OB - naming - Serial position analysis (v5)

```
# 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
```

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

```
# 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
```

```
# 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())
```

```
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	176	12	19	NA	NA	207
2	20	NA	145	11	31	207
3	80	NA	50	75	2	207
4	106	NA	49	19	9	183
5	56	NA	47	12	9	124
6	41	NA	15	17	5	78
7	29	NA	12	3	1	45
8	11	NA	2	3	NA	16
9	5	NA	1	NA	2	8

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.8502415	0.057971	0.0917874	NA	NA	207
2	0.0966184	NA	0.7004831	0.0531401	0.1497585	207
3	0.3864734	NA	0.2415459	0.3623188	0.0096618	207
4	0.5792350	NA	0.2677596	0.1038251	0.0491803	183
5	0.4516129	NA	0.3790323	0.0967742	0.0725806	124
6	0.5256410	NA	0.1923077	0.2179487	0.0641026	78

pos_factor	O	P	V	1	S	total
7	0.6444444	NA	0.2666667	0.0666667	0.0222222	45
8	0.6875000	NA	0.1250000	0.1875000	NA	16
9	0.6250000	NA	0.1250000	NA	0.2500000	8

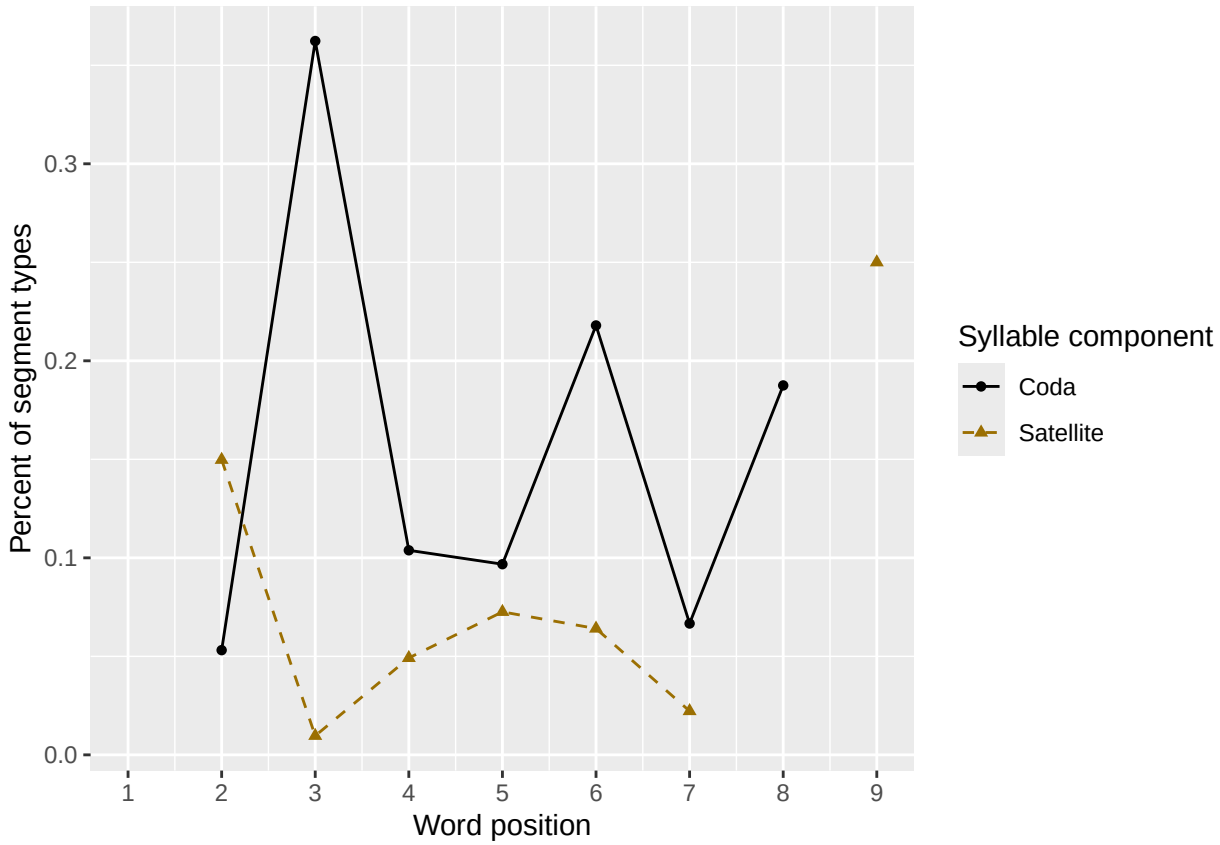
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 4 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 4 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 0.958 0.875 0.792 NA     NA     NA     NA     NA     NA
## 2       5 0.831 0.771 0.729  0.703 NA     NA     NA     NA     NA
## 3       6 0.880 0.743 0.848  0.710  0.630 NA     NA     NA     NA
## 4       7 0.879 0.864 0.742  0.682  0.682  0.561 NA     NA     NA
## 5       8 0.793 0.856 0.730  0.667  0.615  0.563  0.546 NA     NA
## 6       9 1     0.781 0.688  0.406  0.656  0.438  0.438  0.625 NA
## 7      10 0.875 0.65  0.9    0.425  0.125  0.238  0.475  0.262  0.238
```

```
# 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.
```

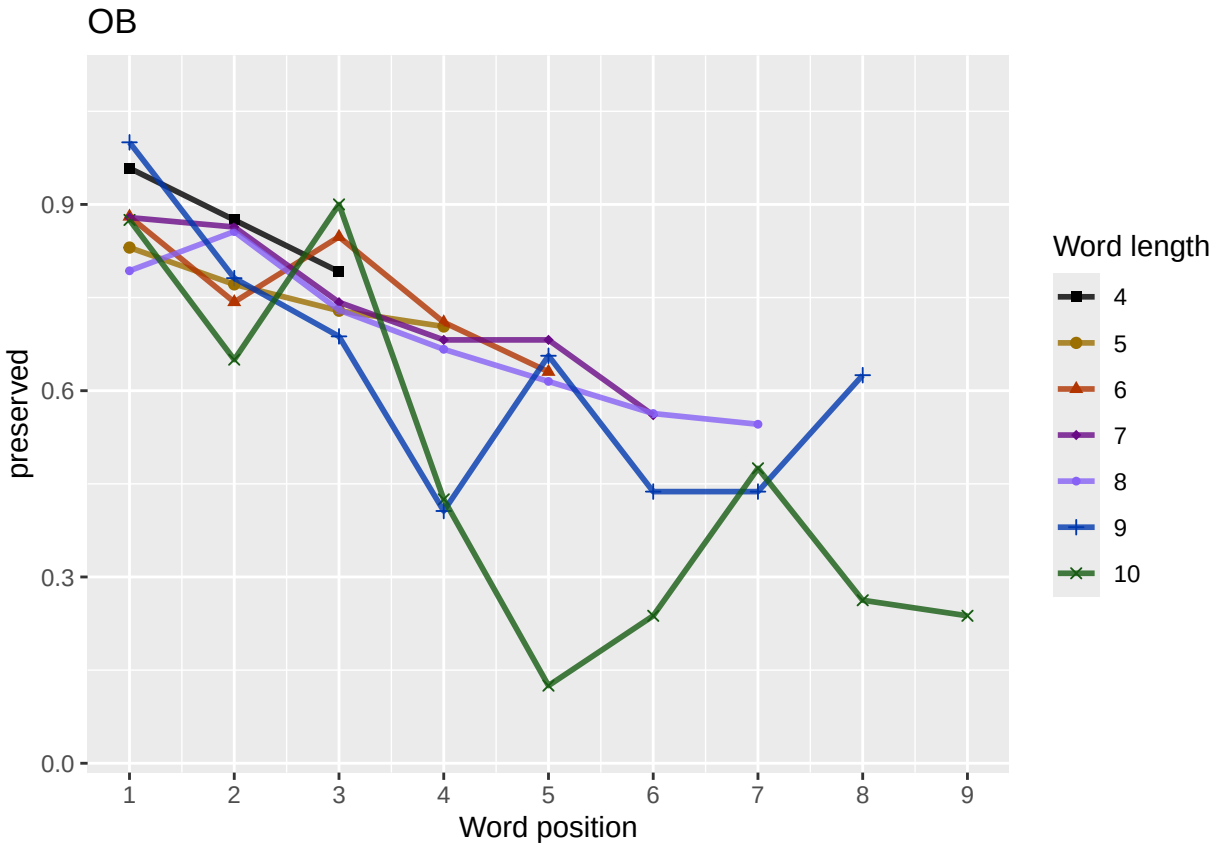
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    24    24    24    NA    NA    NA    NA    NA    NA
## 2       5    59    59    59    59    NA    NA    NA    NA    NA
## 3       6    46    46    46    46    46    NA    NA    NA    NA
## 4       7    33    33    33    33    33    33    NA    NA    NA
## 5       8    29    29    29    29    29    29    29    NA    NA
## 6       9     8     8     8     8     8     8     8     8    NA
## 7      10     8     8     8     8     8     8     8     8     8
```

```
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:  8
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.274198          0.041451          0.092748         -0.497365         -0.004818
##      stimlen:pos
##        -0.029405
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 1085  AIC: 1231
## log likelihood:  -542.6306
## Nagelkerke R2:  0.117655
## % pres/err predicted correctly:  -372.8264
## % of predictable range [ (model-null)/(1-null) ]:  0.09440452
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.60573     -0.09343     -0.28918
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 1090  AIC: 1232
## log likelihood:  -545.2282
## Nagelkerke R2:  0.1109301
## % pres/err predicted correctly:  -374.9085
## % of predictable range [ (model-null)/(1-null) ]:  0.08936078
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.93468     -0.10679      0.01929     -0.44259
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 1089  AIC: 1232
## log likelihood:  -544.6354
## Nagelkerke R2:  0.1124676
## % pres/err predicted correctly:  -373.9211
## % of predictable range [ (model-null)/(1-null) ]:  0.09175274
## **************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.20778     -0.03908     -0.16736     -0.01593
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 1090  AIC: 1233
## log likelihood:  -545.0067
## Nagelkerke R2:  0.1115047
## % pres/err predicted correctly:  -374.9774
## % of predictable range [ (model-null)/(1-null) ]:  0.08919387
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1069      -0.3264
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1236
## log likelihood:  -547.0127
## Nagelkerke R2:  0.1062913
## % pres/err predicted correctly:  -376.9225
## % of predictable range [ (model-null)/(1-null) ]:  0.08448177
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.24537      0.01036     -0.41171
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1238
## log likelihood:  -546.8309
## Nagelkerke R2:  0.1067644
## % pres/err predicted correctly:  -376.5375
## % of predictable range [ (model-null)/(1-null) ]:  0.08541439
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      2.5791      -0.2402
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1140  AIC: 1288
## log likelihood:  -570.1462
## Nagelkerke R2:  0.04474136
## % pres/err predicted correctly:  -395.4661
## % of predictable range [ (model-null)/(1-null) ]:  0.03955988
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9487
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1074 Residual
## Null Deviance:       1173
## Residual Deviance: 1173  AIC: 1330
## log likelihood:  -586.3585
## Nagelkerke R2:  -3.343613e-16
## % pres/err predicted correctly:  -411.7963
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

```
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 * (I(pos^2) + pos)	1231.285	0.0000000	1.0000000	0.3464870	0.1176552	2.2741976	-0.0414513	-0.4973650	-0.0029405	0.0927481	-0.0048183
preserved ~ stimlen + pos	1231.800	0.5144726	0.6773185	0.2678987	0.1109321	2.6057281	-0.0934286	-0.2891773	NA	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	1232.381	1.0949037	0.5784208	0.2004156	0.1124676	2.9346760	-0.1067888	-0.4425920	NA	0.0192874	NA
preserved ~ stimlen * pos	1233.151	1.8729804	0.3920013	0.1358233	0.1115047	2.2077806	-0.0390770	-0.4167357	-0.0159321	NA	NA
preserved ~ pos	1235.894	4.6087735	0.0998200	0.0345863	0.1062912	2.1068695	NA	-0.3264059	NA	NA	NA
preserved ~ I(pos^2) + pos	1237.593	6.3079183	0.0426828	0.0147890	0.1067644	2.2453679	NA	-0.4117057	NA	0.0103572	NA
preserved ~ stimlen	1288.443	57.1577708	0.0000000	0.0000000	0.0447412	1.5790741	-0.2401896	NA	NA	NA	NA
preserved ~ 1	1329.980	98.6953667	0.0000000	0.0000000	0.0000000	0.9486953	NA	NA	NA	NA	NA

