Results of Logarithmic encoding model

Xiuna Zhu, Xuelian Zang & Zhuanghua Shi

24/10/2022

load packages and functions

```
source('mytheme.R')
# model version
modelversion = 'gap_log_rstan'
rstanmodelPath = 'modelrlt'
modelPath = paste0(rstanmodelPath, '/models/', modelversion)
```

Merge the model Result data

1 load all data and model results

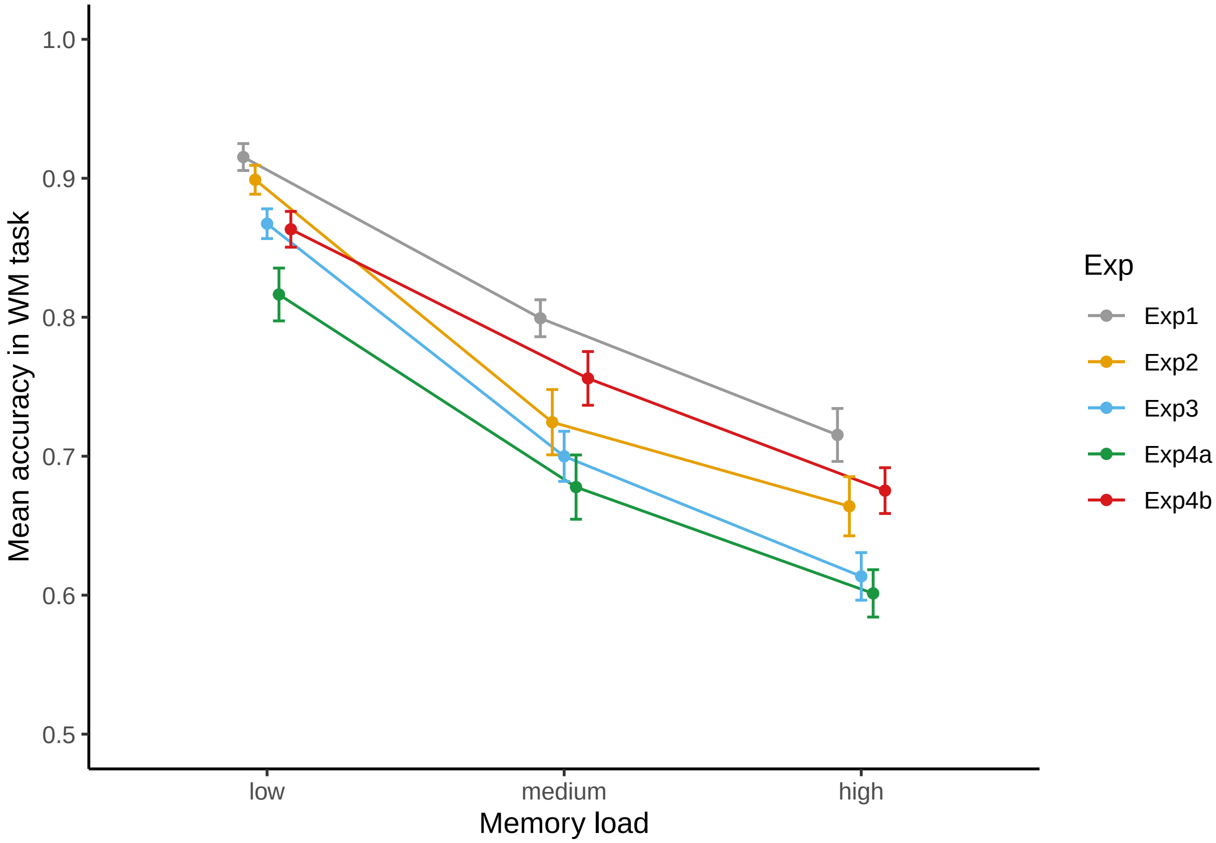
```
AllExpData = read.csv(paste0("../data/AllValidData.csv"))
dur <- sort(unique(AllExpData$curDur))

AllExpData$WMSize <- factor(AllExpData$WMSize, labels = c("low", "medium",  "high"))
# 1: 500ms,  2: 2500, 3 : 2000ms the mean reaction time of WM test
AllExpData$gap <- factor(AllExpData$gap,  labels = c('short','long', 'not sure'))
AllExpData[which(AllExpData$Exp == 'Exp4'),"Exp"] = "Exp4a"
AllExpData[which(AllExpData$Exp == 'Exp5'),"Exp"] = "Exp4b"
```

```
## load model Prediction results
AllDat_predY <- read.csv(paste0(modelPath, "/rlt/AllDat_predY_",modelversion,".csv"))
AllDat_predY$WMSize <- as.factor(AllDat_predY$WMSize)
levels(AllDat_predY$WMSize) = c("low", "medium",  "high")
AllDat_predY$pred_Bias = AllDat_predY$mu_r - AllDat_predY$curDur
AllDat_predY$predErr = AllDat_predY$mu_r - AllDat_predY$repDur
AllDat_predY$relatErr = AllDat_predY$predErr / AllDat_predY$repDur
AllDat_predY[which(AllDat_predY$Exp == "Exp4"),"Exp"] = "Exp4a"
AllDat_predY[which(AllDat_predY$Exp == "Exp5"),"Exp"] = "Exp4b"
AllDat_predY$Exp = as.factor(AllDat_predY$Exp)
AllDat_predY$gap <- factor(AllDat_predY$gap,  labels = c('short','long', 'not sure'))
```

2 Corrct rate

```
WMCrr2 <- ggplot(meanForPlot, aes(WMSize, mean_WMCrr, ymin = mean_WMCrr - se_WMCrr, ymax = mean_WMCrr +
                                  group =Exp, color = Exp, fill = Exp))+
  geom_line(stat = "identity",position = position_dodge(width = 0.2))+
  geom_point(stat = "identity",position = position_dodge(width = 0.2))+
```

```
  geom_errorbar(width=.2,  position = position_dodge(width = 0.2)) +
  coord_cartesian(ylim = c(0.5, 1)) +
  colorSet5+
  labs(x = "Memory load", y = "Mean accuracy in WM task") +
  theme_new
WMCrr2
```




```
ggsave(paste0(getwd(), "/figures/WMCrr2.png"), WMCrr2, width = 4, height = 4)
```

```
### generate WM correct rates
AllExpData$WMCrr <- AllExpData$TPresent == AllExpData$WMRP
m_wmp<- dplyr::group_by(AllExpData, Exp, WMSize, NSub) %>%
  dplyr::summarize(m_WMCrr = mean(WMCrr), n =n(), se_WMCrr = sd(WMCrr)/sqrt(n-1))
```

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

```
ezANOVA(data = WMCrr, dv=m_WMCrr, wid=NSub, within = .(WMSize), between = .(Exp))
```

```
## Warning: Converting "NSub" to factor for ANOVA.

## Warning: Converting "Exp" to factor for ANOVA.

## Warning: The column supplied as the wid variable contains non-unique values
## across levels of the supplied between-Ss variables. Automatically fixing this by
## generating unique wid labels.

## $ANOVA
##       Effect DFn DFd          F            p p<.05        ges
## 2        Exp   4  75   8.816704 6.868286e-06     * 0.25356151
```

```
## 3     WMSize   2 150 526.088822 1.618810e-68     * 0.66068954
## 4 Exp:WMSize   8 150   2.434296 1.672504e-02     * 0.03478551
##
## $`Mauchly's Test for Sphericity`
##       Effect        W          p p<.05
## 3     WMSize 0.907248 0.02728116     *
## 4 Exp:WMSize 0.907248 0.02728116     *
##
## $`Sphericity Corrections`
##       Effect       GGe        p[GG] p[GG]<.05       HFe        p[HF] p[HF]<.05
## 3     WMSize 0.9151207 6.137253e-63         * 0.9368512 2.288774e-64         *
## 4 Exp:WMSize 0.9151207 2.034718e-02         * 0.9368512 1.934824e-02         *
```

2.1 check model parameters

```
### Average Parameters
mm_Baypar <- dplyr::group_by(Bayparlist, Exp) %>%
  dplyr::summarize( m_sig_s2 = mean(sig_s2),
                    m_sig_pr2_log = mean(sig_pr2_log),
                    m_ks= mean(ks),
                    m_kr = mean(kr),
                    m_ls = mean(ls),
                    m_ts = mean(ts),
                    m_mu_pr = mean(mu_pr),
                    m_sig_pr2 = mean(sig_pr2),
                    m_mu_pr_log= mean(mu_pr_log),
                    m_sig_mn2 = mean(sig_mn2),
                    n= n(),
                    se_sig_s2 = sd(sig_s2)/sqrt(n-1),
                    se_sig_mn2 = sd(sig_mn2)/sqrt(n-1),
                    se_sig_pr2_log = sd(sig_pr2_log)/sqrt(n-1),
                    se_ks = sd(ks)/sqrt(n-1),
                    se_kr = sd(kr)/sqrt(n-1),
                    se_ls =sd(ls)/sqrt(n-1),
                    se_mu_pr_log = sd(mu_pr_log)/sqrt(n-1))
mm_Baypar
```

```
## # A tibble: 5 x 19
##   Exp   m_sig_s2 m_sig_pr2_log  m_ks   m_kr   m_ls    m_ts m_mu_pr m_sig_pr2
##   <fct>    <dbl>         <dbl> <dbl>  <dbl>  <dbl>   <dbl>   <dbl>     <dbl>
## 1 Exp1    0.0341         0.139  0     0      0      0.00545   0.966    0.107
## 2 Exp2    0.0801         0.119  0.120 0      0.114  0.0125    1.04     0.0322
## 3 Exp3    0.0388         0.0940 0     0.0318 0      0.00513   0.910    0.0155
## 4 Exp4a   0.0302         0.153  0.421 0.224  0.0141 0.00483   0.982    0.0661
## 5 Exp4b   0.0221         0.104  0.408 0.205  0.0171 0.00484   0.985    0.0143
## # ... with 10 more variables: m_mu_pr_log <dbl>, m_sig_mn2 <dbl>, n <int>,
## #   se_sig_s2 <dbl>, se_sig_mn2 <dbl>, se_sig_pr2_log <dbl>, se_ks <dbl>,
## #   se_kr <dbl>, se_ls <dbl>, se_mu_pr_log <dbl>
```

3 Prediction results

```
#calculate the mean reproduction biases for the five given intervals for all subjects
mpredY_sub <- dplyr::group_by(AllDat_predY, curDur, Exp, NSub, WMSize) %>%
```

```
  dplyr::summarize(n = n(),
                   m_repDur = mean(repDur),
                   sd_repDur = sd(repDur),
                   m_mu_r = mean(mu_r),
                   m_sig_r = mean(sig_r),
                   m_wp = mean(wp),
                   se_wp = sd(wp)/sqrt(n-1),
                   log_lik =mean(log_lik),
                   cv =sd_repDur/ m_repDur,
                   pred_cv = mean(sig_r/mu_r),
                   predRP_err = mean(m_mu_r-m_repDur),
                   predVar_err = mean(m_sig_r-sd_repDur),
                   predRP_rerr = mean(abs(m_mu_r-m_repDur)/m_repDur),
                   predVar_rerr = mean(abs(m_sig_r-sd_repDur)/sd_repDur),
                   predcv_err = pred_cv-cv,
                   predcv_rerr = mean(abs(pred_cv-cv)/cv))
```

```
## `summarise()` has grouped output by 'curDur', 'Exp', 'NSub'. You can override
## using the `.groups` argument.
```

```
write_csv(dplyr::group_by(mpredY_sub, curDur, NSub) %>%
  dplyr::summarize(m_cv = mean(cv))%>%spread(curDur, m_cv), paste0(modelPath, '/rlt/m_cv.csv'))
```

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

```
mpredY_sub$RP_bias = mpredY_sub$m_repDur -mpredY_sub$curDur
mpredY_sub_new <- dplyr::group_by(mpredY_sub, curDur, Exp, NSub) %>%
  dplyr::summarize(m_RP_bias = mean(RP_bias))%>% spread(curDur, m_RP_bias)
```

