OB - repetition - 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	544	35	132	NA	NA	711
2	67	NA	432	100	112	711
3	315	NA	174	206	16	711
4	299	NA	243	71	37	650
5	237	NA	212	74	34	557
6	208	1	135	71	23	438
7	173	NA	104	29	19	325
8	94	NA	53	26	4	177
9	74	NA	2	NA	7	83

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7651195	0.0492264	0.1856540	NA	NA	711
2	0.0942335	NA	0.6075949	0.1406470	0.1575246	711
3	0.4430380	NA	0.2447257	0.2897328	0.0225035	711
4	0.4600000	NA	0.3738462	0.1092308	0.0569231	650
5	0.4254937	NA	0.3806104	0.1328546	0.0610413	557
6	0.4748858	0.0022831	0.3082192	0.1621005	0.0525114	438

pos_factor	O	P	V	1	S	total
7	0.5323077	NA	0.3200000	0.0892308	0.0584615	325
8	0.5310734	NA	0.2994350	0.1468927	0.0225989	177
9	0.8915663	NA	0.0240964	NA	0.0843373	83

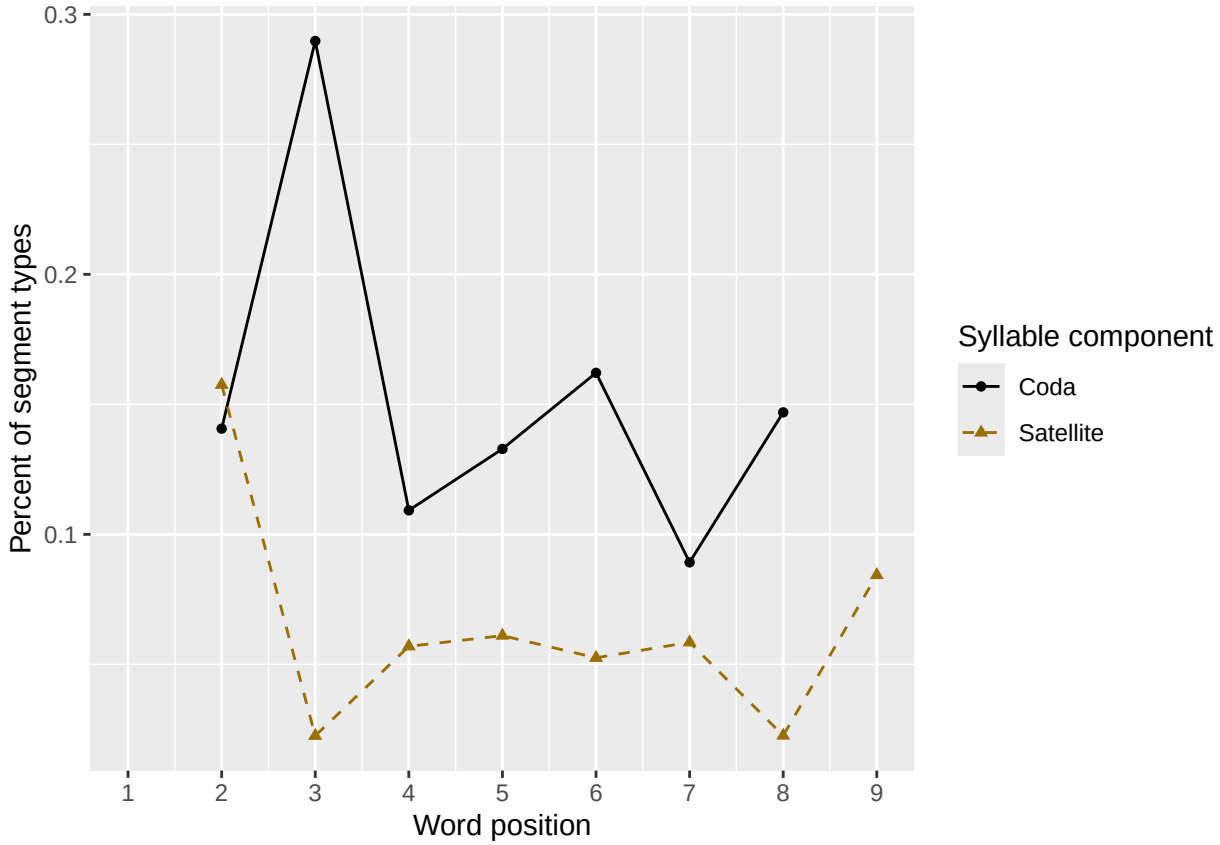
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.984 0.959 0.959 NA     NA     NA     NA     NA     NA
## 2       5 0.978 0.968 0.925  0.882 NA     NA     NA     NA     NA
## 3       6 0.987 0.937 0.912  0.885  0.768 NA     NA     NA     NA
## 4       7 0.978 0.956 0.934  0.854  0.858  0.739 NA     NA     NA
## 5       8 0.964 0.895 0.838  0.800  0.703  0.681  0.582 NA     NA
## 6       9 0.947 0.913 0.865  0.830  0.784  0.713  0.644  0.592 NA
## 7      10 0.935 0.896 0.916  0.744  0.729  0.666  0.633  0.598  0.545
```

```
# 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.
```

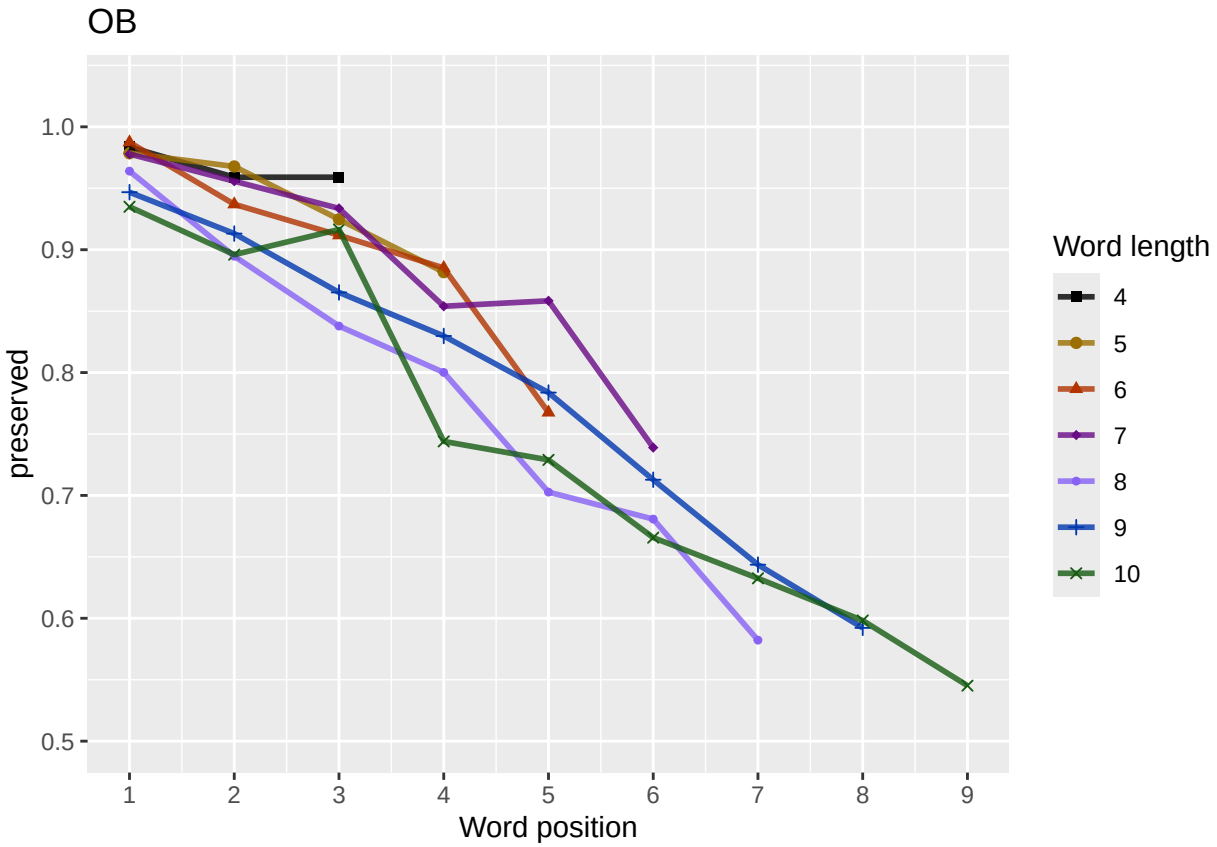
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    61    61    61    NA    NA    NA    NA    NA    NA
## 2       5    93    93    93    93    NA    NA    NA    NA    NA
## 3       6   119   119   119   119   119    NA    NA    NA    NA
## 4       7   113   113   113   113   113   113    NA    NA    NA
## 5       8   148   148   148   148   148   148   148    NA    NA
## 6       9    94    94    94    94    94    94    94    94    NA
## 7      10    83    83    83    83    83    83    83    83    83
```

```
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)
##        6.1606881        -0.2984142         0.0212965        -0.9543988         0.0007292
##      stimlen:pos
##        0.0372486
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 3314  AIC: 3674
## log likelihood:  -1657.089
## Nagelkerke R2:  0.1808021
## % pres/err predicted correctly:  -1070.661
## % of predictable range [ (model-null)/(1-null) ]:  0.1186056
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.09941     -0.14015      0.04312     -0.77730
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 3319  AIC: 3676
## log likelihood:  -1659.702
## Nagelkerke R2:  0.1789571
## % pres/err predicted correctly:  -1071.722
## % of predictable range [ (model-null)/(1-null) ]:  0.1177336
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     6.46406     -0.39353     -0.96830      0.06929
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 3320  AIC: 3678
## log likelihood:  -1660.182
## Nagelkerke R2:  0.178618
## % pres/err predicted correctly:  -1070.607
## % of predictable range [ (model-null)/(1-null) ]:  0.1186506
## **************************
## model index:  6
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.06619      0.03604     -0.75670
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3339  AIC: 3697
## log likelihood:  -1669.441
## Nagelkerke R2:  0.172061
## % pres/err predicted correctly:  -1076.712
## % of predictable range [ (model-null)/(1-null) ]:  0.1136297
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      4.0682      -0.1131      -0.3632
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3341  AIC: 3700
## log likelihood:  -1670.649
## Nagelkerke R2:  0.1712029
## % pres/err predicted correctly:  -1073.161
## % of predictable range [ (model-null)/(1-null) ]:  0.1165498
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.3474      -0.4024
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3355  AIC: 3714
## log likelihood:  -1677.31
## Nagelkerke R2:  0.1664666
## % pres/err predicted correctly:  -1077.069
## % of predictable range [ (model-null)/(1-null) ]:  0.1133354
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      4.0869      -0.3162
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3644  AIC: 4019
## log likelihood:  -1822.099
## Nagelkerke R2:  0.05984294
## % pres/err predicted correctly:  -1169.459
## % of predictable range [ (model-null)/(1-null) ]:  0.03734884
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.573
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4362 Residual
## Null Deviance:       3799
## Residual Deviance: 3799  AIC: 4186
## log likelihood:  -1899.341
## Nagelkerke R2:  1.909818e-16
## % pres/err predicted correctly:  -1214.871
## % 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)	3674.285	0.000000	1.000000	0.6187039	0.0918080	2.1160688	-0.2984142	-1.2954398	0.0372480	0.0212965	0.0007292
preserved ~ stimlen + I(pos^2) + pos	3675.740	0.454582	0.483216	0.2989678	0.0817895	1.7099407	-0.1401541	-1.7773039	NA	0.0431178	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	3678.314	0.034034	0.1330510	0.0823197	0.7178618	6.0464058	-0.3935329	-0.9683007	0.0692890	NA	NA
preserved ~ I(pos^2) + pos	3697.152	22.871998	8.0000108	0.0000067	0.7172064	6.1066192	NA	-0.7567002	NA	0.0360449	NA
preserved ~ stimlen + pos	3699.625	25.340668	3.0000031	0.0000019	0.7171202	6.0968204	-0.1130813	-0.8631589	NA	NA	NA
preserved ~ pos	3713.699	39.413634	4.0000000	0.0000000	0.7166466	6.1347353	NA	-0.4023951	NA	NA	NA
preserved ~ stimlen	4018.936	344.651372	7.2000000	0.0000000	0.5984429	6.0086930	-0.3161878	NA	NA	NA	NA
preserved ~ 1	4185.577	511.284898	8.0000000	0.0000000	0.5400000	5.0573053	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        6.1606881        -0.2984142         0.0212965        -0.9543988         0.0007292
##      stimlen:pos
##        0.0372486
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 3314  AIC: 3674
```