```r
print(BestLPModelFormula)
```

```
## [1] "preserved ~ stimlen * (I(pos^2) + pos)"
```

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.274198          0.041451          0.092748         -0.497365         -0.004818
##      stimlen:pos
##        -0.029405
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 1085  AIC: 1231
```

```r
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.
```

```r
# 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.
```

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

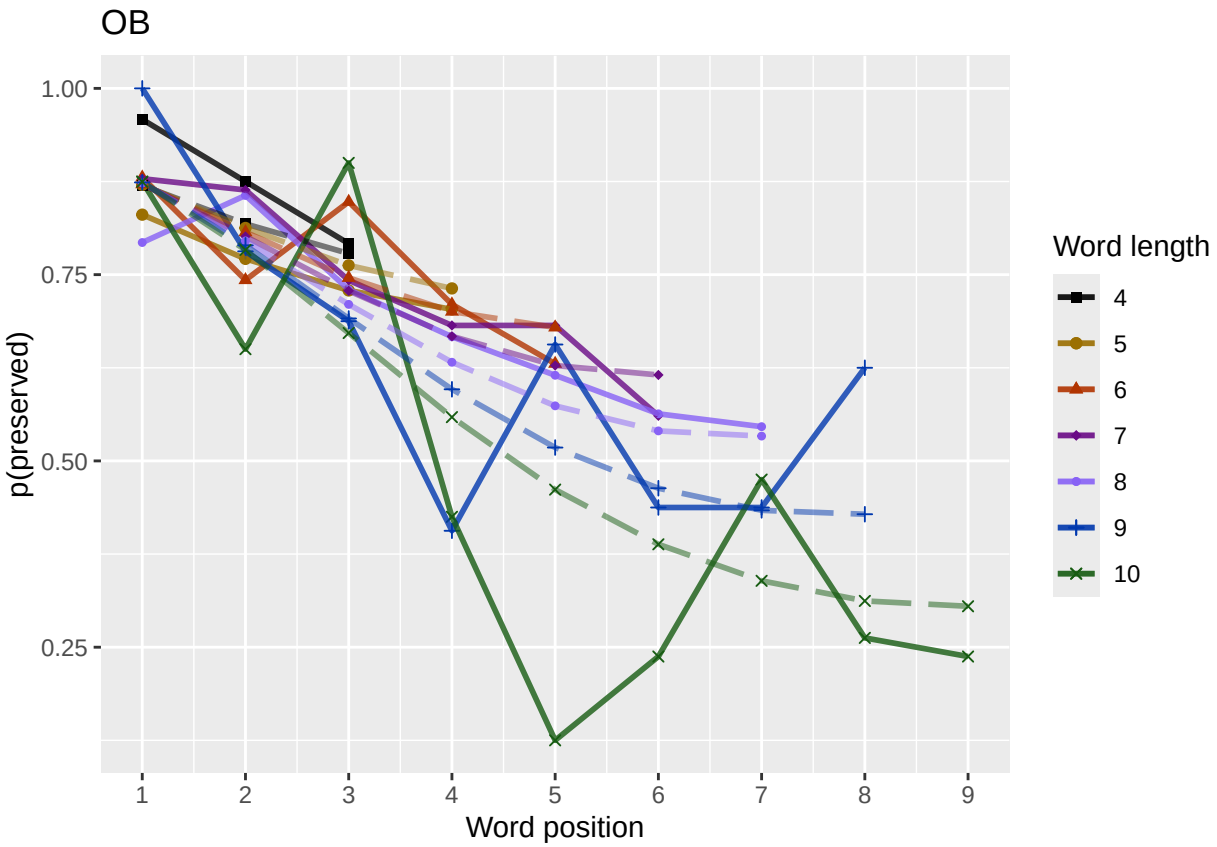
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.870 0.818 0.778 NA     NA     NA     NA     NA     NA
## 2       5 0.871 0.813 0.762  0.732 NA     NA     NA     NA     NA
## 3       6 0.871 0.807 0.746  0.700  0.679 NA     NA     NA     NA
```

```
## 4       7 0.872 0.801 0.728  0.667  0.628  0.615 NA     NA     NA
## 5       8 0.873 0.795 0.710  0.632  0.574  0.540  0.533 NA     NA
## 6       9 0.874 0.789 0.691  0.596  0.518  0.463  0.434  0.428 NA
## 7      10 0.875 0.783 0.672  0.559  0.462  0.388  0.339  0.312  0.305
```

```
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

```
# 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
```

```
# 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       41   207
```

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

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

```
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 41 / 207 = 19.81 percent"
```

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

```
# 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:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.250434          0.049590          0.148758         -0.584864         -0.008099
##      stimlen:pos
##        -0.025753
##
## Degrees of Freedom: 977 Total (i.e. Null);  972 Residual
## Null Deviance:       925.4
## Residual Deviance: 900.7     AIC: 1056
## log likelihood:  -450.3666
## Nagelkerke R2:  0.0407591
## % pres/err predicted correctly:  -302.1336
## % of predictable range [ (model-null)/(1-null) ]:  0.03611337
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.92166     -0.10713      0.04273     -0.45550
##
## Degrees of Freedom: 977 Total (i.e. Null);  974 Residual
## Null Deviance:       925.4
## Residual Deviance: 905.5     AIC: 1057
## log likelihood:  -452.7667
## Nagelkerke R2:  0.0329175
## % pres/err predicted correctly:  -304.5323
## % of predictable range [ (model-null)/(1-null) ]:  0.02848627
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.29971     -0.08351     -0.13493
##
## Degrees of Freedom: 977 Total (i.e. Null);  975 Residual
## Null Deviance:       925.4
## Residual Deviance: 909.3     AIC: 1060
## log likelihood:  -454.6677
## Nagelkerke R2:  0.02667912
## % pres/err predicted correctly:  -305.6497
## % of predictable range [ (model-null)/(1-null) ]:  0.02493324
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.22355      0.03336     -0.42065
##
## Degrees of Freedom: 977 Total (i.e. Null);  975 Residual
## Null Deviance:       925.4
## Residual Deviance: 909.3     AIC: 1062
## log likelihood:  -454.64
## Nagelkerke R2:  0.0267701
## % pres/err predicted correctly:  -306.694
## % of predictable range [ (model-null)/(1-null) ]:  0.02161245
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    2.105574    -0.056734    -0.069528    -0.008679
##
## Degrees of Freedom: 977 Total (i.e. Null);  974 Residual
## Null Deviance:       925.4
## Residual Deviance: 909.2     AIC: 1062
## log likelihood:  -454.6206
## Nagelkerke R2:  0.02683385
## % pres/err predicted correctly:  -305.5601
## % of predictable range [ (model-null)/(1-null) ]:  0.02521787
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.8405      -0.1642
##
## Degrees of Freedom: 977 Total (i.e. Null);  976 Residual
## Null Deviance:       925.4
## Residual Deviance: 911.7     AIC: 1062
## log likelihood:  -455.8572
## Nagelkerke R2:  0.02276344
## % pres/err predicted correctly:  -307.1696
## % of predictable range [ (model-null)/(1-null) ]:  0.02010022
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.2338      -0.1393
```

```
##
## Degrees of Freedom: 977 Total (i.e. Null);  976 Residual
## Null Deviance:       925.4
## Residual Deviance: 917.5     AIC: 1070
## log likelihood:  -458.7271
## Nagelkerke R2:  0.01327617
## % pres/err predicted correctly:  -309.3283
## % of predictable range [ (model-null)/(1-null) ]:  0.01323618
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.308
##
## Degrees of Freedom: 977 Total (i.e. Null);  977 Residual
## Null Deviance:       925.4
## Residual Deviance: 925.4     AIC: 1082
## log likelihood:  -462.7152
## Nagelkerke R2:  -3.629335e-16
## % pres/err predicted correctly:  -313.4909
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
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=FALSE
NoFragLPAICSummary <- subset(NoFragLPAICSummary, select = -c(Row.names))

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	1055.649	0.000000	1.000000	0.5801920	0.0407521	2.1250434	0.0495898	-0.5848641	-1.0257528	0.1487582	-0.0080987
preserved ~ stimlen + I(pos^2) + pos	1057.143	1.493642	0.4738705	0.2749359	0.0329172	1.5921657	-0.1071268	-0.4555013	NA	0.0427267	NA
preserved ~ stimlen + pos	1059.899	4.249678	0.1194522	0.0693052	0.0266729	1.1299712	-0.0835129	-0.1349314	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(posˆ2)	stimlen:I(posˆ2)
preserved ~ I(posˆ2) + pos	1061.674	4.024691	0.0491762	0.0285310	0.0267721	0.1223553	NA	-0.4206525	NA	0.0333552	NA
preserved ~ stimlen * pos	1061.814	4.164838	0.0458482	0.0266008	0.0268328	0.2105574	-0.0567339	-0.0695281	0.0086785	NA	NA
preserved ~ pos	1062.378	4.728724	0.0345840	0.0200654	0.0227631	0.4840464	NA	-0.1642339	NA	NA	NA
preserved ~ stimlen	1070.375	12.725205	0.0006345	0.0003682	0.0132762	0.2233801	-0.1393333	NA	NA	NA	NA
preserved ~ 1	1082.232	24.584188	0.0000017	0.0000010	0.0000000	0.0308344	NA	NA	NA	NA	NA

