-43.3, -21.9)	3.7e-09
Xage_groupOlder:time	Older slope	-106.6 (-116.8, -96.3)	2.4e-74
Xage_groupYoung:trainednow	Young trained now offset	10.3 (-14.0, 34.6)	0.408
Xage_groupOlder:trainednow	Older trained now offset	22.4 (4.6, 40.2)	0.014
Xage_groupYoung:trainedprev	Young trained prev offset	28.5 (1.2, 55.8)	0.042
Xage_groupOlder:trainedprev	Older trained prev offset	26.9 (6.2, 47.6)	0.011

Next, we create Figure 5.

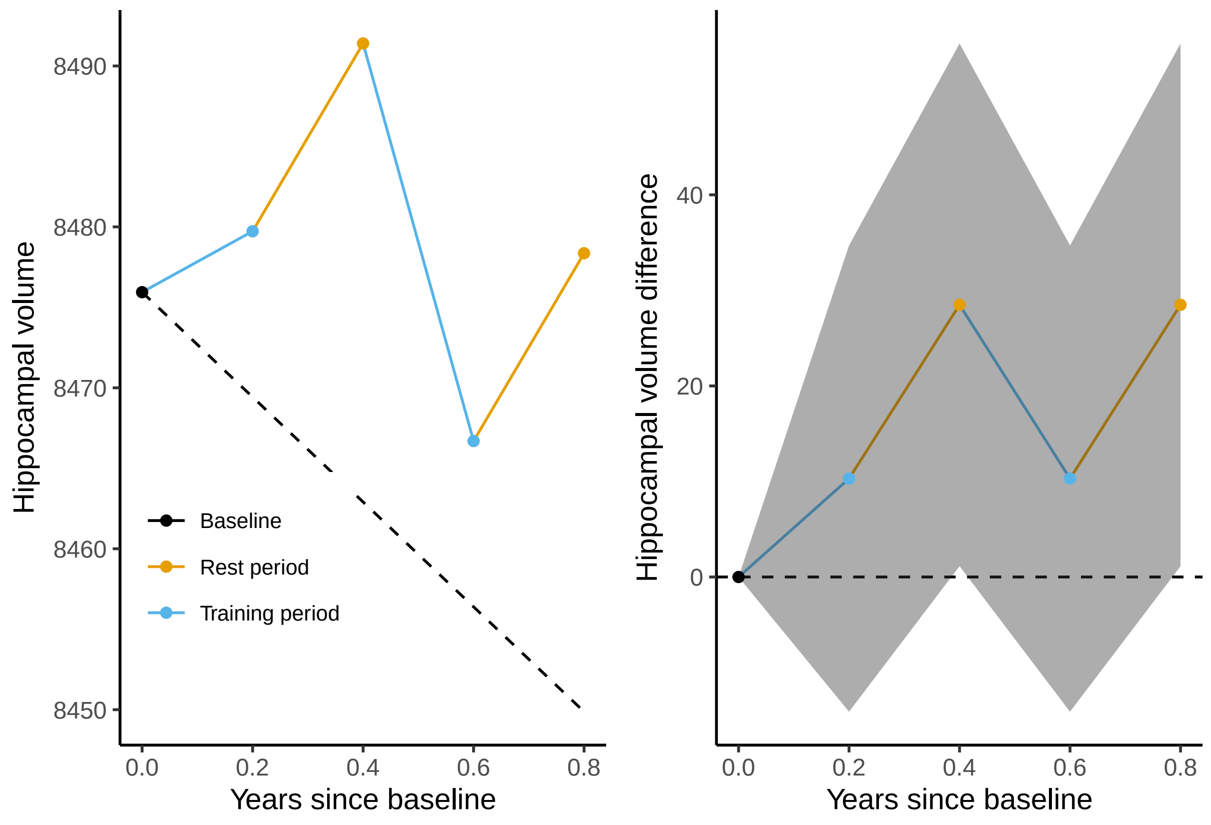
```
grids <- create_hippocampus_grid(hippodat_long)
df <- create_figure_5_data(mod, grids)
create_figure_5(df, group = "Older")
```



```
ggsave("figures/hippocampus_effect.png",
  width = 16, height = 10, units = "cm")
```

We can reproduce the corresponding figure for the young group.

```
create_figure_5(df, group = "Young")
```



Section 2.4

This section shows code for producing the results stated in the section titled "Is there a relationship between memory and hippocampal volume changes following training?".

We start by fitting individual regression models for hippocampus and 100-words as functions of time.

```r
slopedat <- fit_individual_regression_models(
  corrdat_hippo,
  formula1 = hippocampus ~ time,
  formula2 = HundredWords_Total ~ time + retests_dummy1om,
  mod1_name = "hippo_slope",
  mod2_name = "hwt_slope"
  )
```

Then we compute the correlation. For the actual data, there was one extreme outlier, with 100-words slope larger than 300 words/year. This was due to a very short time interval between two consecutive test. We removed this outlier prior to computing the correlation, and hence have the filter `hwt_slope < 300`.

```r
slopedat %>%
  filter((age_group == "Young" & hwt_slope < 300) | age_group == "Older") %>%
  compute_correlation_estimates(slope1 = "hippo_slope", slope2 = "hwt_slope")
```

Age group	What	Correlation
Young	Slope correlation	-0.08 (-0.33, 0.17)
Older	Slope correlation	0.03 (-0.17, 0.23)

Correlation between hippocampal volume slope during training and overall memory performance

Next, we compute the correlation between hippocampus slope and overall memory performance.

```
df <- compute_hippo_slope_overall_memory_fits(corrdat_hippo2)
compute_hippo_slope_overall_memory_correlation(df)
```

Age group	What	Correlation
Young	Pearson correlation	0.16 (p=0.348, CI: [-0.18, 0.46])
Older	Pearson correlation	0.01 (p=0.963, CI: [-0.25, 0.26])

Correlation between hippocampal volume slope during training and memory at follow-up

Finally, we compute the correlation between hippocampal volume slope during training and memory performance at follow-up.

```
df <- compute_hippo_slope_memory_followup(corrdat_hippo3)
compute_hippo_slope_memory_followup_correlation(df)
```

Age group	What	Correlation
Young	Pearson correlation	-0.12 (p=0.470, CI: [-0.44, 0.21])
Older	Pearson correlation	0.12 (p=0.352, CI: [-0.14, 0.36])