EM - repetition - Serial position analysis (v5)

```r
# do this in calling R script
# library(ggplot2)
# library(fmsb) # for TestModels
# library(lme4)
# library(kableExtra)
# library(MASS) # for dropterm
# library(tidyverse)
# library(dominanceanalysis)
# library(cowplot) # for multiple plots in one figure
# library(formatters) # to wrap strings for plot titles
# library(pals) # for continous color palettes
```

```r
# load needed functions
# source(paste0(RootDir,"/src/function_library/sp_functions.R"))
# do this in the calling R script
```

```r
# for testing - set by calling script normally
# ppt_parms <- read.csv(config$patient_param_file)
# CurPat <- ppt_parms$patient
# CurTask <- ppt_parms$task
# MinLength <- ppt_parms$min_length
# MaxLength <- ppt_parms$max_length
# RemoveFinalPosition<-ppt_parms$remove_final_position
```

```r
# create patient directory if it doesn't already exist
if(!dir.exists(paste0(RootDir,"/output/",CurPat))){
  dir.create(paste0(RootDir,"/output/",CurPat),showWarnings = TRUE)
}
TablesDir <- paste0(RootDir,"/output/",CurPat,"/tables/")
if(!dir.exists(TablesDir)){
  dir.create(TablesDir,showWarnings = TRUE)
}
FigDir <- paste0(RootDir,"/output/",CurPat,"/fig/")
if(!dir.exists(FigDir)){
  dir.create(FigDir,showWarnings = TRUE)
}
ReportsDir <- paste0(RootDir,"/output/",CurPat,"/reports/")
if(!dir.exists(ReportsDir)){
  dir.create(ReportsDir,showWarnings = TRUE)
}
results_report_DF <- data.frame(Description=character(), ParameterValue=double())
```

```r
ModelDatFilename<-paste0(RootDir,"/data/",CurPat,"/",CurPat,"_",CurTask,"_posdat.csv")
if(!file.exists(ModelDatFilename)){
  print("**ERROR** Input data file not found")
  print(sprintf("\tfile: ",ModelDatFilename))
  print(sprintf("\troot dir: ",RootDir))
}
PosDat<-read.csv(ModelDatFilename)
```

```
# may already be done in datafile
# if(RemoveFinalPosition){
#   PosDat <- PosDat[PosDat$pos < PosDat$stimlen,] # remove final position
# }
PosDat <- PosDat[PosDat$stimlen >= MinLength,] # include only lengths with sufficient N
PosDat <- PosDat[PosDat$stimlen <= MaxLength,]
# include only correct and nonword errors (not no response, word or w+nw errors)
PosDat <- PosDat[(PosDat$err_type == "correct") | (PosDat$err_type == "nonword"), ]
PosDat <- PosDat[(PosDat$pos) != (PosDat$stimlen), ]
# make sure final positions have been removed
PosDat$stim_number <- NumberStimuli(PosDat)
PosDat$raw_log_freq <- PosDat$log_freq
PosDat$log_freq <- PosDat$log_freq - mean(PosDat$log_freq) # centre frequency
PosDat$CumErr <- CalcCumErrFromPreserved(PosDat)
```

```
# plot distribution of complex onsets and codas
# across serial position in the stimuli -- do complexities concentrate
# on one part of the serial position curve?
# syll_component = 1 is coda
# syll_component = S is satellite

# get counts of syllable components in different serial positions
syll_comp_dist_N <- PosDat %>% mutate(pos_factor = as.factor(pos), syll_component_factor = as.factor(sy

syll_comp_dist_perc <- syll_comp_dist_N %>% ungroup() %>%
  mutate(across(O:S, ~ . / total))

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	533	33	126	NA	NA	692
2	64	NA	426	95	107	692
3	309	NA	167	201	15	692
4	294	NA	240	67	38	639
5	231	NA	210	73	36	550
6	207	1	139	72	22	441
7	178	NA	104	29	19	330
8	92	NA	55	26	4	177
9	76	NA	2	NA	7	85

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7702312	0.0476879	0.1820809	NA	NA	692
2	0.0924855	NA	0.6156069	0.1372832	0.1546243	692
3	0.4465318	NA	0.2413295	0.2904624	0.0216763	692
4	0.4600939	NA	0.3755869	0.1048513	0.0594679	639
5	0.4200000	NA	0.3818182	0.1327273	0.0654545	550
6	0.4693878	0.0022676	0.3151927	0.1632653	0.0498866	441

pos_factor	O	P	V	1	S	total
7	0.5393939	NA	0.3151515	0.0878788	0.0575758	330
8	0.5197740	NA	0.3107345	0.1468927	0.0225989	177
9	0.8941176	NA	0.0235294	NA	0.0823529	85

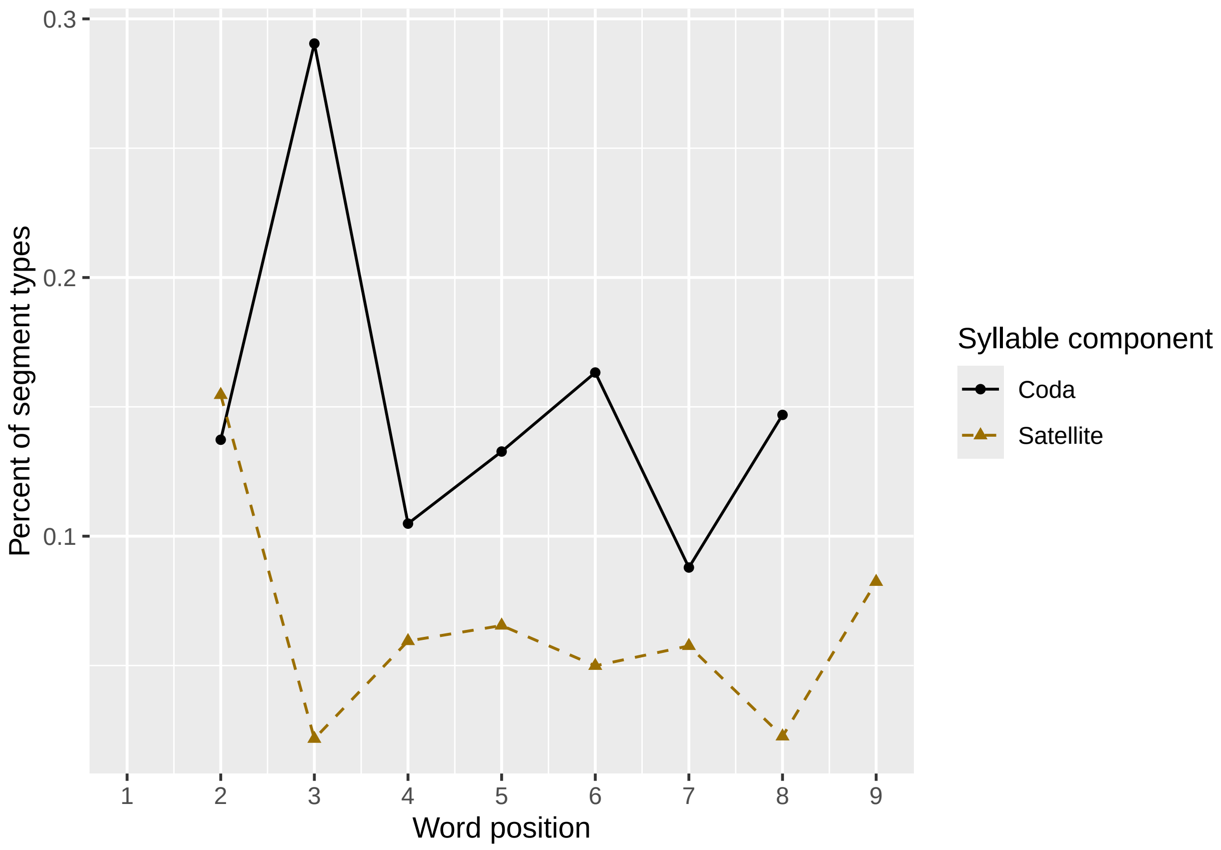
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# pivot to long format for plotting
syll_comp_plot_data <- syll_comp_perc %>% pivot_longer(cols=seq(1:(ncol(syll_comp_perc)-1)),
            names_to="syll_component", values_to="percent") %>% filter(syll_component %in% c("Coda","Sa
# plot satellite and coda occurances across position

syll_comp_plot <- ggplot(syll_comp_plot_data,
                         aes(x=pos,y=percent,group=syll_component,
                             color=syll_component,
                             linetype = syll_component,
                             shape = syll_component)) +
  geom_line() + geom_point() + scale_color_manual(name="Syllable component", values = palette_values) +
syll_comp_plot <- syll_comp_plot + scale_y_continuous(name="Percent of segment types") + scale_linetype_
  scale_shape_discrete(name="Syllable component")
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_pos_of_syll_complexities_in_stimuli.png"),plot=syll_comp_plot
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
syll_comp_plot
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range (`geom_line()`).
## Removed 3 rows containing missing values or values outside the scale range (`geom_point()`).
```



```
# len/pos table
pos_len_summary <- PosDat %>% group_by(stimlen,pos) %>% summarise(preserved = mean(preserved))
```

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

```
min_preserved_raw <- min(as.numeric(unlist(pos_len_summary$preserved)))
max_preserved_raw <- max(as.numeric(unlist(pos_len_summary$preserved)))
preserved_range <- (max_preserved_raw - min_preserved_raw)
min_preserved <- min_preserved_raw - (0.1 * preserved_range)
max_preserved <- max_preserved_raw + (0.1 * preserved_range)
pos_len_table <- pos_len_summary %>% pivot_wider(names_from = pos, values_from = preserved)
write.csv(pos_len_table,paste0(TablesDir,CurPat,"_",CurTask,"_preserved_percentages_by_len_position.csv
pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.623 0.868 0.774 NA     NA     NA     NA     NA     NA
## 2       5 0.482 0.747 0.561  0.538 NA     NA     NA     NA     NA
## 3       6 0.437 0.652 0.589  0.739  0.655 NA     NA     NA     NA
## 4       7 0.427 0.624 0.484  0.546  0.781  0.781 NA     NA     NA
## 5       8 0.476 0.683 0.558  0.657  0.599  0.798  0.726 NA     NA
## 6       9 0.320 0.554 0.483  0.645  0.507  0.540  0.806  0.724 NA
## 7      10 0.393 0.626 0.549  0.610  0.633  0.562  0.665  0.858  0.816
```

```
# len/pos table
pos_len_N <- PosDat %>% group_by(stimlen,pos) %>% summarise(preserved = mean(preserved),N=n()) %>% dply
```

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

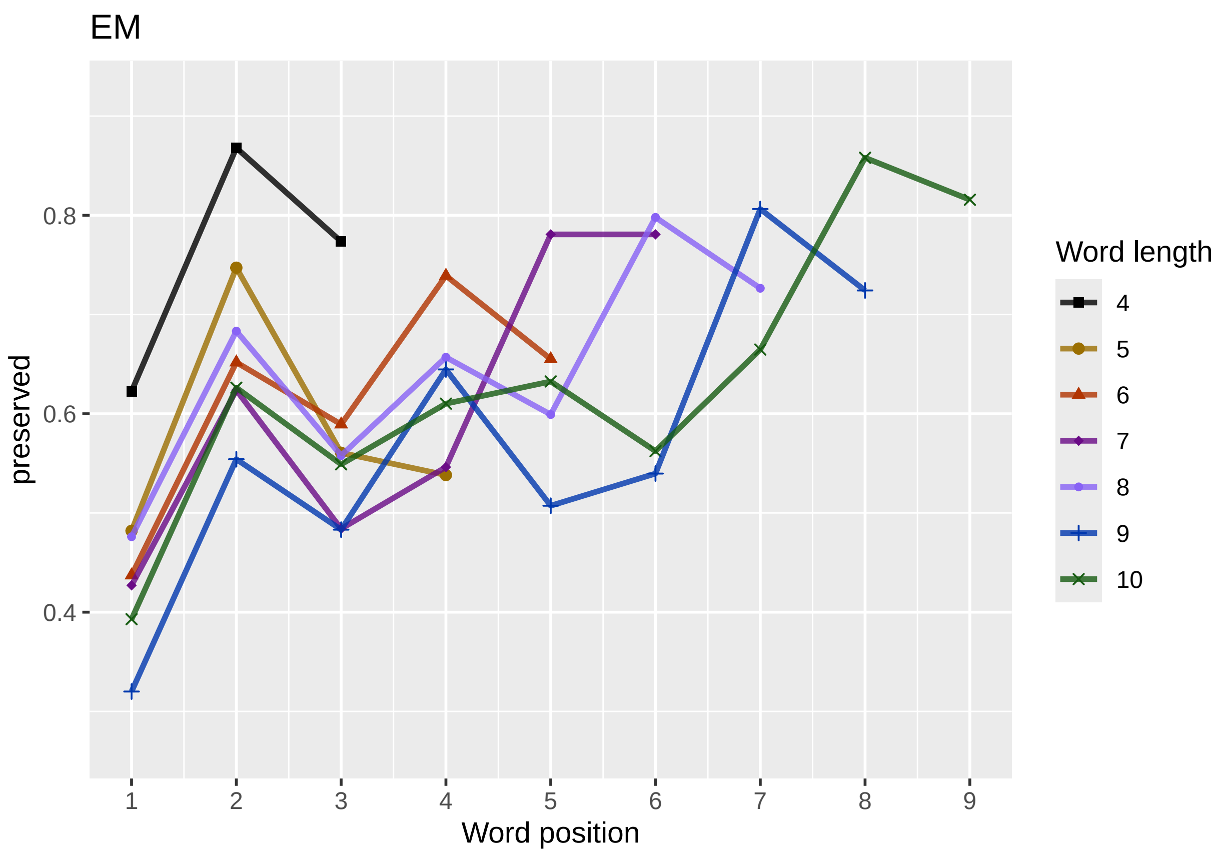
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1       4    53    53    53    NA    NA    NA    NA    NA    NA
## 2       5    89    89    89    89    NA    NA    NA    NA    NA
## 3       6   109   109   109   109   109    NA    NA    NA    NA
## 4       7   111   111   111   111   111   111    NA    NA    NA
## 5       8   153   153   153   153   153   153   153    NA    NA
## 6       9    92    92    92    92    92    92    92    92    NA
## 7      10    85    85    85    85    85    85    85    85    85
```

```
obs_linetypes <- c("solid","solid","solid","solid",
                      "solid","solid","solid","solid","solid")
pred_linetypes <- "dashed"
pos_len_summary$stimlen<-factor(pos_len_summary$stimlen)
pos_len_summary$pos<-factor(pos_len_summary$pos)
# len_pos_plot <- ggplot(pos_len_summary,aes(x=pos,y=preserved,group=stimlen,shape=stimlen,color=stimle
len_pos_plot <- plot_len_pos_obs_predicted(PosDat,
                                           paste0(CurPat),
                                           NULL,
                                           c(min_preserved,max_preserved),
                                           palette_values,
                                           shape_values,
                                           obs_linetypes,
                                           pred_linetypes)
```

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale for y is already present. Adding another scale for y, which will replace the existing
## scale.
```