```
# 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.
```

```
# 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.
```

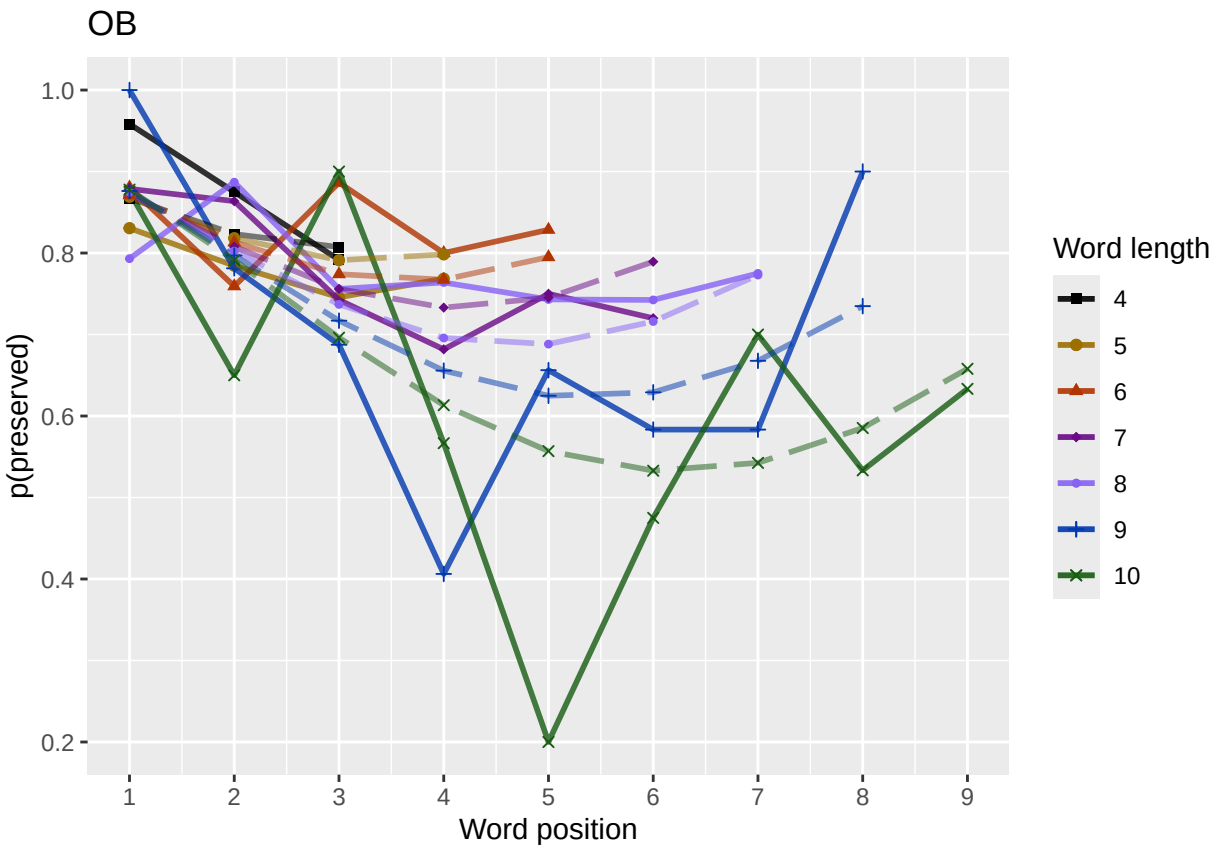
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_fi
nofrag_fitted_pos_len_table
```

```
## # 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 0.867 0.823 0.807 NA     NA     NA     NA     NA     NA
## 2       5 0.869 0.818 0.791  0.798 NA     NA     NA     NA     NA
## 3       6 0.871 0.813 0.774  0.767  0.795 NA     NA     NA     NA
## 4       7 0.873 0.808 0.756  0.733  0.745  0.789 NA     NA     NA
## 5       8 0.874 0.802 0.737  0.696  0.688  0.716  0.773 NA     NA
## 6       9 0.876 0.797 0.717  0.656  0.625  0.629  0.668  0.735 NA
## 7      10 0.878 0.791 0.696  0.613  0.557  0.533  0.542  0.585  0.658
```

```
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: 0.04 - 1.09"
```

```
# 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] -0.02751203
```

```
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] -0.04626169
```

```
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
  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"
```

```
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:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.274198          0.041451          0.092748         -0.497365         -0.004818
##      stimlen:pos
##        -0.029405
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 1085  AIC: 1231
## log likelihood:  -542.6306
## Nagelkerke R2:  0.117655
## % pres/err predicted correctly:  -372.8264
## % of predictable range [ (model-null)/(1-null) ]:  0.09440452
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.60573     -0.09343     -0.28918
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
```

```
## Residual Deviance: 1090  AIC: 1232
## log likelihood:  -545.2282
## Nagelkerke R2:  0.1109301
## % pres/err predicted correctly:  -374.9085
## % of predictable range [ (model-null)/(1-null) ]:  0.08936078
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.93468     -0.10679      0.01929     -0.44259
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 1089  AIC: 1232
## log likelihood:  -544.6354
## Nagelkerke R2:  0.1124676
## % pres/err predicted correctly:  -373.9211
## % of predictable range [ (model-null)/(1-null) ]:  0.09175274
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.44201     -0.06777     -0.29020      0.08733
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 1087  AIC: 1233
## log likelihood:  -543.5935
## Nagelkerke R2:  0.1151659
## % pres/err predicted correctly:  -374.0919
## % of predictable range [ (model-null)/(1-null) ]:  0.09133904
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     2.77274     -0.08118      0.01941     -0.44459      0.08746
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1070 Residual
## Null Deviance:       1173
## Residual Deviance: 1086  AIC: 1233
## log likelihood:  -542.995
## Nagelkerke R2:  0.1167134
```

```
## % pres/err predicted correctly:  -373.1079
## % of predictable range [ (model-null)/(1-null) ]:  0.09372267
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.20778     -0.03908     -0.16736     -0.01593
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 1090  AIC: 1233
## log likelihood:  -545.0067
## Nagelkerke R2:  0.1115047
## % pres/err predicted correctly:  -374.9774
## % of predictable range [ (model-null)/(1-null) ]:  0.08919387
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.45747          -0.07162           0.17795          -0.28975          -0.01323
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1070 Residual
## Null Deviance:       1173
## Residual Deviance: 1087  AIC: 1234
## log likelihood:  -543.4822
## Nagelkerke R2:  0.1154538
## % pres/err predicted correctly:  -373.9132
## % of predictable range [ (model-null)/(1-null) ]:  0.09177179
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      2.0745      -0.3153       0.1047
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 1089  AIC: 1234
## log likelihood:  -544.4627
## Nagelkerke R2:  0.1129151
## % pres/err predicted correctly:  -375.239
## % of predictable range [ (model-null)/(1-null) ]:  0.08856005
## ***************************
```

```
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.44802      -0.06671      -0.29476       0.13334      -0.01243
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1070 Residual
## Null Deviance:       1173
## Residual Deviance: 1087  AIC: 1234
## log likelihood:  -543.4683
## Nagelkerke R2:  0.1154898
## % pres/err predicted correctly:  -374.0034
## % of predictable range [ (model-null)/(1-null) ]:  0.09155331
## **************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          2.77823          -0.08416           0.16552           0.01895          -0.44056
## stimlen:log_freq
##         -0.01136
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 1086  AIC: 1235
## log likelihood:  -542.913
## Nagelkerke R2:  0.1169253
## % pres/err predicted correctly:  -372.9685
## % of predictable range [ (model-null)/(1-null) ]:  0.09406044
## **************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1069      -0.3264
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1236
## log likelihood:  -547.0127
## Nagelkerke R2:  0.1062913
## % pres/err predicted correctly:  -376.9225
## % of predictable range [ (model-null)/(1-null) ]:  0.08448177
## **************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.08773      -0.31996       0.15553      -0.01377
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 1089  AIC: 1236
## log likelihood:  -544.3095
## Nagelkerke R2:  0.113312
## % pres/err predicted correctly:  -375.1353
## % of predictable range [ (model-null)/(1-null) ]:  0.08881127
## **************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)               pos           log_freq
##           2.68201           -0.07722            0.01133           -0.39224           -0.09555
## I(pos^2):log_freq       pos:log_freq
##          -0.01591            0.12341
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1068 Residual
## Null Deviance:       1173
## Residual Deviance: 1084  AIC: 1236
## log likelihood:  -541.9101
## Nagelkerke R2:  0.1195144
## % pres/err predicted correctly:  -372.0839
## % of predictable range [ (model-null)/(1-null) ]:  0.09620327
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.456975         -0.069613          0.180175         -0.293186         -0.008680
##     log_freq:pos
##        -0.009008
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 1087  AIC: 1236
## log likelihood:  -543.4297
## Nagelkerke R2:  0.1155896
## % pres/err predicted correctly:  -373.9076
## % of predictable range [ (model-null)/(1-null) ]:  0.09178551
## **************************
```

```
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.24537      0.01036     -0.41171
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1238
## log likelihood:  -546.8309
## Nagelkerke R2:  0.1067644
## % pres/err predicted correctly:  -376.5375
## % of predictable range [ (model-null)/(1-null) ]:  0.08541439
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          2.680060          -0.076782          -0.103671           0.011277          -0.392068
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##          0.001275          -0.016010           0.123711
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1067 Residual
## Null Deviance:       1173
## Residual Deviance: 1084  AIC: 1238
## log likelihood:  -541.9094
## Nagelkerke R2:  0.1195164
## % pres/err predicted correctly:  -372.0914
## % of predictable range [ (model-null)/(1-null) ]:  0.0961851
## ***************************
## 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
##          2.179522           0.004718          -0.367978          -0.073535          -0.016491
##      pos:log_freq
##          0.125142
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 1086  AIC: 1239
## log likelihood:  -542.9655
## Nagelkerke R2:  0.1167896
## % pres/err predicted correctly:  -373.5817
## % of predictable range [ (model-null)/(1-null) ]:  0.09257483
```

```
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5791      -0.2402
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1140  AIC: 1288
## log likelihood:  -570.1462
## Nagelkerke R2:  0.04474136
## % pres/err predicted correctly:  -395.4661
## % of predictable range [ (model-null)/(1-null) ]:  0.03955988
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.42297     -0.21625      0.08282
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 1137  AIC: 1289
## log likelihood:  -568.597
## Nagelkerke R2:  0.04894639
## % pres/err predicted correctly:  -394.6444
## % of predictable range [ (model-null)/(1-null) ]:  0.04155044
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          2.44098          -0.22040           0.18650          -0.01501
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 1137  AIC: 1291
## log likelihood:  -568.4436
## Nagelkerke R2:  0.0493621
## % pres/err predicted correctly:  -394.5509
## % of predictable range [ (model-null)/(1-null) ]:  0.04177708
## **************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9487
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1074 Residual
## Null Deviance:       1173
## Residual Deviance: 1173  AIC: 1330
## log likelihood:  -586.3585
## Nagelkerke R2:  -3.343613e-16
## % pres/err predicted correctly:  -411.7963
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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.name
kable(FLPAICSummary)
```