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

```
write_csv(mpredY_sub_new%>%filter(Exp == 'Exp1'), paste0(modelPath, '/rlt/RP_Bias_exp1.csv'))
write_csv(mpredY_sub_new%>%filter(Exp == 'Exp2'), paste0(modelPath, '/rlt/RP_Bias_exp2.csv'))
write_csv(mpredY_sub_new%>%filter(Exp == 'Exp3'), paste0(modelPath, '/rlt/RP_Bias_exp3.csv'))
write_csv(mpredY_sub_new%>%filter(Exp == 'Exp4a'), paste0(modelPath, '/rlt/RP_Bias_exp4a.csv'))
write_csv(mpredY_sub_new%>%filter(Exp == 'Exp4b'), paste0(modelPath, '/rlt/RP_Bias_exp4b.csv'))


mpredY_sub_WMsize <- dplyr::group_by(mpredY_sub, WMSize, Exp, NSub) %>%
  dplyr::summarize(m_RP_bias = mean(RP_bias))%>% spread(WMSize, m_RP_bias)
```

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

```
write_csv(mpredY_sub_WMsize%>%filter(Exp == 'Exp3'), paste0(modelPath, '/rlt/RP_Bias_WMsize_exp3.csv'))
write_csv(mpredY_sub_WMsize%>%filter(Exp == 'Exp4a'), paste0(modelPath, '/rlt/RP_Bias_WMsize_exp4a.csv'
write_csv(mpredY_sub_WMsize%>%filter(Exp == 'Exp4b'), paste0(modelPath, '/rlt/RP_Bias_WMsize_exp4b.csv'
```

```
#### predicted data
m_predY <- mpredY_sub%>%
  dplyr::group_by(Exp, curDur, WMSize) %>%
  dplyr::summarize(m_m_repDur = mean(m_repDur),
                   m_sd_repDur = mean(sd_repDur),
                   m_m_sig_r =mean(m_sig_r),
                   m_m_mu_r = mean(m_mu_r),
```

```
                   m_m_wp = mean(m_wp),
                   n = n(),
                   m_se_wp = sd(se_wp)/sqrt(n-1),
                   log_lik =mean(log_lik),
                   mpredRP_err = mean(predRP_err),
                   mpredVar_err = mean(predVar_err),
                   mpredRP_rerr = mean(predRP_rerr),
                   mpredVar_rerr = mean(predVar_rerr),
                   cv= mean(cv),
                   pred_cv = mean(pred_cv),
                   mpredcv_err = mean(predcv_err),
                   mpredcv_rerr = mean(predcv_rerr))
m_predY_acc =  mpredY_sub%>%
  dplyr::group_by(Exp) %>%
  dplyr::summarize(mpred_rerr = mean(predRP_rerr)*100,
                   mpredVar_rerr = mean(predVar_rerr)*100,
                   mpredcv_rerr = mean(predcv_rerr)*100)
m_predY_acc
```

```
## # A tibble: 5 x 4
##   Exp   mpred_rerr mpredVar_rerr mpredcv_rerr
##   <fct>      <dbl>         <dbl>        <dbl>
## 1 Exp1        3.33          17.8         17.7
## 2 Exp2        4.17          13.6         13.9
## 3 Exp3        3.40          16.1         15.9
## 4 Exp4a       3.47          18.5         18.5
## 5 Exp4b       2.81          16.5         15.7
```

4 WAIC and LOO-CV

```
#extract waic and loo-cv from parameter list
m_WAIC <- dplyr::group_by(Bayparlist, Exp) %>%
  dplyr::summarize(n =n(),
                   m_looic = mean(looic),
                   m_waic = mean(waic),
                   se_waic = sd(waic)/sqrt(n-1),
                   se_looic = sd(looic)/sqrt(n-1),
                   m_p_loo = mean(p_loo),
                   m_elpd_loo = mean(elpd_loo),
                   m_se_looic = mean(se_looic),
                   m_se_p_loo = mean(se_p_loo),
                   m_p_waic = mean(p_waic),
                   m_se_waic = mean(se_waic))

m_WAIC
```

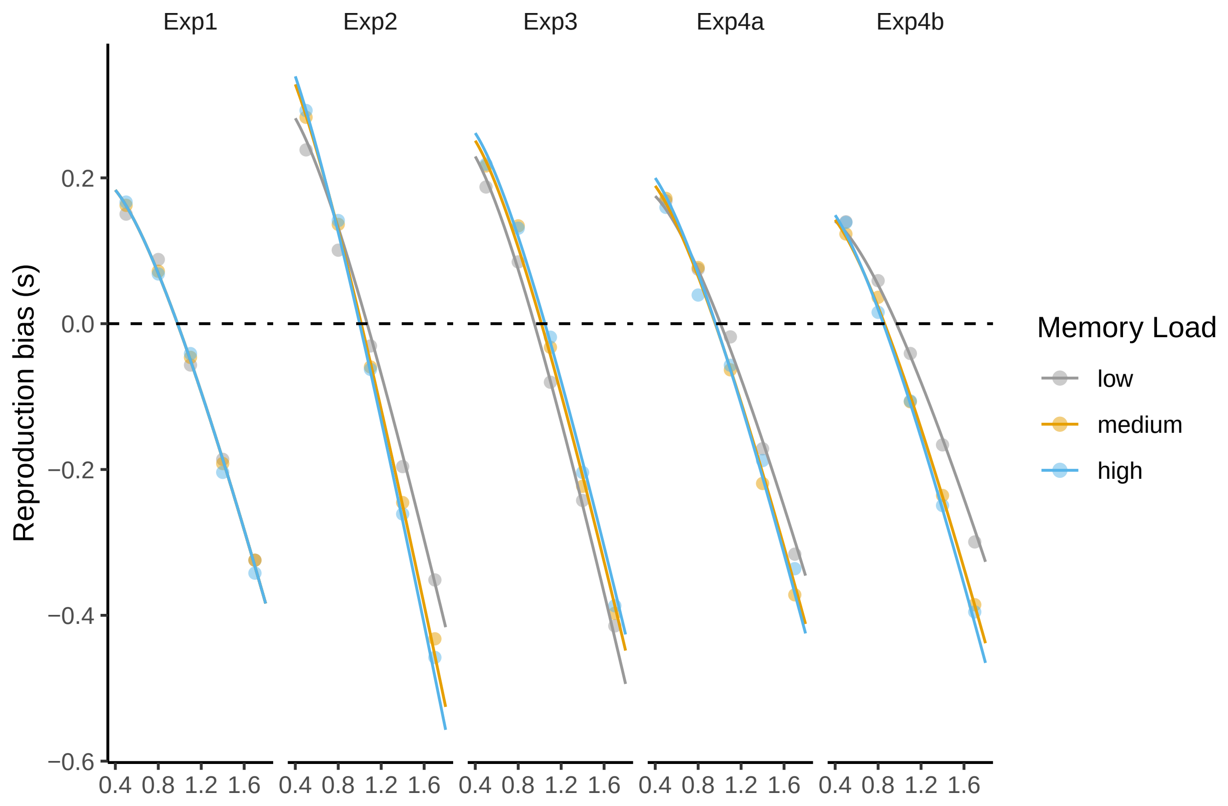
```
## # A tibble: 5 x 12
##   Exp       n m_looic m_waic se_waic se_looic m_p_loo m_elpd_loo m_se_looic
##   <fct> <int>   <dbl>  <dbl>   <dbl>    <dbl>   <dbl>      <dbl>      <dbl>
## 1 Exp1     16    374.   125.    38.2     38.2    305.      -187.       38.2
## 2 Exp2     16    557.   311.    40.5     40.1    303.      -278.       40.1
## 3 Exp3     16    421.   174.    21.7     21.4    304.      -210.       21.4
## 4 Exp4a    16    418.   170.    36.2     36.0    305.      -209.       36.0
```

```
## 5 Exp4b    16    813.   317.   103.     103.     609.      -406.      103.
## # ... with 3 more variables: m_se_p_loo <dbl>, m_p_waic <dbl>, m_se_waic <dbl>
```

```
#load test results
AllDat_newY <- read.csv(paste0(modelPath, "/rlt/AllDat_newY_",modelversion,".csv"))
AllDat_newY$WMSize <- as.factor(AllDat_newY$WMSize)
levels(AllDat_newY$WMSize) = c("low", "medium",  "high")
AllDat_newY[which(AllDat_newY$Exp == "Exp4"), "Exp"] = "Exp4a"
AllDat_newY[which(AllDat_newY$Exp == "Exp5"), "Exp"] = "Exp4b"
AllDat_newY$Exp = as.factor(AllDat_newY$Exp)
```

4.1 RP biase

```
  RP_bias  <- ggplot(data = m_predY, aes(x = curDur, y = m_m_repDur - curDur,color=WMSize, shape = as.fa
  geom_point(size=2, alpha = 0.5)+
  geom_line(data= m_newY, aes(x=curDur, y=m_mu_r-curDur, color=WMSize)) +
  geom_hline(yintercept = 0, linetype='dashed')+
  facet_grid(cols = vars(Exp)) +
  labs(x=" ", y="Reproduction bias (s)", shape=" ", color = "Memory Load")+theme_new+
  colorSet3+guides(shape="none")

ggsave(paste0(getwd(), "/", modelPath, "/figures/RP_bias.png"), RP_bias, width = 6, height = 6)

RP_bias
```




```
fig = ggplot(data = AllExpData %>% filter(Exp == 'Exp3') %>%
  dplyr::group_by(curDur, WMSize, NSub) %>%
  dplyr::summarize(n = n(),
```

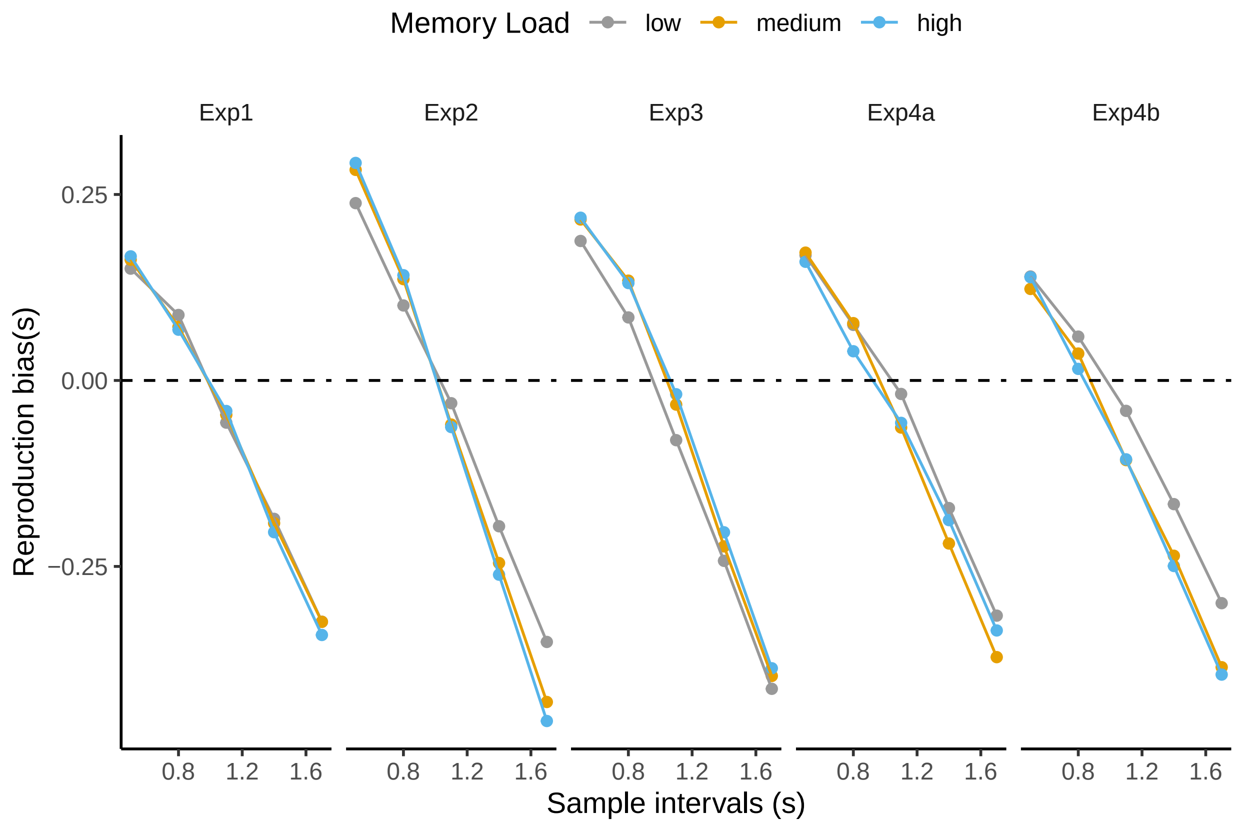
```r
                 m_repDur = mean(repDur),
                 se_repDur = sd(repDur)/sqrt(n-1)), aes(x=curDur, y = m_repDur-curDur, group = WMSize
  geom_point()+
  geom_line()+
  #geom_errorbar(width=.2,  aes(ymin = m_repDur - se_repDur, ymax = m_repDur + se_repDur)) +
  geom_hline(yintercept = 0, linetype='dashed')+
  facet_wrap(~NSub) +
  labs(x="Sample intervals (s)", y="Reproduction bias in Exp 4b(s)", shape=" ", color = "Memory Load")+
  theme_new+colorSet3+guides(shape="none")+
   scale_x_continuous(breaks=seq(0, 1.6, 0.4))+ theme(legend.position="top")
```