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_percent_preserved_by_length_pos.png"),
       plot=len_pos_plot,device="png",unit="cm",width=15,height=11)
len_pos_plot
```



Length and position

```
# length and position

LPModelEquations<-c("preserved ~ 1",
            "preserved ~ stimlen",
            "preserved ~ pos",
            "preserved ~ stimlen + pos",
            "preserved ~ stimlen*pos",
            "preserved ~ I(pos^2)+pos",
            "preserved ~ stimlen + I(pos^2) + pos",
            "preserved ~ stimlen * (I(pos^2) + pos)"
)

LPRes<-TestModels(LPModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  4
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      0.5902      -0.1088       0.1952
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5204  AIC: 5620
## log likelihood:  -2601.84
## Nagelkerke R2:  0.04702635
## % pres/err predicted correctly:  -1860.154
## % of predictable range [ (model-null)/(1-null) ]:  0.03525783
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    0.800533    -0.134508     0.123872     0.008441
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5203  AIC: 5621
## log likelihood:  -2601.52
## Nagelkerke R2:  0.04722832
## % pres/err predicted correctly:  -1860.032
## % of predictable range [ (model-null)/(1-null) ]:  0.03532102
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    0.667962    -0.110930     0.004872     0.154833
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5203  AIC: 5621
## log likelihood:  -2601.598
## Nagelkerke R2:  0.04717953
## % pres/err predicted correctly:  -1860.064
## % of predictable range [ (model-null)/(1-null) ]:  0.03530463
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##     (Intercept)          stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        0.408006        -0.084538         -0.038086          0.403771          0.004559
##     stimlen:pos
##       -0.026092
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4292 Residual
## Null Deviance:       5350
## Residual Deviance: 5202  AIC: 5625
## log likelihood:  -2601.075
## Nagelkerke R2:  0.04750949
## % pres/err predicted correctly:  -1859.622
## % of predictable range [ (model-null)/(1-null) ]:  0.0355337
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     -0.1285       0.1633
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5647
## log likelihood:  -2616.075
## Nagelkerke R2:  0.03800482
## % pres/err predicted correctly:  -1874.037
## % of predictable range [ (model-null)/(1-null) ]:  0.02806124
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##  -0.1404349   -0.0009136    0.1709600
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5649
## log likelihood:  -2616.066
## Nagelkerke R2:  0.03801037
## % pres/err predicted correctly:  -1873.992
## % of predictable range [ (model-null)/(1-null) ]:  0.0280847
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##      0.4863
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4297 Residual
## Null Deviance:       5350
## Residual Deviance: 5350  AIC: 5769
## log likelihood:  -2675.026
## Nagelkerke R2:  -3.118626e-16
## % pres/err predicted correctly:  -1928.172
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     0.60613     -0.01551
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5349  AIC: 5771
## log likelihood:  -2674.685
## Nagelkerke R2:  0.0002231365
## % pres/err predicted correctly:  -1927.778
## % of predictable range [ (model-null)/(1-null) ]:  0.0002043344
## ***************************
```

```r
BestLPModel<-LPRes$ModelResult[[1]]
BestLPModelFormula<-LPRes$Model[[1]]

LPAICSummary<-data.frame(Model=LPRes$Model,
                       AIC=LPRes$AIC,
                       row.names = LPRes$Model)
LPAICSummary$DeltaAIC<-LPAICSummary$AIC-LPAICSummary$AIC[1]
LPAICSummary$AICexp<-exp(-0.5*LPAICSummary$DeltaAIC)
LPAICSummary$AICwt<-LPAICSummary$AICexp/sum(LPAICSummary$AICexp)
LPAICSummary$NagR2<-LPRes$NagR2

LPAICSummary <- merge(LPAICSummary,LPRes$CoefficientValues, by='row.names',sort=FALSE)
LPAICSummary <- subset(LPAICSummary, select = -c(Row.names))

write.csv(LPAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_len_pos_model_summary.csv"),row.names = FA
kable(LPAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	5619.719	0.000000	1.000000	0.4955783	0.0047026	0.5902308	-0.1088072	0.1951700	NA	NA	NA
preserved ~ stimlen * pos	5621.120	1.401054	0.4963237	0.2459673	0.0047228	0.8005328	-0.1345078	0.1238724	0.0084414	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	5621.367	1.648490	0.4385660	0.2173438	0.0471795	0.6679620	-0.1109295	0.1548335	NA	0.0048723	NA
preserved ~ stimlen * (I(pos^2) + pos)	5624.698	4.978967	0.0829528	0.0411096	0.0475095	0.4080057	-0.0845381	0.4037707	-0.0260917	-0.0380861	0.0045592
preserved ~ pos	5646.712	26.992946	0.0000014	0.0000007	0.0380048	-0.1285371	NA	0.1632952	NA	NA	NA
preserved ~ I(pos^2) + pos	5648.667	28.948140	0.0000005	0.0000003	0.0380104	-0.1404349	NA	0.1709600	NA	-0.0009136	NA
preserved ~ 1	5769.397	149.6784	0.0000000	0.0000000	0.0000000	0.4862643	NA	NA	NA	NA	NA
preserved ~ stimlen	5770.861	151.1427	0.0000000	0.0000000	0.0002231	0.6061301	-0.0155118	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

```
## [1] "preserved ~ stimlen + pos"
```

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      0.5902      -0.1088       0.1952
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5204  AIC: 5620
```

```
PosDat$LPFitted<-fitted(BestLPModel)
fitted_len_pos_plot <- plot_len_pos_obs_predicted(PosDat, PosDat$patient[1],
                          NULL,palette_values=palette_values)
```

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

```
# len/pos table
fitted_pos_len_summary <- PosDat %>% group_by(stimlen,pos) %>% summarise(fitted = mean(LPFitted))
```

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

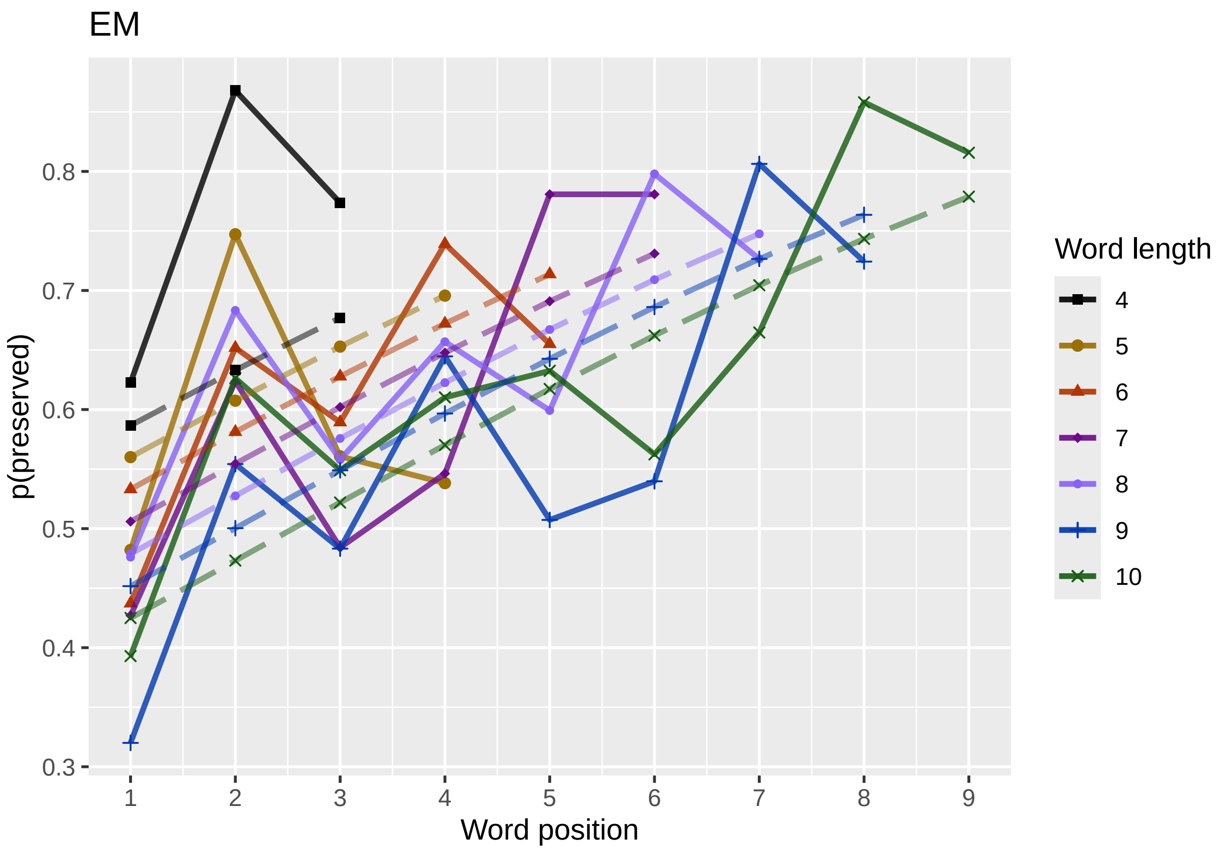
```
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.587 0.633 0.677 NA     NA     NA     NA     NA     NA
## 2       5 0.560 0.607 0.653  0.696 NA     NA     NA     NA     NA
## 3       6 0.533 0.581 0.628  0.672  0.714 NA     NA     NA     NA
## 4       7 0.506 0.555 0.602  0.648  0.691  0.731 NA     NA     NA
## 5       8 0.479 0.528 0.576  0.623  0.667  0.709  0.748 NA     NA
## 6       9 0.452 0.500 0.549  0.597  0.643  0.686  0.727  0.764 NA
## 7      10 0.425 0.473 0.522  0.570  0.617  0.662  0.704  0.743  0.779
```

```
fitted_pos_len_summary$stimlen<-factor(fitted_pos_len_summary$stimlen)
fitted_pos_len_summary$pos<-factor(fitted_pos_len_summary$pos)
# fitted_len_pos_plot <- ggplot(pos_len_summary,aes(x=pos,y=preserved,group=stimlen,shape=stimlen,color
# fitted_len_pos_plot <- fitted_len_pos_plot + geom_line(data=fitted_pos_len_summary, aes(x=pos,y=fitte
# geom_point(data=fitted_pos_len_summary,aes(x=pos,y=fitted,group=stimlen,shape=stimlen)) + ggtitle(pas

fitted_len_pos_plot <- plot_len_pos_obs_predicted(PosDat,
                                          paste0(PosDat$patient[1]),
                                          "LPFitted",
                                          NULL,
                                          palette_values,
                                          shape_values,
                                          obs_linetypes,
                                          pred_linetypes = c("longdash")
                                          )
```

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

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_percent_preserved_by_length_pos_wfit.png"),plot=fitted_len_pos
fitted_len_pos_plot
```



length and position without fragments to see if this changes position^2 influence

```r
# first number responses, then count resp with fragments - below we will eliminate fragments
# and re-run models

# number responses
resp_num<-0
prev_pos<-9999 # big number to initialize (so first position is smaller)
resp_num_array <- integer(length = nrow(PosDat))
for(i in seq(1,nrow(PosDat))){
  if(PosDat$pos[i] <= prev_pos){ # equal for resp length 1
    resp_num <- resp_num + 1
  }
  resp_num_array[i]<-resp_num
  prev_pos <- PosDat$pos[i]
}
PosDat$resp_num <- resp_num_array
```

```r
# count responses with fragments
resp_with_frag <- PosDat %>% group_by(resp_num) %>% summarise(frag = as.logical(sum(fragment)))
num_frag <- resp_with_frag %>% summarise(frag_sum = sum(frag), N=n())
num_frag
```

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1        4   692
```

```r
num_frag$percent_with_frag[1] <- (num_frag$frag_sum[1] / num_frag$N[1])*100
```

```
## Warning: Unknown or uninitialised column: `percent_with_frag`.
```

```r
print(sprintf("The number of responses with fragments was %d / %d = %.2f percent",
              num_frag$frag_sum[1],num_frag$N[1],num_frag$percent_with_frag[1]))
```

```
## [1] "The number of responses with fragments was 4 / 692 = 0.58 percent"
```

```r
write.csv(num_frag,paste0(TablesDir,CurPat,"_",CurTask,"_percent_fragments.csv"),row.names = FALSE)
```

```r
# length and position models for data without fragments
NoFragData <- PosDat %>% filter(fragment != 1)

NoFrag_LPRes<-TestModels(LPModelEquations,NoFragData)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen          pos
##      0.5698      -0.1076       0.2005
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4282 Residual
## Null Deviance:       5329
## Residual Deviance: 5176  AIC: 5588
## log likelihood:  -2587.915
## Nagelkerke R2:  0.04922994
## % pres/err predicted correctly:  -1848.767
## % of predictable range [ (model-null)/(1-null) ]:  0.03691055
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     0.83478     -0.14002      0.11036      0.01069
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4281 Residual
## Null Deviance:       5329
## Residual Deviance: 5175  AIC: 5589
## log likelihood:  -2587.411
## Nagelkerke R2:  0.04954898
## % pres/err predicted correctly:  -1848.549
## % of predictable range [ (model-null)/(1-null) ]:  0.03702427
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    0.668490    -0.110265     0.006242     0.148998
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4281 Residual
## Null Deviance:       5329
## Residual Deviance: 5175  AIC: 5589
## log likelihood:  -2587.525
## Nagelkerke R2:  0.04947681
## % pres/err predicted correctly:  -1848.577
## % of predictable range [ (model-null)/(1-null) ]:  0.03700949
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         0.359940         -0.079120         -0.045951          0.448605          0.005541
##      stimlen:pos
##        -0.031356
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4279 Residual
## Null Deviance:       5329
## Residual Deviance: 5174  AIC: 5592
## log likelihood:  -2586.756
## Nagelkerke R2:  0.04996309
## % pres/err predicted correctly:  -1847.973
## % of predictable range [ (model-null)/(1-null) ]:  0.03732406
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     -0.1415       0.1692
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4283 Residual
## Null Deviance:       5329
## Residual Deviance: 5204  AIC: 5614
## log likelihood:  -2601.786
## Nagelkerke R2:  0.04042246
## % pres/err predicted correctly:  -1862.355
## % of predictable range [ (model-null)/(1-null) ]:  0.02983583
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##  -0.1350980    0.0004931    0.1650332
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4282 Residual
## Null Deviance:       5329
## Residual Deviance: 5204  AIC: 5616
## log likelihood:  -2601.784
## Nagelkerke R2:  0.04042405
## % pres/err predicted correctly:  -1862.37
## % of predictable range [ (model-null)/(1-null) ]:  0.02982795
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.4932
```