Model	AIC DeltaAIC AICexp AICwt NagR2 (Intercept) stimlen log_freq … I(pos^2) … stimlen:I(pos^2) [overlapping, illegible]										
preserved ~ stimlen * (I(pos^2) + pos)	1231.[illegible]	[illegible]	NA	-0.4973650	NA	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]
preserved ~ stimlen + pos	1231.[illegible] -0.0934286	NA	NA	-0.2891773	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	1232.[illegible] -0.1067888	NA	NA	-0.4425920	NA	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen + pos + log_freq	1232.[illegible] -0.0677693	[illegible]	[illegible]	-0.2902015	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	1233.[illegible] -0.0811750	[illegible]	[illegible]	-0.4445851	NA	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen * pos	1233.[illegible] -0.0390774	NA	NA	-0.1673578	NA	NA	NA	NA	NA	-0.0159321	NA

Model	AIC DeltaAIC AICexp AICw NagR2 (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 + pos	1234.2954982280098571173543574695 -0.0716198	0.1779519	-0.0132272	-0.2897451	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	1234.3294407262826643721549105745NA56	0.1047083	NA	-0.3153131	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	1234.3056542376910245316548980244 -0.0667097	0.1333417	NA	-0.2940761	-0.0124334	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	1234.3859583165643380831462757823 -0.0841565	0.1655247	-0.0113602	-0.2405629	NA	NA	0.0189507	NA	NA	NA	NA
preserved ~ pos	1235.4640807309820991706291068625	NA	NA	-0.3264059	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	1236.0850009807175073432187728NA5	0.1555328	NA	-0.3199599	-0.0937705	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	1236.0938035910339809178516820093 -0.0772229	-0.2755497	NA	-0.3922361	0.1234115	NA	0.0113347	-0.0159098	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	1236.2072202185534017096253856975 -0.0696126	0.1801748	-0.0086797	-0.2931856	NA	-0.0090082	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1237.6530791842682851076244536NA9	NA	NA	-0.4117057	NA	NA	0.0103572	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	1238.6872195253009661127516800596 -0.0767821	-0.[illegible]36710	0.0012752	-0.3920678	NA	0.1237	0.0012767	NA	-0.0160103	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	1238.7582547262186525146789795NA19	-0.0735347	NA	-0.3679779	0.1251424	NA	0.0047182	-0.0164905	NA	NA	NA
preserved ~ stimlen	1288.5713507708000000000417490741 -0.2401896	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	1289.5759648025800000000489462229668 -0.2162456	0.0828237	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	1290.5845985064800000000493624109803 -0.2204039	0.1865048	-0.0150145	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	1329.986953000000000000000094869053	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

```
## [1] "preserved ~ stimlen * (I(pos^2) + pos)"
```

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.274198          0.041451          0.092748         -0.497365         -0.004818
##      stimlen:pos
##        -0.029405
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 1085  AIC: 1231
```

```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.
```

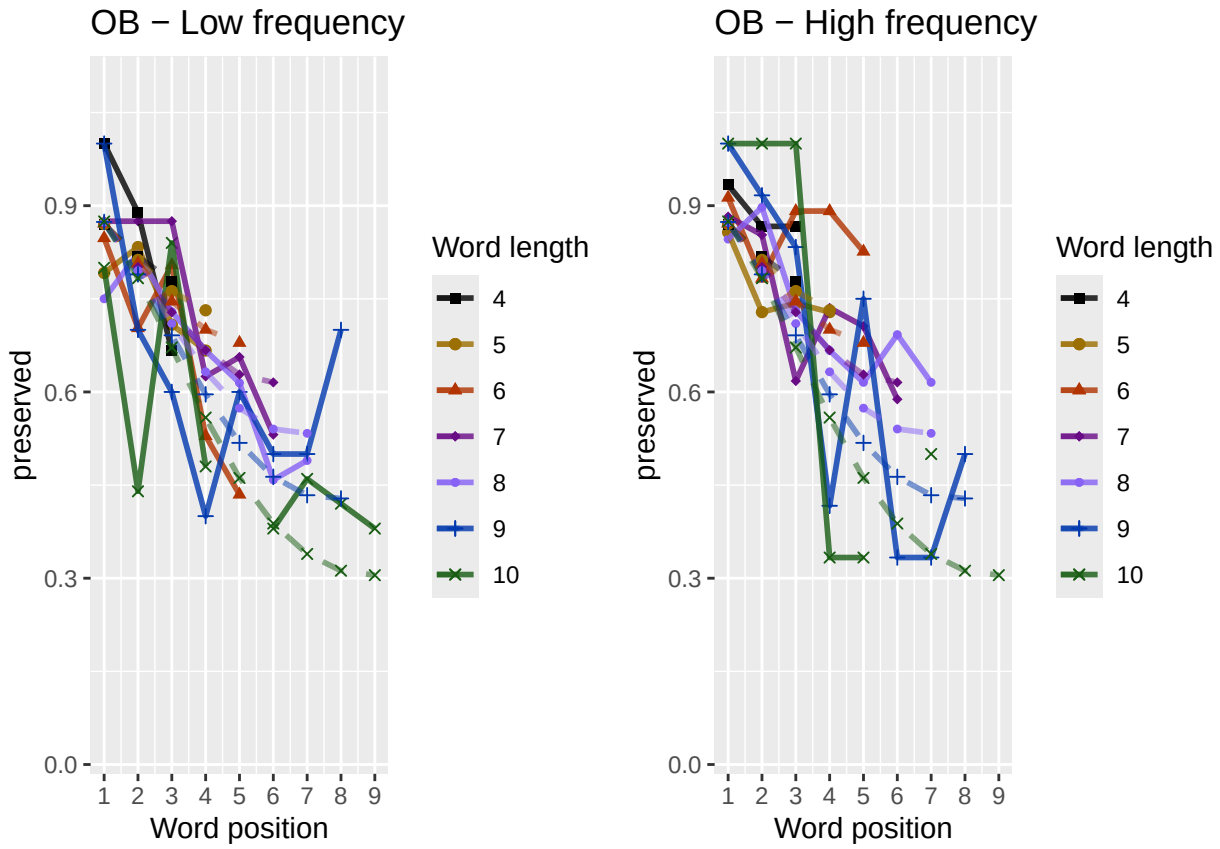
```r
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```

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

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_frequency_effect_length_pos_wfit.png"),device="png",unit="cm"
print(Both_Plots)
```




```
# 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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)       CumErr
##       1.586       -1.011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 973.2     AIC: 1077
## log likelihood:  -486.6214
## Nagelkerke R2:  0.2550256
## % pres/err predicted correctly:  -325.9679
## % of predictable range [ (model-null)/(1-null) ]:  0.2079195
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1069      -0.3264
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1236
## log likelihood:  -547.0127
## Nagelkerke R2:  0.1062913
## % pres/err predicted correctly:  -376.9225
## % of predictable range [ (model-null)/(1-null) ]:  0.08448177
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.24537      0.01036     -0.41171
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1238
## log likelihood:  -546.8309
## Nagelkerke R2:  0.1067644
## % pres/err predicted correctly:  -376.5375
## % of predictable range [ (model-null)/(1-null) ]:  0.08541439
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5791      -0.2402
```

```
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1140  AIC: 1288
## log likelihood:  -570.1462
## Nagelkerke R2:  0.04474136
## % pres/err predicted correctly:  -395.4661
## % of predictable range [ (model-null)/(1-null) ]:  0.03955988
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9487
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1074 Residual
## Null Deviance:       1173
## Residual Deviance: 1173  AIC: 1330
## log likelihood:  -586.3585
## Nagelkerke R2:  -3.343613e-16
## % pres/err predicted correctly:  -411.7963
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.05066     -0.05645
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1171  AIC: 1330
## log likelihood:  -585.5799
## Nagelkerke R2:  0.002179638
## % pres/err predicted correctly:  -411.1815
## % of predictable range [ (model-null)/(1-null) ]:  0.001489387
## ***************************
```

```
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,
                      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 ~ CumErr	1076.549	0.0000	1	1	0.2550256	6.5863632	NA	-1.011198	NA	NA	NA
preserved ~ pos	1235.894	159.3445	0	0	0.1062913	3.1068695	NA	NA	NA	-0.3264059	NA
preserved ~ (I(pos^2) + pos)	1237.593	161.0437	0	0	0.1067642	2.2453679	NA	NA	0.0103572	-0.4117057	NA
preserved ~ stimlen	1288.443	211.8935	0	0	0.0447412	2.5790741	NA	NA	NA	NA	-0.2401896
preserved ~ 1	1329.980	253.4311	0	0	0.0000000	0.9486953	NA	NA	NA	NA	NA
preserved ~ CumPres	1330.175	253.6258	0	0	0.0021796	1.0506617	-0.0564507	NA	NA	NA	NA

```
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	0.6871429	140
O	0.7187977	524
P	0.9166667	12
S	0.6169492	59
V	0.7490196	340

```
#  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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       1.594       -1.111
##
## Degrees of Freedom: 1003 Total (i.e. Null);  1002 Residual
## Null Deviance:       1091
## Residual Deviance: 898.6     AIC: 992
## log likelihood:  -449.2952
## Nagelkerke R2:  0.2632383
## % pres/err predicted correctly:  -298.7427
## % of predictable range [ (model-null)/(1-null) ]:  0.2173457
```

```
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1727      -0.3367
##
## Degrees of Freedom: 1003 Total (i.e. Null);  1002 Residual
## Null Deviance:       1091
## Residual Deviance: 1014  AIC: 1142
## log likelihood:  -507.1521
## Nagelkerke R2:  0.1110552
## % pres/err predicted correctly:  -348.5078
## % of predictable range [ (model-null)/(1-null) ]:  0.08740455
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.37462      0.01493     -0.45996
##
## Degrees of Freedom: 1003 Total (i.e. Null);  1001 Residual
## Null Deviance:       1091
## Residual Deviance: 1014  AIC: 1144
## log likelihood:  -506.8122
## Nagelkerke R2:  0.1120013
## % pres/err predicted correctly:  -347.9774
## % of predictable range [ (model-null)/(1-null) ]:  0.08878969
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.4651      -0.2204
##
## Degrees of Freedom: 1003 Total (i.e. Null);  1002 Residual
## Null Deviance:       1091
## Residual Deviance: 1066  AIC: 1200
## log likelihood:  -532.8011
## Nagelkerke R2:  0.03777103
## % pres/err predicted correctly:  -368.9597
## % of predictable range [ (model-null)/(1-null) ]:  0.034003
## **************************
## model index:  1
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.10136     -0.07756
##
## Degrees of Freedom: 1003 Total (i.e. Null);  1002 Residual
## Null Deviance:       1091
## Residual Deviance: 1088  AIC: 1232
## log likelihood:  -544.2026
## Nagelkerke R2:  0.003973035
## % pres/err predicted correctly:  -380.9116
## % of predictable range [ (model-null)/(1-null) ]:  0.002795429
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9675
##
## Degrees of Freedom: 1003 Total (i.e. Null);  1003 Residual
## Null Deviance:       1091
## Residual Deviance: 1091  AIC: 1233
## log likelihood:  -545.526
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -381.9822
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
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 ~ CumErr	991.9537	0.0000	1	1	0.2632383	3.5942980	NA	-1.110695	NA	NA	NA
preserved ~ pos	1142.368	150.4144	0	0	0.1110552	2.1727176	NA	NA	NA	-0.3367371	NA
preserved ~ (I(pos^2) + pos)	1143.672	151.7183	0	0	0.1120012	3.3746170	NA	NA	0.0149261	-0.4599594	NA
preserved ~ stimlen	1199.550	207.5965	0	0	0.0377712	0.4651257	NA	NA	NA	NA	-0.2204216
preserved ~ CumPres	1232.148	240.1947	0	0	0.0039730	0.1013577	-0.0775609	NA	NA	NA	NA
preserved ~ 1	1233.159	241.2057	0	0	0.0000000	0.9674864	NA	NA	NA	NA	NA