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

```r
RP_bias_obs  <- ggplot(data = AllExpData %>%
  dplyr::group_by(Exp, curDur, WMSize, NSub) %>%
  dplyr::summarize(n = n(),
                 m_repDur = mean(repDur),
                 se_repDur = sd(repDur)/sqrt(n-1)) %>%
  dplyr::group_by(Exp, curDur, WMSize) %>%
  dplyr::summarize(m_m_repDur = mean(m_repDur),
                 m_se_repDur = mean(se_repDur)), aes(x = curDur, y = m_m_repDur-curDur,  color=as.fac
  geom_point()+
  geom_line()+
  #geom_errorbar(width=.2,  aes(ymin = m_m_repDur-curDur - m_se_repDur, ymax = m_m_repDur -curDur + m_s
  geom_hline(yintercept = 0, linetype='dashed')+
  facet_grid(cols = vars(Exp)) +
  labs(x="Sample intervals (s)", y="Reproduction bias(s)", shape=" ", color = "Memory Load")+
  theme_new+colorSet3+guides(shape="none")+
   scale_x_continuous(breaks=seq(0, 1.6, 0.4))+ theme(legend.position="top")
```

```
## `summarise()` has grouped output by 'Exp', 'curDur', 'WMSize'. You can override using the `.groups` a
## `summarise()` has grouped output by 'Exp', 'curDur'. You can override using the `.groups` argument.
```

```r
ggsave(paste0(getwd(), "/", modelPath, "/figures/RP_bias_obs.png"), RP_bias_obs, width = 6, height = 4)

RP_bias_obs
```



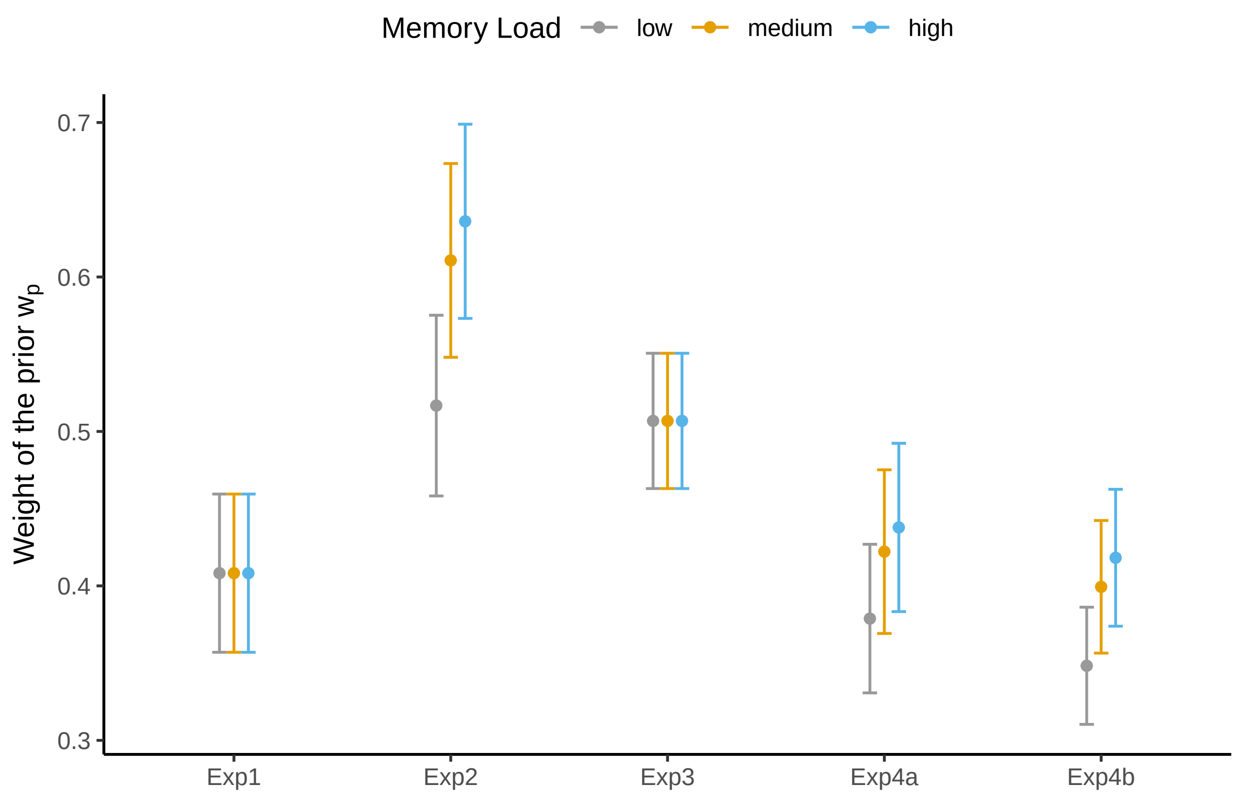
weight of prior

```
plt_wp <- ggplot(data = AllDat_predY %>%dplyr::group_by(NSub, Exp, WMSize) %>% dplyr::summarise(m_wp = m
  geom_line(stat = "identity",position = position_dodge(width = 0.2))+
  geom_point(stat = "identity",position = position_dodge(width = 0.2))+
  geom_errorbar(width=.2,  position = position_dodge(width = 0.2)) +
  #coord_cartesian(ylim = c(0.5, 1)) +
  colorSet5+
  labs(x = "", y = TeX("Weight of the prior $w_p$"), color = 'Memory Load') +
  theme_new + theme(legend.position="top")
```

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

```
plt_wp
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```




```
ggsave(paste0(getwd(), "/", modelPath, "/figures/plt_wp.png"), plt_wp, width = 3, height = 3)
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

5 Indifference Point and slope

5.1 Indifference point and slope

5.1.1 Observed data

```
#Observed Indifference Point for Exp.4b
obs_model <- function(df) {
  lm(repDur ~ curDur, data = df)
}
#Observed Indifference Point
obs_Inp_list <- AllDat_predY %>%
  dplyr::group_by(NSub, Exp, WMSize, gap) %>% nest()  %>%
  mutate(model = map(data, obs_model)) %>%  # linear regression
  mutate(slope = map(model, broom::tidy)) %>%  # get estimates
  unnest(slope, .drop = TRUE) %>% # remove raw data
  select(-std.error,-statistic, -p.value) %>%  # remove unnessary columns
  spread(term, estimate) %>%   # spread stimates
  dplyr::rename(Intercept = `(Intercept)`, slope = curDur)  # rename columns
```

```
## Warning: The `.drop` argument of `unnest()` is deprecated as of tidyr 1.0.0.
## All list-columns are now preserved.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

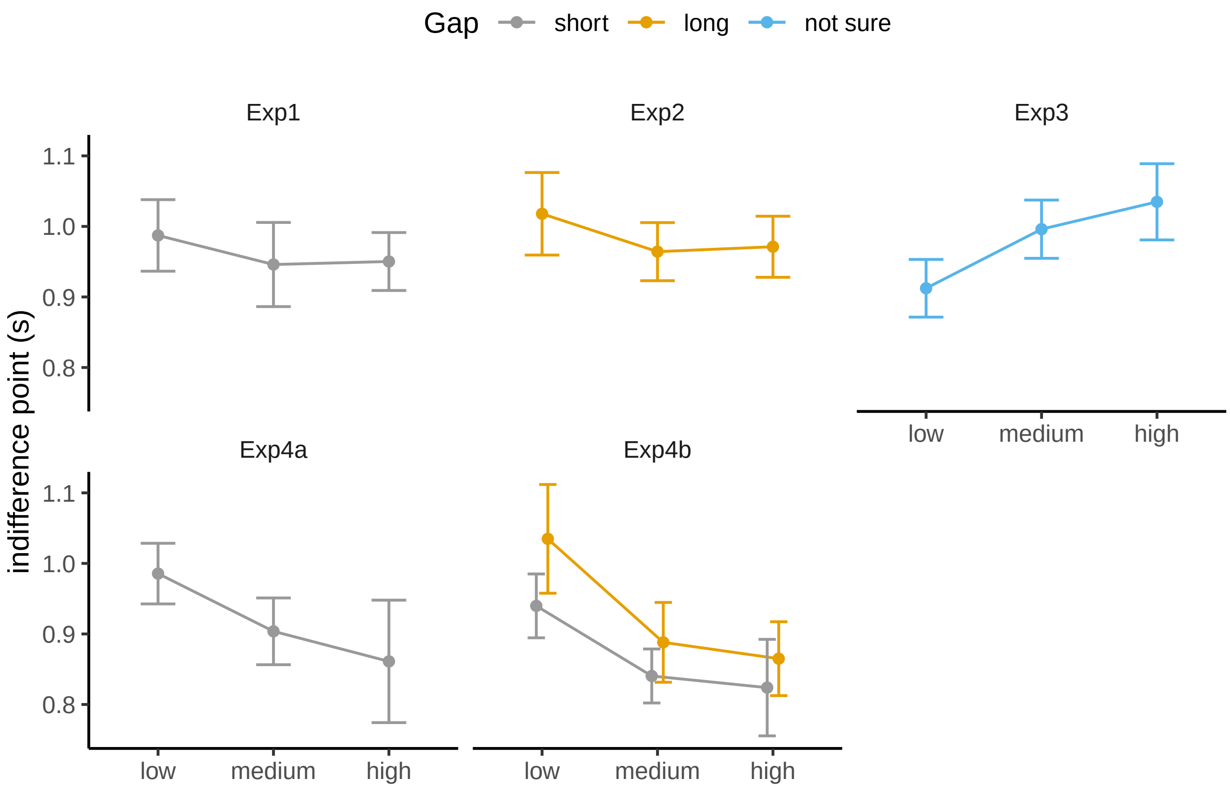
```r
obs_Inp_list$model = NULL
obs_Inp_list$data = NULL
obs_Inp_list$inP = obs_Inp_list$Intercept /(1-obs_Inp_list$slope)
```

```r
obs_Inp_list_no_gap <- AllDat_predY %>%
  dplyr::group_by(NSub, Exp, WMSize) %>% nest()  %>%
  mutate(model = map(data, obs_model)) %>%  # linear regression
  mutate(slope = map(model, broom::tidy)) %>%  # get estimates
  unnest(slope, .drop = TRUE) %>% # remove raw data
  select(-std.error,-statistic, -p.value) %>%
  spread(term, estimate) %>%   # spread stimates
  dplyr::rename(Intercept = `(Intercept)`, slope = curDur)  # rename columns
obs_Inp_list_no_gap$model = NULL
obs_Inp_list_no_gap$data = NULL
obs_Inp_list_no_gap$inP = obs_Inp_list_no_gap$Intercept /(1-obs_Inp_list_no_gap$slope)
```

```r
m_obs_Inp_list = obs_Inp_list %>% group_by(Exp, WMSize, gap)%>%
  dplyr::summarise(n=n(),
                   m_Intercept = mean(Intercept),
                   se_Intercept= sd(Intercept)/sqrt(n-1),
                   m_inP = mean(inP),
                   se_inP = sd(inP)/sqrt(n-1),
                   m_slope = mean(slope),
                   se_slope = sd(slope)/sqrt(n-1))
```

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