```
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4284 Residual
## Null Deviance:       5329
## Residual Deviance: 5329  AIC: 5745
## log likelihood:  -2664.322
## Nagelkerke R2:  -3.120166e-16
## % pres/err predicted correctly:  -1919.66
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     0.59098     -0.01266
##
## Degrees of Freedom: 4284 Total (i.e. Null);  4283 Residual
## Null Deviance:       5329
## Residual Deviance: 5328  AIC: 5747
## log likelihood:  -2664.096
## Nagelkerke R2:  0.0001483008
## % pres/err predicted correctly:  -1919.391
## % of predictable range [ (model-null)/(1-null) ]:  0.0001400206
## ***************************
```

```
NoFragBestLPModel<-NoFrag_LPRes$ModelResult[[1]]
NoFragBestLPModelFormula<-NoFrag_LPRes$Model[[1]]

NoFragLPAICSummary<-data.frame(Model=NoFrag_LPRes$Model,
                        AIC=NoFrag_LPRes$AIC,
                        row.names = NoFrag_LPRes$Model)
NoFragLPAICSummary$DeltaAIC<-NoFragLPAICSummary$AIC-NoFragLPAICSummary$AIC[1]
NoFragLPAICSummary$AICexp<-exp(-0.5*NoFragLPAICSummary$DeltaAIC)
NoFragLPAICSummary$AICwt<-NoFragLPAICSummary$AICexp/sum(NoFragLPAICSummary$AICexp)
NoFragLPAICSummary$NagR2<-NoFrag_LPRes$NagR2

NoFragLPAICSummary <- merge(NoFragLPAICSummary,NoFrag_LPRes$CoefficientValues, by='row.names',sort=FALS
NoFragLPAICSummary <- subset(NoFragLPAICSummary, select = -c(Row.names))

write.csv(NoFragLPAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_no_fragments_len_pos_model_summary.c
kable(NoFragLPAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	5587.778	0.0000000	1.0000000	0.4436410	0.0492299	0.5697783	-0.1075961	0.2005051	NA	NA	NA
preserved ~ stimlen * pos	5588.777	0.9987892	0.6068980	0.2692409	0.0495490	0.8347832	-0.1400156	0.1103552	0.0106879	NA	NA
preserved ~ stimlen + I(pos^2) + pos	5589.085	1.3069603	0.5202321	0.2307964	0.0494768	0.6684896	-0.1102651	0.1489976	NA	0.0062415	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	5591.906	4.1280640	0.1269401	0.0563163	0.0499631	0.3599403	-0.0791198	0.4486054	-0.0313558	-0.0459509	0.0055406
preserved ~ pos	5614.082	26.3035249	0.0000019	0.0000009	0.0404225	-0.1414770	NA	0.1691560	NA	NA	NA
preserved ~ I(pos^2) + pos	5616.085	28.3066941	0.0000007	0.0000003	0.0404240	-0.1350980	NA	0.1650332	NA	0.0004931	NA
preserved ~ 1	5745.032	157.2536268	0.0000000	0.0000000	0.0000000	0.4932479	NA	NA	NA	NA	NA
preserved ~ stimlen	5746.713	158.9348440	0.0000000	0.0000000	0.0001483	0.5909841	-0.0126551	NA	NA	NA	NA

```r
# plot no fragment data

NoFragData$LPFitted<-fitted(NoFragBestLPModel)
nofrag_obs_len_pos_plot <- plot_len_pos_obs_predicted(NoFragData, NoFragData$patient[1],
                          NULL,palette_values=palette_values)
```

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

```r
# len/pos table
nofrag_fitted_pos_len_summary <- NoFragData %>% group_by(stimlen,pos) %>% summarise(fitted = mean(LPFit
```

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

```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

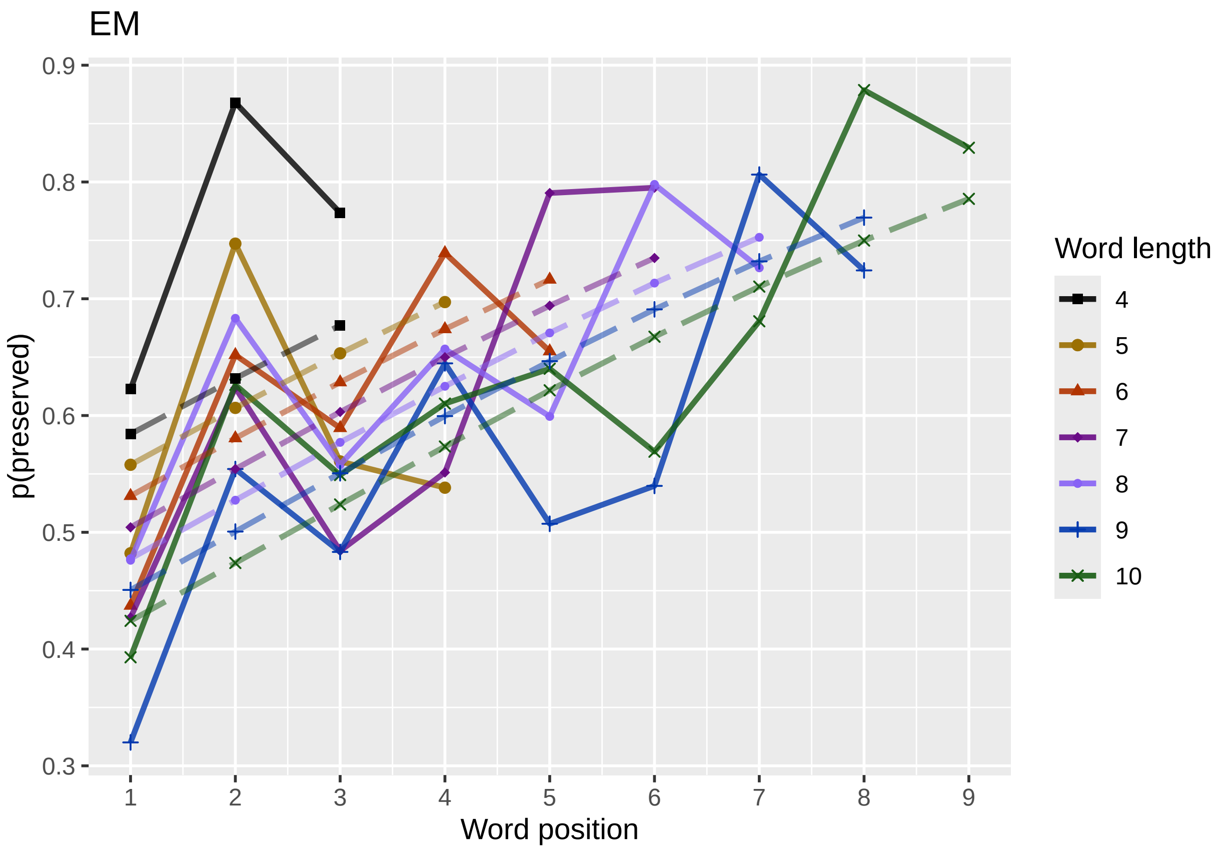
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.584 0.632 0.677 NA     NA     NA     NA     NA     NA
## 2       5 0.558 0.607 0.653  0.697 NA     NA     NA     NA     NA
## 3       6 0.531 0.581 0.628  0.674  0.716 NA     NA     NA     NA
## 4       7 0.504 0.554 0.603  0.650  0.694  0.735 NA     NA     NA
## 5       8 0.477 0.527 0.577  0.625  0.671  0.713  0.753 NA     NA
## 6       9 0.451 0.501 0.551  0.600  0.647  0.691  0.732  0.769 NA
## 7      10 0.424 0.474 0.524  0.573  0.622  0.667  0.710  0.750  0.786
```

```r
fitted_pos_len_summary$stimlen<-factor(nofrag_fitted_pos_len_summary$stimlen)
fitted_pos_len_summary$pos<-factor(nofrag_fitted_pos_len_summary$pos)
# fitted_len_pos_plot <- ggplot(pos_len_summary,aes(x=pos,y=preserved,group=stimlen,shape=stimlen,color
# fitted_len_pos_plot <- fitted_len_pos_plot + geom_line(data=fitted_pos_len_summary, aes(x=pos,y=fitte
# geom_point(data=fitted_pos_len_summary,aes(x=pos,y=fitted,group=stimlen,shape=stimlen)) + ggtitle(pas

nofrag_fitted_len_pos_plot <- plot_len_pos_obs_predicted(NoFragData,
                                  paste0(NoFragData$patient[1]),
                                  "LPFitted",
                                  NULL,
                                  palette_values,
                                  shape_values,
                                  obs_linetypes,
                                  pred_linetypes = c("longdash")
```

```
                                        )
```

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

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"no_fragments_percent_preserved_by_length_pos_wfit.png"),plot=no
nofrag_fitted_len_pos_plot
```



back to full data

```
results_report_DF <- AddReportLine(results_report_DF,"min preserved",min_preserved)
results_report_DF <- AddReportLine(results_report_DF,"max preserved",max_preserved)
print(sprintf("Min/max preserved range: %.2f - %.2f",min_preserved,max_preserved))
```

```
## [1] "Min/max preserved range: 0.27 - 0.92"
```

```
# find the average difference between lengths and the average difference
# between positions taking the other factor into account

# choose fitted or raw -- we use fitted values because they are smoothed averages
# that take into account the influence of the other variable
table_to_use <- fitted_pos_len_table
# take away column with length information
table_to_use <- table_to_use[,2:ncol(table_to_use)]

# take averages for positions
# don't want downward estimates influenced by return upward of U
# therefore, for downward influence, use only the values before the min
# take the difference between each value (differences between position proportion correct) **NOTE** pro
# average the difference in probabilities

# in case min is close to the end or we are not using a min (for non-U-shaped)
# functions, we need to get differences _first_ and then average those (e.g.
# if there is an effect of length and we just average position data,
# later positions) will have a lower average (because of the length effect)
# even if all position functions go upward

table_pos_diffs <- t(diff(t(as.matrix(table_to_use))))
ncol_pos_diff_matrix <- ncol(table_pos_diffs)
first_col_mean <- mean(as.numeric(unlist(table_pos_diffs[,1])))
potential_u_shape <- 0
if(first_col_mean < 0){ # downward, so potential U-shape
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs >= 0] <- NA
  potential_u_shape <- 1
}else{
  # upward - use the positive values
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs <= 0] <- NA
}
average_pos_diffs <- apply(filtered_table_pos_diffs,2,mean,na.rm=TRUE)
OA_mean_pos_diff <- mean(average_pos_diffs,na.rm=TRUE)

table_len_diffs <- diff(as.matrix(table_to_use))
average_len_diffs <- apply(table_len_diffs,1,mean,na.rm=TRUE)
OA_mean_len_diff <- mean(average_len_diffs,na.rm=TRUE)
CurrentLabel<-"mean change in probability for each additional length: "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional length: "
```

```
print(OA_mean_len_diff)
```

```
## [1] -0.02524279
```

```
results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,OA_mean_len_diff)

CurrentLabel<-"mean change in probability for each additional position (excluding  U-shape): "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional position (excluding  U-shape): "
```

```
print(OA_mean_pos_diff)
```

```
## [1] 0.04273433
```

```
results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,OA_mean_pos_diff)
```

```
if(potential_u_shape){ # downward, so potential U-shape
  # take only positive values from the table
  filtered_pos_upward_u<-table_pos_diffs
  filtered_pos_upward_u[table_pos_diffs <= 0] <- NA
```

```r
  average_pos_u_diffs <- apply(filtered_pos_upward_u,
                               2,mean,na.rm=TRUE)
  OA_mean_pos_u_diff <- mean(average_pos_u_diffs,na.rm=TRUE)
  if(is.nan(OA_mean_pos_u_diff) | (OA_mean_pos_u_diff <= 0)){
    print("No U-shape in this participant")
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
    potential_u_shape <- FALSE
  }else{
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 1)

    CurrentLabel<-"Average upward change after U minimum"
    print(CurrentLabel)
    print(OA_mean_pos_u_diff)
    results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, OA_mean_pos_u_diff)

    CurrentLabel<-"Proportion of average downward change"
    prop_ave_down <- abs(OA_mean_pos_u_diff/OA_mean_pos_diff)
    results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, prop_ave_down)
  }
}else{
  print("No U-shape in this participant")
  results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
}
```

```
## [1] "No U-shape in this participant"
```

```r
if(potential_u_shape){
  n_rows<-nrow(table_to_use)
  downward_dist <- numeric(n_rows)
  upward_dist <- numeric(n_rows)
  for(i in seq(1,n_rows)){
    current_row <- as.numeric(unlist(table_to_use[i,1:9]))
    current_row <- current_row[!is.na(current_row)]
    current_row_len <- length(current_row)
    row_min <- min(current_row,na.rm=TRUE)
    min_pos <- which(current_row == row_min)
    left_max <- max(current_row[1:min_pos],na.rm=TRUE)
    right_max <- max(current_row[min_pos:current_row_len])
    left_diff <- left_max - row_min
    right_diff <- right_max - row_min
    downward_dist[i]<-left_diff
    upward_dist[i]<-right_diff
  }
  print("differences from left max to min for each row: ")
  print(downward_dist)
  print("differences from min to right max for each row: ")
  print(upward_dist)

  # Use the max return upward (min of upward_dist) and then the
  # downward distance from the left for the same row

  print(" ")
  biggest_return_upward_row <- which(upward_dist == max(upward_dist))
  CurrentLabel <- "row with biggest return upward"
  print(CurrentLabel)
  print(biggest_return_upward_row)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, biggest_return_upward_row)

  print(" ")
  CurrentLabel<-"downward distance for row with the largest upward value"
  print(CurrentLabel)
  print(downward_dist[biggest_return_upward_row])
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, downward_dist[biggest_return_upwa

  CurrentLabel<-"return upward value"
  print(upward_dist[biggest_return_upward_row])
   results_report_DF <- AddReportLine(results_report_DF,
                                      CurrentLabel,
                                      upward_dist[biggest_return_upward_row])

  print(" ")
  # percentage return upward
  percentage_return_upward <- upward_dist[biggest_return_upward_row]/downward_dist[biggest_return_upwar
  CurrentLabel <- "Return upward as a proportion of the downward distance:"
  print(CurrentLabel)
  print(percentage_return_upward)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,
                        percentage_return_upward)
}else{
  print("no U-shape in this participant")
}
```