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

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

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

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

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

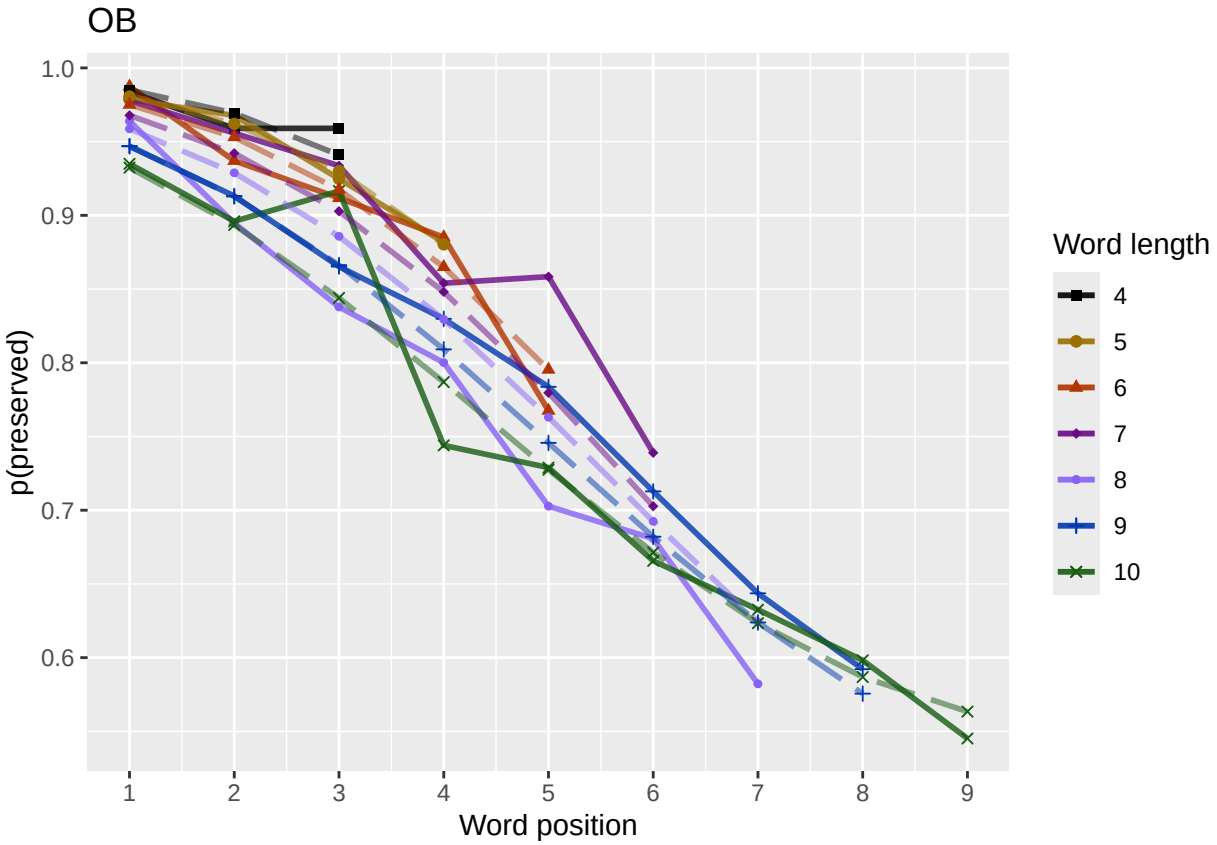
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.985 0.969 0.941 NA     NA     NA     NA     NA     NA
## 2       5 0.981 0.962 0.930  0.880 NA     NA     NA     NA     NA
## 3       6 0.975 0.953 0.917  0.865  0.795 NA     NA     NA     NA
## 4       7 0.968 0.942 0.903  0.848  0.780  0.703 NA     NA     NA
```

```
## 5       8 0.959 0.929 0.886  0.829  0.763  0.692  0.624 NA     NA
## 6       9 0.947 0.913 0.866  0.809  0.746  0.682  0.624  0.576 NA
## 7      10 0.932 0.894 0.844  0.787  0.727  0.671  0.623  0.587  0.563
```

```
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      141   711
```

```
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 141 / 711 = 19.83 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)
##         5.494589         -0.237110         -0.040672         -0.250190          0.009584
##      stimlen:pos
##        -0.035777
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3974 Residual
## Null Deviance:       2341
## Residual Deviance: 2213  AIC: 2570
## log likelihood:  -1106.253
## Nagelkerke R2:  0.07122122
## % pres/err predicted correctly:  -649.7157
## % of predictable range [ (model-null)/(1-null) ]:  0.0361722
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##      5.5532      -0.2180       0.0577      -0.6849
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3976 Residual
## Null Deviance:       2341
## Residual Deviance: 2216  AIC: 2571
## log likelihood:  -1108.158
## Nagelkerke R2:  0.06913545
## % pres/err predicted correctly:  -650.5928
## % of predictable range [ (model-null)/(1-null) ]:  0.03487298
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     6.76087     -0.46453     -0.83251      0.07637
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3976 Residual
## Null Deviance:       2341
## Residual Deviance: 2224  AIC: 2581
## log likelihood:  -1111.996
## Nagelkerke R2:  0.06492659
## % pres/err predicted correctly:  -652.2369
## % of predictable range [ (model-null)/(1-null) ]:  0.03243788
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos
##      4.3817      -0.1869      -0.1655
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3977 Residual
## Null Deviance:       2341
## Residual Deviance: 2239  AIC: 2597
## log likelihood:  -1119.749
## Nagelkerke R2:  0.05640024
## % pres/err predicted correctly:  -654.571
## % of predictable range [ (model-null)/(1-null) ]:  0.02898065
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.90601      0.04636     -0.64615
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3977 Residual
## Null Deviance:       2341
## Residual Deviance: 2249  AIC: 2606
## log likelihood:  -1124.571
## Nagelkerke R2:  0.05108119
## % pres/err predicted correctly:  -656.7521
## % of predictable range [ (model-null)/(1-null) ]:  0.02575
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.1350      -0.2221
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3978 Residual
## Null Deviance:       2341
## Residual Deviance: 2264  AIC: 2622
## log likelihood:  -1132.21
## Nagelkerke R2:  0.0426262
## % pres/err predicted correctly:  -658.9907
## % of predictable range [ (model-null)/(1-null) ]:  0.02243426
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.3394      -0.2651
```