```r
plt_InP_linear_gap<- ggplot(data = m_obs_Inp_list, aes(x=WMSize, y=m_inP, group = gap, color = gap))+
  geom_line(stat = "identity",position = position_dodge(width = 0.2))+
  geom_point(stat = "identity",position = position_dodge(width = 0.2))+
  geom_errorbar(width=.3,  aes(ymin = m_inP - se_inP, ymax = m_inP + se_inP), position = position_dodge
  labs(colour = "Gap")+colorSet3+
  facet_wrap(~Exp)+
  xlab(' ')+ylab("indifference point (s)")+guides(shape="none")+
  theme(legend.position = "top")

ggsave(paste0(getwd(), "/", modelPath, "/figures/plt_InP_linear_gap.png"), plt_InP_linear_gap, width = 

plt_InP_linear_gap
```

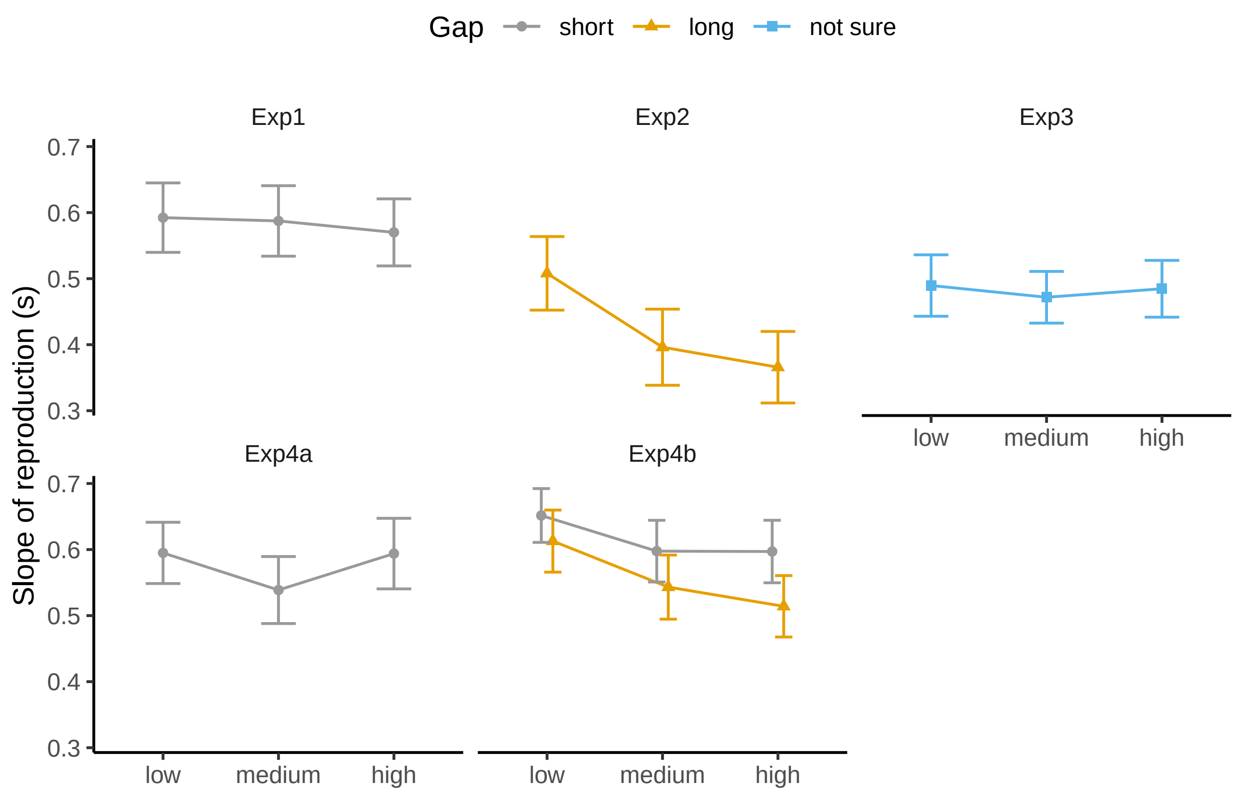



```
ezANOVA(data = obs_Inp_list%>%filter(Exp =='Exp4b'), dv= inP, wid=NSub, within= .(gap, WMSize) )
```

```
## Warning: Converting "NSub" to factor for ANOVA.

## Warning: You have removed one or more levels from variable "gap". Refactoring
## for ANOVA.

## $ANOVA
##        Effect DFn DFd        F           p p<.05         ges
## 2         gap   1  15 4.065468 0.062041074       0.019543220
## 3      WMSize   2  30 8.762649 0.001006806     * 0.078253848
## 4 gap:WMSize   2  30 0.584382 0.563670241       0.003061788
##
## $`Mauchly's Test for Sphericity`
##        Effect         W           p p<.05
## 3      WMSize 0.4515805 0.003829536     *
## 4 gap:WMSize 0.5210587 0.010428128     *
##
## $`Sphericity Corrections`
##        Effect       GGe       p[GG] p[GG]<.05       HFe       p[HF] p[HF]<.05
## 3      WMSize 0.6458198 0.004991546         * 0.6808339 0.004255437         *
## 4 gap:WMSize 0.6761593 0.502620038           0.7194299 0.512209537
```

```
plt_RP_slope_linear_gap<- ggplot(data = m_obs_Inp_list, aes(x= WMSize, y=m_slope, group = gap,color = ga
  geom_line(stat = "identity",position = position_dodge(width = 0.2))+
  geom_point(stat = "identity",position = position_dodge(width = 0.2))+
  geom_errorbar(width=.3,  aes(ymin = m_slope - se_slope, ymax = m_slope + se_slope), position = positio
  facet_wrap(~Exp)+
  labs(colour = "Gap", shape = "Gap")+colorSet3+
```

```
  xlab(' ')+ylab("Slope of reproduction (s)")+
  theme(legend.position = "top")

ggsave(paste0(getwd(), "/", modelPath, "/figures/plt_RP_slope_linear_gap.png"), plt_RP_slope_linear_gap

plt_RP_slope_linear_gap
```




```
# plot the observed indifference points and slopes of RP
plt_obs_InP_slope_err<- ggplot(data = obs_Inp_list %>% group_by(Exp, WMSize)%>%
  dplyr::summarise(n=n(),
                   m_inP = mean(inP),
                   se_inP = sd(inP)/sqrt(n-1),
                   m_slope = mean(slope),
                   se_slope = sd(slope)/sqrt(n-1)), aes(x= m_slope, y=m_inP, color = WMSize))+
  geom_line(stat = "identity")+
  geom_point(stat = "identity")+
  geom_errorbar(width = 0.02, aes(ymin = m_inP - se_inP, ymax = m_inP + se_inP)) +
  geom_errorbarh(height =0.02, aes(xmin = m_slope - se_slope, xmax = m_slope + se_slope)) +
  theme_new+
  labs(colour = "Memory Load")+colorSet3+
  facet_grid(~Exp)+
  xlab('slope of reproduction')+ylab("indifference point (s)")+guides(shape="none")+
  theme(legend.position = "top")
```

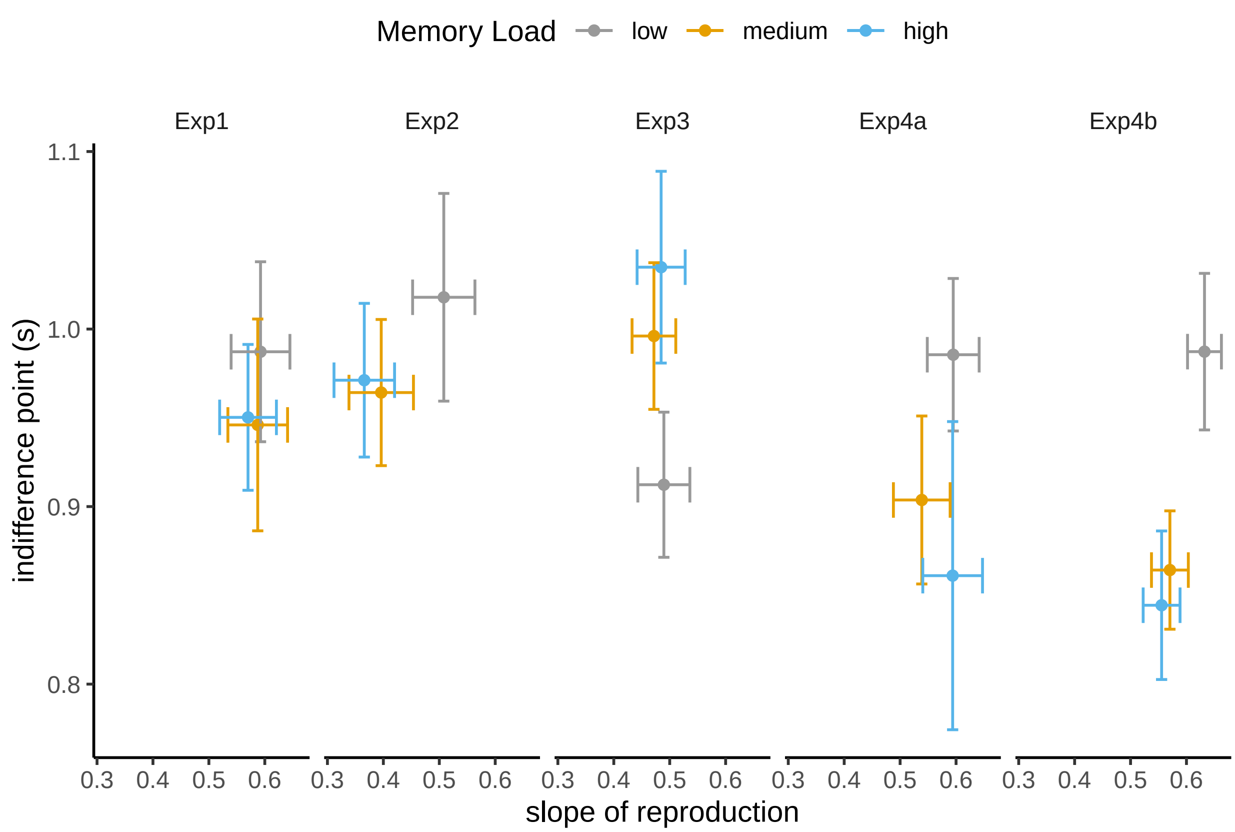
```
## `summarise()` has grouped output by 'Exp'. You can override using the `.groups`
## argument.
```

```
ggsave(paste0(getwd(), "/", modelPath, "/figures/plt_obs_InP_slope_err.png"), plt_obs_InP_slope_err, wi
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

```
plt_obs_InP_slope_err
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```



5.1.2 Predicated data

```
#Predicated Indifference Point for Exp.4b
pred_model <- function(df) {
  lm(mu_r ~ curDur, data = df)
}

pred_Inp_list <- AllDat_predY %>%
  dplyr::group_by(NSub, Exp, WMSize, gap) %>% nest()  %>%
  mutate(model = map(data, pred_model)) %>%  # linear regression
  mutate(slope = map(model, broom::tidy)) %>%  # get estimates
  unnest(slope, .drop = TRUE) %>% # remove raw data
  select(-std.error,-statistic, -p.value) %>%  # remove unnessary columns
  spread(term, estimate) %>%   # spread stimates
  dplyr::rename(Intercept = `(Intercept)`, pred_slope = curDur)  # rename columns
pred_Inp_list$model = NULL
pred_Inp_list$data = NULL
pred_Inp_list$pred_inP = pred_Inp_list$Intercept /(1-pred_Inp_list$pred_slope)
```

```
pred_Inp_slope_no_gap <- AllDat_predY %>%
  dplyr::group_by(NSub, Exp, WMSize) %>% nest()  %>%
  mutate(model = map(data, pred_model)) %>%  # linear regression
  mutate(slope = map(model, broom::tidy)) %>%  # get estimates
  unnest(slope, .drop = TRUE) %>% # remove raw data
  select(-std.error,-statistic, -p.value) %>%  # remove unnessary columns
  spread(term, estimate) %>%   # spread stimates
  dplyr::rename(Intercept = `(Intercept)`, pred_slope = curDur)  # rename columns
pred_Inp_slope_no_gap$model = NULL
pred_Inp_slope_no_gap$data = NULL
pred_Inp_slope_no_gap$pred_inP = pred_Inp_slope_no_gap$Intercept /(1-pred_Inp_slope_no_gap$pred_slope)
```

```
m_pred_Inp_slope_no_gap = pred_Inp_slope_no_gap %>% group_by(Exp, WMSize)%>%
  dplyr::summarise(n=n(),
                   m_Intercept = mean(Intercept),
                   se_Intercept= sd(Intercept)/sqrt(n-1),
                   m_pred_inP = mean(pred_inP),
                   se_pred_inP = sd(pred_inP)/sqrt(n-1),
                   m_pred_slope = mean(pred_slope),
                   se_pred_slope = sd(pred_slope)/sqrt(n-1))
```

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