```
## [1] "no U-shape in this participant"
```

```
FLPModelEquations<-c(
  # models with log frequency, but no three-way interactions (due to difficulty interpreting and small
            "preserved ~ stimlen*log_freq",
            "preserved ~ stimlen+log_freq",
            "preserved ~ pos*log_freq",
            "preserved ~ pos+log_freq",
            "preserved ~ stimlen*log_freq + pos*log_freq",
            "preserved ~ stimlen*log_freq + pos",
            "preserved ~ stimlen + pos*log_freq",
            "preserved ~ stimlen + pos + log_freq",
            "preserved ~ (I(pos^2)+pos)*log_freq",
            "preserved ~ stimlen*log_freq + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen*log_freq + I(pos^2) + pos",
            "preserved ~ stimlen + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen + I(pos^2) + pos + log_freq",

            # models without frequency
            "preserved ~ 1",
            "preserved ~ stimlen",
            "preserved ~ pos",
            "preserved ~ stimlen + pos",
            "preserved ~ stimlen*pos",
            "preserved ~ I(pos^2)+pos",
            "preserved ~ stimlen + I(pos^2) + pos",
            "preserved ~ stimlen * (I(pos^2) + pos)"
)

FLPRes<-TestModels(FLPModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)          stimlen         log_freq              pos  stimlen:log_freq
##          0.42477         -0.09006          0.22148          0.19546          -0.02344
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4293 Residual
## Null Deviance:       5350
## Residual Deviance: 5193  AIC: 5613
## log likelihood:  -2596.501
## Nagelkerke R2:  0.05039498
## % pres/err predicted correctly:  -1855.17
## % of predictable range [ (model-null)/(1-null) ]:  0.03784149
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)          stimlen         log_freq         I(pos^2)              pos
##         0.489473        -0.091905         0.219305         0.003992         0.162421
## stimlen:log_freq
##        -0.023177
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4292 Residual
## Null Deviance:       5350
## Residual Deviance: 5193  AIC: 5615
## log likelihood:  -2596.339
## Nagelkerke R2:  0.05049678
## % pres/err predicted correctly:  -1855.09
## % of predictable range [ (model-null)/(1-null) ]:  0.03788263
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##        0.4247325        -0.0900155         0.2215437         0.1953694        -0.0232720
##     log_freq:pos
##       -0.0003738
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4292 Residual
## Null Deviance:       5350
## Residual Deviance: 5193  AIC: 5615
## log likelihood:  -2596.5
## Nagelkerke R2:  0.05039555
## % pres/err predicted correctly:  -1855.178
## % of predictable range [ (model-null)/(1-null) ]:  0.03783702
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     0.49215     -0.09615      0.19542      0.03996
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5198  AIC: 5618
## log likelihood:  -2599.201
## Nagelkerke R2:  0.04869253
## % pres/err predicted correctly:  -1857.97
## % of predictable range [ (model-null)/(1-null) ]:  0.03638982
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##         4.899e-01         -9.194e-02          2.207e-01          4.026e-03          1.622e-01
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##        -2.330e-02          9.333e-05         -5.797e-04
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4290 Residual
## Null Deviance:       5350
## Residual Deviance: 5193  AIC: 5619
## log likelihood:  -2596.339
## Nagelkerke R2:  0.05049712
## % pres/err predicted correctly:  -1855.088
## % of predictable range [ (model-null)/(1-null) ]:  0.03788386
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     0.482682     -0.094456      0.193583      0.066793     -0.007261
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4293 Residual
## Null Deviance:       5350
## Residual Deviance: 5198  AIC: 5619
## log likelihood:  -2598.8
## Nagelkerke R2:  0.0489454
## % pres/err predicted correctly:  -1857.713
## % of predictable range [ (model-null)/(1-null) ]:  0.03652291
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    0.568283    -0.098243     0.004763     0.155995     0.039880
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4293 Residual
## Null Deviance:       5350
## Residual Deviance: 5198  AIC: 5620
## log likelihood:  -2598.97
## Nagelkerke R2:  0.0488384
## % pres/err predicted correctly:  -1857.881
## % of predictable range [ (model-null)/(1-null) ]:  0.0364359
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      0.5902      -0.1088       0.1952
```

```
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5204  AIC: 5620
## log likelihood:  -2601.84
## Nagelkerke R2:  0.04702635
## % pres/err predicted correctly:  -1860.154
## % of predictable range [ (model-null)/(1-null) ]:  0.03525783
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    0.800533    -0.134508     0.123872     0.008441
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5203  AIC: 5621
## log likelihood:  -2601.52
## Nagelkerke R2:  0.04722832
## % pres/err predicted correctly:  -1860.032
## % of predictable range [ (model-null)/(1-null) ]:  0.03532102
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    0.667962    -0.110930     0.004872     0.154833
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5203  AIC: 5621
## log likelihood:  -2601.598
## Nagelkerke R2:  0.04717953
## % pres/err predicted correctly:  -1860.064
## % of predictable range [ (model-null)/(1-null) ]:  0.03530463
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          0.545328          -0.096172           0.003812           0.161646           0.051117
## I(pos^2):log_freq       pos:log_freq
##         -0.001036           0.001999
##
```

```
## Degrees of Freedom: 4297 Total (i.e. Null);  4291 Residual
## Null Deviance:       5350
## Residual Deviance: 5197  AIC: 5623
## log likelihood:  -2598.6
## Nagelkerke R2:  0.04907184
## % pres/err predicted correctly:  -1857.612
## % of predictable range [ (model-null)/(1-null) ]:  0.03657553
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        0.408006         -0.084538         -0.038086          0.403771          0.004559
##      stimlen:pos
##       -0.026092
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4292 Residual
## Null Deviance:       5350
## Residual Deviance: 5202  AIC: 5625
## log likelihood:  -2601.075
## Nagelkerke R2:  0.04750949
## % pres/err predicted correctly:  -1859.622
## % of predictable range [ (model-null)/(1-null) ]:  0.0355337
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)          pos      log_freq  pos:log_freq
##     -0.14795      0.16737       0.09863      -0.01048
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5217  AIC: 5638
## log likelihood:  -2608.693
## Nagelkerke R2:  0.04269066
## % pres/err predicted correctly:  -1867.581
## % of predictable range [ (model-null)/(1-null) ]:  0.03140784
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##    -0.15092      0.16946      0.06056
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5219  AIC: 5638
## log likelihood:  -2609.53
## Nagelkerke R2:  0.04216023
## % pres/err predicted correctly:  -1868.211
## % of predictable range [ (model-null)/(1-null) ]:  0.03108161
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##        -0.163800          -0.001400           0.178342           0.078758          -0.001569
##      pos:log_freq
##         0.002499
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4292 Residual
## Null Deviance:       5350
## Residual Deviance: 5217  AIC: 5641
## log likelihood:  -2608.587
## Nagelkerke R2:  0.04275811
## % pres/err predicted correctly:  -1867.41
## % of predictable range [ (model-null)/(1-null) ]:  0.03149632
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     -0.1285       0.1633
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5647
## log likelihood:  -2616.075
## Nagelkerke R2:  0.03800482
## % pres/err predicted correctly:  -1874.037
## % of predictable range [ (model-null)/(1-null) ]:  0.02806124
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##  -0.1404349   -0.0009136    0.1709600
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5649
## log likelihood:  -2616.066
## Nagelkerke R2:  0.03801037
## % pres/err predicted correctly:  -1873.992
## % of predictable range [ (model-null)/(1-null) ]:  0.0280847
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         0.445405          0.002801          0.217449         -0.023024
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5339  AIC: 5764
## log likelihood:  -2669.452
## Nagelkerke R2:  0.00363815
## % pres/err predicted correctly:  -1922.957
## % of predictable range [ (model-null)/(1-null) ]:  0.00270337
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     0.51166     -0.00320      0.03858
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5344  AIC: 5769
## log likelihood:  -2672.137
## Nagelkerke R2:  0.001886865
## % pres/err predicted correctly:  -1925.762
## % of predictable range [ (model-null)/(1-null) ]:  0.001249372
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.4863
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4297 Residual
## Null Deviance:       5350
## Residual Deviance: 5350  AIC: 5769
## log likelihood:  -2675.026
## Nagelkerke R2:  -3.118626e-16
## % pres/err predicted correctly:  -1928.172
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     0.60613     -0.01551
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5349  AIC: 5771
## log likelihood:  -2674.685
## Nagelkerke R2:  0.0002231365
## % pres/err predicted correctly:  -1927.778
## % of predictable range [ (model-null)/(1-null) ]:  0.0002043344
## ***************************
```

```
BestFLPModel<-FLPRes$ModelResult[[1]]
BestFLPModelFormula<-FLPRes$Model[[1]]

FLPAICSummary<-data.frame(Model=FLPRes$Model,
                       AIC=FLPRes$AIC,row.names=FLPRes$Model)
FLPAICSummary$DeltaAIC<-FLPAICSummary$AIC-FLPAICSummary$AIC[1]
FLPAICSummary$AICexp<-exp(-0.5*FLPAICSummary$DeltaAIC)
FLPAICSummary$AICwt<-FLPAICSummary$AICexp/sum(FLPAICSummary$AICexp)
FLPAICSummary$NagR2<-FLPRes$NagR2

FLPAICSummary <- merge(FLPAICSummary,FLPRes$CoefficientValues,
                          by='row.names',sort=FALSE)
FLPAICSummary <- subset(FLPAICSummary, select = -c(Row.names))

write.csv(FLPAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_freq_len_pos_model_summary.csv"),row.names
kable(FLPAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + pos	5613.04[illegible]	0.0000000	1.0000000	0.47886[illegible]	[illegible]	-0.0900630	0.2214799	-0.0234421	0.195[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	5614.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0919047	0.2193048	-0.0231773	0.162[illegible]	NA	0.0039[illegible]	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	5615.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0900155	0.2215437	-0.0232720	0.195[illegible]	-0.0003738	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + pos + log_freq	5617.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0961470	0.0399630	NA	0.1954212	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	5618.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0919440	0.2207277	-0.0233013	0.1622344	NA	-0.0005797	0.0040258	NA	9.33e-05	NA	NA
preserved ~ stimlen + pos * log_freq	5618.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0944555	0.0667928	NA	0.1935828	-0.0072608	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	5619.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0982428	0.0398797	NA	0.1559953	NA	NA	0.0047627	NA	NA	NA	NA
preserved ~ stimlen + pos	5619.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1088072	NA	NA	0.1951700	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	5621.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1345078	NA	NA	0.1238724	NA	NA	NA	NA	NA	0.0084414	NA
preserved ~ stimlen + I(pos^2) + pos	5621.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1109295	NA	NA	0.1548335	NA	NA	0.0048723	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	5622.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0961715	0.0511171	NA	[illegible]	[illegible]	NA	0.0038123	-0.0010364	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	5624.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0845381	NA	NA	0.4037707	NA	NA	-0.0380861	NA	NA	-0.0260917	0.0045592
preserved ~ pos * log_freq	5637.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1479493	NA	0.0980273	NA	0.1673684	-0.0104759	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	5637.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1509228	NA	0.0605624	NA	0.1694597	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	5641.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1637996	NA	0.0787583	NA	[illegible]	[illegible]	NA	[illegible]	-0.0015686	NA	NA	NA
preserved ~ pos	5646.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1285371	NA	NA	NA	0.1632952	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	5648.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1404349	NA	NA	NA	0.1709600	NA	NA	-0.0009136	NA	NA	NA	NA
preserved ~ stimlen * log_freq	5764.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0452802	0.1374491	-0.0230240	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC Delta**AIC**expAICwNagR2(Intercept)stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)		
preserved ~ stimlen + log_freq	5769.229.1849850.000000185616574 -0.0032004	0.0385846		NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	5769.396.35281000000000.486643	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	5770.8671.8071020000000.261301 -0.0155118	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

```
## [1] "preserved ~ stimlen * log_freq + pos"
```

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          0.42477          -0.09006           0.22148           0.19546          -0.02344
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4293 Residual
## Null Deviance:       5350
## Residual Deviance: 5193  AIC: 5613
```

```r
# do a median split on frequency to plot hf/ls effects (analysis is continuous)
median_freq <- median(PosDat$log_freq)
PosDat$freq_bin[PosDat$log_freq >= median_freq]<-"hf"
PosDat$freq_bin[PosDat$log_freq <  median_freq]<-"lf"
```

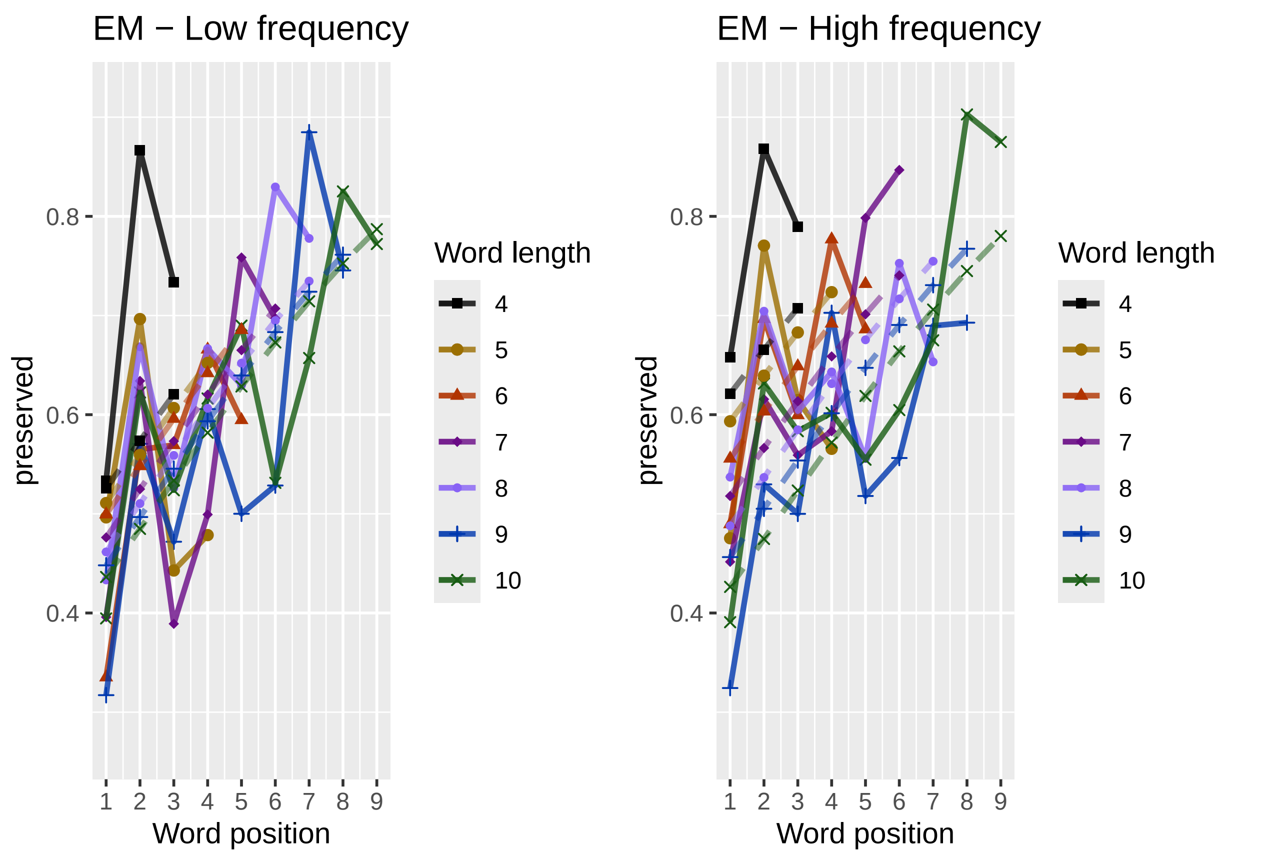
```r
PosDat$FLPFitted<-fitted(BestFLPModel)