```
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3978 Residual
## Null Deviance:       2341
## Residual Deviance: 2278  AIC: 2643
## log likelihood:  -1139.109
## Nagelkerke R2:  0.03496408
## % pres/err predicted correctly:  -662.2595
## % of predictable range [ (model-null)/(1-null) ]:  0.01759261
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        2.25
##
## Degrees of Freedom: 3979 Total (i.e. Null);  3979 Residual
## Null Deviance:       2341
## Residual Deviance: 2341  AIC: 2712
## log likelihood:  -1170.287
## Nagelkerke R2:  -9.988297e-16
## % pres/err predicted correctly:  -674.1369
## % 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)	2569.822	0.0000000	1.0000000	0.6055094	0.0712254	2.2494589	-0.2371104	-0.4250190	0.0357769	0.0406724	0.0095836
preserved ~ stimlen + I(pos^2) + pos	2570.693	0.8704728	0.6471108	0.3918323	0.0691354	2.2531670	-0.2180198	-0.6848483	NA	0.0576977	NA
preserved ~ stimlen * pos	2580.680	10.8574165	0.0043888	0.0026574	0.0649266	2.2760867	-0.4645320	-0.8325089	0.0763689	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	2596.826	27.004120	0.0000014	0.0000008	0.0564004	2.381721	-0.1869234	-0.1655373	NA	NA	NA
preserved ~ I(pos^2) + pos	2605.523	35.700868	0.0000000	0.0000000	0.0510832	1.906013	NA	-0.6461512	NA	0.0463614	NA
preserved ~ pos	2622.384	52.561925	0.0000000	0.0000000	0.0426262	2.135007	NA	-0.2221436	NA	NA	NA
preserved ~ stimlen	2642.566	72.743553	0.0000000	0.0000000	0.0349641	1.339373	-0.2650637	NA	NA	NA	NA
preserved ~ 1	2712.107	142.284376	0.0000000	0.0000000	0.0000000	2.249585	NA	NA	NA	NA	NA

```r
# plot no fragment data

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

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

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

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

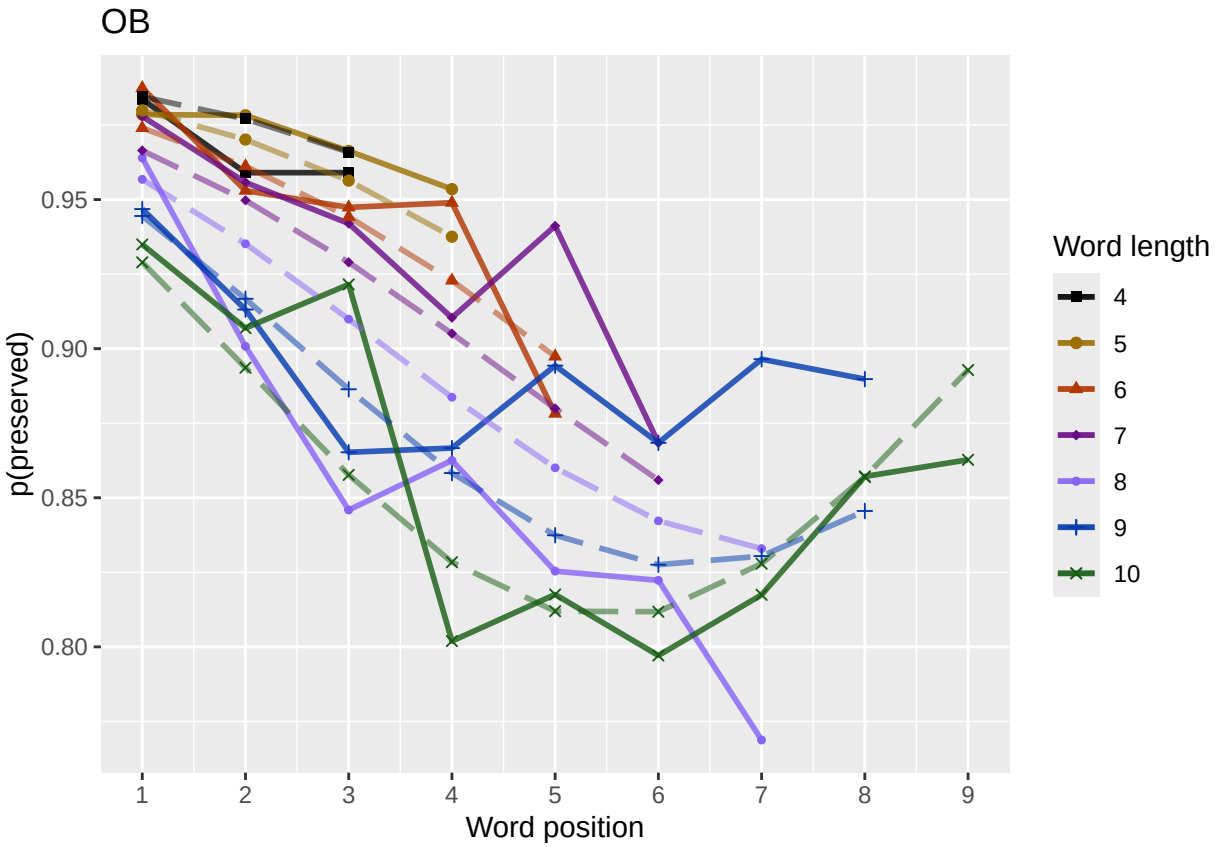
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.984 0.977 0.966 NA     NA     NA     NA     NA     NA
## 2       5 0.980 0.970 0.956  0.938 NA     NA     NA     NA     NA
## 3       6 0.974 0.961 0.944  0.923  0.897 NA     NA     NA     NA
## 4       7 0.966 0.950 0.929  0.905  0.880  0.856 NA     NA     NA
## 5       8 0.957 0.935 0.910  0.884  0.860  0.842  0.833 NA     NA
## 6       9 0.944 0.917 0.886  0.858  0.837  0.828  0.830  0.846 NA
## 7      10 0.929 0.894 0.858  0.828  0.812  0.812  0.828  0.857  0.893
```

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

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

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

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



back to full data

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

```
## [1] "Min/max preserved range: 0.50 - 1.03"
```

```
# 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] -0.01185636
```

```
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] -0.04705421
```

```
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:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          4.87373          -0.11518           0.33979           0.04178          -0.76467
## stimlen:log_freq
##         -0.03145
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 3303  AIC: 3665
## log likelihood:  -1651.406
## Nagelkerke R2:  0.1848073
## % pres/err predicted correctly:  -1067.555
## % of predictable range [ (model-null)/(1-null) ]:  0.1211602
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     4.91873     -0.11628      0.04262     -0.77351      0.08085
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4358 Residual
## Null Deviance:       3799
```

```
## Residual Deviance: 3307  AIC: 3668
## log likelihood:  -1653.521
## Nagelkerke R2:  0.1833178
## % pres/err predicted correctly:  -1068.114
## % of predictable range [ (model-null)/(1-null) ]:  0.1207007
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          4.896089          -0.118409           0.382796           0.042872          -0.770339
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.038413           0.001855          -0.008004
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4355 Residual
## Null Deviance:       3799
## Residual Deviance: 3302  AIC: 3668
## log likelihood:  -1651.015
## Nagelkerke R2:  0.1850819
## % pres/err predicted correctly:  -1067.085
## % of predictable range [ (model-null)/(1-null) ]:  0.1215471
## ****************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##         4.914e+00         -1.155e-01          4.248e-02         -7.729e-01          8.809e-02
## I(pos^2):log_freq       pos:log_freq
##        -8.778e-05         -9.248e-04
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4356 Residual
## Null Deviance:       3799
## Residual Deviance: 3307  AIC: 3671
## log likelihood:  -1653.508
## Nagelkerke R2:  0.1833268
## % pres/err predicted correctly:  -1068.159
## % of predictable range [ (model-null)/(1-null) ]:  0.1206638
## ***************************
## 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)
##        6.1606881         -0.2984142          0.0212965         -0.9543988          0.0007292
##      stimlen:pos
##        0.0372486
```

```
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 3314  AIC: 3674
## log likelihood:  -1657.089
## Nagelkerke R2:  0.1808021
## % pres/err predicted correctly:  -1070.661
## % of predictable range [ (model-null)/(1-null) ]:  0.1186056
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.09941     -0.14015      0.04312     -0.77730
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 3319  AIC: 3676
## log likelihood:  -1659.702
## Nagelkerke R2:  0.1789571
## % pres/err predicted correctly:  -1071.722
## % of predictable range [ (model-null)/(1-null) ]:  0.1177336
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     6.46406     -0.39353     -0.96830      0.06929
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 3320  AIC: 3678
## log likelihood:  -1660.182
## Nagelkerke R2:  0.178618
## % pres/err predicted correctly:  -1070.607
## % of predictable range [ (model-null)/(1-null) ]:  0.1186506
## ***************************
## 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
##         4.0573173         0.0363886        -0.7532672         0.1294952        -0.0002957
##      pos:log_freq
##        -0.0045654
##
```

```
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 3319  AIC: 3685
## log likelihood:  -1659.628
## Nagelkerke R2:  0.179009
## % pres/err predicted correctly:  -1071.203
## % of predictable range [ (model-null)/(1-null) ]:  0.1181603
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          3.86780          -0.08809           0.35988          -0.36360          -0.03380
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4358 Residual
## Null Deviance:       3799
## Residual Deviance: 3323  AIC: 3687
## log likelihood:  -1661.651
## Nagelkerke R2:  0.1775793
## % pres/err predicted correctly:  -1069.013
## % of predictable range [ (model-null)/(1-null) ]:  0.1199614
## ****************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         3.868342         -0.089946          0.360835         -0.360709         -0.037875
##     log_freq:pos
##         0.006583
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 3323  AIC: 3689
## log likelihood:  -1661.5
## Nagelkerke R2:  0.1776857
## % pres/err predicted correctly:  -1068.764
## % of predictable range [ (model-null)/(1-null) ]:  0.1201661
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     3.89473     -0.08872     -0.36436      0.08296
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
```

```
## Null Deviance:       3799
## Residual Deviance: 3328  AIC: 3691
## log likelihood:  -1664.178
## Nagelkerke R2:  0.1757911
## % pres/err predicted correctly:  -1069.795
## % of predictable range [ (model-null)/(1-null) ]:  0.1193185
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     3.888210     -0.086577     -0.366863      0.109454     -0.005377
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4358 Residual
## Null Deviance:       3799
## Residual Deviance: 3328  AIC: 3692
## log likelihood:  -1664.052
## Nagelkerke R2:  0.1758802
## % pres/err predicted correctly:  -1069.896
## % of predictable range [ (model-null)/(1-null) ]:  0.1192354
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.06619      0.03604     -0.75670
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3339  AIC: 3697
## log likelihood:  -1669.441
## Nagelkerke R2:  0.172061
## % pres/err predicted correctly:  -1076.712
## % of predictable range [ (model-null)/(1-null) ]:  0.1136297
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     3.32325     -0.39346      0.09681
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3336  AIC: 3699
## log likelihood:  -1668.049
```

```
## Nagelkerke R2:  0.1730484
## % pres/err predicted correctly:  -1071.937
## % of predictable range [ (model-null)/(1-null) ]:  0.1175568
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      4.0682      -0.1131      -0.3632
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3341  AIC: 3700
## log likelihood:  -1670.649
## Nagelkerke R2:  0.1712029
## % pres/err predicted correctly:  -1073.161
## % of predictable range [ (model-null)/(1-null) ]:  0.1165498
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     3.336150     -0.396623      0.142292     -0.009329
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 3335  AIC: 3700
## log likelihood:  -1667.666
## Nagelkerke R2:  0.1733196
## % pres/err predicted correctly:  -1071.99
## % of predictable range [ (model-null)/(1-null) ]:  0.1175132
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.3474      -0.4024
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3355  AIC: 3714
## log likelihood:  -1677.31
## Nagelkerke R2:  0.1664666
## % pres/err predicted correctly:  -1077.069
## % of predictable range [ (model-null)/(1-null) ]:  0.1133354
```

```
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          3.90145          -0.29348           0.36055          -0.03441
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 3627  AIC: 4007
## log likelihood:  -1813.266
## Nagelkerke R2:  0.06655232
## % pres/err predicted correctly:  -1166.147
## % of predictable range [ (model-null)/(1-null) ]:  0.04007288
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     3.92758     -0.29444      0.07638
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3632  AIC: 4011
## log likelihood:  -1816.103
## Nagelkerke R2:  0.06440025
## % pres/err predicted correctly:  -1166.783
## % of predictable range [ (model-null)/(1-null) ]:  0.03954954
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.0869      -0.3162
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3644  AIC: 4019
## log likelihood:  -1822.099
## Nagelkerke R2:  0.05984294
## % pres/err predicted correctly:  -1169.459
## % of predictable range [ (model-null)/(1-null) ]:  0.03734884
## **************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.573
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4362 Residual
## Null Deviance:       3799
## Residual Deviance: 3799  AIC: 4186
## log likelihood:  -1899.341
## Nagelkerke R2:  1.909818e-16
## % pres/err predicted correctly:  -1214.871
## % 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.names
kable(FLPAICSummary)
```