```
# plot the observed indifference points and slopes of RP
plt_pred_InP_slope_err<- ggplot(data = m_pred_Inp_slope_no_gap, aes(x= m_pred_slope, y=m_pred_inP, colo
  geom_line(stat = "identity")+
  geom_errorbar(width = 0.02, aes(ymin = m_pred_inP - se_pred_inP, ymax = m_pred_inP + se_pred_inP)) +
  geom_errorbarh(height =0.02, aes(xmin = m_pred_slope - se_pred_slope, xmax = m_pred_slope + se_pred_sl
  geom_point(data = obs_Inp_list%>% group_by(Exp, WMSize)%>%
  dplyr::summarise(n=n(),
                   m_inP = mean(inP),
                   se_inP = sd(inP)/sqrt(n-1),
                   m_slope = mean(slope),
                   se_slope = sd(slope)/sqrt(n-1)), aes(x= m_slope, y =m_inP, color = WMSize))+
  theme_new+
```

```
  labs(colour = "Memory Load")+colorSet3+
  facet_grid(~Exp)+
  xlab('slope of reproduction')+ylab("indifference point (s)")+guides(shape="none")+
  theme(legend.position = "top")
```

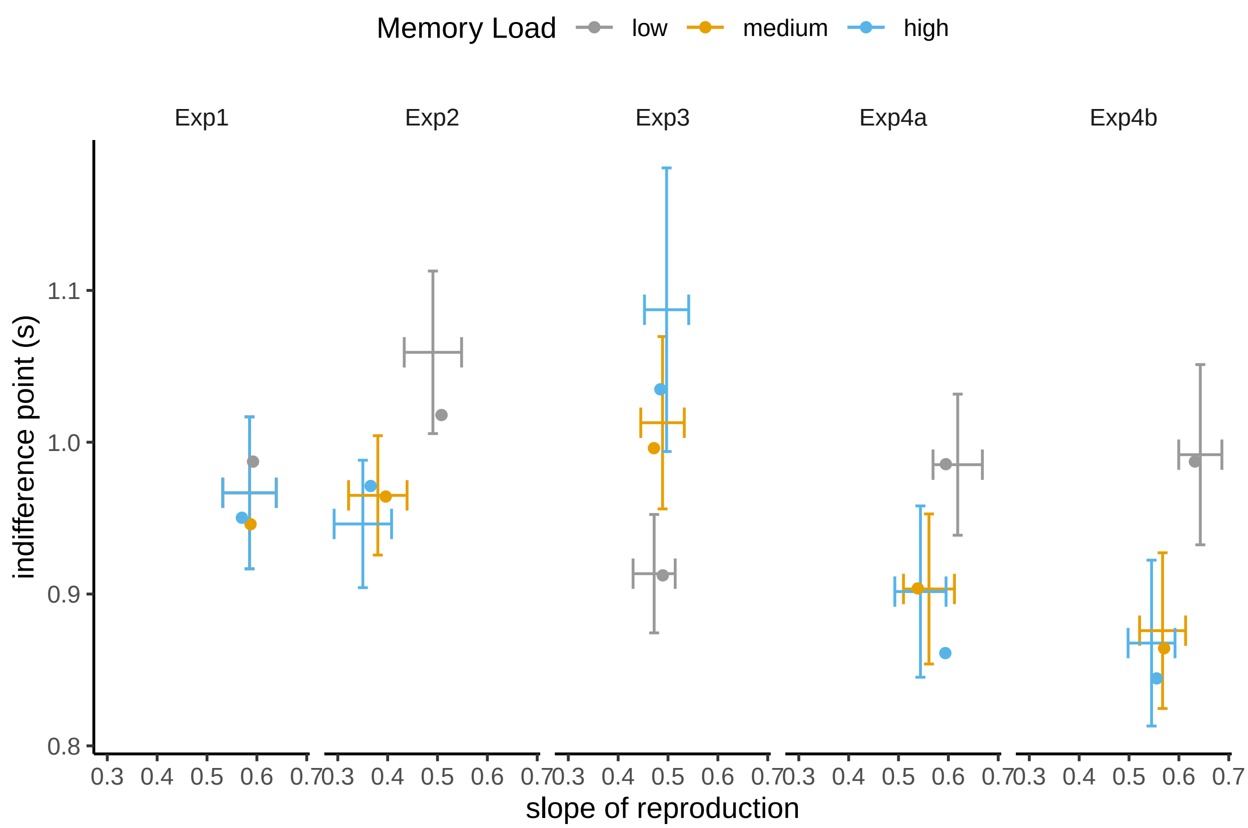
```
## `summarise()` has grouped output by 'Exp'. You can override using the `.groups`
## argument.
```

```
ggsave(paste0(getwd(), "/", modelPath, "/figures/plt_pred_InP_slope_err.png"), plt_pred_InP_slope_err, w
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

```
plt_pred_InP_slope_err
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```




```r
InP_obs<-  ggplot(data = obs_Inp_list_no_gap %>%dplyr::group_by(WMSize, Exp) %>%dplyr::summarise(m_inP =
  geom_line(stat = "identity",position = position_dodge(width = 0.2))+
  geom_point(stat = "identity",position = position_dodge(width = 0.2))+
  geom_errorbar(width=.2,  aes(ymin = m_inP - se_inP, ymax = m_inP + se_inP), position = position_dodge
  labs(colour = "Memory Load")+colorSet3+
  xlab(' ')+ylab("observed indifference point (s)")+guides(shape="none")+
  theme(legend.position = "top")
```

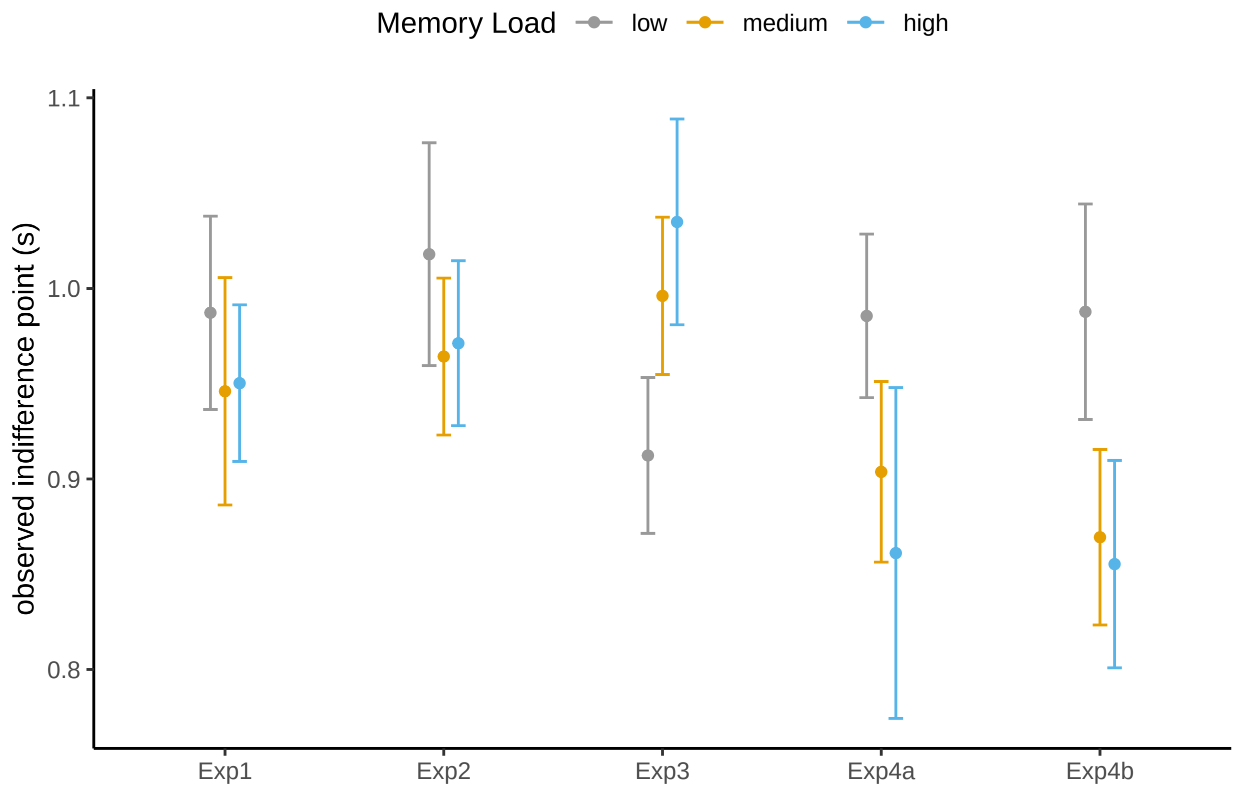
```
## `summarise()` has grouped output by 'WMSize'. You can override using the
## `.groups` argument.
```

```r
ggsave(paste0(getwd(), "/", modelPath, "/figures/InP_obs.png"), InP_obs, width = 3, height = 3)
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

```r
InP_obs
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

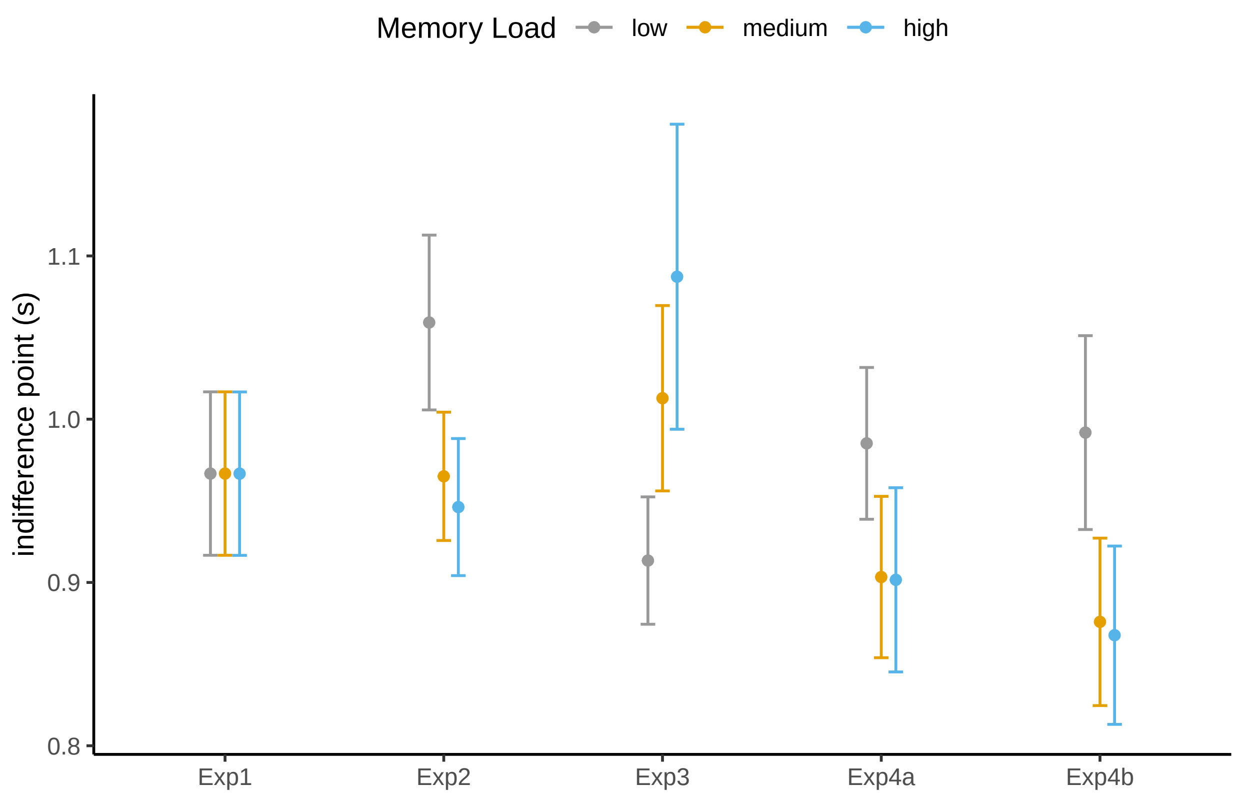