HFDat <- PosDat[PosDat$freq_bin == "hf",]
LFDat <- PosDat[PosDat$freq_bin == "lf",]

HF_Plot <- plot_len_pos_obs_predicted(HFDat,paste0(CurPat," - High frequency"),"FLPFitted",c(min_preser
```

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale for y is already present. Adding another scale for y, which will replace the existing
## scale.
```

```r
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale for y is already present. Adding another scale for y, which will replace the existing
## scale.
```

```r
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_frequency_effect_length_pos_wfit.png"),device="png",unit="cm"
print(Both_Plots)
```



```r
# only main effects
MEModelEquations<-c(
  "preserved ~ CumPres",
  "preserved ~ CumErr",
  "preserved ~ (I(pos^2)+pos)",
  "preserved ~ pos",
  "preserved ~ stimlen",
  "preserved ~ 1"
)
MERes<-TestModels(MEModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)      CumPres
##     0.05626      0.28718
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5182  AIC: 5592
## log likelihood:  -2591.038
## Nagelkerke R2:  0.05383288
## % pres/err predicted correctly:  -1850.655
## % of predictable range [ (model-null)/(1-null) ]:  0.04018159
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     -0.1285       0.1633
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5647
## log likelihood:  -2616.075
## Nagelkerke R2:  0.03800482
## % pres/err predicted correctly:  -1874.037
## % of predictable range [ (model-null)/(1-null) ]:  0.02806124
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##  -0.1404349   -0.0009136    0.1709600
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5649
## log likelihood:  -2616.066
## Nagelkerke R2:  0.03801037
## % pres/err predicted correctly:  -1873.992
## % of predictable range [ (model-null)/(1-null) ]:  0.0280847
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      0.3844       0.0812
##
```

```
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5340  AIC: 5761
## log likelihood:  -2669.982
## Nagelkerke R2:  0.003292856
## % pres/err predicted correctly:  -1924.171
## % of predictable range [ (model-null)/(1-null) ]:  0.002073937
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.4863
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4297 Residual
## Null Deviance:       5350
## Residual Deviance: 5350  AIC: 5769
## log likelihood:  -2675.026
## Nagelkerke R2:  -3.118626e-16
## % pres/err predicted correctly:  -1928.172
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     0.60613     -0.01551
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5349  AIC: 5771
## log likelihood:  -2674.685
## Nagelkerke R2:  0.0002231365
## % pres/err predicted correctly:  -1927.778
## % of predictable range [ (model-null)/(1-null) ]:  0.0002043344
## ***************************
```

```
BestMEModel<-MERes$ModelResult[[ BestModelIndexL1 ]]
BestMEModelFormula<-MERes$Model[[BestModelIndexL1]]

MEAICSummary<-data.frame(Model=MERes$Model,
                       AIC=MERes$AIC,row.names=MERes$Model)
MEAICSummary$DeltaAIC<-MEAICSummary$AIC-MEAICSummary$AIC[1]
MEAICSummary$AICexp<-exp(-0.5*MEAICSummary$DeltaAIC)
MEAICSummary$AICwt<-MEAICSummary$AICexp/sum(MEAICSummary$AICexp)
MEAICSummary$NagR2<-MERes$NagR2

MEAICSummary <- merge(MEAICSummary,MERes$CoefficientValues,
```

```r
                          by='row.names',sort=FALSE)
MEAICSummary <- subset(MEAICSummary, select = -c(Row.names))

write.csv(MEAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_main_effects_model_summary.csv"),row.names
kable(MEAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	5591.734	0.00000	1	1	0.0538320	0.0562566	0.2871761	NA	NA	NA	NA
preserved ~ pos	5646.712	54.97797	0	0	0.0380048	-0.1285371	NA	NA	NA	0.1632952	NA
preserved ~ (I(pos^2) + pos)	5648.667	56.93316	0	0	0.0380104	-0.1404349	NA	NA	-0.0009136	0.1709600	NA
preserved ~ CumErr	5761.359	169.62505	0	0	0.0032920	0.3843637	NA	0.0812011	NA	NA	NA
preserved ~ 1	5769.397	177.66343	0	0	0.0000000	0.4862643	NA	NA	NA	NA	NA
preserved ~ stimlen	5770.861	179.12774	0	0	0.0002230	0.6061301	NA	NA	NA	NA	-0.0155118

```r
if(DoSimulations){
      BestMEModelFormulaRnd <- BestMEModelFormula
    if(grepl("CumPres",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumPres","RndCumPres",BestMEModelFormulaRnd)
    }else if(grepl("CumErr",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumErr","RndCumErr",BestMEModelFormulaRnd)
    }

    RndModelAIC<-numeric(length=RandomSamples)
    for(rindex in seq(1,RandomSamples)){
    # Shuffle cumulative values
      PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
      PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
      BestModelRnd <- glm(as.formula(BestMEModelFormulaRnd),
                          family="binomial",data=PosDat)
      RndModelAIC[rindex] <- BestModelRnd$aic
    }
    ModelNames<-c(paste0("***",BestMEModelFormula),
                  rep(BestMEModelFormulaRnd,RandomSamples))
    AICValues <- c(BestMEModel$aic,RndModelAIC)
    BestMEModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
    BestMEModelRndDF <- BestMEModelRndDF %>% arrange(AIC)
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random average"),
                                         AIC=c(mean(RndModelAIC))))
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random SD"),
                                         AIC=c(sd(RndModelAIC))))

    write.csv(BestMEModelRndDF,
          paste0(TablesDir,CurPat,"_",CurTask,
                "_best_main_effects_model_with_random_cum_term.csv"),
          row.names = FALSE)
}
```

```
syll_component_summary <- PosDat %>%
  group_by(syll_component) %>% summarise(MeanPres = mean(preserved),
                                         N = n())
write.csv(syll_component_summary,paste0(TablesDir,CurPat,"_",CurTask,"_syllable_component_summary.csv")
kable(syll_component_summary)
```

syll_component	MeanPres	N
1	0.4482922	563
O	0.5405880	1984
P	0.0294118	34
S	0.3389785	248
V	0.8519071	1469

```
#  main effects models for data without satellite positions

keep_components = c("O","V","1")
OV1Data <- PosDat[PosDat$syll_component %in% keep_components,]
OV1Data <- OV1Data %>% select(stim_number,
                              stimlen,stim,pos,
                              preserved,syll_component)
OV1Data$CumPres <- CalcCumPres(OV1Data)
OV1Data$CumErr <- CalcCumErrFromPreserved(OV1Data)

SimpSyllMEAICSummary <- EvaluateSubsetData(OV1Data,MEModelEquations)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.1676       0.2835
##
## Degrees of Freedom: 4015 Total (i.e. Null);  4014 Residual
## Null Deviance:       4893
## Residual Deviance: 4749  AIC: 5152
## log likelihood:  -2374.456
## Nagelkerke R2:  0.05016865
## % pres/err predicted correctly:  -1690.359
## % of predictable range [ (model-null)/(1-null) ]:  0.03706066
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     -0.0149       0.1580
##
## Degrees of Freedom: 4015 Total (i.e. Null);  4014 Residual
## Null Deviance:       4893
## Residual Deviance: 4792  AIC: 5200
## log likelihood:  -2396.228
## Nagelkerke R2:  0.03523771
## % pres/err predicted correctly:  -1710.731
## % of predictable range [ (model-null)/(1-null) ]:  0.02546214
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##  -0.0005554    0.0011090    0.1487417
##
## Degrees of Freedom: 4015 Total (i.e. Null);  4013 Residual
## Null Deviance:       4893
## Residual Deviance: 4792  AIC: 5202
## log likelihood:  -2396.216
## Nagelkerke R2:  0.03524586
## % pres/err predicted correctly:  -1710.762
## % of predictable range [ (model-null)/(1-null) ]:  0.02544471
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      0.5146       0.0614
##
## Degrees of Freedom: 4015 Total (i.e. Null);  4014 Residual
## Null Deviance:       4893
## Residual Deviance: 4889  AIC: 5302
## log likelihood:  -2444.423
## Nagelkerke R2:  0.001603667
## % pres/err predicted correctly:  -1753.96
## % of predictable range [ (model-null)/(1-null) ]:  0.0008508065
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       0.582
##
## Degrees of Freedom: 4015 Total (i.e. Null);  4015 Residual
## Null Deviance:       4893
## Residual Deviance: 4893  AIC: 5305
## log likelihood:  -2446.692
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1755.454
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    0.615397    -0.004323
##
## Degrees of Freedom: 4015 Total (i.e. Null);  4014 Residual
## Null Deviance:       4893
## Residual Deviance: 4893  AIC: 5307
## log likelihood:  -2446.668
## Nagelkerke R2:  1.717429e-05
## % pres/err predicted correctly:  -1755.417
## % of predictable range [ (model-null)/(1-null) ]:  2.125885e-05
## ***************************
```

```
write.csv(SimpSyllMEAICSummary,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV1_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	5152.089	0.00000	1	1	0.0501687	0.1675634	0.2834953	NA	NA	NA	NA
preserved ~ pos	5200.016	47.92700	0	0	0.0352377	-0.0148965	NA	NA	NA	0.1580180	NA
preserved ~ (I(pos^2) + pos)	5202.034	49.94438	0	0	0.0352459	-0.0005554	NA	NA	0.001109	0.1487417	NA
preserved ~ CumErr	5302.494	150.40461	0	0	0.0016037	0.5145937	NA	0.0614044	NA	NA	NA
preserved ~ 1	5305.282	153.19264	0	0	0.0000000	0.5819917	NA	NA	NA	NA	NA
preserved ~ stimlen	5307.300	155.21018	0	0	0.0000172	0.6153969	NA	NA	NA	NA	-0.004323

```
#  main effects models for data without satellite or coda positions
# (tradeoff -- takes away more complex segments, in addition to satellites, but
#  also reduces data)
```

```r
keep_components = c("O","V")
OVData <- PosDat[PosDat$syll_component %in% keep_components,]
OVData <- OVData %>% select(stim_number,
                            stimlen,stim,pos,
                            preserved,syll_component)
OVData$CumPres <- CalcCumPres(OVData)
OVData$CumErr <- CalcCumErrFromPreserved(OVData)

SimpSyllMEAICSummary2<-EvaluateSubsetData(OVData,MEModelEquations)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.3541       0.2865
##
## Degrees of Freedom: 3452 Total (i.e. Null);  3451 Residual
## Null Deviance:       4066
## Residual Deviance: 3964  AIC: 4320
## log likelihood:  -1982.137
## Nagelkerke R2:  0.0420348
## % pres/err predicted correctly:  -1399.597
## % of predictable range [ (model-null)/(1-null) ]:  0.03151001
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    -0.06468     -0.01907      0.30628
##
## Degrees of Freedom: 3452 Total (i.e. Null);  3450 Residual
## Null Deviance:       4066
## Residual Deviance: 3984  AIC: 4346
## log likelihood:  -1991.763
## Nagelkerke R2:  0.0341906
## % pres/err predicted correctly:  -1409.775
## % of predictable range [ (model-null)/(1-null) ]:  0.02447179
## ***************************
## model index:  4
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.1729       0.1471
##
## Degrees of Freedom: 3452 Total (i.e. Null);  3451 Residual
## Null Deviance:       4066
## Residual Deviance: 3990  AIC: 4351
## log likelihood:  -1994.954
## Nagelkerke R2:  0.0315804
## % pres/err predicted correctly:  -1412.869
## % of predictable range [ (model-null)/(1-null) ]:  0.02233273
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      0.5983       0.1476
##
## Degrees of Freedom: 3452 Total (i.e. Null);  3451 Residual
## Null Deviance:       4066
## Residual Deviance: 4052  AIC: 4418
## log likelihood:  -2026.047
## Nagelkerke R2:  0.005892751
## % pres/err predicted correctly:  -1441.63
## % of predictable range [ (model-null)/(1-null) ]:  0.002444859
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.7219
##
## Degrees of Freedom: 3452 Total (i.e. Null);  3452 Residual
## Null Deviance:       4066
## Residual Deviance: 4066  AIC: 4430
## log likelihood:  -2033.101
## Nagelkerke R2:  -3.208837e-16
## % pres/err predicted correctly:  -1445.166
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen
##    0.783548    -0.007993
##
## Degrees of Freedom: 3452 Total (i.e. Null);  3451 Residual
## Null Deviance:       4066
## Residual Deviance: 4066  AIC: 4432
## log likelihood:  -2033.032
## Nagelkerke R2:  5.792857e-05
## % pres/err predicted correctly:  -1444.978
## % of predictable range [ (model-null)/(1-null) ]:  0.0001298916
## ***************************
```