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

```
#  also reduces data)

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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       1.554       -1.237
##
## Degrees of Freedom: 863 Total (i.e. Null);  862 Residual
## Null Deviance:       929.1
## Residual Deviance: 779.8     AIC: 863.6
## log likelihood:  -389.8964
## Nagelkerke R2:  0.2408698
## % pres/err predicted correctly:  -259.3253
## % of predictable range [ (model-null)/(1-null) ]:  0.1989114
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1971      -0.3395
##
## Degrees of Freedom: 863 Total (i.e. Null);  862 Residual
## Null Deviance:       929.1
## Residual Deviance: 858.7     AIC: 968.7
## log likelihood:  -429.3323
## Nagelkerke R2:  0.1188096
## % pres/err predicted correctly:  -293.7107
## % of predictable range [ (model-null)/(1-null) ]:  0.09309836
```

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.38953      0.01485     -0.46080
##
## Degrees of Freedom: 863 Total (i.e. Null);  861 Residual
## Null Deviance:       929.1
## Residual Deviance: 858.1     AIC: 970.2
## log likelihood:  -429.0267
## Nagelkerke R2:  0.1197988
## % pres/err predicted correctly:  -293.2121
## % of predictable range [ (model-null)/(1-null) ]:  0.09463293
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.4348      -0.2121
##
## Degrees of Freedom: 863 Total (i.e. Null);  862 Residual
## Null Deviance:       929.1
## Residual Deviance: 908.6     AIC: 1024
## log likelihood:  -454.2873
## Nagelkerke R2:  0.03561113
## % pres/err predicted correctly:  -313.5492
## % of predictable range [ (model-null)/(1-null) ]:  0.03205027
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.2335      -0.1581
##
## Degrees of Freedom: 863 Total (i.e. Null);  862 Residual
## Null Deviance:       929.1
## Residual Deviance: 921.8     AIC: 1045
## log likelihood:  -460.9167
## Nagelkerke R2:  0.01268911
## % pres/err predicted correctly:  -320.8754
## % of predictable range [ (model-null)/(1-null) ]:  0.009505677
## ***************************
## model index:  6
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9981
##
## Degrees of Freedom: 863 Total (i.e. Null);  863 Residual
## Null Deviance:       929.1
## Residual Deviance: 929.1     AIC: 1051
## log likelihood:  -464.5433
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -323.9644
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

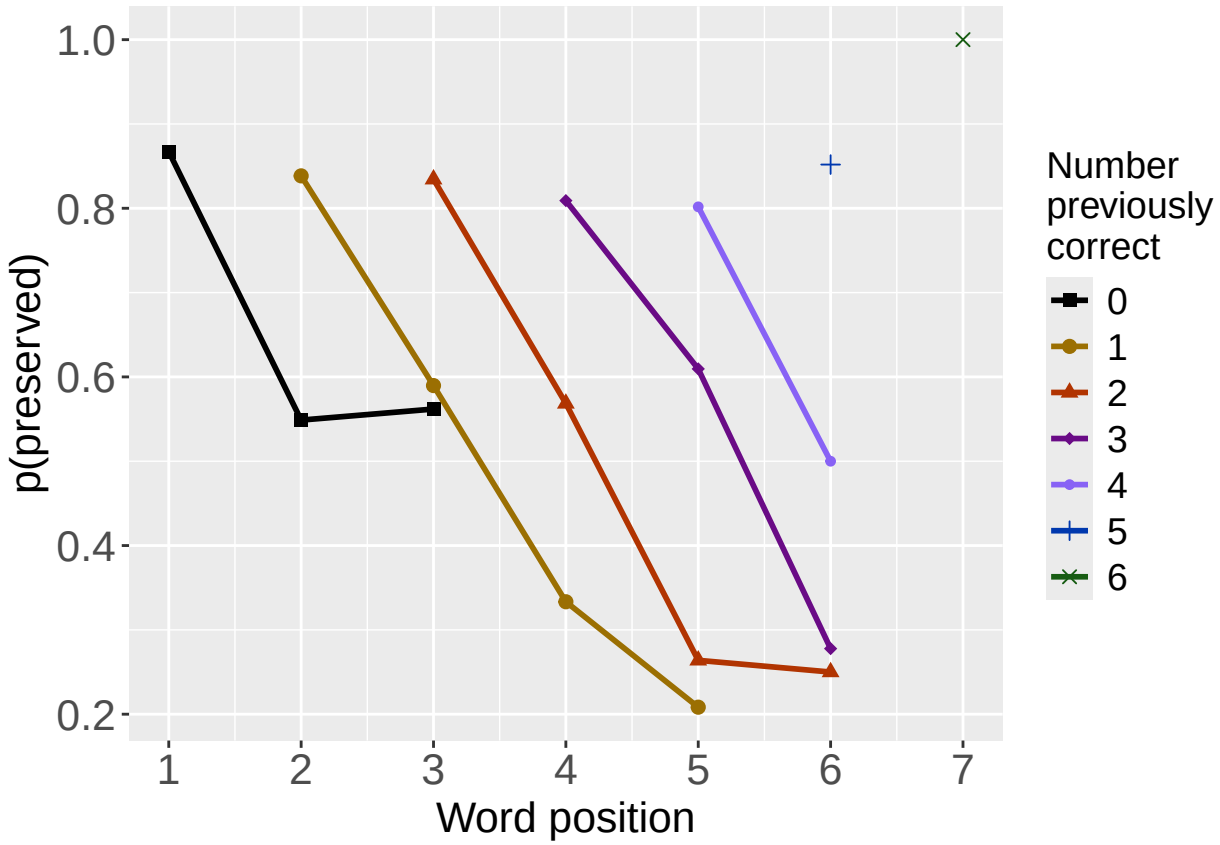
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(posˆ2)	pos	stimlen
preserved ~ CumErr	863.6064	0.0000	1	1	0.2408698	1.5539721	NA	-1.236625	NA	NA	NA
preserved ~ pos	968.7153	105.1089	0	0	0.1188090	2.1970994	NA	NA	NA	-0.3394832	NA
preserved ~ (I(posˆ2) + pos)	970.1668	106.5603	0	0	0.1197988	2.3895259	NA	NA	0.0148481	-0.4607979	NA
preserved ~ stimlen	1024.428	160.8225	0	0	0.0356112	2.4347561	NA	NA	NA	NA	-0.2121295
preserved ~ CumPres	1044.955	181.3488	0	0	0.0126891	1.2334595	-0.1581112	NA	NA	NA	NA
preserved ~ 1	1051.235	187.6294	0	0	0.0000000	0.9981291	NA	NA	NA	NA	NA

```
# 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.
```

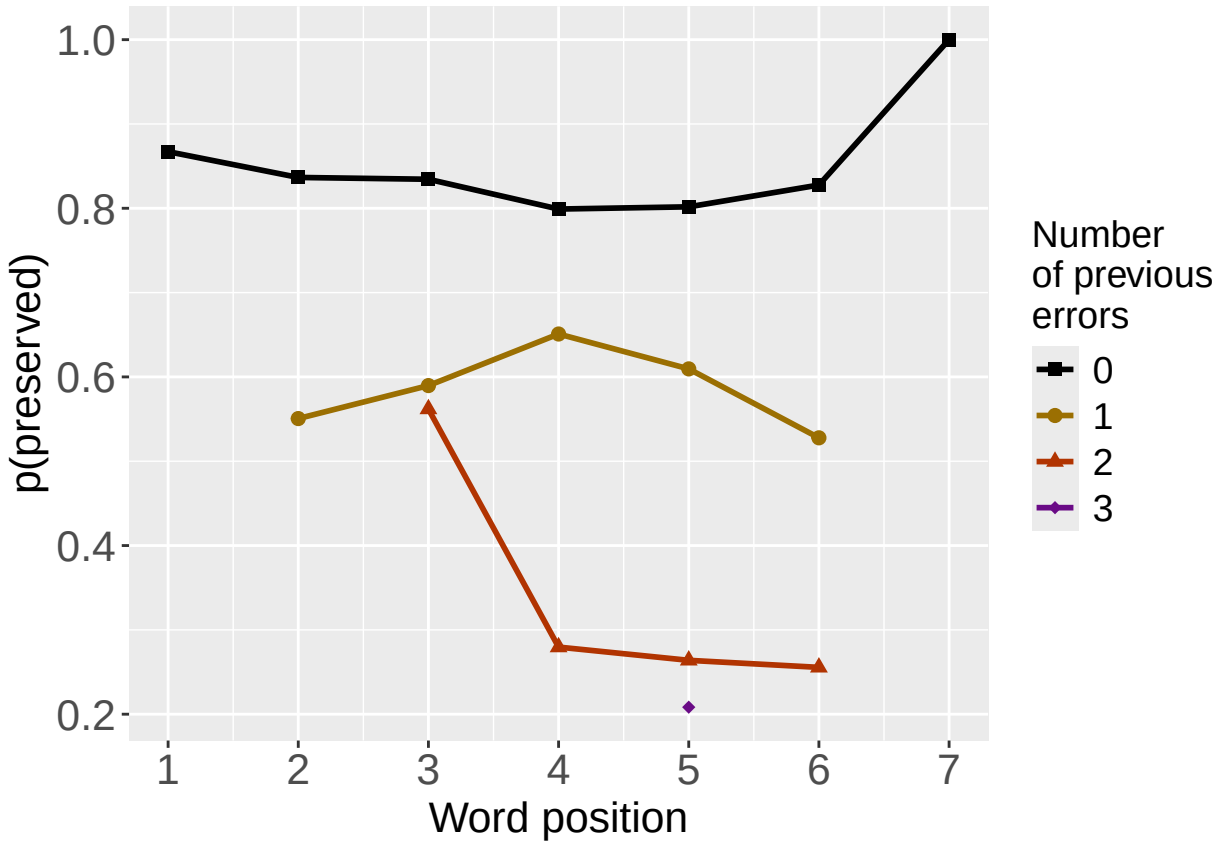
```
print(PrevCorPlot)
```




```
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
```