Model	AIC DeltaAIC AICexp AICwt 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 + I(pos^2) + pos	3664.6850000.0000000.6710378418873725	-0.1151840	0.3397928	-0.0314529	-0.7646697	NA	NA	0.0417759 NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	3667.2.5871072370.7539610834978726	-0.1162784	0.080852 NA		-0.7735126	NA	NA	0.0426238 NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	3667.3.49634820561367164081996089	-0.1184091	0.3827960	-0.0384127	-0.7703385	NA	-0.0080045	0.0428717 NA	0.0018554	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	3671.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1155359	0.0880029	NA	-0.7729340	-0.0409248	NA	0.0424752	-0.0000878	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	3674.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2984142	NA	NA	-0.9543988	NA	NA	0.0212965	NA	NA	0.0372486	0.0607292
preserved ~ stimlen + I(pos^2) + pos	3675.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1401541	NA	NA	-0.7773039	NA	NA	0.0431178	NA	NA	NA	NA
preserved ~ stimlen * pos	3678.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3935329	NA	NA	-0.9683007	NA	NA	NA	NA	NA	0.0692890	NA
preserved ~ (I(pos^2) + pos) * log_freq	3685.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1294052	NA	-0.7532670	-0.0245654	NA	0.0363886	-0.0002957	NA	NA	NA
preserved ~ stimlen * log_freq + pos	3687.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0880935	0.3598777	-0.0337970	-0.3636001	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	3688.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0899455	0.3608352	-0.0378753	-0.3607087	NA	0.0065829	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	3690.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0887193	0.0829605	NA	-0.3643620	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	3692.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0865774	0.1094543	NA	-0.3668602	-0.0253772	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	3697.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.7567002	NA	NA	0.0360149	NA	NA	NA	NA
preserved ~ pos + log_freq	3699.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.0968098	NA	-0.3934632	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	3699.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1130813	NA	NA	-0.3631589	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	3699.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1422016	NA	-0.3960229	-0.0293290	NA	NA	NA	NA	NA	NA
preserved ~ pos	3713.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.4023951	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	4006.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2934761	0.3605468	-0.0344074	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAICexpAICwNagR2(Intercept)stimlenlog_freqstimlen:log_freqpos log_freq:I(pos^2)... stimlen:I(pos^2)										
preserved ~ stimlen + log_freq	4011.346 1.326 1.000000000.06410927579	-0.2944384	0.07638NA01	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	4018.336.2513280000.005948426930	-0.3161878	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	4185.520.884850000000001573053NA		NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          4.87373          -0.11518           0.33979           0.04178          -0.76467
## stimlen:log_freq
##         -0.03145
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 3303  AIC: 3665
```

```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"
```

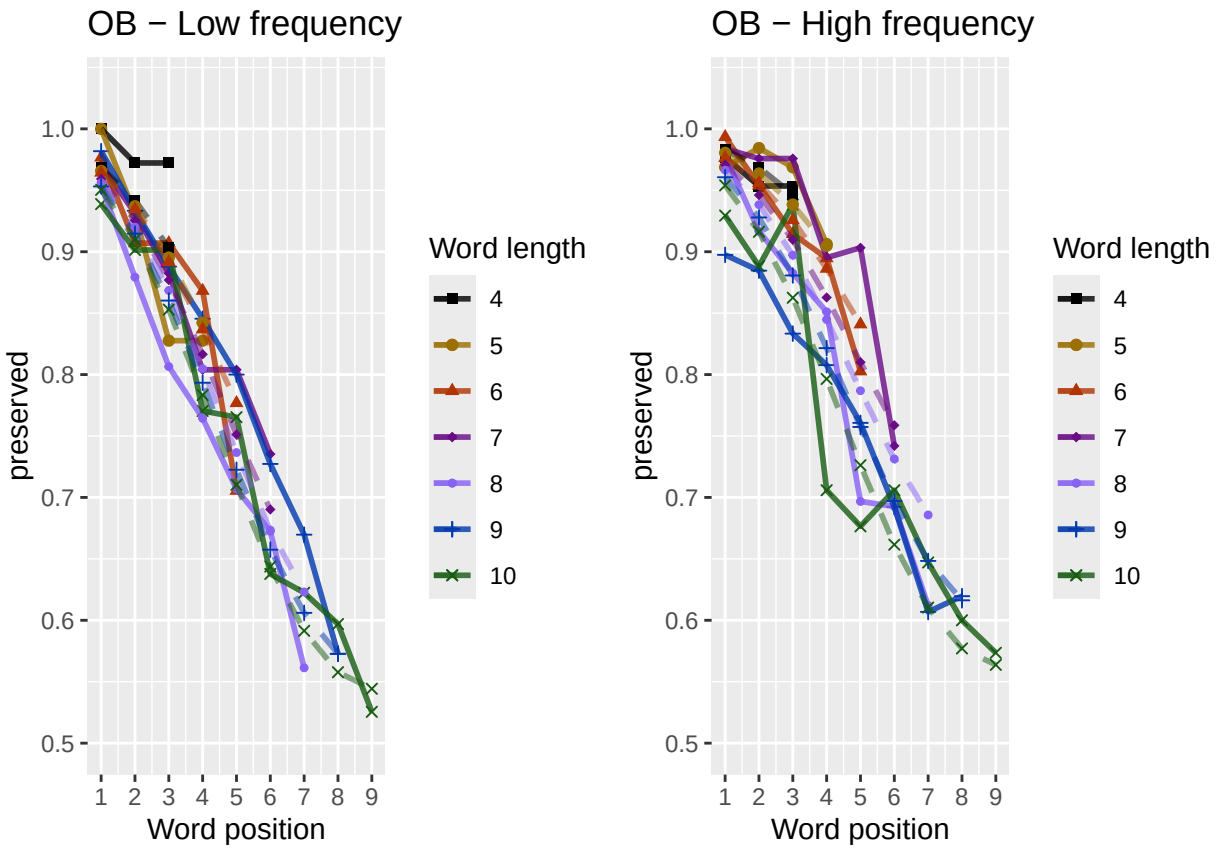
```r
PosDat$FLPFitted<-fitted(BestFLPModel)

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.
```

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




```
# 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
##       2.333       -1.541
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 2791  AIC: 3040
## log likelihood:  -1395.522
## Nagelkerke R2:  0.3547444
## % pres/err predicted correctly:  -834.6394
## % of predictable range [ (model-null)/(1-null) ]:  0.3127234
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.06619      0.03604     -0.75670
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3339  AIC: 3697
## log likelihood:  -1669.441
## Nagelkerke R2:  0.172061
## % pres/err predicted correctly:  -1076.712
## % of predictable range [ (model-null)/(1-null) ]:  0.1136297
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.3474      -0.4024
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3355  AIC: 3714
## log likelihood:  -1677.31
## Nagelkerke R2:  0.1664666
## % pres/err predicted correctly:  -1077.069
## % of predictable range [ (model-null)/(1-null) ]:  0.1133354
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)      stimlen
##      4.0869      -0.3162
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3644  AIC: 4019
## log likelihood:  -1822.099
## Nagelkerke R2:  0.05984294
## % pres/err predicted correctly:  -1169.459
## % of predictable range [ (model-null)/(1-null) ]:  0.03734884
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##       1.993       -0.157
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3741  AIC: 4128
## log likelihood:  -1870.331
## Nagelkerke R2:  0.0227243
## % pres/err predicted correctly:  -1199.653
## % of predictable range [ (model-null)/(1-null) ]:  0.01251576
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.573
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4362 Residual
## Null Deviance:       3799
## Residual Deviance: 3799  AIC: 4186
## log likelihood:  -1899.341
## Nagelkerke R2:  1.909818e-16
## % pres/err predicted correctly:  -1214.871
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
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	3039.690	0.0000	1	1	0.3547442	2.333274	NA	-1.54082	NA	NA	NA
preserved ~ (I(pos^2) + pos)	3697.157	657.4675	0	0	0.1720610	1.066192	NA	NA	0.0360449	-0.7567002	NA
preserved ~ pos	3713.699	674.0092	0	0	0.1664666	1.347353	NA	NA	NA	-0.4023951	NA
preserved ~ stimlen	4018.936	979.2469	0	0	0.0598429	1.086930	NA	NA	NA	NA	-0.3161878
preserved ~ CumPres	4127.850	1088.1601	0	0	0.0227243	3.993030	-0.1570258	NA	NA	NA	NA
preserved ~ 1	4185.570	1145.8804	0	0	0.0000000	1.573053	NA	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	0.8439919	577
O	0.8095558	2011
P	1.0000000	36
S	0.7513228	252
V	0.8562094	1487

```
#  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
##       2.337       -1.670
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       3506
## Residual Deviance: 2561  AIC: 2803
## log likelihood:  -1280.262
## Nagelkerke R2:  0.3588945
```

```
## % pres/err predicted correctly:  -763.283
## % of predictable range [ (model-null)/(1-null) ]:  0.3168694
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.33587      0.04006     -0.82587
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       3506
## Residual Deviance: 3037  AIC: 3378
## log likelihood:  -1518.323
## Nagelkerke R2:  0.188576
## % pres/err predicted correctly:  -978.8835
## % of predictable range [ (model-null)/(1-null) ]:  0.1241616
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.5070      -0.4271
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       3506
## Residual Deviance: 3054  AIC: 3395
## log likelihood:  -1526.854
## Nagelkerke R2:  0.1820959
## % pres/err predicted correctly:  -979.4495
## % of predictable range [ (model-null)/(1-null) ]:  0.1236557
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.1252      -0.3178
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       3506
## Residual Deviance: 3363  AIC: 3720
## log likelihood:  -1681.413
## Nagelkerke R2:  0.05986324
## % pres/err predicted correctly:  -1076.363
## % of predictable range [ (model-null)/(1-null) ]:  0.03703297
## **************************
```

```
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0261      -0.1701
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       3506
## Residual Deviance: 3448  AIC: 3817
## log likelihood:  -1723.894
## Nagelkerke R2:  0.02461075
## % pres/err predicted correctly:  -1102.385
## % of predictable range [ (model-null)/(1-null) ]:  0.01377391
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.596
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4074 Residual
## Null Deviance:       3506
## Residual Deviance: 3506  AIC: 3875
## log likelihood:  -1753.035
## Nagelkerke R2:  -3.848254e-16
## % pres/err predicted correctly:  -1117.795
## % 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	2802.664	0.0000	1	1	0.3588942	3.337215	NA	-1.670113	NA	NA	NA
preserved ~ (I(pos^2) + pos)	3377.536	574.8663	0	0	0.1885760	0.335868	NA	NA	0.0400613	-0.8258675	NA
preserved ~ pos	3395.297	592.6329	0	0	0.1820959	0.506992	NA	NA	NA	-0.4270551	NA
preserved ~ stimlen	3719.644	916.9800	0	0	0.0598632	2.125195	NA	NA	NA	NA	-0.3177784
preserved ~ CumPres	3817.003	1014.3389	0	0	0.0246108	2.026096	-0.170077	NA	NA	NA	NA
preserved ~ 1	3874.738	1072.0744	0	0	0.0000000	0.595982	NA	NA	NA	NA	NA