```
InP_pred<-  ggplot(data = m_pred_Inp_slope_no_gap, aes(x= Exp, y=m_pred_inP, color = WMSize))+
  geom_line(stat = "identity",position = position_dodge(width = 0.2))+
  geom_point(stat = "identity",position = position_dodge(width = 0.2))+
  geom_errorbar(width=.2,  aes(ymin = m_pred_inP - se_pred_inP, ymax = m_pred_inP + se_pred_inP), posit
  labs(colour = "Memory Load")+colorSet3+
  xlab(' ')+ylab("indifference point (s)")+guides(shape="none")+
  theme(legend.position = "top")


ggsave(paste0(getwd(), "/", modelPath, "/figures/InP_pred.png"), InP_pred, width = 3, height = 3)
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

```
InP_pred
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```




```
### Calculate predication error
Inp_list_no_gap = left_join(obs_Inp_list_no_gap, pred_Inp_slope_no_gap, by = c("NSub", "Exp", "WMSize"))
Inp_list_no_gap$InP_err = Inp_list_no_gap$pred_inP -Inp_list_no_gap$inP
Inp_list_no_gap$InP_rerr = 100*Inp_list_no_gap$InP_err/ Inp_list_no_gap$inP


Inp_list_no_gap$slope_err = Inp_list_no_gap$pred_slope - Inp_list_no_gap$slope
Inp_list_no_gap$slope_rerr = 100* Inp_list_no_gap$slope_err/Inp_list_no_gap$slope

m_Inp_list_no_gap = Inp_list_no_gap %>% dplyr::group_by(Exp) %>% dplyr::summarise(m_InP_rerr = mean(InP_

m_Inp_list_no_gap$InP_auc = 100- m_Inp_list_no_gap$m_InP_rerr_abs
m_Inp_list_no_gap$slope_auc = 100-  m_Inp_list_no_gap$m_slope_rerr_abs
```

6 plot figures

```
#plot the predicated indifference points and slope of predicated RP
plt_pred_InP_slope_err<- ggplot(data = obs_Inp_list_no_gap%>% dplyr::group_by(Exp, WMSize)%>%
  dplyr::summarise(n=n(),
                   m_inP = mean(inP),
                   se_inP = sd(inP)/sqrt(n-1),
                   m_slope = mean(slope),
                   se_slope = sd(slope)/sqrt(n-1)), aes(x= m_slope, y=m_inP, color = WMSize))+
  geom_line(stat = "identity")+
  geom_point(stat = "identity")+
  geom_errorbar(width = 0.02, aes(ymin = m_inP - se_inP, ymax = m_inP + se_inP)) +
```

```
  geom_errorbarh(height =0.02, aes(xmin = m_slope - se_slope, xmax = m_slope + se_slope)) +
  theme_new+
  labs(colour = "Memory Load")+colorSet3+
  facet_grid(~Exp)+
  xlab('slope of reproduction')+ylab("indifference point (s)")+guides(shape="none")+
  theme(legend.position = "top")
```

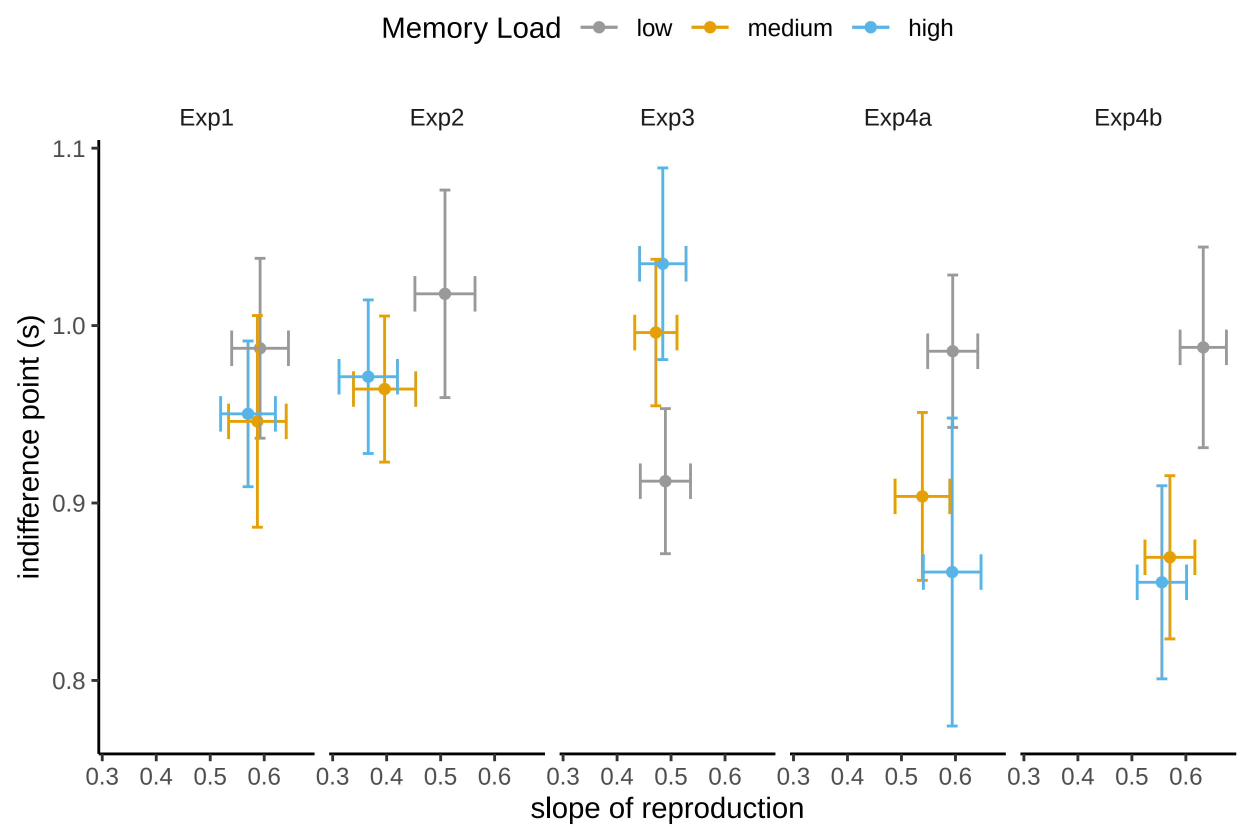
```
## `summarise()` has grouped output by 'Exp'. You can override using the `.groups`
## argument.
```

```
ggsave(paste0(getwd(), "/", modelPath, "/figures/plt_pred_InP_slope_err.png"), plt_pred_InP_slope_err, 
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

```
plt_pred_InP_slope_err
```

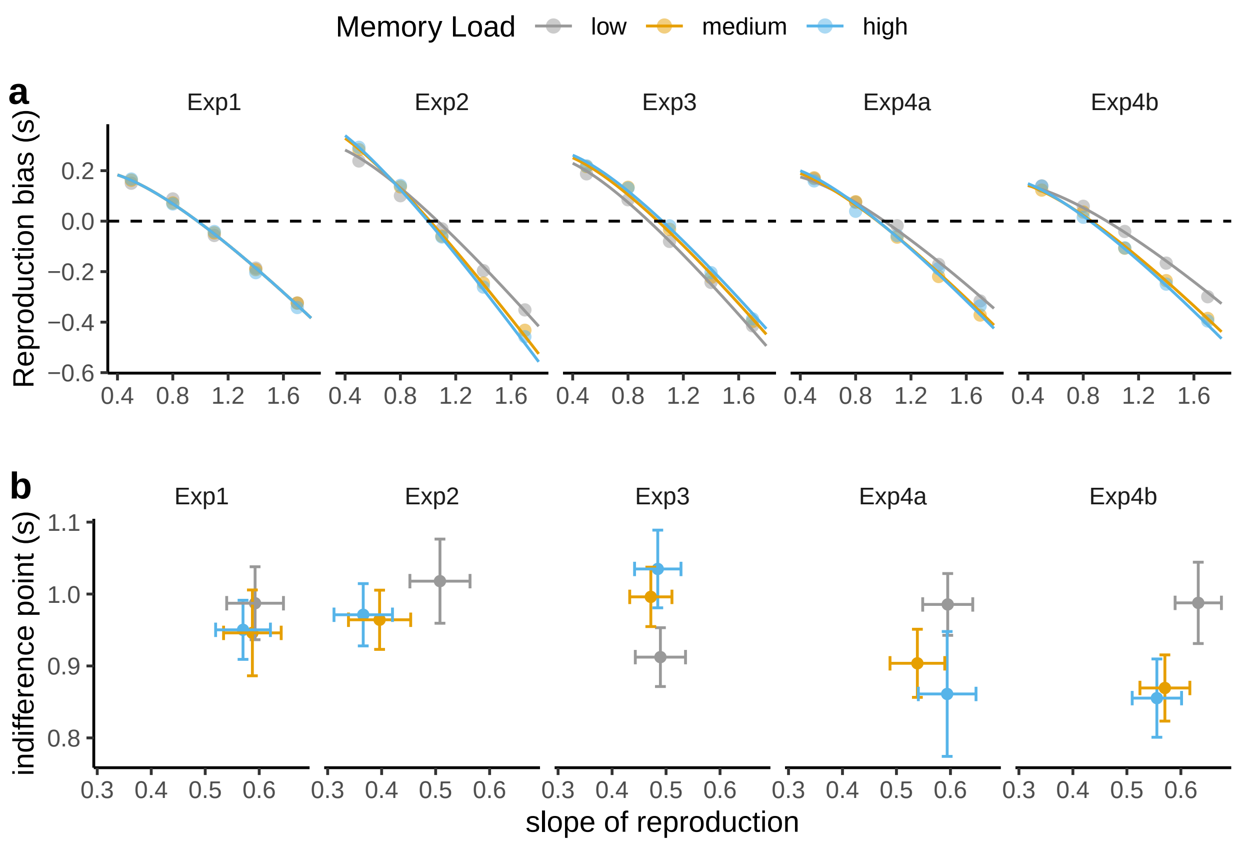
```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```




```
## Figures in the MS
fig3<-ggarrange(RP_bias, plt_pred_InP_slope_err, common.legend = TRUE, ncol=1, nrow=2,  labels = c("a",
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

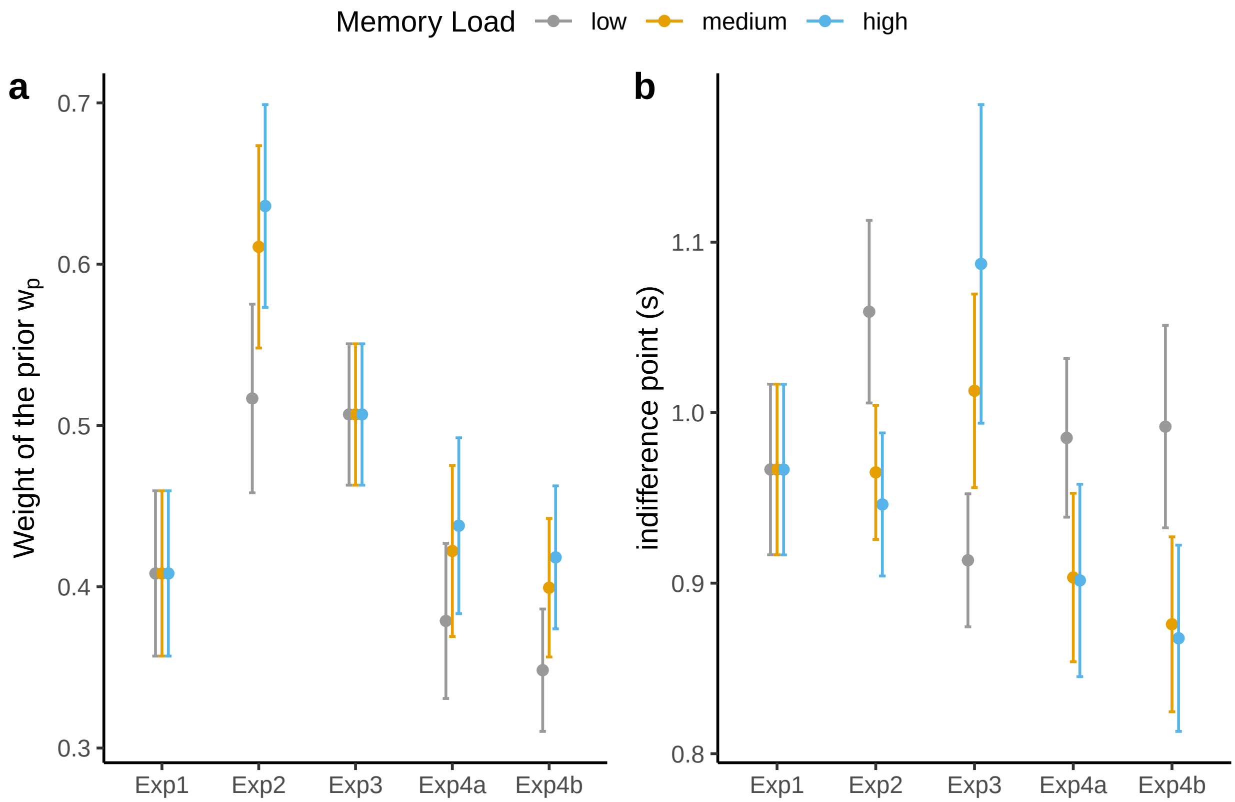
```
ggsave(paste0(getwd(), "/", modelPath, "/figures/fig3.png"), fig3, width = 6, height = 5)
fig3
```




```
## combine InP and wp
fig4<-ggarrange(plt_wp, InP_pred, common.legend = TRUE, ncol=2, nrow=1,  labels = c("a", "b"))
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```