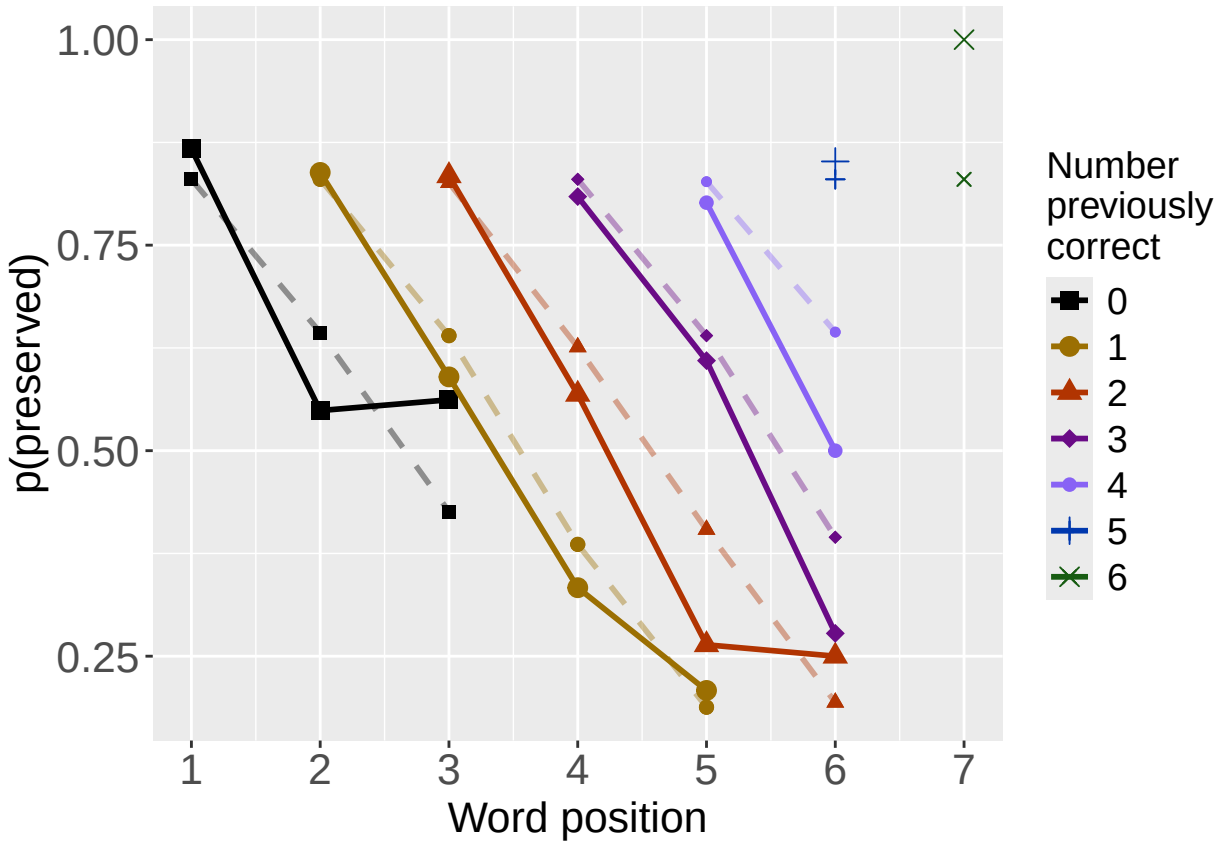
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

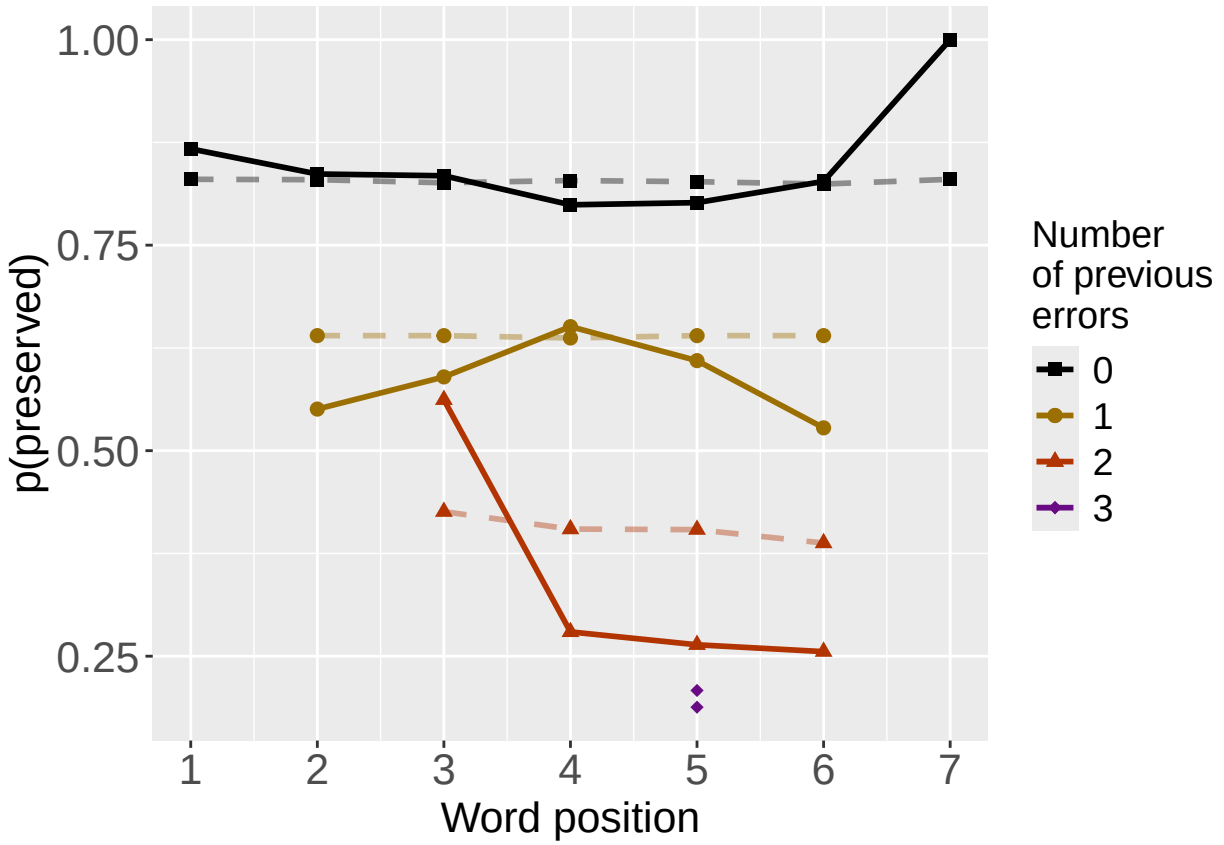
```
print(PrevCorPlot)
```




```
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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      2.3071      -1.0288       0.0570      -0.4541
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 965   AIC: 1073
## log likelihood:  -482.4927
## Nagelkerke R2:  0.2645965
## % pres/err predicted correctly:  -323.4142
## % of predictable range [ (model-null)/(1-null) ]:  0.2141058
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       1.586       -1.011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 973.2     AIC: 1077
## log likelihood:  -486.6214
## Nagelkerke R2:  0.2550256
## % pres/err predicted correctly:  -325.9679
## % of predictable range [ (model-null)/(1-null) ]:  0.2079195
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.24537      0.01036     -0.41171
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1238
## log likelihood:  -546.8309
## Nagelkerke R2:  0.1067644
## % pres/err predicted correctly:  -376.5375
## % of predictable range [ (model-null)/(1-null) ]:  0.08541439
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1072.574	0.000000	1.0000000	0.8795082	0.2645965	2.307076	-1.028789	0.0569979	-0.4541383

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1076.549	3.975562	0.1369991	0.1204918	0.2550256	1.586363	-1.011198	NA	NA
preserved ~ I(posˆ2) + pos	1237.593	165.019221	0.0000000	0.0000000	0.1067644	2.245368	NA	0.0103572	-0.4117057

```
########
# 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)       CumErr      stimlen
##     1.93395     -0.98568     -0.05425
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 972   AIC: 1076
## log likelihood:  -486.0165
## Nagelkerke R2:  0.2564324
## % pres/err predicted correctly:  -325.2726
## % of predictable range [ (model-null)/(1-null) ]:  0.2096038
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       1.586       -1.011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 973.2     AIC: 1077
## log likelihood:  -486.6214
## Nagelkerke R2:  0.2550256
## % pres/err predicted correctly:  -325.9679
## % of predictable range [ (model-null)/(1-null) ]:  0.2079195
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5791      -0.2402
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1140  AIC: 1288
## log likelihood:  -570.1462
## Nagelkerke R2:  0.04474136
## % pres/err predicted correctly:  -395.4661
## % of predictable range [ (model-null)/(1-null) ]:  0.03955988
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	1075.977	0.0000000	1.0000000	0.5710055	0.2564324	1.933951	-0.9856793	-0.0542510
preserved ~ CumErr	1076.549	0.5719092	0.7512967	0.4289945	0.2550256	1.586363	-1.0111985	NA
preserved ~ stimlen	1288.443	212.4654213	0.0000000	0.0000000	0.0447414	2.579074	NA	-0.2401896

```
########
# 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)
}
```

```
## 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)       CumErr
##       1.586       -1.011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
```

```
## Null Deviance:       1173
## Residual Deviance: 973.2     AIC: 1077
## log likelihood:  -486.6214
## Nagelkerke R2:  0.2550256
## % pres/err predicted correctly:  -325.9679
## % of predictable range [ (model-null)/(1-null) ]:  0.2079195
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     1.62161     -1.00924     -0.02011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 973.1     AIC: 1078
## log likelihood:  -486.5438
## Nagelkerke R2:  0.2552061
## % pres/err predicted correctly:  -325.9018
## % of predictable range [ (model-null)/(1-null) ]:  0.2080796
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.05066     -0.05645
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1171  AIC: 1330
## log likelihood:  -585.5799
## Nagelkerke R2:  0.002179638
## % pres/err predicted correctly:  -411.1815
## % of predictable range [ (model-null)/(1-null) ]:  0.001489387
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr	1076.549	0.000000	1.0000000	0.7184369	0.2550256	1.586363	-1.011198	NA
preserved ~ CumErr + CumPres	1078.423	1.873442	0.3919107	0.2815631	0.2552061	1.621613	-1.009243	-0.0201138
preserved ~ CumPres	1330.175	253.625822	0.0000000	0.0000000	0.0021796	1.050662	NA	-0.0564507