```r
#  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
##       2.268       -1.855
##
## Degrees of Freedom: 3497 Total (i.e. Null);  3496 Residual
## Null Deviance:       3038
## Residual Deviance: 2271  AIC: 2474
## log likelihood:  -1135.608
## Nagelkerke R2:  0.3392641
## % pres/err predicted correctly:  -680.757
## % of predictable range [ (model-null)/(1-null) ]:  0.2974763
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.33753      0.04247     -0.84385
##
## Degrees of Freedom: 3497 Total (i.e. Null);  3495 Residual
## Null Deviance:       3038
## Residual Deviance: 2618  AIC: 2898
## log likelihood:  -1309.17
## Nagelkerke R2:  0.1949296
```

```
## % pres/err predicted correctly:  -846.313
## % of predictable range [ (model-null)/(1-null) ]:  0.1268773
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.4706      -0.4221
##
## Degrees of Freedom: 3497 Total (i.e. Null);  3496 Residual
## Null Deviance:       3038
## Residual Deviance: 2636  AIC: 2917
## log likelihood:  -1318.195
## Nagelkerke R2:  0.1870254
## % pres/err predicted correctly:  -847.0359
## % of predictable range [ (model-null)/(1-null) ]:  0.1261324
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.9973      -0.3047
##
## Degrees of Freedom: 3497 Total (i.e. Null);  3496 Residual
## Null Deviance:       3038
## Residual Deviance: 2922  AIC: 3216
## log likelihood:  -1460.892
## Nagelkerke R2:  0.05647444
## % pres/err predicted correctly:  -935.746
## % of predictable range [ (model-null)/(1-null) ]:  0.03472009
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.1611      -0.2656
##
## Degrees of Freedom: 3497 Total (i.e. Null);  3496 Residual
## Null Deviance:       3038
## Residual Deviance: 2945  AIC: 3248
## log likelihood:  -1472.434
## Nagelkerke R2:  0.04544248
## % pres/err predicted correctly:  -943.3688
## % of predictable range [ (model-null)/(1-null) ]:  0.02686515
## ***************************
```

```
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.581
##
## Degrees of Freedom: 3497 Total (i.e. Null);  3497 Residual
## Null Deviance:       3038
## Residual Deviance: 3038  AIC: 3341
## log likelihood:  -1519.188
## Nagelkerke R2:  1.912651e-16
## % pres/err predicted correctly:  -969.4398
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

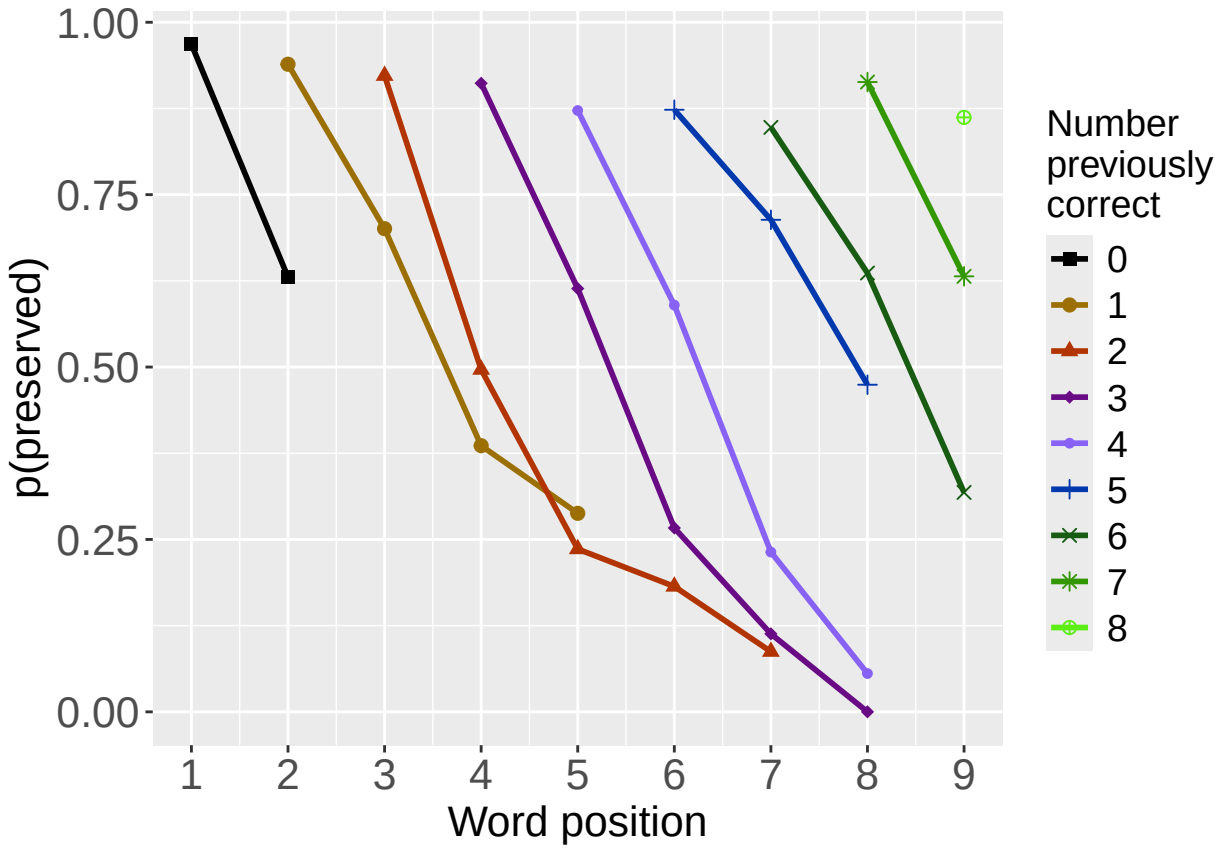
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2474.301	0.0000	1	1	0.3392642	2.268289	NA	-1.854521	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2898.279	423.9780	0	0	0.1949296	4.337531	NA	NA	0.0424713	-0.8438523	NA
preserved ~ pos	2916.685	442.3838	0	0	0.1870253	3.470609	NA	NA	NA	-0.4220585	NA
preserved ~ stimlen	3216.319	742.0179	0	0	0.0564743	3.997337	NA	NA	NA	NA	-0.3046737
preserved ~ CumPres	3247.676	773.3752	0	0	0.0454423	3.161114	-0.2655781	NA	NA	NA	NA
preserved ~ 1	3341.106	866.8055	0	0	0.0000000	1.581298	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.
```

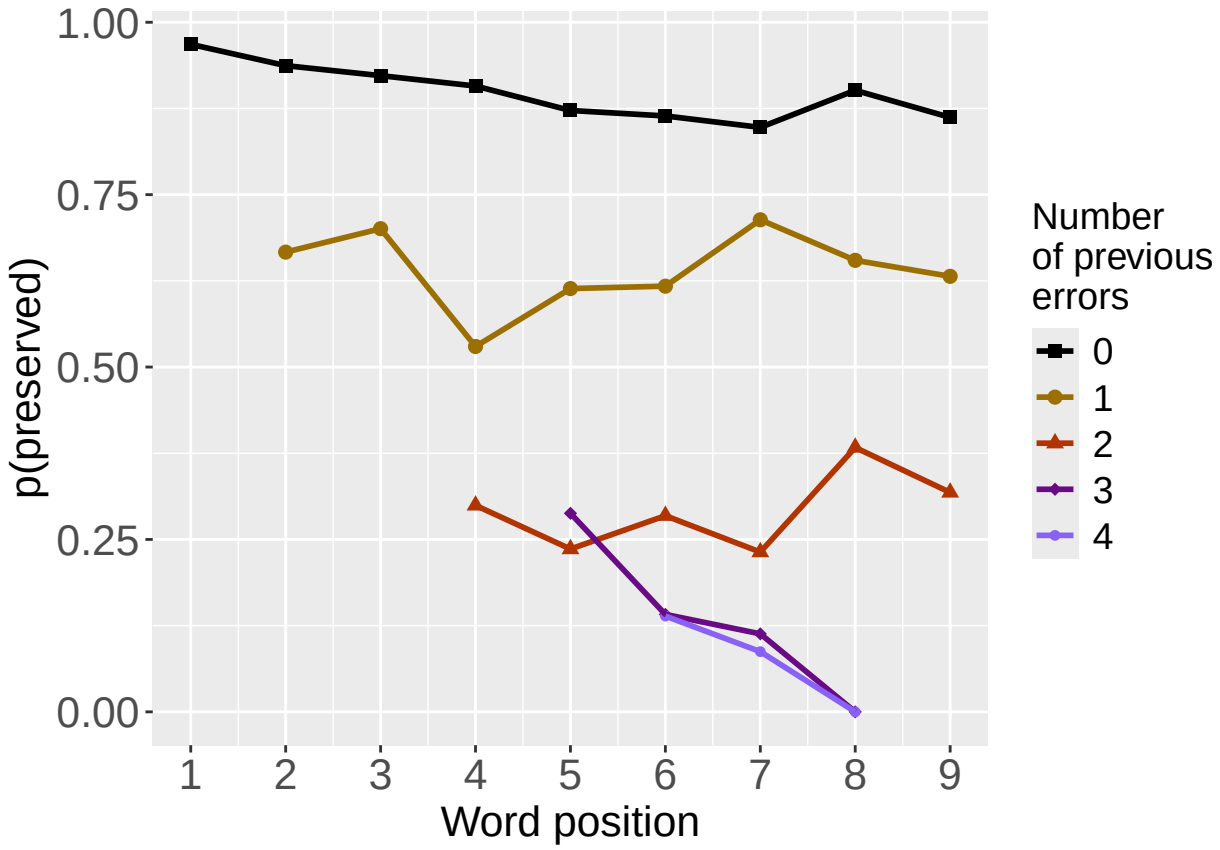
```
print(PrevCorPlot)
```