```
ggsave(paste0(getwd(), "/", modelPath, "/figures/fig4.png"), fig4, width = 6, height = 3)
fig4
```



7 Model prediction error

```
m_predErr_sub<- mpredY_sub%>%
  dplyr::group_by(Exp, WMSize, NSub) %>% dplyr::summarise(
    mpredRP_err=mean(predRP_err),
    mpredVar_err=mean(predVar_err),
    mpredcv_err = mean(predcv_err),
    mpredRP_rerr = mean(predRP_rerr),
    mpredVar_rerr = mean(predVar_rerr),
    mpredcv_rerr = mean(predcv_rerr))
```

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

```
m_predErr<- m_predY%>%
  dplyr::group_by(Exp, WMSize) %>% dplyr::summarise(
    mmpredcv_err = mean(mpredcv_err),
    mmpredRP_err=mean(mpredRP_err),
    mmpredVar_err=mean(mpredVar_err),
    mmpredRP_rerr = mean(mpredRP_rerr),
    mmpredVar_rerr = mean(mpredVar_rerr),
    mmpredcv_rerr = mean(mpredcv_rerr))
```

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

```
m_predErr
```

```
## # A tibble: 15 x 8
## # Groups:   Exp [5]
##     Exp   WMSize mmpredcv_err mmpredRP_err mmpredVar_err mmpredRP_rerr
##     <fct> <fct>         <dbl>        <dbl>         <dbl>         <dbl>
##   1 Exp1  low        0.00594     -0.00193       0.00675         0.0283
##   2 Exp1  medium     0.0128      -0.00205       0.0117          0.0384
##   3 Exp1  high       0.00520      0.00274       0.00487         0.0333
##   4 Exp2  low        0.000861     0.0141        0.00576         0.0393
##   5 Exp2  medium     0.00862     -0.00541       0.00727         0.0364
##   6 Exp2  high       0.0135      -0.0111        0.0112          0.0494
##   7 Exp3  low        0.00725     -0.00433       0.00584         0.0361
##   8 Exp3  medium     0.0102      -0.00196       0.0113          0.0322
##   9 Exp3  high       0.00276      0.00645       0.00401         0.0339
##  10 Exp4a low        0.00833     -0.000218      0.00670         0.0303
##  11 Exp4a medium     0.00276      0.00211       0.00406         0.0354
##  12 Exp4a high       0.0144      -0.00363       0.0147          0.0383
##  13 Exp4b low        0.0153      -0.00151       0.0132          0.0210
##  14 Exp4b medium     0.0156       0.00147       0.0141          0.0321
##  15 Exp4b high       0.00117     -0.00130       0.000249        0.0310
## # ... with 2 more variables: mmpredVar_rerr <dbl>, mmpredcv_rerr <dbl>
```

8 Model comparison (logarithmic vs. linear)

```
m_predErr_sub$model = 'logarithmic'
m_predErr$model = 'logarithmic'
linear_model = 'gap_linear_rstan'
m_predErr_linear = read.csv(paste0(getwd(), "/", rstanmodelPath, '/models/', linear_model, "/rlt/m_predE
m_predErr_linear$X = NULL
m_predErr_sub_linear = read.csv(paste0(getwd(), "/", rstanmodelPath, '/models/', linear_model, "/rlt/m_p
m_predErr_sub_linear$X = NULL

m_predErr_sub_all = rbind(m_predErr_sub, m_predErr_sub_linear)
m_predErr_all = rbind(m_predErr, m_predErr_linear)
```

```
m_predErr_all$WMSize = as.factor(m_predErr_all$WMSize)
levels(m_predErr_all$WMSize) =  c("low", "medium",  "high")
temp = m_predErr_all %>% filter(model == 'logarithmic') %>%summarise(abs_mmpredcv_err = abs(mmpredcv_err
```

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

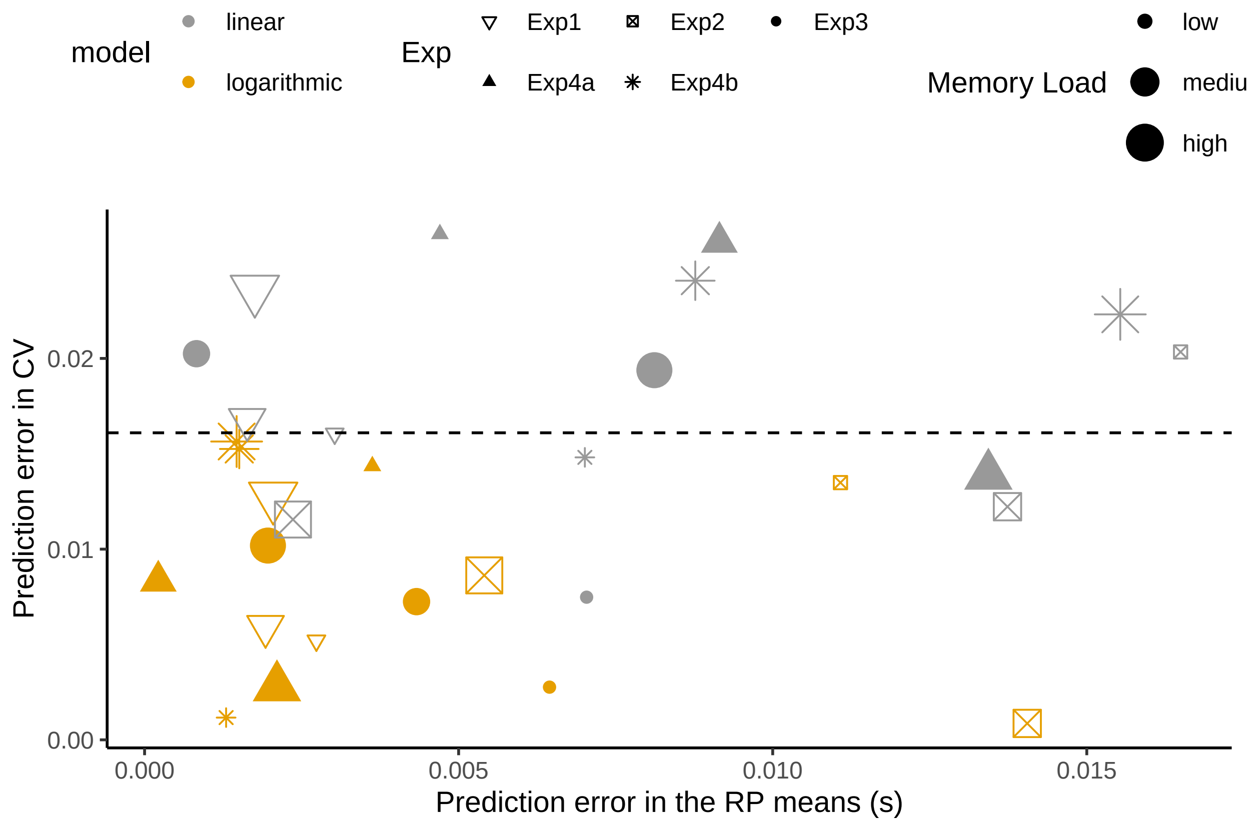
```
plt_Err_CV_all = ggplot(m_predErr_all, aes(abs(mmpredRP_err), abs(mmpredcv_err), group = interaction(mod
  geom_point() +
  geom_hline(yintercept = round(max(temp$abs_mmpredcv_err), 4)+0.0005, linetype='dashed')+
  xlab('Prediction error in the RP means (s)')+ ylab('Prediction error in CV')+colorSet3+
  scale_shape_manual(values = c(6, 7, 16, 17,8)) +
  theme_new+
  theme(legend.position = 'top')+
 labs(size = 'Memory Load')+
  guides(colour = guide_legend(order = 1, nrow=2,byrow=TRUE),
         shape = guide_legend(order =2, nrow=2,byrow=TRUE),
            size = guide_legend(order = 3, nrow=3,byrow=TRUE))
```

```
ggsave(paste0(getwd(), "/", modelPath, "/figures/plt_Err_CV_all.png"), plt_Err_CV_all, width = 7, heigh
```

```
## Warning: Using size for a discrete variable is not advised.
```

```
plt_Err_CV_all
```

```
## Warning: Using size for a discrete variable is not advised.
```




```
m_predY_acc =  m_predErr_sub_all%>%
  dplyr::group_by(Exp, model) %>%
  dplyr::summarize(mmpredRP_rerr = mean(mpredRP_rerr)*100,
                   mmpredVar_rerr = mean(mpredVar_rerr)*100,
                   mmpredcv_rerr = mean(mpredcv_rerr)*100,
                   mmpredRP_acc = (1-mean(mpredRP_rerr))*100,
                   mmpredVar_acc = (1-mean(mpredVar_rerr))*100,
                   mmpredCV_acc = (1-mean(mpredcv_rerr))*100)
```

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

```
m_predY_acc
```

```
## # A tibble: 10 x 8
## # Groups:   Exp [5]
##    Exp   model       mmpredRP_rerr mmpredVar_rerr mmpredcv_rerr mmpredRP_acc
##    <chr> <chr>               <dbl>          <dbl>         <dbl>        <dbl>
##  1 Exp1  linear               3.59           25.3          25.3         96.4
##  2 Exp1  logarithmic          3.33           17.8          17.7         96.7
##  3 Exp2  linear               4.66           18.6          19.0         95.3
```

```
##   4 Exp2  logarithmic          4.17           13.6           13.9          95.8
##   5 Exp3  linear               4.06           22.2           22.5          95.9
##   6 Exp3  logarithmic          3.40           16.1           15.9          96.6
##   7 Exp4a linear               3.91           28.2           28.7          96.1
##   8 Exp4a logarithmic          3.47           18.5           18.5          96.5
##   9 Exp4b linear               4.10           24.1           23.0          95.9
##  10 Exp4b logarithmic          2.81           16.5           15.7          97.2
## # ... with 2 more variables: mmpredVar_acc <dbl>, mmpredCV_acc <dbl>
```

9 Export data for spss

```
obs_Inp_slope_Exp1_jasp <- obs_Inp_list_no_gap %>% filter(Exp == 'Exp1') %>% select(c("WMSize", "NSub",
```

```
## Adding missing grouping variables: `Exp`
```

```
write.csv(obs_Inp_slope_Exp1_jasp, paste0(modelPath, '/rlt/obs_Inp_slope_Exp1_jasp.csv'))

obs_Inp_slope_Exp2_jasp <- obs_Inp_list_no_gap %>% filter(Exp == 'Exp2') %>% select(c("WMSize", "NSub",
```

```
## Adding missing grouping variables: `Exp`
```

```
write.csv(obs_Inp_slope_Exp2_jasp, paste0(modelPath, '/rlt/obs_Inp_slope_Exp2_jasp.csv'))

obs_Inp_slope_Exp3_jasp <- obs_Inp_list_no_gap %>% filter(Exp == 'Exp3') %>% select(c("WMSize", "NSub",
```

```
## Adding missing grouping variables: `Exp`
```

```
write.csv(obs_Inp_slope_Exp3_jasp, paste0(modelPath, '/rlt/obs_Inp_slope_Exp3_jasp.csv'))

obs_Inp_slope_Exp4a_jasp <- obs_Inp_list_no_gap %>% filter(Exp =='Exp4a') %>% select(c("WMSize", "NSub"
  pivot_wider(names_from = c("WMSize"), values_from = c(inP, slope), names_sep="_")
```