```
########
# 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)       CumErr
##       1.586       -1.011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 973.2     AIC: 1077
## log likelihood:  -486.6214
## Nagelkerke R2:  0.2550256
## % pres/err predicted correctly:  -325.9679
## % of predictable range [ (model-null)/(1-null) ]:  0.2079195
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     1.64173     -0.98913     -0.02011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 973.1     AIC: 1078
## log likelihood:  -486.5438
## Nagelkerke R2:  0.2552061
## % pres/err predicted correctly:  -325.9018
## % of predictable range [ (model-null)/(1-null) ]:  0.2080796
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.1069      -0.3264
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 1094  AIC: 1236
## log likelihood:  -547.0127
## Nagelkerke R2:  0.1062913
## % pres/err predicted correctly:  -376.9225
## % of predictable range [ (model-null)/(1-null) ]:  0.08448177
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr	1076.549	0.000000	1.0000000	0.7184369	0.2550256	1.586363	-1.0111985	NA
preserved ~ CumErr + pos	1078.423	1.873442	0.3919107	0.2815631	0.2552061	1.641727	-0.9891291	-0.0201138
preserved ~ pos	1235.894	159.344515	0.0000000	0.0000000	0.1062913	2.106870	NA	-0.3264059

```
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)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + I(pos^2) + pos	1072.574	0.0000000	1.0000000	0.8795081	0.2264596	2.5307076	-1.0287890	0.0569979	-0.4541383	NA	NA
preserved ~ CumErr + stimlen	1075.977	0.0000000	1.0000000	0.5710055	0.2564321	1.4933951	-0.9856793	NA	NA	-0.0542510	NA
preserved ~ CumErr	1076.549	3.9755616	0.1369991	0.1120491	0.2550256	1.586363	-1.0111985	NA	NA	NA	NA
preserved ~ CumErr	1076.549	0.5719092	0.7512967	0.4289945	0.2550256	1.586363	-1.0111985	NA	NA	NA	NA
preserved ~ CumErr	1076.549	0.0000000	1.0000000	0.7184369	0.2550256	1.586363	-1.0111985	NA	NA	NA	NA
preserved ~ CumErr	1076.549	0.0000000	1.0000000	0.7184369	0.2550256	1.586363	-1.0111985	NA	NA	NA	NA
preserved ~ CumErr + CumPres	1078.423	1.8734425	0.3919107	0.2815631	0.2552061	1.621613	-1.0092429	NA	NA	NA	-0.0201138
preserved ~ CumErr + pos	1078.423	1.8734425	0.3919107	0.2815631	0.2552061	1.641727	-0.9891291	NA	-0.0201138	NA	NA
preserved ~ pos	1235.894	159.3445148	0.0000000	0.0000000	0.1062913	2.106870	NA	NA	-0.3264059	NA	NA
preserved ~ I(pos^2) + pos	1237.593	165.0192212	0.0000000	0.0000000	0.1067642	2.245368	NA	0.0103572	-0.4117057	NA	NA
preserved ~ stimlen	1288.443	212.4654213	0.0000000	0.0000000	0.0447412	1.579074	NA	NA	NA	-0.2401896	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumPres	1330.173	253.625821	0.5000000	0.0000000	0.0002179	1.6050662	NA	NA	NA	NA	-0.0564507

```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!
## 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)       CumErr     I(pos^2)          pos      stimlen
##     2.90540     -1.02504      0.06502     -0.48372     -0.09224
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1070 Residual
## Null Deviance:       1173
## Residual Deviance: 962   AIC: 1069
## log likelihood:  -481.0217
## Nagelkerke R2:  0.2679889
## % pres/err predicted correctly:  -321.9974
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.2175379
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     2.79096     -1.01966      0.06470     -0.48369     -0.07452      0.06097
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1069 Residual
## Null Deviance:       1173
## Residual Deviance: 960.7     AIC: 1072
## log likelihood:  -480.3283
## Nagelkerke R2:  0.2695847
## % pres/err predicted correctly:  -321.598
## % of predictable range [ (model-null)/(1-null) ]:  0.2185056
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      2.3071      -1.0288       0.0570      -0.4541
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 965   AIC: 1073
## log likelihood:  -482.4927
## Nagelkerke R2:  0.2645965
## % pres/err predicted correctly:  -323.4142
## % of predictable range [ (model-null)/(1-null) ]:  0.2141058
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     2.30717     -1.02084      0.05857     -0.46122      0.07947
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1070 Residual
## Null Deviance:       1173
## Residual Deviance: 962.4     AIC: 1073
## log likelihood:  -481.2184
## Nagelkerke R2:  0.2675359
## % pres/err predicted correctly:  -322.5361
## % of predictable range [ (model-null)/(1-null) ]:  0.2162329
## ***************************
## model index:  2
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9487
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1074 Residual
## Null Deviance:       1173
## Residual Deviance: 1173  AIC: 1330
## log likelihood:  -586.3585
## Nagelkerke R2:  -3.343613e-16
## % pres/err predicted correctly:  -411.7963
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
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)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen	1069.381	0.000000	1.000000	0.5981863	0.2679829	0.9053967	-1.025039	0.0650226	-0.4837176	NA	-0.0922416
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	1071.535	2.154147	0.3405368	0.2037092	0.2695827	0.7909611	-1.019658	0.0647047	-0.4836863	0.0609731	-0.0745208
preserved ~ CumErr + I(pos^2) + pos	1072.574	3.193129	0.2025904	0.1211874	0.2645923	0.3070757	-1.028789	0.0569979	-0.4541383	NA	NA
preserved ~ CumErr + I(pos^2) + pos + log_freq	1073.488	4.102209	0.1285928	0.0769205	0.2675329	0.3071721	-1.020836	0.0585732	-0.4612244	0.0794659	NA
preserved ~ 1	1329.982	260.599798	0.0000000	0.0000000	0.0000000	0.0486953	NA	NA	NA	NA	NA

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

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

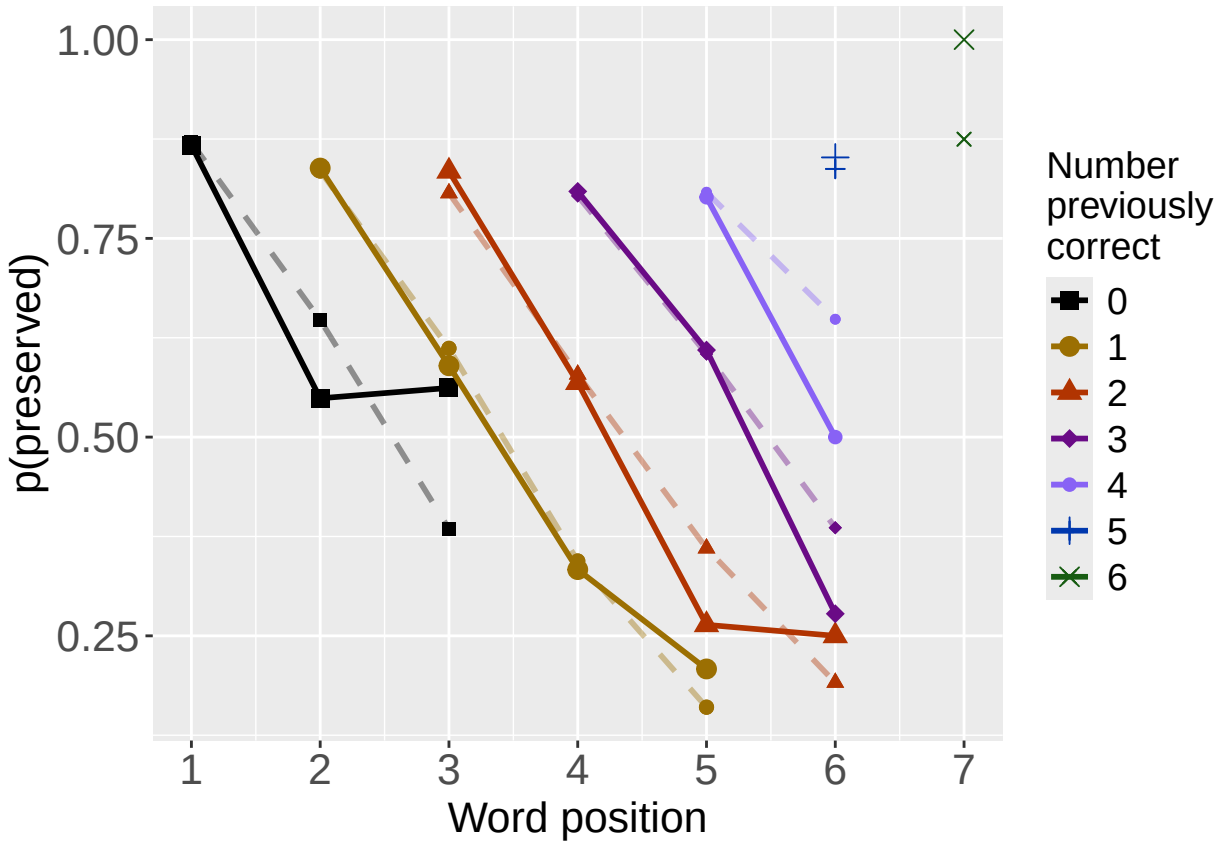
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen
##          Df Deviance    AIC
## CumErr    1  1089.27 1194.6
## I(pos^2)  1   972.03 1077.4
## pos       1   970.99 1076.3
## stimlen   1   964.99 1070.3
## <none>        962.04 1069.4
```