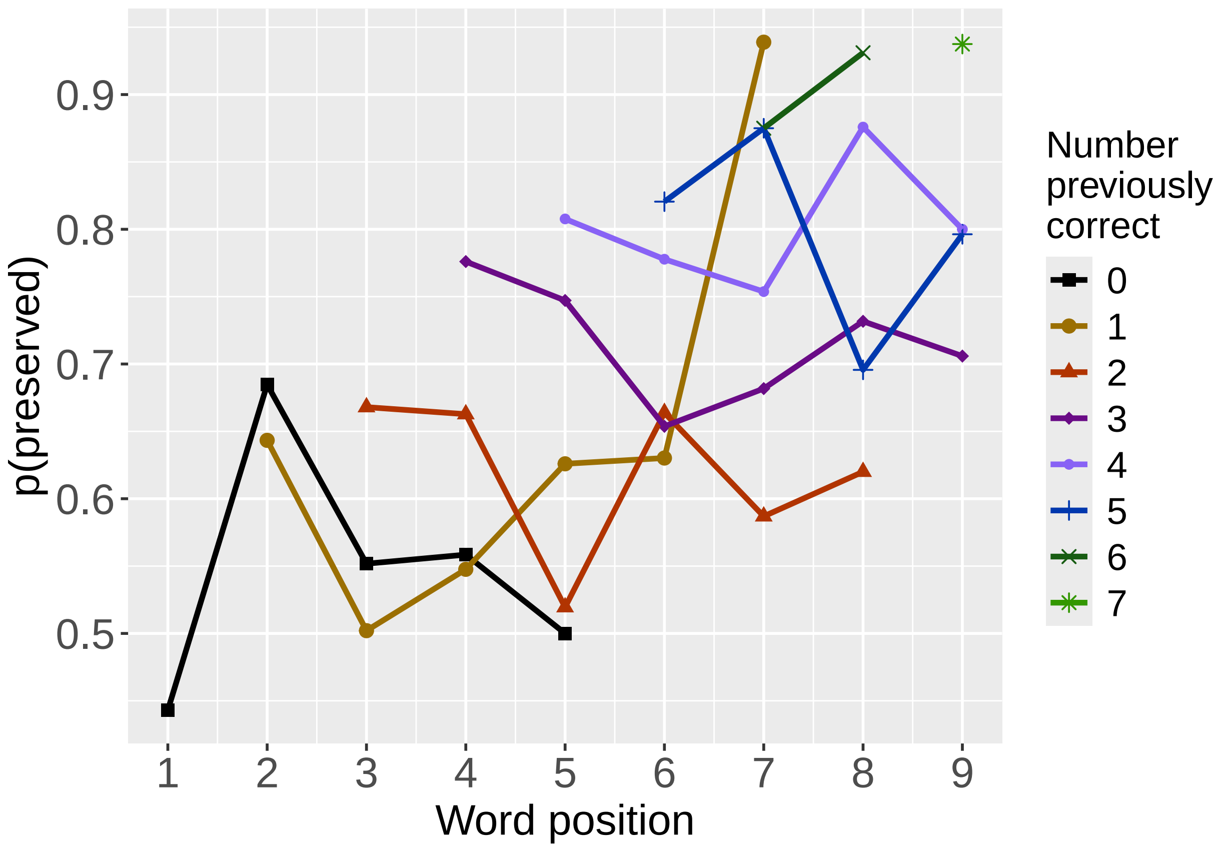
```r
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	4319.737	0.00000	1.0e+00	0.9999981	0.0420348	0.3541368	0.2865436	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	4346.237	26.50075	1.8e-06	0.0000018	0.0341906	-0.0646802	NA	NA	-0.0190746	0.3062772	NA
preserved ~ pos	4351.109	31.37193	2.0e-07	0.0000002	0.0315804	0.1728520	NA	NA	NA	0.1470586	NA
preserved ~ CumErr	4418.436	98.69922	0.0e+00	0.0000000	0.0058928	0.5983414	NA	0.1476479	NA	NA	NA
preserved ~ 1	4430.165	110.42862	0.0e+00	0.0000000	0.0000000	0.7219288	NA	NA	NA	NA	NA
preserved ~ stimlen	4432.134	112.39695	0.0e+00	0.0000000	0.0000579	0.7835484	NA	NA	NA	NA	-0.0079931

```r
# plot prev err and prev cor plots
PrevCorPlot<-PlotObsPreviousCorrect(PosDat,palette_values,shape_values)
```

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

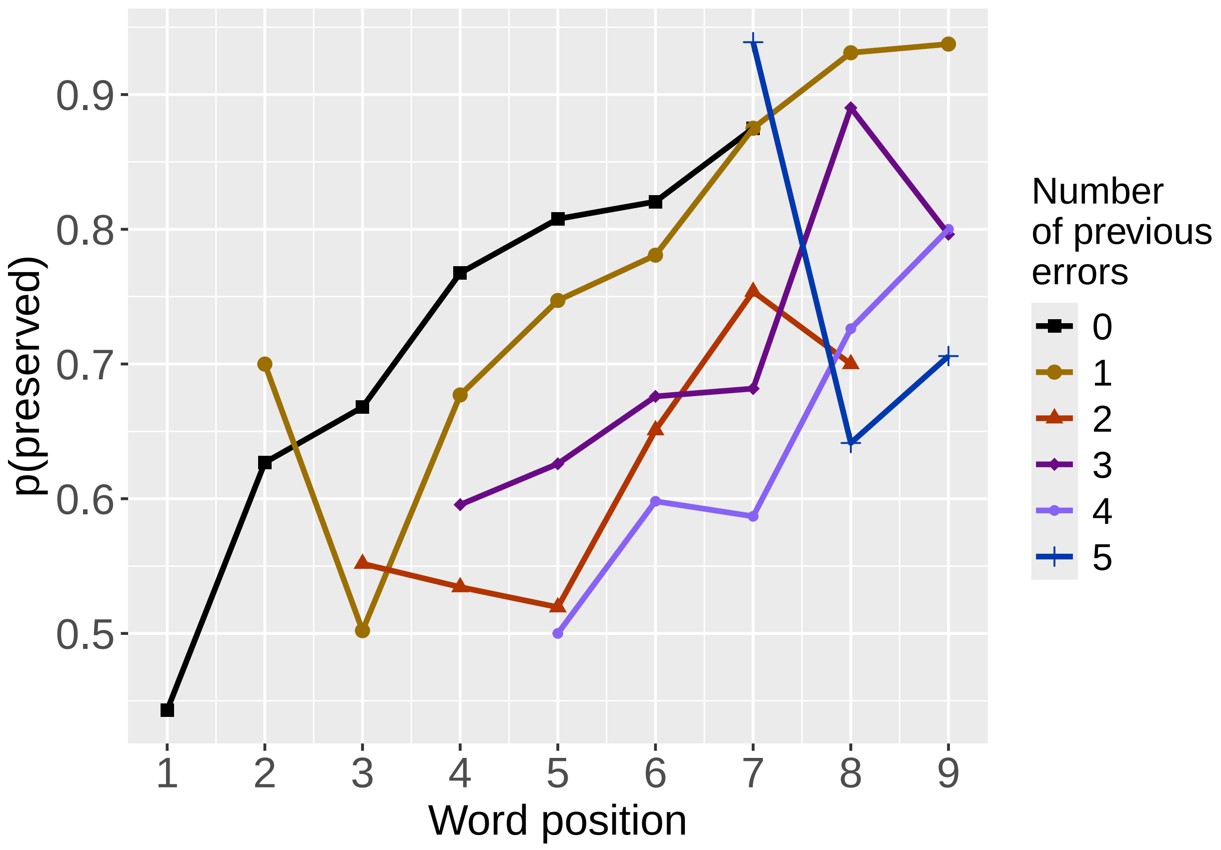
```r
print(PrevCorPlot)
```



```
PrevErrPlot<-PlotObsPreviousError(PosDat,palette_values,shape_values)
```

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

```
print(PrevErrPlot)
```



```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevCorPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevCorPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

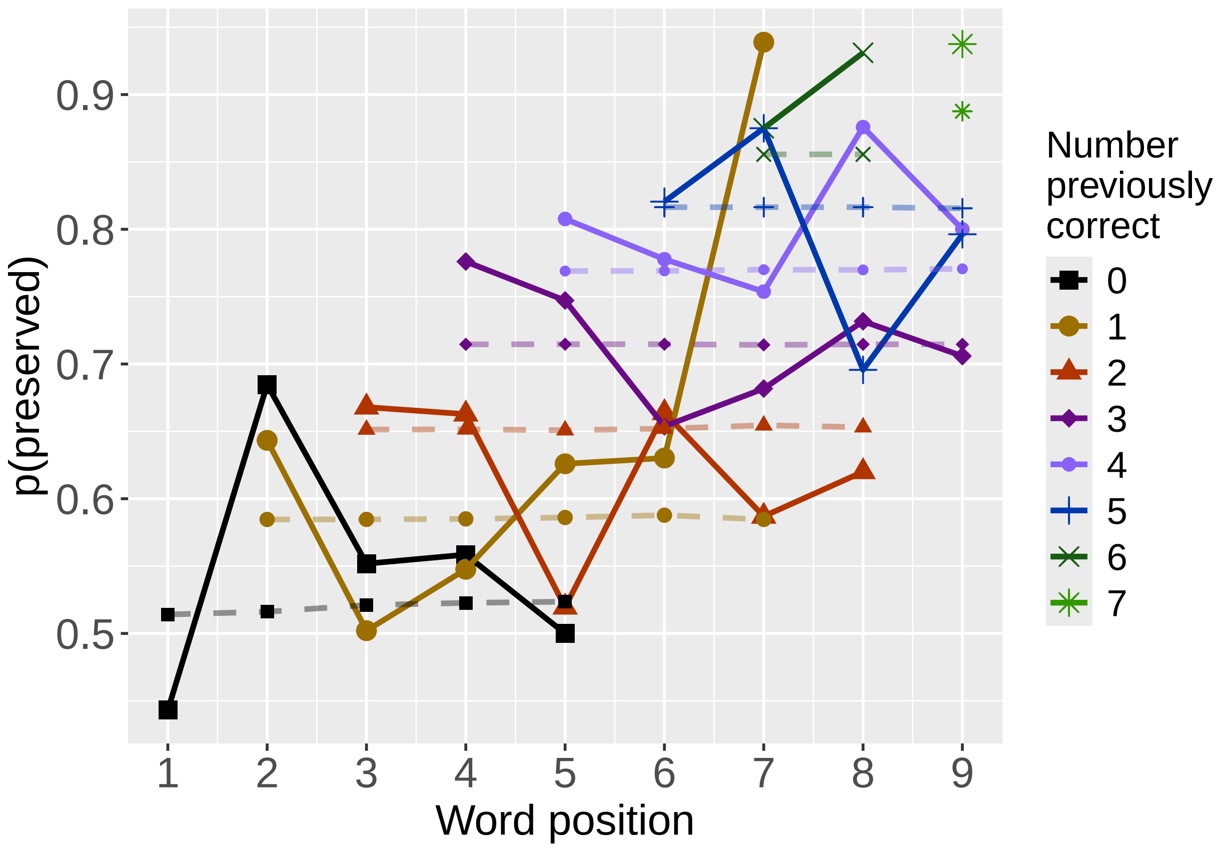
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

```
# plot prev err and prev cor with predicted values
MEModel<-MERes$ModelResult[[ BestModelIndexL1 ]]
PosDat$MEPred<-fitted(MEModel)

PrevCorPlot<-PlotObsPredPreviousCorrect(PosDat,"preserved","MEPred",palette_values,shape_values)
```

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

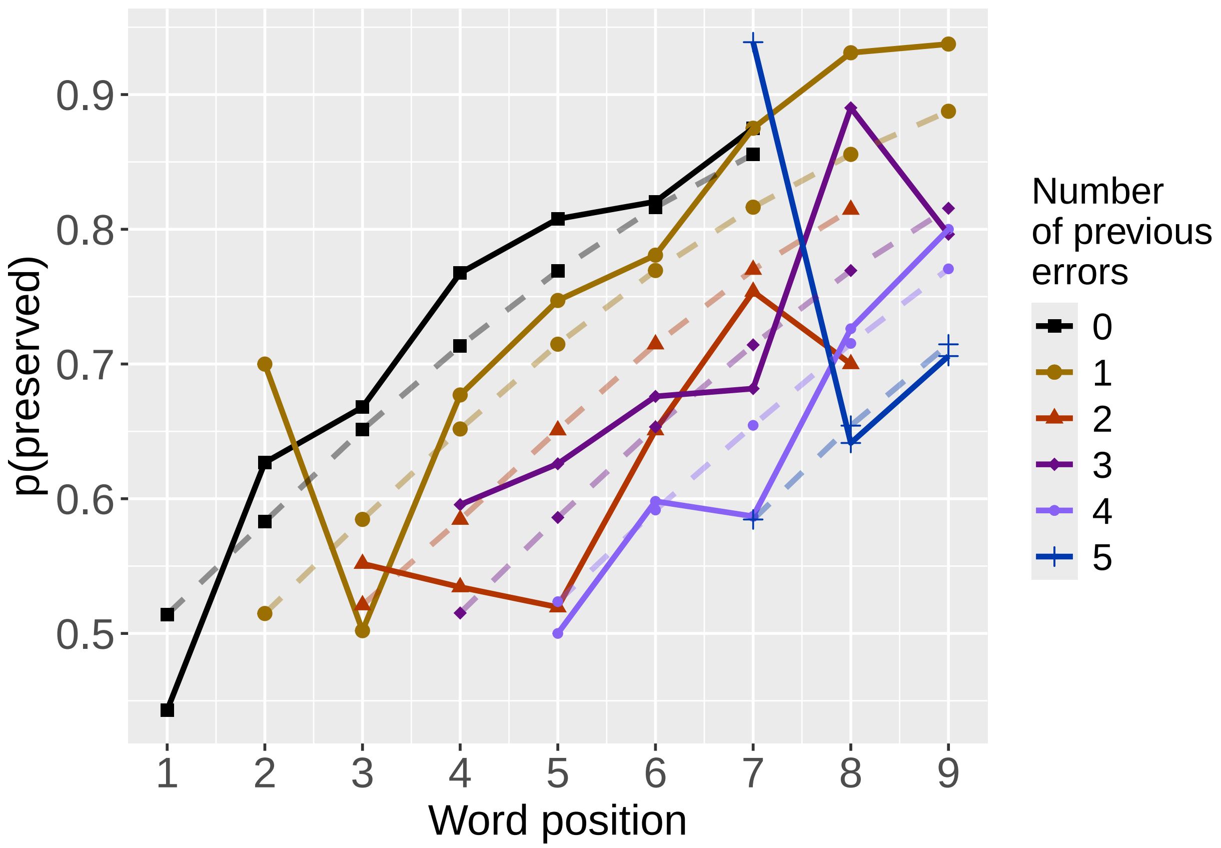
```
print(PrevCorPlot)
```



```
PrevErrPlot<-PlotObsPredPreviousError(PosDat,"preserved","MEPred",palette_values,shape_values)
```

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

```
print(PrevErrPlot)
```



```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"PrevCorPlot_ObsPredME")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevCorPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"PrevErrPlot_ObsPredME")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

```r
CumAICSummary <- NULL
```

```r
########
# level 2 -- Add position squared (quadratic with position)
########