```
## Adding missing grouping variables: `Exp`
```

```
write.csv(obs_Inp_slope_Exp4a_jasp, paste0(modelPath, '/rlt/obs_Inp_slope_Exp4a_jasp.csv'))

obs_Inp_list_Exp4b_jasp <- obs_Inp_list %>% filter(Exp == 'Exp4b') %>% select(c("WMSize", "NSub", "gap"
```

```
## Adding missing grouping variables: `Exp`
```

```
write.csv(obs_Inp_list_Exp4b_jasp, paste0(modelPath, '/rlt/obs_Inp_slope_Exp4b_jasp.csv'))
```

```
pred_Inp_slope_Exp1_jasp <- pred_Inp_slope_no_gap %>% filter(Exp == 'Exp1') %>% select(c("WMSize", "NSub
```

```
## Adding missing grouping variables: `Exp`
```

```
write.csv(pred_Inp_slope_Exp1_jasp, paste0(modelPath, '/rlt/pred_Inp_slope_Exp1_jasp.csv'))

pred_Inp_slope_Exp2_jasp <- pred_Inp_slope_no_gap %>% filter(Exp == 'Exp2') %>% select(c("WMSize", "NSub
```

```
## Adding missing grouping variables: `Exp`
```

```
write.csv(pred_Inp_slope_Exp2_jasp, paste0(modelPath, '/rlt/pred_Inp_slope_Exp2_jasp.csv'))

pred_Inp_slope_Exp3_jasp <- pred_Inp_slope_no_gap %>% filter(Exp == 'Exp3') %>% select(c("WMSize", "NSub
```

```
## Adding missing grouping variables: `Exp`
```

```r
write.csv(pred_Inp_slope_Exp3_jasp, paste0(modelPath, '/rlt/pred_Inp_slope_Exp3_jasp.csv'))

pred_Inp_slope_Exp4a_jasp <- pred_Inp_slope_no_gap %>% filter(Exp =='Exp4a') %>% select(c("WMSize", "NSu
  pivot_wider(names_from = c("WMSize"), values_from = c(pred_inP, pred_slope), names_sep="_")
```

```
## Adding missing grouping variables: `Exp`
```

```r
write.csv(pred_Inp_slope_Exp4a_jasp, paste0(modelPath, '/rlt/pred_Inp_slope_Exp4a_jasp.csv'))

pred_Inp_slope_Exp4b_jasp <- pred_Inp_list %>% filter(Exp == 'Exp4b') %>% select(c("WMSize", "NSub", "ga
```

```
## Adding missing grouping variables: `Exp`
```

```r
write.csv(pred_Inp_slope_Exp4b_jasp, paste0(modelPath, '/rlt/pred_Inp_slope_Exp4b_jasp.csv'))
```

10 Linear Mixed-Effects Models

```r
library(tidyverse)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:rstatix':
##
##     select

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

```r
library(lme4)
library(lmerTest)
```

```
##
## Attaching package: 'lmerTest'

## The following object is masked from 'package:lme4':
##
##     lmer

## The following object is masked from 'package:stats':
##
##     step
```

```r
library(sjPlot)
AllValidData <- read_csv("../data/AllValidData.csv")
```

```
## Rows: 34483 Columns: 15

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): Exp
## dbl (14): WMSize, DurLevel, TPresent, NT, NSub, curDur, repDur, WMRP, valid,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
AllValidData$exp = substr(AllValidData$Exp, 4, 4)
AllValidData$Sub = as.numeric(AllValidData$exp)*100 + AllValidData$NSub
AllValidData$memory = (AllValidData$WMSize-1)/2  # memory load 0, 1, 2
```

Preliminary investigation using standard linear model to get first impression.

```r
mRep = AllValidData %>% filter(repDur > 0.2, repDur < 4) %>%
  group_by(Exp, Sub, WMSize, curDur) %>%
  summarise(Rep = mean(repDur)) %>%
  mutate(bias = Rep - curDur)
```

```
## `summarise()` has grouped output by 'Exp', 'Sub', 'WMSize'. You can override
## using the `.groups` argument.
```

```r
mRep$cond = mRep$Exp #avoid confusion from output Exp --> Cond
mRep$memory = (mRep$WMSize-1)/2  # memory load 0, 1, 2

lm(Rep ~ Exp*curDur*WMSize,
            data = mRep)
```

```
##
## Call:
## lm(formula = Rep ~ Exp * curDur * WMSize, data = mRep)
##
## Coefficients:
##           (Intercept)                ExpExp2                ExpExp3
##              0.376351               0.099875               0.089245
##               ExpExp4                ExpExp5                 curDur
##              0.037578              -0.032409               0.600837
##                WMSize         ExpExp2:curDur         ExpExp3:curDur
##              0.005844              -0.072969              -0.113054
##        ExpExp4:curDur         ExpExp5:curDur         ExpExp2:WMSize
##             -0.023689               0.037591               0.027468
##        ExpExp3:WMSize         ExpExp4:WMSize         ExpExp5:WMSize
##              0.006122              -0.011628              -0.002018
##         curDur:WMSize  ExpExp2:curDur:WMSize  ExpExp3:curDur:WMSize
##             -0.006202              -0.028602               0.004657
## ExpExp4:curDur:WMSize  ExpExp5:curDur:WMSize
##              0.006096              -0.010697
```

It seems that working memory impacts on the central tendency, as the interaction term WMSize x Duration varied across experiments.

To make a formal analysis, we use linear mixed model. Given that Experiment 1 was a baseline checking the duration reproduciton was not impacted by the sequential presentation, we leave this experiment out for cross-experiment analysis.

Given that we are interested how memory manipulation influence duration reproduction on the encoding and reproduction stages, the cross-experiment analysis mainly focuses on the manipulation stage.

Here are the experimental design:

Exp. 2: Memory load on the encoding stage Exp. 3: Memory load on the reproduction stage Exp. 4a (coded: 4): Memory load on both stages Exp. 4b (coded: 5): Memory load on both stage + additional 2 sec gap.

We are interested the following comparisons: 1. encoding vs. reproduction (Exp. 2 vs. Exp. 3) 2. individual impact vs. combination (exp. 2 + exp. 3 vs. Exp. 4a) 3. Gap effect (Exp. 4a vs. 4b)

Thus we construct the contrast matrix as follows:

$$\begin{matrix} 1 & -1 & 0 & 0 \\ 0.5 & 0.5 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{matrix}$$

In addition, we assume in each experiment, the slope and intercept of the reproduction could be covaried across different memory loads for individuals.

```
# contrast among experiment.
# we want to compare
# 1. E2 vs E3 (encoding vs. reprod)
# 2. combination of two phases vs. spanning: E2 + E3 vs. E4a
# 3. gap: E4a vs. E4b

contr = rbind(c(1, -1, 0, 0),
             c(0.5, 0.5, -1, 0),
             c(0, 0, 1, -1))
cmat = ginv(contr)

# Linear mixed model
mod1 = lmer(bias ~ cond*curDur*memory +
              (curDur * memory|Sub),
            contrasts = list(cond = cmat),
            data = mRep %>% filter(cond != 'Exp1'), REML = FALSE)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00600736 (tol = 0.002, component 1)
```

```
summary(mod1)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: bias ~ cond * curDur * memory + (curDur * memory | Sub)
##    Data: mRep %>% filter(cond != "Exp1")
##
##      AIC      BIC   logLik deviance df.resid
##  -2415.0  -2283.6   1234.5  -2469.0      933
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2847 -0.5847 -0.0061  0.6132  2.6875
##
## Random effects:
##  Groups   Name          Variance  Std.Dev. Corr
##  Sub      (Intercept)   3.162e-02 0.177827
##           curDur        3.035e-02 0.174199 -0.93
##           memory        6.281e-05 0.007925  0.82 -0.96
##           curDur:memory 4.030e-04 0.020074  0.09 -0.01 -0.25
##  Residual               2.666e-03 0.051629
## Number of obs: 960, groups:  Sub, 64
##
## Fixed effects:
##                        Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)            0.435753   0.023403  64.219084  18.619  < 2e-16 ***
```

```
## cond1                 0.031977   0.066194  64.218987   0.483  0.63069
## cond2                 0.085405   0.057326  64.219105   1.490  0.14117
## cond3                 0.060378   0.066194  64.219004   0.912  0.36511
## curDur               -0.455532   0.022643  64.079747 -20.118  < 2e-16 ***
## memory                0.021660   0.005757 462.438276   3.762  0.00019 ***
## cond1:curDur          0.006826   0.064044  64.079687   0.107  0.91545
## cond2:curDur         -0.087391   0.055464  64.079758  -1.576  0.12004
## cond3:curDur         -0.044487   0.064044  64.079690  -0.695  0.48980
## cond1:memory          0.042693   0.016283 462.438182   2.622  0.00903 **
## cond2:memory          0.056846   0.014102 462.438317   4.031 6.49e-05 ***
## cond3:memory         -0.019219   0.016283 462.438273  -1.180  0.23848
## curDur:memory        -0.026677   0.005425 115.608789  -4.917 2.93e-06 ***
## cond1:curDur:memory  -0.066517   0.015345 115.608618  -4.335 3.13e-05 ***
## cond2:curDur:memory  -0.036137   0.013289 115.608865  -2.719  0.00755 **
## cond3:curDur:memory   0.033586   0.015345 115.608775   2.189  0.03063 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation matrix not shown by default, as p = 16 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
## optimizer (nloptwrap) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.00600736 (tol = 0.002, component 1)
```

```
anova(mod1)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##                      Sum Sq Mean Sq NumDF  DenDF  F value    Pr(>F)
## cond                0.01910 0.00637     3  64.22   2.3881 0.0770323 .
## curDur              1.07882 1.07882     1  64.08 404.7324 < 2.2e-16 ***
## memory              0.03773 0.03773     1 462.44  14.1556 0.0001899 ***
## cond:curDur         0.01694 0.00565     3  64.08   2.1182 0.1065836
## cond:memory         0.06690 0.02230     3 462.44   8.3659 1.996e-05 ***
## curDur:memory       0.06445 0.06445     1 115.61  24.1785 2.930e-06 ***
## cond:curDur:memory 0.07132 0.02377     3 115.61   8.9194 2.300e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tab_model(mod1)
```

bias

Predictors

Estimates

CI

p

(Intercept)

0.44

0.39 – 0.48

<0.001

cond1

0.03

-0.10 – 0.16

0.629

cond2

0.09

-0.03 – 0.20

0.137

cond3

0.06

-0.07 – 0.19

0.362

curDur

-0.46

-0.50 – -0.41

<0.001

memory

0.02

0.01 – 0.03

<0.001

cond1 * curDur

0.01

-0.12 – 0.13

0.915

cond2 * curDur

-0.09

-0.20 – 0.02

0.115

cond3 * curDur

-0.04

-0.17 – 0.08

0.487

cond1 * memory

0.04

0.01 – 0.07

0.009

cond2 * memory

0.06

0.03 – 0.08

<0.001

cond3 * memory

-0.02

-0.05 – 0.01

0.238

curDur * memory

-0.03

-0.04 – -0.02

<0.001

(cond1 * curDur) * memory

-0.07

-0.10 – -0.04

<0.001

(cond2 * curDur) * memory

-0.04

-0.06 – -0.01

0.007

(cond3 * curDur) * memory

0.03

0.00 – 0.06

0.029

Random Effects

2

0.00

00 Sub

0.03

11 Sub.curDur

0.03

11 Sub.memory

0.00

11 Sub.curDur:memory

0.00

01

-0.93

0.82

0.09

ICC

0.81

N Sub

64

Observations

960

Marginal R2 / Conditional R2

0.757 / 0.953

Interpretation of the results

-The overall central tendency across all experiments was significant. The mean central tendency index was 0.45.

-The main effect of Memory Load was also significant. Increase memory load one level elevated the bias for 22 ms. (Overestimation)

-There was no significant difference of mean central tendency effect for those comparisons we were interested in (the above three comparison).

-Memory load, however, impacted differently. There was different impact of memory between Exp. 2 and 3., Exp2 + Exp 3 vs. Exp. 4. But there was no much difference between Exp 4 and 5 (4a vs. 4b)

-There was a strong interaction between memory and duration (central tendency). Increase one memory load level, the central tendency would increase 2.6% on average.

-The impact of memory on central tendency significantly differ among three contrasts (comparisons). 1. Difference between Exp 2 vs. 3. 2. Exp. 4 differred from the combination of Exp. 2 and 3. (i.e., no complete cancel each other on the central tendency when memory load was imposed on both stages) 3. Adding a gap inbetween increased 3.3% central tendency.