```
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
```

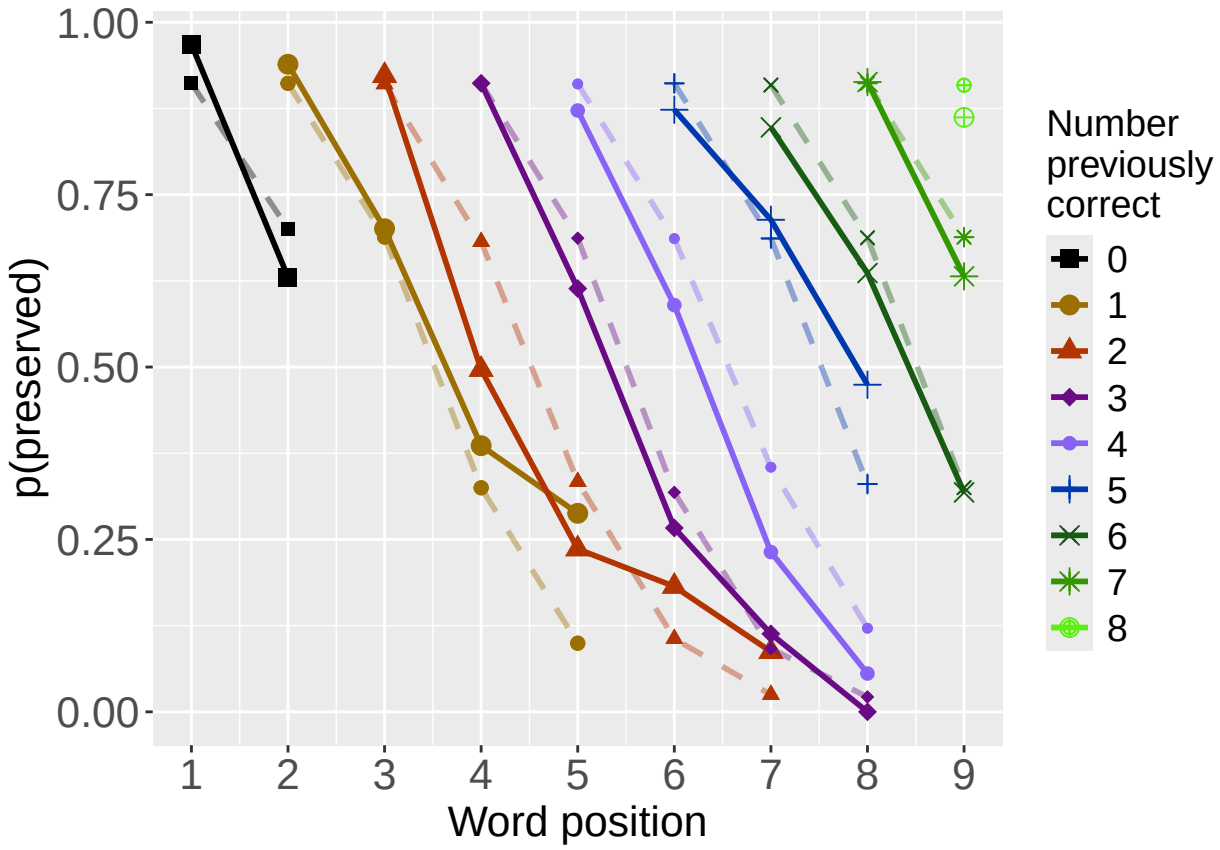
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

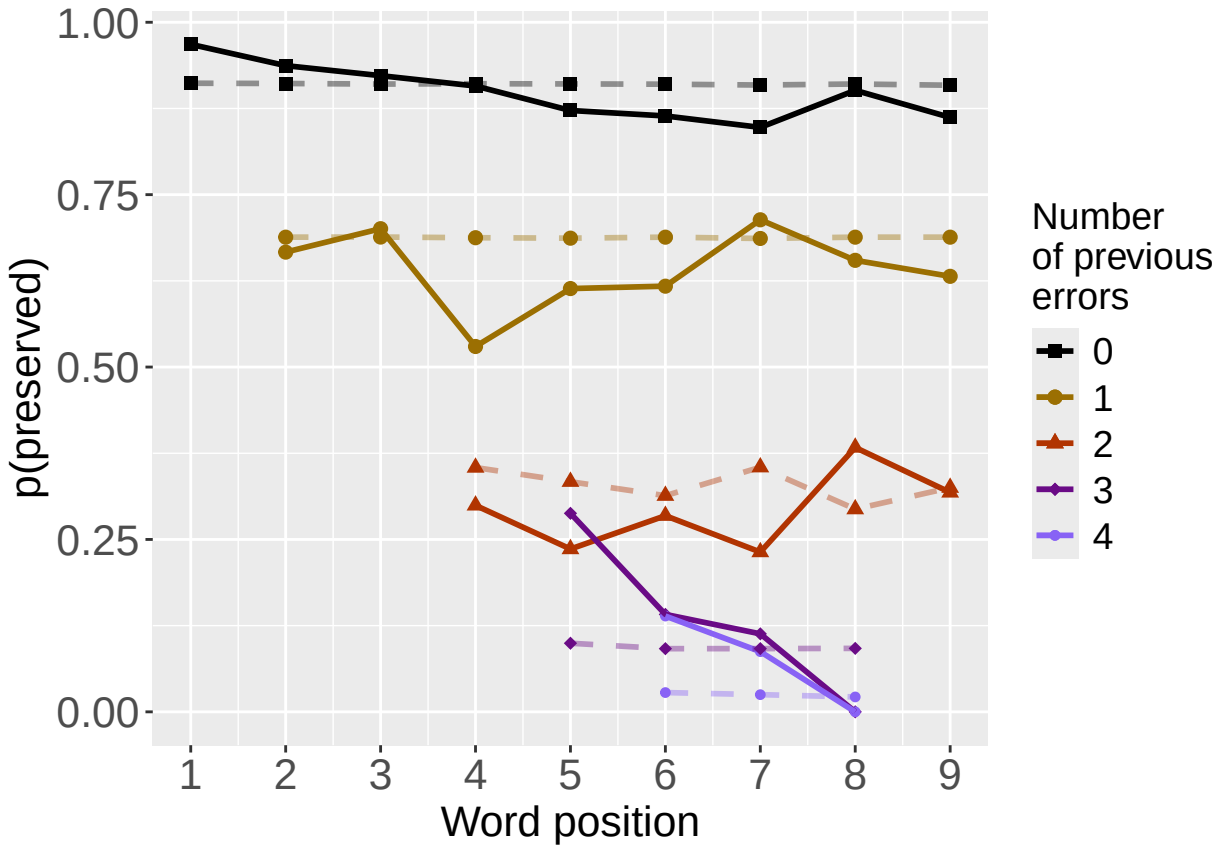
```
print(PrevCorPlot)
```




```
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
##     3.94673     -1.34432      0.05909     -0.70661
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 2721  AIC: 2975
## log likelihood:  -1360.259
## Nagelkerke R2:  0.3766391
## % pres/err predicted correctly:  -824.0081
## % of predictable range [ (model-null)/(1-null) ]:  0.3214671
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.333       -1.541
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 2791  AIC: 3040
## log likelihood:  -1395.522
## Nagelkerke R2:  0.3547444
## % pres/err predicted correctly:  -834.6394
## % of predictable range [ (model-null)/(1-null) ]:  0.3127234
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.06619      0.03604     -0.75670
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 3339  AIC: 3697
## log likelihood:  -1669.441
## Nagelkerke R2:  0.172061
## % pres/err predicted correctly:  -1076.712
## % of predictable range [ (model-null)/(1-null) ]:  0.1136297
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	2974.523	0.00000	1	1	0.3766391	3.946730	-1.344325	0.0590873	-0.7066069

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	3039.690	65.16665	0	0	0.3547444	2.333274	-1.540820	NA	NA
preserved ~ I(posˆ2) + pos	3697.157	722.63417	0	0	0.1720610	4.066192	NA	0.0360449	-0.7567002

```
########
# 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
##      3.3205      -1.4781      -0.1287
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 2773  AIC: 3023
## log likelihood:  -1386.638
## Nagelkerke R2:  0.3602938
## % pres/err predicted correctly:  -832.7657
## % of predictable range [ (model-null)/(1-null) ]:  0.3142644
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.333       -1.541
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 2791  AIC: 3040
## log likelihood:  -1395.522
## Nagelkerke R2:  0.3547444
## % pres/err predicted correctly:  -834.6394
## % of predictable range [ (model-null)/(1-null) ]:  0.3127234
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.0869      -0.3162
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3644  AIC: 4019
## log likelihood:  -1822.099
## Nagelkerke R2:  0.05984294
## % pres/err predicted correctly:  -1169.459
## % of predictable range [ (model-null)/(1-null) ]:  0.03734884
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	3023.336	0.00000	1.0000000	0.9997189	0.3602938	3.320521	-1.478082	-0.1286546
preserved ~ CumErr	3039.690	16.35327	0.0002811	0.0002811	0.3547444	2.333274	-1.540820	NA
preserved ~ stimlen	4018.936	995.60016	0.0000000	0.0000000	0.0598429	4.086930	NA	-0.3161878

```
########
# 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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      2.7550      -1.5057      -0.1554
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
```