```
################################
# 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.
```

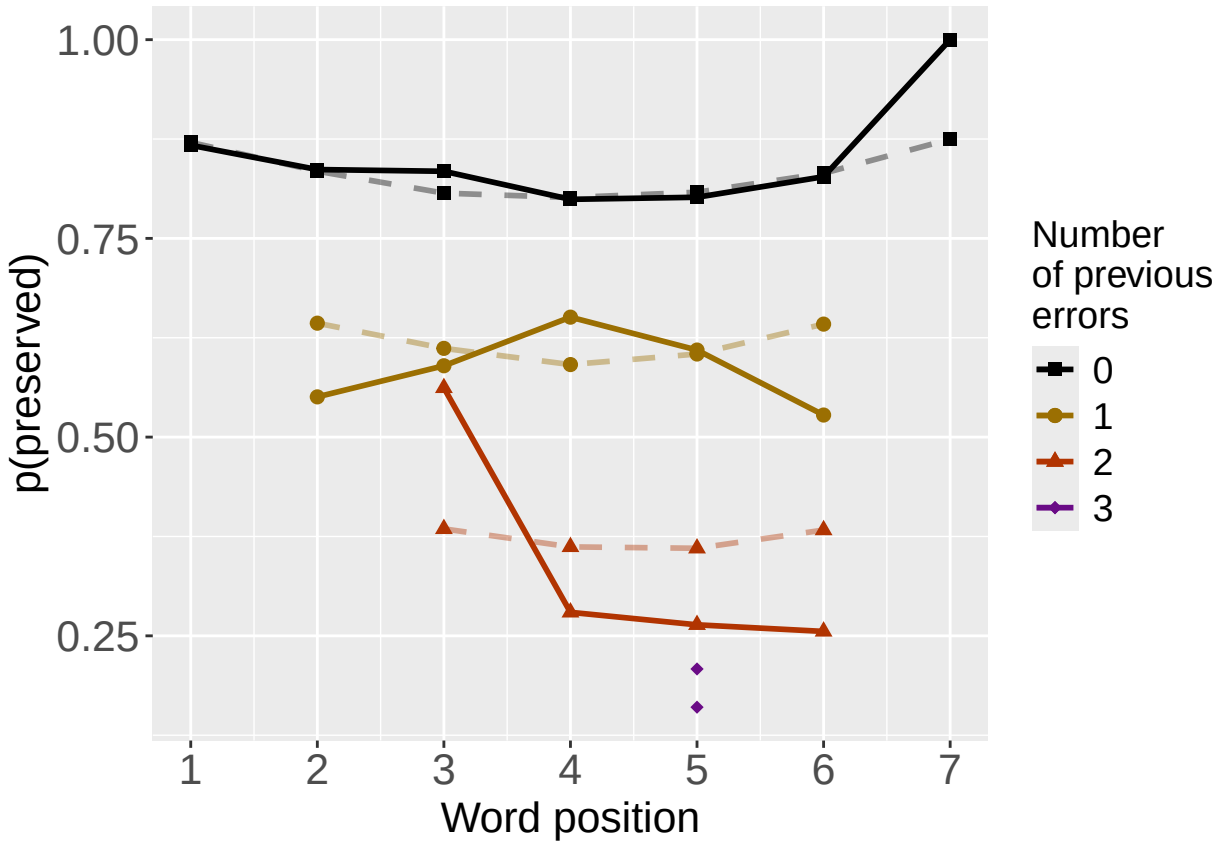
```
print(PrevCorPlot)
```




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

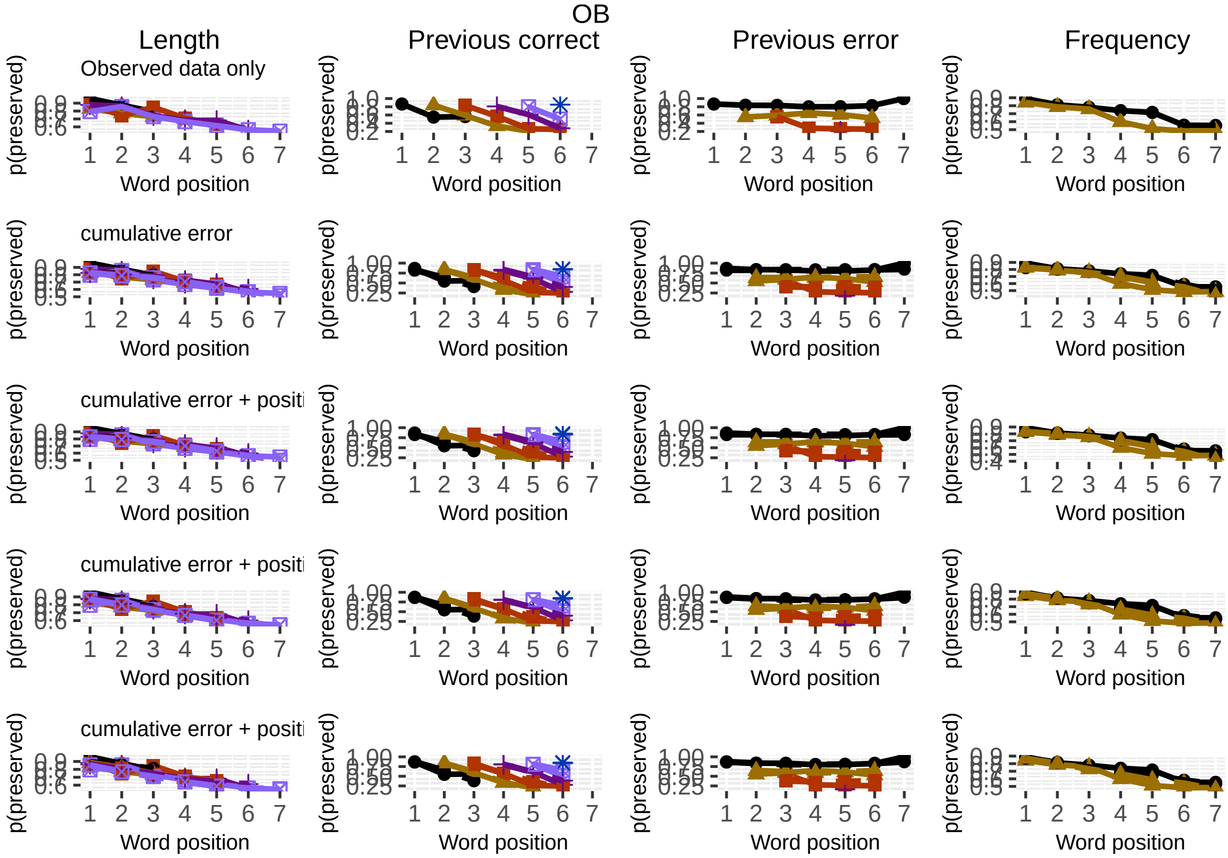
```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)
##    1.554049    -1.042042     0.003466
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1072 Residual
## Null Deviance:       1173
## Residual Deviance: 973   AIC: 1078
## log likelihood:  -486.4787
```

```
## Nagelkerke R2:  0.2553577
## % pres/err predicted correctly:  -325.8888
## % of predictable range [ (model-null)/(1-null) ]:  0.2081111
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       1.586       -1.011
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1073 Residual
## Null Deviance:       1173
## Residual Deviance: 973.2     AIC: 1077
## log likelihood:  -486.6214
## Nagelkerke R2:  0.2550256
## % pres/err predicted correctly:  -325.9679
## % of predictable range [ (model-null)/(1-null) ]:  0.2079195
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      2.3071      -1.0288       0.0570      -0.4541
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1071 Residual
## Null Deviance:       1173
## Residual Deviance: 965   AIC: 1073
## log likelihood:  -482.4927
## Nagelkerke R2:  0.2645965
## % pres/err predicted correctly:  -323.4142
## % of predictable range [ (model-null)/(1-null) ]:  0.2141058
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     2.90540     -1.02504      0.06502     -0.48372     -0.09224
##
## Degrees of Freedom: 1074 Total (i.e. Null);  1070 Residual
## Null Deviance:       1173
## Residual Deviance: 962   AIC: 1069
## log likelihood:  -481.0217
## Nagelkerke R2:  0.2679889
## % pres/err predicted correctly:  -321.9974
## % of predictable range [ (model-null)/(1-null) ]:  0.2175379
```

```
## ***************************
## `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.
## `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 1 row 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 1 row containing missing values or values outside the scale range (`geom_point()`).
## Removed 1 row 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 1 row containing missing values or values outside the scale range (`geom_point()`).
## Removed 1 row 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 1 row containing missing values or values outside the scale range (`geom_point()`).
## Removed 1 row 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 1 row containing missing values or values outside the scale range (`geom_point()`).
## Removed 1 row 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!
## 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!
```

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

	CumErr	I(pos^2)	pos	stimlen
McFadden	0.1438752	0.0222747	0.0252043	0.0109076
SquaredCorrelation	0.1534374	0.0257684	0.0291477	0.0127369
Nagelkerke	0.2163341	0.0363313	0.0410958	0.0179580
Estrella	0.1724009	0.0271444	0.0307172	0.0133138

```
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!
## 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
## CumErr + I(pos^2) + pos + stimlen CumErr + I(pos^2) + pos + stimlen  962.0434
## CumErr + I(pos^2) + pos                     CumErr + I(pos^2) + pos  964.9855
## CumErr + I(pos^2)                                 CumErr + I(pos^2)  972.9573
## CumErr                                                       CumErr  973.2428
## null                                                           null 1172.7170
##                                  deviance_explained percent_explained
## CumErr + I(pos^2) + pos + stimlen          210.6736          17.96457
## CumErr + I(pos^2) + pos                    207.7316          17.71370
## CumErr + I(pos^2)                          199.7597          17.03392
## CumErr                                     199.4742          17.00958
## null                                         0.0000           0.00000
##                                  percent_of_explained_deviance increment_in_explained
## CumErr + I(pos^2) + pos + stimlen                    100.00000              1.3964904
## CumErr + I(pos^2) + pos                               98.60351              3.7839949
## CumErr + I(pos^2)                                     94.81951              0.1355004
## CumErr                                                94.68401             94.6840143
## null                                                        NA              0.0000000
```

```
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")
```

```
NagContributions <- t(DAContributionAverage[3,])
NagTotal<-sum(NagContributions)
NagPercents <- NagContributions / NagTotal
NagResultTable <- data.frame(NagContributions = NagContributions, NagPercents = NagPercents)
```

	deviance	deviance_explained
CumErr + I(pos^2) + pos + stimlen	962.0434	210.6736
CumErr + I(pos^2) + pos	964.9855	207.7316
CumErr + I(pos^2)	972.9573	199.7597
CumErr	973.2428	199.4742
null	1172.7170	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + I(pos^2) + pos + stimlen	17.96457	100.00000	1.3964904
CumErr + I(pos^2) + pos	17.71370	98.60351	3.7839949
CumErr + I(pos^2)	17.03392	94.81951	0.1355004
CumErr	17.00958	94.68401	94.6840143
null	0.00000	NA	0.0000000

```
names(NagResultTable)<-c("NagContributions","NagPercents")
write.csv(NagResultTable,paste0(TablesDir,CurPat,"_",CurTask,"_nagelkerke_contributions_table.csv"),row.names = TRUE)
NagPercents
```

```
##          Nagelkerke
## CumErr   0.69400304
## I(pos^2) 0.11655138
## pos      0.13183600
## stimlen  0.05760958
```

```
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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

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

```
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_CumErr
## 1                     preserved ~ CumErr       0.8896471       973.2428 0.000000000
## 2             preserved ~ CumErr+I(pos^2)       0.8989122       972.9573 0.009265095
## 3         preserved ~ CumErr+I(pos^2)+pos       0.9141261       964.9855 0.024479055
## 4 preserved ~ CumErr+I(pos^2)+pos+stimlen       0.9158097       962.0434 0.026162671
##   diff_CumErr+I(pos^2) diff_CumErr+I(pos^2)+pos diff_CumErr+I(pos^2)+pos+stimlen
## 1         -0.009265095             -0.024479055                     -0.026162671
## 2          0.000000000             -0.015213959                     -0.016897576
## 3          0.015213959              0.000000000                     -0.001683616
## 4          0.016897576              0.001683616                      0.000000000
```

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.8896471	973.2428
preserved ~ CumErr+I(pos^2)	0.8989122	972.9573
preserved ~ CumErr+I(pos^2)+pos	0.9141261	964.9855
preserved ~ CumErr+I(pos^2)+pos+stimlen	0.9158097	962.0434

model	diff_CumErr	diff_CumErr+I(pos^2)	diff_CumErr+I(pos^2)+pos
preserved ~ CumErr	0.0000000	-0.0092651	-0.0244791
preserved ~ CumErr+I(pos^2)	0.0092651	0.0000000	-0.0152140
preserved ~ CumErr+I(pos^2)+pos	0.0244791	0.0152140	0.0000000
preserved ~ CumErr+I(pos^2)+pos+stimlen	0.0261627	0.0168976	0.0016836

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

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