# After establishing the primary variable, see about additions

BestMEFactor<-BestMEModelFormula
OtherFactor<-"I(pos^2) + pos"
OtherFactorName<-"quadratic_by_pos"

if(!grepl("pos",BestMEFactor,fixed = TRUE)){ # if best model does not include pos
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor,OtherFactorName)
  CumAICSummary <- AICSummary
  kable(AICSummary)
}
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     0.05626      0.28718
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5182  AIC: 5592
## log likelihood:  -2591.038
## Nagelkerke R2:  0.05383288
## % pres/err predicted correctly:  -1850.655
## % of predictable range [ (model-null)/(1-null) ]:  0.04018159
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos
##    0.012838     0.279619    -0.002418     0.026453
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5182  AIC: 5596
## log likelihood:  -2590.95
## Nagelkerke R2:  0.05388811
## % pres/err predicted correctly:  -1850.44
## % of predictable range [ (model-null)/(1-null) ]:  0.04029318
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##  -0.1404349   -0.0009136    0.1709600
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5649
## log likelihood:  -2616.066
## Nagelkerke R2:  0.03801037
## % pres/err predicted correctly:  -1873.992
## % of predictable range [ (model-null)/(1-null) ]:  0.0280847
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos
preserved ~ CumPres	5591.734	0.000000	1.000000	0.8698708	0.0538329	0.0562566	0.2871761	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos
preserved ~ CumPres + I(pos^2) + pos	5595.533	3.799634	0.149596	0.1301292	0.0538881	0.0128378	0.2796192	-0.0024184	0.0264535
preserved ~ I(pos^2) + pos	5648.667	56.933160	0.000000	0.0000000	0.0380104	-0.1404349	NA	-0.0009136	0.1709600

```r
########
# level 2 -- Add length
########

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

if(!grepl(OtherFactor,BestMEFactor,fixed = TRUE)){
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor)
  CumAICSummary <- ConcatAndAddMissingColumns(CumAICSummary,AICSummary)
  kable(AICSummary)
}
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres      stimlen
##      0.7142       0.3154      -0.0908
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5161  AIC: 5572
## log likelihood:  -2580.481
## Nagelkerke R2:  0.06045165
## % pres/err predicted correctly:  -1841.021
## % of predictable range [ (model-null)/(1-null) ]:  0.04517568
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     0.05626      0.28718
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5182  AIC: 5592
## log likelihood:  -2591.038
## Nagelkerke R2:  0.05383288
## % pres/err predicted correctly:  -1850.655
## % of predictable range [ (model-null)/(1-null) ]:  0.04018159
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     0.60613     -0.01551
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5349  AIC: 5771
## log likelihood:  -2674.685
## Nagelkerke R2:  0.0002231365
## % pres/err predicted correctly:  -1927.778
## % of predictable range [ (model-null)/(1-null) ]:  0.0002043344
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	stimlen
preserved ~ CumPres + stimlen	5572.166	0.00000	1.00e+00	0.9999436	0.0604516	0.7142150	0.3153941	-0.0908023
preserved ~ CumPres	5591.734	19.56741	5.64e-05	0.0000564	0.0538329	0.0562566	0.2871761	NA
preserved ~ stimlen	5770.861	198.69515	0.00e+00	0.0000000	0.0002231	0.6061301	NA	-0.0155118

```
########
# level 2 -- add cumulative preserved
########

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

if(!grepl(OtherFactor,BestMEFactor,fixed = TRUE)){
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor)
  CumAICSummary <- ConcatAndAddMissingColumns(CumAICSummary,AICSummary)
  kable(AICSummary)
}
```

```
########
# level 2 -- Add linear position (NOT quadratic)
########

BestMEFactor<-BestMEModelFormula
OtherFactor<-"pos"

if(!grepl(OtherFactor,BestMEFactor,fixed = TRUE)){
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor)
  CumAICSummary <- ConcatAndAddMissingColumns(CumAICSummary,AICSummary)
  kable(AICSummary)
}
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     0.05626      0.28718
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5182  AIC: 5592
## log likelihood:  -2591.038
## Nagelkerke R2:  0.05383288
## % pres/err predicted correctly:  -1850.655
## % of predictable range [ (model-null)/(1-null) ]:  0.04018159
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos
##    0.044029     0.279458     0.006311
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5182  AIC: 5594
## log likelihood:  -2591.01
## Nagelkerke R2:  0.05385042
## % pres/err predicted correctly:  -1850.611
## % of predictable range [ (model-null)/(1-null) ]:  0.04020445
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     -0.1285       0.1633
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5232  AIC: 5647
## log likelihood:  -2616.075
## Nagelkerke R2:  0.03800482
## % pres/err predicted correctly:  -1874.037
## % of predictable range [ (model-null)/(1-null) ]:  0.02806124
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	pos
preserved ~ CumPres	5591.734	0.000000	1.0000000	0.7304098	0.0538329	0.0562566	0.2871761	NA
preserved ~ CumPres + pos	5593.727	1.993406	0.3690944	0.2695902	0.0538504	0.0440294	0.2794581	0.0063108
preserved ~ pos	5646.712	54.977966	0.0000000	0.0000000	0.0380048	-0.1285371	NA	0.1632952

```r
CumAICSummary <- CumAICSummary %>% arrange(AIC)
BestModelFormulaL2 <- CumAICSummary$Model[BestModelIndexL2]
write.csv(CumAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_main_effects_plus_one_model_summary.csv")
kable(CumAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos	stimlen
preserved ~ CumPres + stimlen	5572.160	0.000000	1.0000000	0.9999436	0.0604516	0.7142150	0.3153941	NA	NA	-0.0908023
preserved ~ CumPres	5591.734	0.000000	1.0000000	0.8698708	0.0538329	0.0562566	0.2871761	NA	NA	NA
preserved ~ CumPres	5591.734	19.567406	0.0000564	0.0000564	0.0538329	0.0562566	0.2871761	NA	NA	NA
preserved ~ CumPres	5591.734	0.000000	1.0000000	0.7304098	0.0538329	0.0562566	0.2871761	NA	NA	NA
preserved ~ CumPres + pos	5593.727	1.993406	0.3690944	0.2695902	0.0538504	0.0440294	0.2794581	NA	0.0063108	NA
preserved ~ CumPres + I(pos^2) + pos	5595.533	3.799634	0.1495960	0.1301292	0.0538880	0.0128378	0.2796192	-0.0024184	0.0264535	NA
preserved ~ pos	5646.712	54.977966	0.0000000	0.0000000	0.0380048	-0.1285371	NA	NA	0.1632952	NA
preserved ~ I(pos^2) + pos	5648.667	56.933160	0.0000000	0.0000000	0.0380104	-0.1404349	NA	-0.0009136	0.1709600	NA
preserved ~ stimlen	5770.861	198.695146	0.0000000	0.0000000	0.0002230	0.6061301	NA	NA	NA	-0.0155118

```r
# explore influence of frequency and length

if(grepl("stimlen",BestModelFormulaL2)){ # if length is already in the formula
  Level3ModelEquations<-c(
      BestModelFormulaL2, # best model from level 2
    "preserved ~ 1", # null model
      paste0(BestModelFormulaL2," + log_freq")
  )
}else if(grepl("log_freq",BestModelFormulaL2)){ # if log_freq is already in the formula
    Level3ModelEquations<-c(
      BestModelFormulaL2, # best model from level 2
    "preserved ~ 1", # null model
      paste0(BestModelFormulaL2," + stimlen")
  )
}else{
  Level3ModelEquations<-c(
      BestModelFormulaL2, # best model from level 2
    "preserved ~ 1", # null model
      paste0(BestModelFormulaL2," + log_freq"),
      paste0(BestModelFormulaL2," + stimlen"),
      paste0(BestModelFormulaL2," + stimlen + log_freq")
  )
}

Level3Res<-TestModels(Level3ModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres      stimlen     log_freq
##     0.63213      0.31448     -0.07992      0.03386
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5157  AIC: 5572
## log likelihood:  -2578.599
## Nagelkerke R2:  0.06162797
## % pres/err predicted correctly:  -1839.534
## % of predictable range [ (model-null)/(1-null) ]:  0.04594613
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres      stimlen
##      0.7142       0.3154      -0.0908
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5161  AIC: 5572
## log likelihood:  -2580.481
## Nagelkerke R2:  0.06045165
## % pres/err predicted correctly:  -1841.021
## % of predictable range [ (model-null)/(1-null) ]:  0.04517568
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.4863
##
```

```
## Degrees of Freedom: 4297 Total (i.e. Null);  4297 Residual
## Null Deviance:       5350
## Residual Deviance: 5350  AIC: 5769
## log likelihood:  -2675.026
## Nagelkerke R2:  -3.118626e-16
## % pres/err predicted correctly:  -1928.172
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
BestModelL3<-Level3Res$ModelResult[[ BestModelIndexL3 ]]
BestModelFormulaL3<-Level3Res$Model[[BestModelIndexL3]]

AICSummary<-data.frame(Model=Level3Res$Model,
                       AIC=Level3Res$AIC,
                       row.names = Level3Res$Model)
AICSummary$DeltaAIC<-AICSummary$AIC-AICSummary$AIC[1]
AICSummary$AICexp<-exp(-0.5*AICSummary$DeltaAIC)
AICSummary$AICwt<-AICSummary$AICexp/sum(AICSummary$AICexp)
AICSummary$NagR2<-Level3Res$NagR2

AICSummary <- merge(AICSummary,Level3Res$CoefficientValues,
                          by='row.names',sort=FALSE)
AICSummary <- subset(AICSummary, select = -c(Row.names))

write.csv(AICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_main_effects_plus_one_len_freq_summary.csv")

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	stimlen	log_freq
preserved ~ CumPres + stimlen + log_freq	5571.728	0.0000000	1.0000000	0.5545981	0.0616280	0.6321301	0.3144787	-0.0799223	0.0338628
preserved ~ CumPres + stimlen	5572.166	0.4385332	0.8031076	0.4454019	0.0604516	0.7142150	0.3153941	-0.0908023	NA
preserved ~ 1	5769.397	197.6693737	0.0000000	0.0000000	0.0000000	0.4862643	NA	NA	NA

```r
# BestModelIndex is set by the parameter file and is used to choose
# a model that is essentially equivalent to the lowest AIC model, but
# simpler (and with a slightly higher AIC value, so not in first position)
BestModelL3<-Level3Res$ModelResult[[ BestModelIndexL3 ]]
BestModelFormulaL3<-Level3Res$Model[BestModelIndexL3]

BestModel<-BestModelL3
BestModelDeletions<-arrange(dropterm(BestModel),desc(AIC))
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
# order by terms that increase AIC the most when they are the one dropped
BestModelDeletions
```

```
## Single term deletions
##
## Model:
```

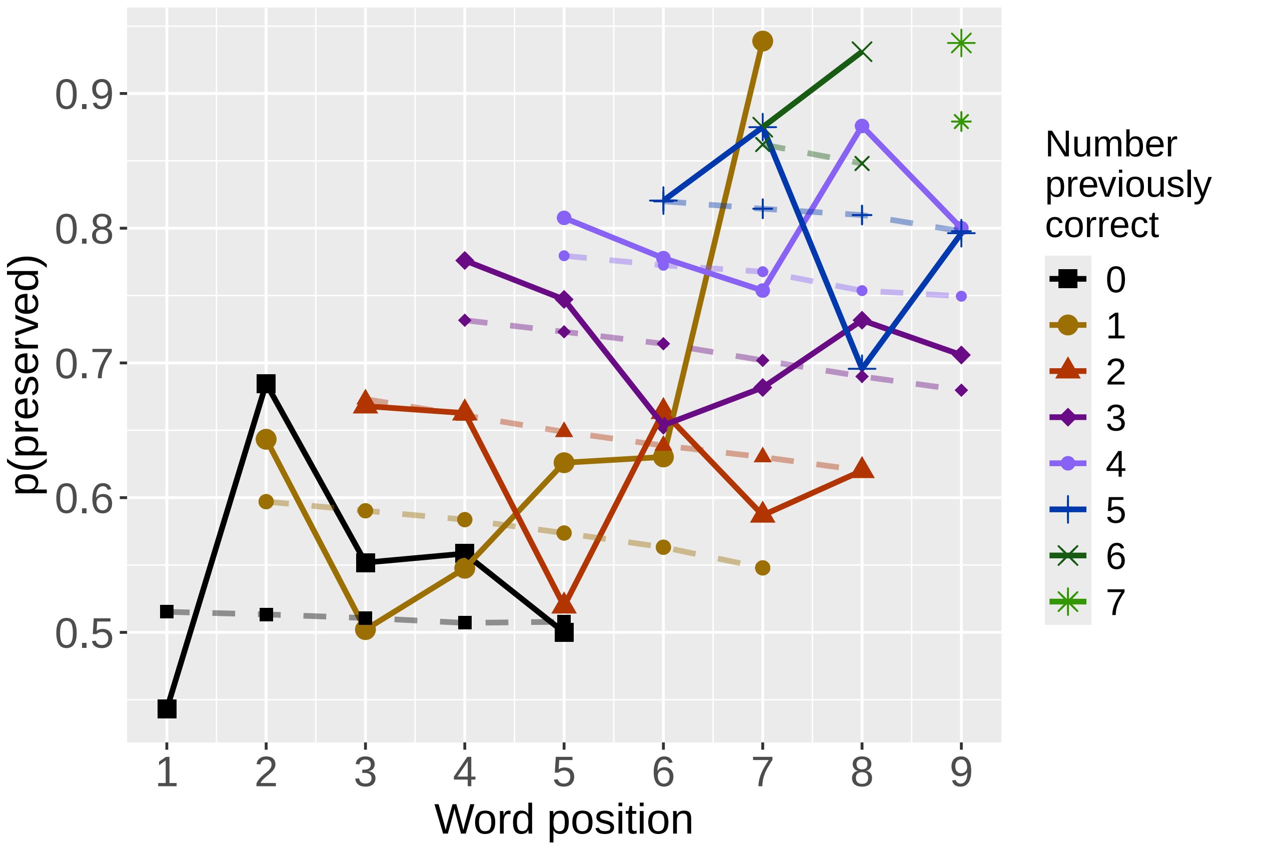
```
## preserved ~ CumPres + stimlen + log_freq
##          Df Deviance    AIC
## CumPres   1   5344.3 5756.8
## stimlen   1   5172.3 5584.8
## log_freq  1   5161.0 5573.5
## <none>        5157.2 5571.7
```

```
################################
# Single deletions from best model
################################

write.csv(BestModelDeletions,paste0(TablesDir,CurPat,"_",CurTask,"_best_model_single_term_deletions.csv
```

```
# plot prev err and prev cor with predicted values
PosDat$OAPred<-fitted(BestModel)

PrevCorPlot<-PlotObsPredPreviousCorrect(PosDat,"preserved","OAPred",palette_values,shape_values)
```