```
## Null Deviance:       3799
## Residual Deviance: 2752  AIC: 3007
## log likelihood:  -1376.035
## Nagelkerke R2:  0.3668874
## % pres/err predicted correctly:  -830.4595
## % of predictable range [ (model-null)/(1-null) ]:  0.3161611
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.333       -1.541
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 2791  AIC: 3040
## log likelihood:  -1395.522
## Nagelkerke R2:  0.3547444
## % pres/err predicted correctly:  -834.6394
## % of predictable range [ (model-null)/(1-null) ]:  0.3127234
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##       1.993       -0.157
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3741  AIC: 4128
## log likelihood:  -1870.331
## Nagelkerke R2:  0.0227243
## % pres/err predicted correctly:  -1199.653
## % of predictable range [ (model-null)/(1-null) ]:  0.01251576
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	3006.927	0.00000	1e+00	0.9999999	0.3668874	2.754963	-1.50569	-0.1553836
preserved ~ CumErr	3039.690	32.76233	1e-07	0.0000001	0.3547444	2.333274	-1.54082	NA
preserved ~ CumPres	4127.850	1120.92245	0e+00	0.0000000	0.0227243	1.993030	NA	-0.1570258

```
########
# 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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.9103      -1.3503      -0.1554
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 2752  AIC: 3007
## log likelihood:  -1376.035
## Nagelkerke R2:  0.3668874
## % pres/err predicted correctly:  -830.4595
## % of predictable range [ (model-null)/(1-null) ]:  0.3161611
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.333       -1.541
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 2791  AIC: 3040
## log likelihood:  -1395.522
## Nagelkerke R2:  0.3547444
## % pres/err predicted correctly:  -834.6394
## % of predictable range [ (model-null)/(1-null) ]:  0.3127234
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      3.3474      -0.4024
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 3355  AIC: 3714
## log likelihood:  -1677.31
## Nagelkerke R2:  0.1664666
## % pres/err predicted correctly:  -1077.069
## % of predictable range [ (model-null)/(1-null) ]:  0.1133354
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	3006.927	0.00000	1e+00	0.9999999	0.3668874	2.910347	-1.350307	-0.1553836
preserved ~ CumErr	3039.690	32.76233	1e-07	0.0000001	0.3547444	2.333274	-1.540820	NA
preserved ~ pos	3713.699	706.77148	0e+00	0.0000000	0.1664666	3.347353	NA	-0.4023951

```
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	2974.520	0.00000	1.000000	1.000000	0.3766398	1.946730	-1.344325	0.0590873	-0.7066069	NA	NA
preserved ~ CumErr + pos	3006.927	0.00000	1.000000	0.9999999	0.3668874	2.910347	-1.350307	NA	-0.1553836	NA	NA
preserved ~ CumErr + CumPres	3006.927	0.00000	1.000000	0.9999999	0.3668874	2.754963	-1.505690	NA	NA	NA	-0.1553836
preserved ~ CumErr + stimlen	3023.330	0.00000	1.000000	0.9997189	0.3602938	3.320521	-1.478082	NA	NA	-0.1286546	NA
preserved ~ CumErr	3039.690	65.16665	0.000000	0.0000000	0.3547444	2.333274	-1.540820	NA	NA	NA	NA
preserved ~ CumErr	3039.690	106.35327	0.000281	1.000281	0.3547444	2.333274	-1.540820	NA	NA	NA	NA
preserved ~ CumErr	3039.690	32.76233	0.000000	1.000000	0.3547444	2.333274	-1.540820	NA	NA	NA	NA
preserved ~ CumErr	3039.690	32.76233	0.000000	1.000000	0.3547444	2.333274	-1.540820	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	3697.157	722.63417	0.000000	0.0000000	0.1720614	0.066192	NA	0.0360449	-0.7567002	NA	NA
preserved ~ pos	3713.699	706.77148	0.000000	0.0000000	0.1664666	3.347353	NA	NA	-0.4023951	NA	NA
preserved ~ stimlen	4018.930	995.60016	0.000000	0.0000000	0.0059842	4.086930	NA	NA	NA	-0.3161878	NA
preserved ~ CumPres	4127.850	1120.92245	0.000000	0.0000000	0.0022724	3.993030	NA	NA	NA	NA	-0.1570258

```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:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     4.55167     -1.32657      0.06377     -0.71981     -0.08245      0.06890
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4357 Residual
## Null Deviance:       3799
## Residual Deviance: 2705  AIC: 2963
## log likelihood:  -1352.321
## Nagelkerke R2:  0.3815188
## % pres/err predicted correctly:  -820.9265
## % of predictable range [ (model-null)/(1-null) ]:  0.3240016
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     4.70649     -1.33185      0.06406     -0.72126     -0.10325
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4358 Residual
## Null Deviance:       3799
## Residual Deviance: 2711  AIC: 2966
## log likelihood:  -1355.733
## Nagelkerke R2:  0.3794239
## % pres/err predicted correctly:  -822.4943
## % of predictable range [ (model-null)/(1-null) ]:  0.3227122
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     3.94243     -1.33544      0.05994     -0.70802      0.08294
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4358 Residual
## Null Deviance:       3799
## Residual Deviance: 2710  AIC: 2968
## log likelihood:  -1355.038
## Nagelkerke R2:  0.3798509
## % pres/err predicted correctly:  -821.5249
## % of predictable range [ (model-null)/(1-null) ]:  0.3235094
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     3.94673     -1.34432      0.05909     -0.70661
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 2721  AIC: 2975
## log likelihood:  -1360.259
## Nagelkerke R2:  0.3766391
## % pres/err predicted correctly:  -824.0081
## % of predictable range [ (model-null)/(1-null) ]:  0.3214671
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       1.573
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4362 Residual
## Null Deviance:       3799
## Residual Deviance: 3799  AIC: 4186
## log likelihood:  -1899.341
## Nagelkerke R2:  1.909818e-16
## % pres/err predicted correctly:  -1214.871
## % 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 + log_freq	2963.30	0.0000000	1.0000000	0.7292701	0.3815188	1.551671	-1.326573	0.0637665	-0.7198126	0.0688955	-0.0824480
preserved ~ CumErr + I(pos^2) + pos + stimlen	2965.77	2.4729990	0.2903990	0.2117706	0.3794239	1.706487	-1.331851	0.0640643	-0.7212603	NA	-0.1032474
preserved ~ CumErr + I(pos^2) + pos + log_freq	2968.42	5.1235060	0.0771609	0.0562774	0.3798589	1.942425	-1.335445	0.0599423	-0.7080161	0.0829436	NA
preserved ~ CumErr + I(pos^2) + pos	2974.52	11.2184390	0.0036639	0.0026720	0.3766391	1.946730	-1.344325	0.0590873	-0.7066069	NA	NA
preserved ~ 1	4185.57	1222.2655120	0.0000000	0.0000000	0.0000000	1.573053	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!
## 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
```

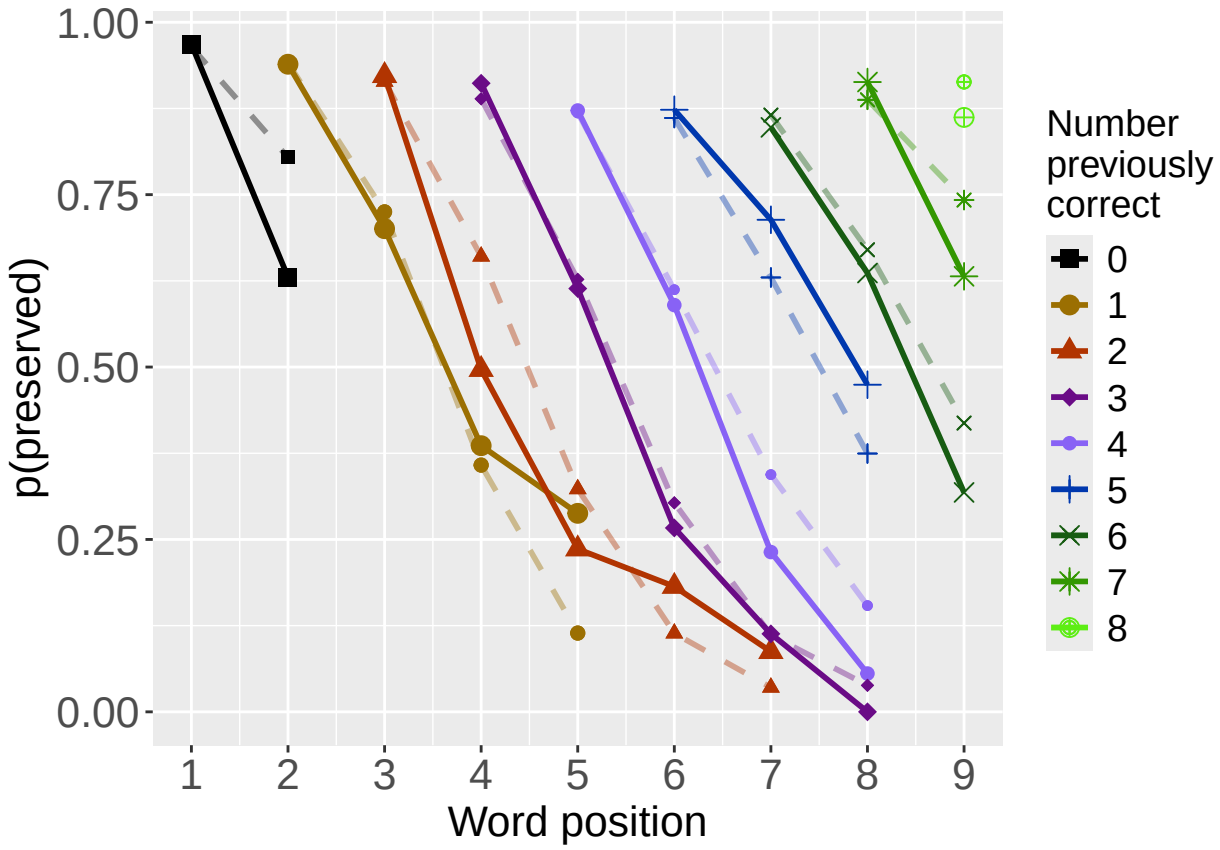
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   3307.0 3563.7
## pos       1   2755.8 3012.5
## I(pos^2)  1   2740.5 2997.1
## log_freq  1   2711.5 2968.1
## stimlen   1   2710.1 2966.7
## <none>        2704.6 2963.3
```

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

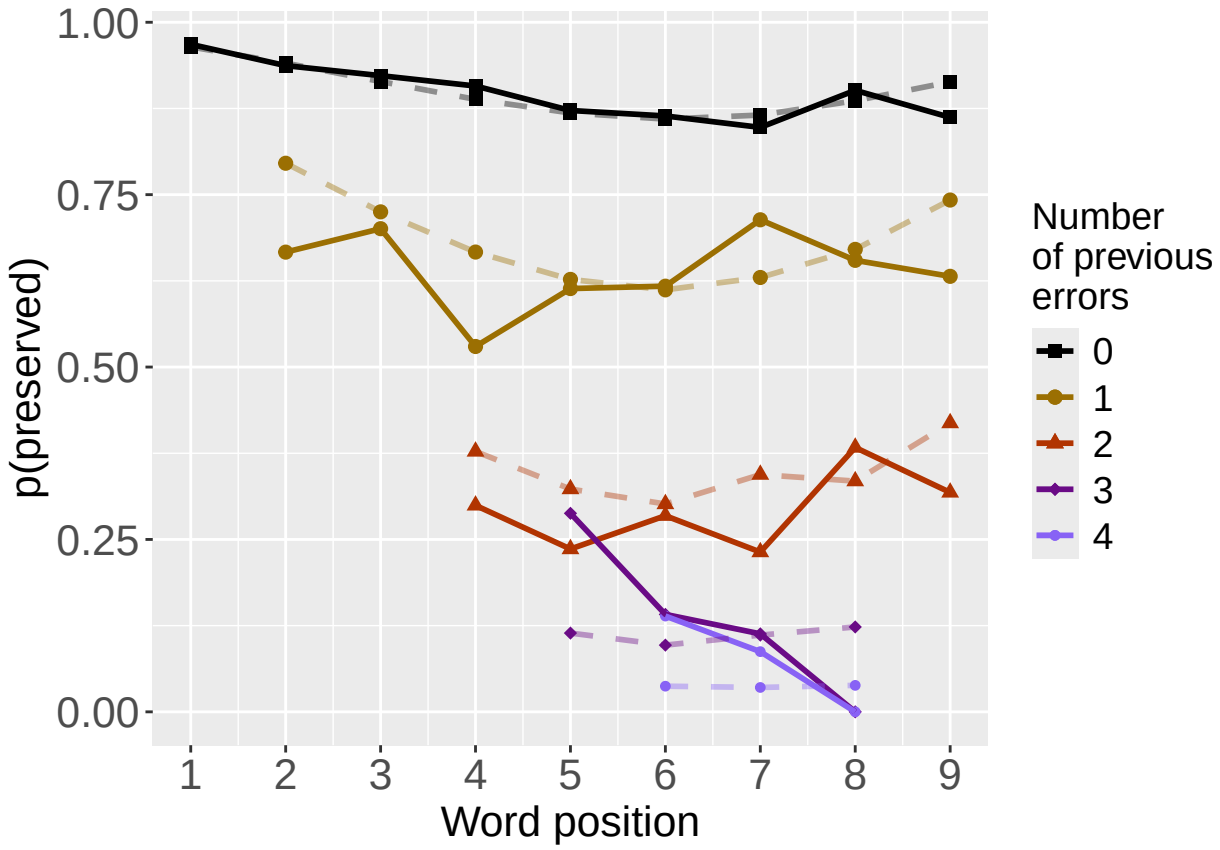
```
print(PrevCorPlot)
```




```
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!
```

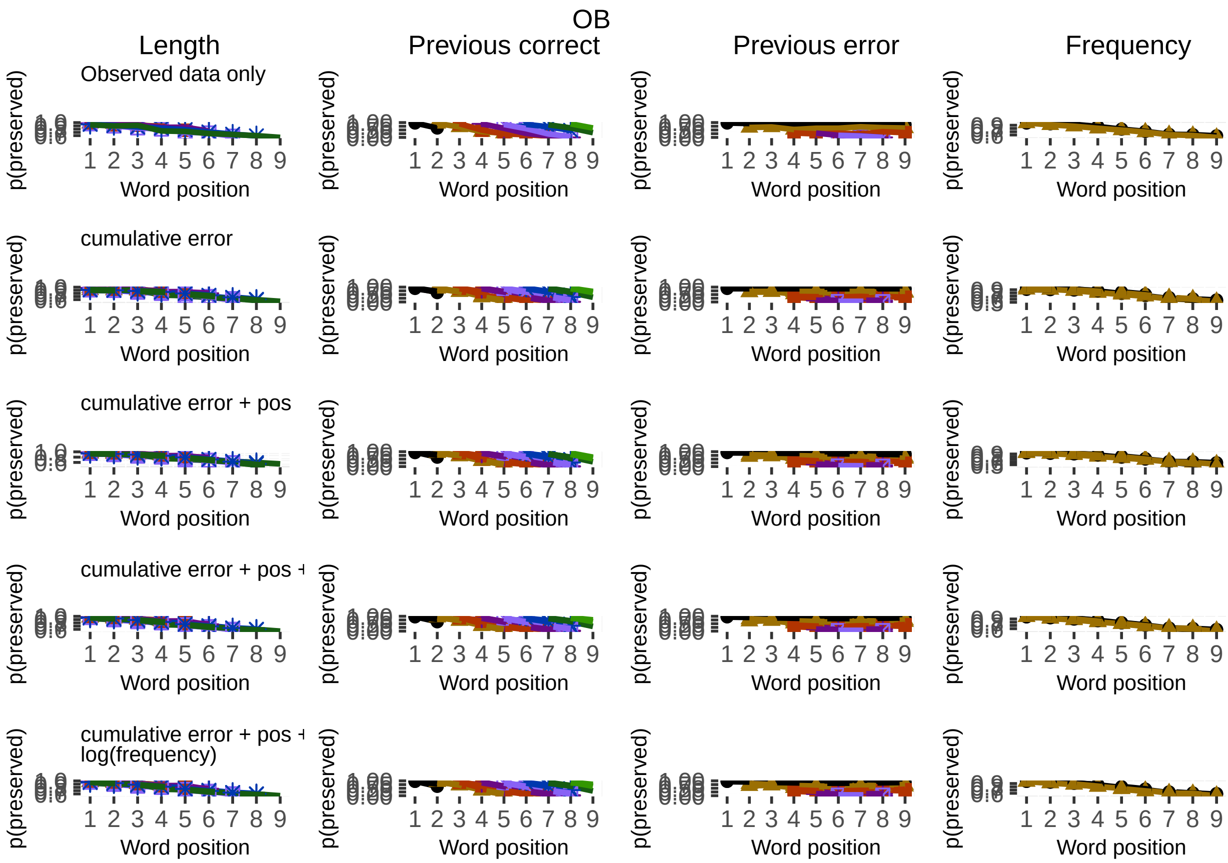
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.333       -1.541
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4361 Residual
## Null Deviance:       3799
## Residual Deviance: 2791  AIC: 3040
## log likelihood:  -1395.522
```

```
## Nagelkerke R2:  0.3547444
## % pres/err predicted correctly:  -834.6394
## % of predictable range [ (model-null)/(1-null) ]:  0.3127234
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.9103      -1.3503      -0.1554
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4360 Residual
## Null Deviance:       3799
## Residual Deviance: 2752  AIC: 3007
## log likelihood:  -1376.035
## Nagelkerke R2:  0.3668874
## % pres/err predicted correctly:  -830.4595
## % of predictable range [ (model-null)/(1-null) ]:  0.3161611
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)
##     3.94673     -1.34432     -0.70661      0.05909
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4359 Residual
## Null Deviance:       3799
## Residual Deviance: 2721  AIC: 2975
## log likelihood:  -1360.259
## Nagelkerke R2:  0.3766391
## % pres/err predicted correctly:  -824.0081
## % of predictable range [ (model-null)/(1-null) ]:  0.3214671
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)     log_freq
##     3.94243     -1.33544     -0.70802      0.05994      0.08294
##
## Degrees of Freedom: 4362 Total (i.e. Null);  4358 Residual
## Null Deviance:       3799
## Residual Deviance: 2710  AIC: 2968
## log likelihood:  -1355.038
## Nagelkerke R2:  0.3798509
## % pres/err predicted correctly:  -821.5249
## % of predictable range [ (model-null)/(1-null) ]:  0.3235094
```

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).
```

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




```r
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$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$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	log_freq
McFadden	0.2021827	0.0339689	0.0430819	0.0116709	0.0036444
SquaredCorrelation	0.1640443	0.0302296	0.0378649	0.0106790	0.0032369
Nagelkerke	0.1640443	0.0302296	0.0378649	0.0106790	0.0032369
Estrella	0.1954524	0.0326900	0.0414851	0.0112171	0.0035085

```
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 + pos + I(pos^2) + log_freq CumErr + pos + I(pos^2) + log_freq 2710.075
## CumErr + pos + I(pos^2)                       CumErr + pos + I(pos^2) 2720.518
## CumErr + pos                                            CumErr + pos 2752.070
## CumErr                                                        CumErr 2791.044
## null                                                            null 3798.682
##                                    deviance_explained percent_explained
## CumErr + pos + I(pos^2) + log_freq          1088.606          28.65748
## CumErr + pos + I(pos^2)                     1078.164          28.38258
## CumErr + pos                                1046.612          27.55197
## CumErr                                      1007.638          26.52600
## null                                           0.000           0.00000
##                                    percent_of_explained_deviance increment_in_explained
## CumErr + pos + I(pos^2) + log_freq                     100.00000              0.9592436
## CumErr + pos + I(pos^2)                                 99.04076              2.8984259
## CumErr + pos                                            96.14233              3.5801120
## CumErr                                                  92.56222             92.5622185
## 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 + pos + I(pos^2) + log_freq	2710.075	1088.606
CumErr + pos + I(pos^2)	2720.518	1078.164
CumErr + pos	2752.070	1046.612
CumErr	2791.044	1007.638
null	3798.682	0.000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + I(pos^2) + log_freq	28.65748	100.00000	0.9592436
CumErr + pos + I(pos^2)	28.38258	99.04076	2.8984259
CumErr + pos	27.55197	96.14233	3.5801120
CumErr	26.52600	92.56222	92.5622185
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   0.66669871
## I(pos^2) 0.12285734
## pos      0.15388800
## stimlen  0.04340093
## log_freq 0.01315502
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.9830898	2791.044
preserved ~ CumErr+pos	0.9852406	2752.070
preserved ~ CumErr+pos+I(pos^2)	0.9914515	2720.518
preserved ~ CumErr+pos+I(pos^2)+log_freq	0.9917867	2710.075

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

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

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

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



```
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       0.9830898       2791.044 0.000000000
## 2                  preserved ~ CumErr+pos       0.9852406       2752.070 0.002150820
## 3           preserved ~ CumErr+pos+I(pos^2)       0.9914515       2720.518 0.008361695
## 4 preserved ~ CumErr+pos+I(pos^2)+log_freq       0.9917867       2710.075 0.008696918
##   diff_CumErr+pos diff_CumErr+pos+I(pos^2) diff_CumErr+pos+I(pos^2)+log_freq
## 1    -0.002150820            -0.0083616955                     -0.0086969182
## 2     0.000000000            -0.0062108757                     -0.0065460985
## 3     0.006210876             0.0000000000                     -0.0003352228
## 4     0.006546098             0.0003352228                      0.0000000000
```

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

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

model	diff_CumErr	diff_CumErr+pos	diff_CumErr+pos+I(pos^2)
preserved ~ CumErr	0.0000000	-0.0021508	-0.0083617
preserved ~ CumErr+pos	0.0021508	0.0000000	-0.0062109
preserved ~ CumErr+pos+I(pos^2)	0.0083617	0.0062109	0.0000000
preserved ~ CumErr+pos+I(pos^2)+log_freq	0.0086969	0.0065461	0.0003352

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