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

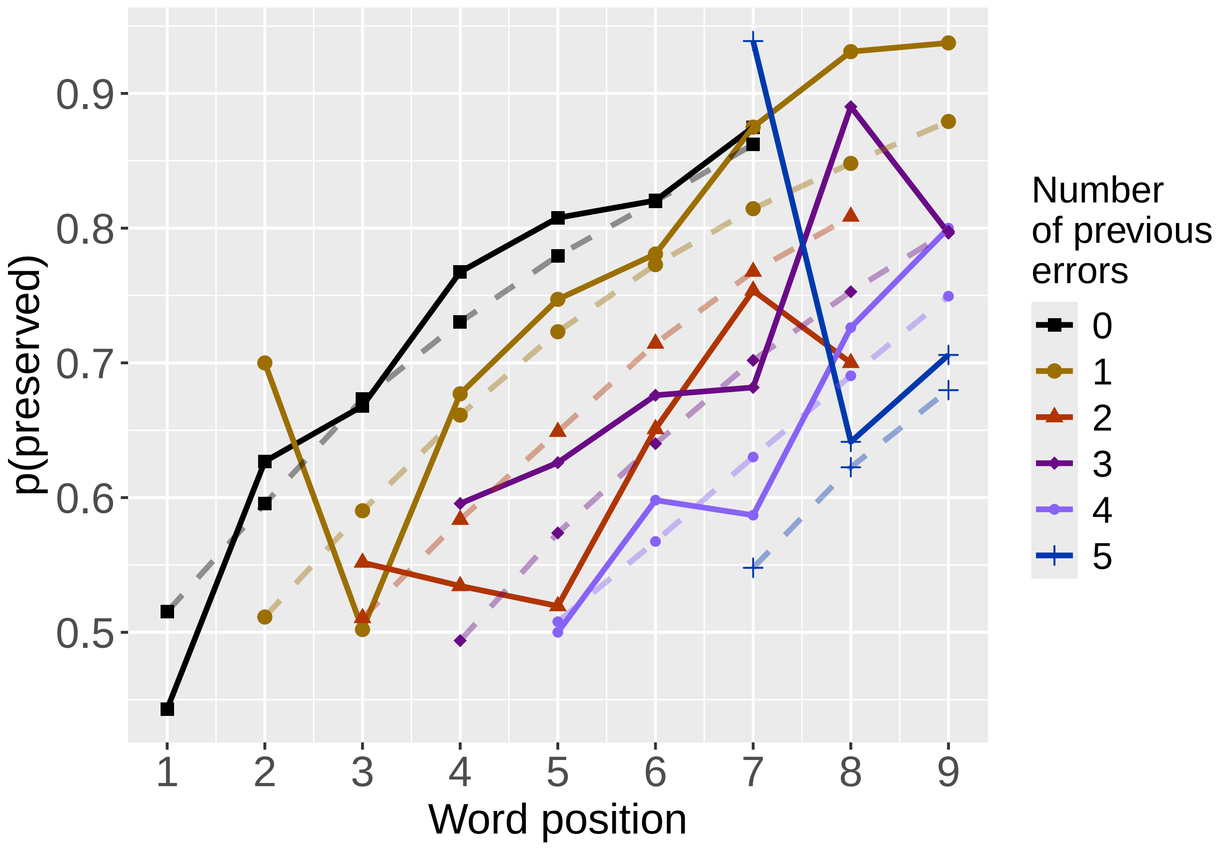
```
print(PrevCorPlot)
```



```
PrevErrPlot<-PlotObsPredPreviousError(PosDat,"preserved","OAPred",palette_values,shape_values)
```

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

```
print(PrevErrPlot)
```



```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_ObsPred_BestFullModel")
ggsave(filename=paste(PlotName,"_prev_correct.tif",sep=""),plot=PrevCorPlot,device="tiff",compression =
```

```
## Saving 6.5 x 4.5 in image
```

```
ggsave(filename=paste(PlotName,"_prev_error.tif",sep=""),plot=PrevErrPlot,device="tiff",compression = "
```

```
## Saving 6.5 x 4.5 in image
```

```
if(DoSimulations){
  BestModelFormulaL3Rnd <- BestModelFormulaL3
  if(grepl("CumPres",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumPres","RndCumPres",BestModelFormulaL3Rnd)
  }
  if(grepl("CumErr",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumErr","RndCumErr",BestModelFormulaL3Rnd)
  }

  RndModelAIC<-numeric(length=RandomSamples)
  for(rindex in seq(1,RandomSamples)){
  # Shuffle cumulative values
    PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
    PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
    BestModelRnd <- glm(as.formula(BestModelFormulaL3Rnd),
                        family="binomial",data=PosDat)
    RndModelAIC[rindex] <- BestModelRnd$aic
  }
  ModelNames<-c(paste0("***",BestModelFormulaL3),
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random average"),
                                     AIC=c(mean(RndModelAIC))))
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random SD"),
                                     AIC=c(sd(RndModelAIC))))
  write.csv(BestModelRndDF,paste0(TablesDir,CurPat,"_",CurTask,
                                  "_best_model_with_random_cum_term.csv"),
            row.names = FALSE)
}
```

```
# plot influence factors
num_models <- nrow(BestModelDeletions) - 1
# minus 1 because last
# model is always <none> and we don't want to include that

if(num_models > 4){
  last_model_num <- 4
}else{
  last_model_num <- num_models
}

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_FactorPlots")

FactorPlot<-PlotModelSetWithFreq(PosDat,FinalModelSet,
                                 palette_values,FinalModelSet,PptID=CurPat)
```

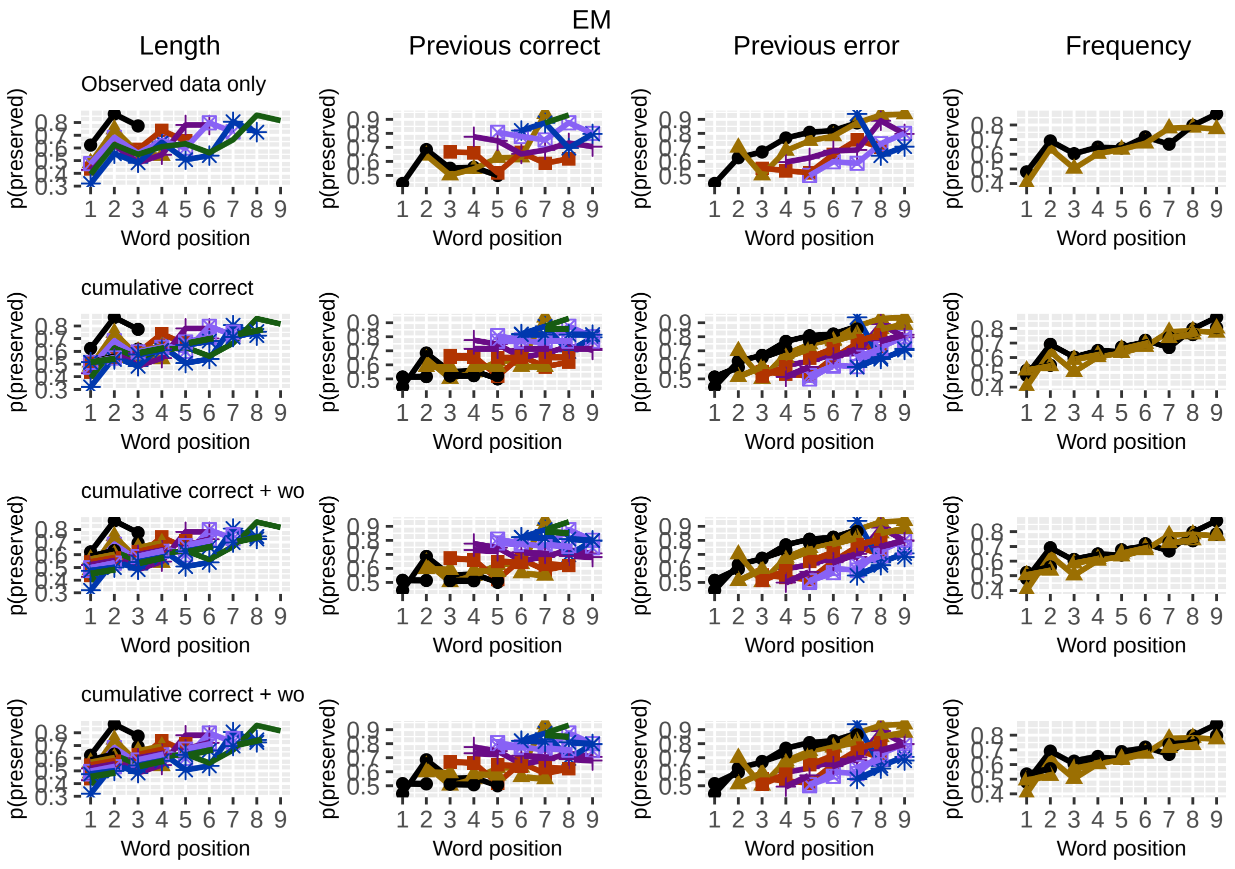
```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     0.05626      0.28718
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4296 Residual
## Null Deviance:       5350
## Residual Deviance: 5182  AIC: 5592
## log likelihood:  -2591.038
## Nagelkerke R2:  0.05383288
## % pres/err predicted correctly:  -1850.655
## % of predictable range [ (model-null)/(1-null) ]:  0.04018159
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres      stimlen
##      0.7142       0.3154      -0.0908
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4295 Residual
## Null Deviance:       5350
## Residual Deviance: 5161  AIC: 5572
## log likelihood:  -2580.481
## Nagelkerke R2:  0.06045165
## % pres/err predicted correctly:  -1841.021
## % of predictable range [ (model-null)/(1-null) ]:  0.04517568
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres      stimlen     log_freq
##     0.63213      0.31448     -0.07992      0.03386
##
## Degrees of Freedom: 4297 Total (i.e. Null);  4294 Residual
## Null Deviance:       5350
## Residual Deviance: 5157  AIC: 5572
## log likelihood:  -2578.599
## Nagelkerke R2:  0.06162797
## % pres/err predicted correctly:  -1839.534
## % of predictable range [ (model-null)/(1-null) ]:  0.04594613
## ***************************
```

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 9 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 3 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 3 rows containing missing values or values outside the scale range (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 3 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 3 rows containing missing values or values outside the scale range (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 3 rows containing missing values or values outside the scale range (`geom_point()`).
```

```
ggsave(paste0(PlotName,".tif"),plot=FactorPlot,width = 360,height=400,units="mm",device="tiff",compress
# use \blandscape and \elandscape to make markdown plots landscape if needed
FactorPlot
```



```
DA.Result<-dominanceAnalysis(BestModel)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
DAContributionAverage<-ConvertDominanceResultToTable(DA.Result)
write.csv(DAContributionAverage,paste0(TablesDir,CurPat,"_",CurTask,"_dominance_analysis_table.csv"),ro
kable(DAContributionAverage)
```

	CumPres	stimlen	log_freq
McFadden	0.0330124	0.0015902	0.0007113
SquaredCorrelation	0.0433071	0.0020418	0.0009328
Nagelkerke	0.0433071	0.0020418	0.0009328
Estrella	0.0440403	0.0021096	0.0009488

	deviance	deviance_explained
CumPres + stimlen + log_freq	5157.199	192.8538
CumPres + stimlen	5160.962	189.0906
CumPres	5182.076	167.9771
null	5350.053	0.0000

```
deviance_differences_df<-GetDevianceSet(FinalModelSet,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
write.csv(deviance_differences_df,paste0(TablesDir,CurPat,"_",CurTask,"_deviance_differences_table.csv"),row.names = FALSE)
deviance_differences_df
```

```
##                                                     model deviance deviance_explained
## CumPres + stimlen + log_freq CumPres + stimlen + log_freq 5157.199           192.8538
## CumPres + stimlen                       CumPres + stimlen 5160.962           189.0906
## CumPres                                           CumPres 5182.076           167.9771
## null                                                 null 5350.053             0.0000
##                              percent_explained percent_of_explained_deviance
## CumPres + stimlen + log_freq          3.604709                     100.00000
## CumPres + stimlen                     3.534368                      98.04864
## CumPres                               3.139727                      87.10071
## null                                  0.000000                            NA
##                              increment_in_explained
## CumPres + stimlen + log_freq               1.951357
## CumPres + stimlen                         10.947932
## CumPres                                   87.100712
## null                                       0.000000
```

```
kable(deviance_differences_df[,2:3], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```

```
kable(deviance_differences_df[,c(4:6)], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumPres + stimlen + log_freq	3.604709	100.00000	1.951356
CumPres + stimlen	3.534368	98.04864	10.947931
CumPres	3.139727	87.10071	87.100712
null	0.000000	NA	0.000000

```
NagContributions <- t(DAContributionAverage[3,])
NagTotal<-sum(NagContributions)
NagPercents <- NagContributions / NagTotal
NagResultTable <- data.frame(NagContributions = NagContributions, NagPercents = NagPercents)
names(NagResultTable)<-c("NagContributions","NagPercents")
write.csv(NagResultTable,paste0(TablesDir,CurPat,"_",CurTask,"_nagelkerke_contributions_table.csv"),row.names = TRUE)
NagPercents
```

```
##          Nagelkerke
## CumPres  0.93572761
## stimlen  0.04411679
## log_freq 0.02015560
```

```
sse_results_list<-compare_SS_accounted_for(FinalModelSet,"preserved ~ 1",PosDat,N_cutoff=10)
```

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

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length

## Warning in (null_cumcor_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length

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

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length
## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length

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

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length
## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length
```

```r
sse_table<-sse_results_table(sse_results_list)
write.csv(sse_table,paste0(TablesDir,CurPat,"_",CurTask,"_sse_results_table.csv"),row.names = TRUE)
sse_table
```

```
##                                  model p_accounted_for model_deviance diff_CumPres
## 1                  preserved ~ CumPres       0.5806939       5182.076  0.000000000
## 2 preserved ~ CumPres+stimlen+log_freq       0.5819271       5157.199  0.001233122
## 3          preserved ~ CumPres+stimlen       0.5861256       5160.962  0.005431631
```

model	p_accounted_for	model_deviance
preserved ~ CumPres	0.5806939	5182.076
preserved ~ CumPres+stimlen+log_freq	0.5819271	5157.199
preserved ~ CumPres+stimlen	0.5861256	5160.962

model	diff_CumPres	diff_CumPres+stimlen+log_freq	diff_CumPres+stimlen
preserved ~ CumPres	0.0000000	-0.0012331	-0.0054316
preserved ~ CumPres+stimlen+log_freq	0.0012331	0.0000000	-0.0041985
preserved ~ CumPres+stimlen	0.0054316	0.0041985	0.0000000

```
##   diff_CumPres+stimlen+log_freq diff_CumPres+stimlen
## 1                  -0.001233122         -0.005431631
## 2                   0.000000000         -0.004198509
## 3                   0.004198509          0.000000000
```

```r
kable(sse_table[,1:3], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```

```r
write.csv(results_report_DF,paste0(TablesDir,CurPat,"_",CurTask,"_results_report_df.csv"),row.names = FALSE)
```

```r
kable(sse_table[,c(1,4:6)], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```