RM - 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	556	36	133	NA	NA	725
2	68	NA	443	101	113	725
3	319	NA	176	214	16	725
4	309	NA	244	71	39	663
5	240	NA	215	72	39	566
6	208	1	142	72	23	446
7	181	NA	103	29	19	332
8	92	NA	55	26	4	177
9	76	NA	2	NA	7	85

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7668966	0.0496552	0.1834483	NA	NA	725
2	0.0937931	NA	0.6110345	0.1393103	0.1558621	725
3	0.4400000	NA	0.2427586	0.2951724	0.0220690	725
4	0.4660633	NA	0.3680241	0.1070890	0.0588235	663
5	0.4240283	NA	0.3798587	0.1272085	0.0689046	566
6	0.4663677	0.0022422	0.3183857	0.1614350	0.0515695	446

pos_factor	O	P	V	1	S	total
7	0.5451807	NA	0.3102410	0.0873494	0.0572289	332
8	0.5197740	NA	0.3107345	0.1468927	0.0225989	177
9	0.8941176	NA	0.0235294	NA	0.0823529	85

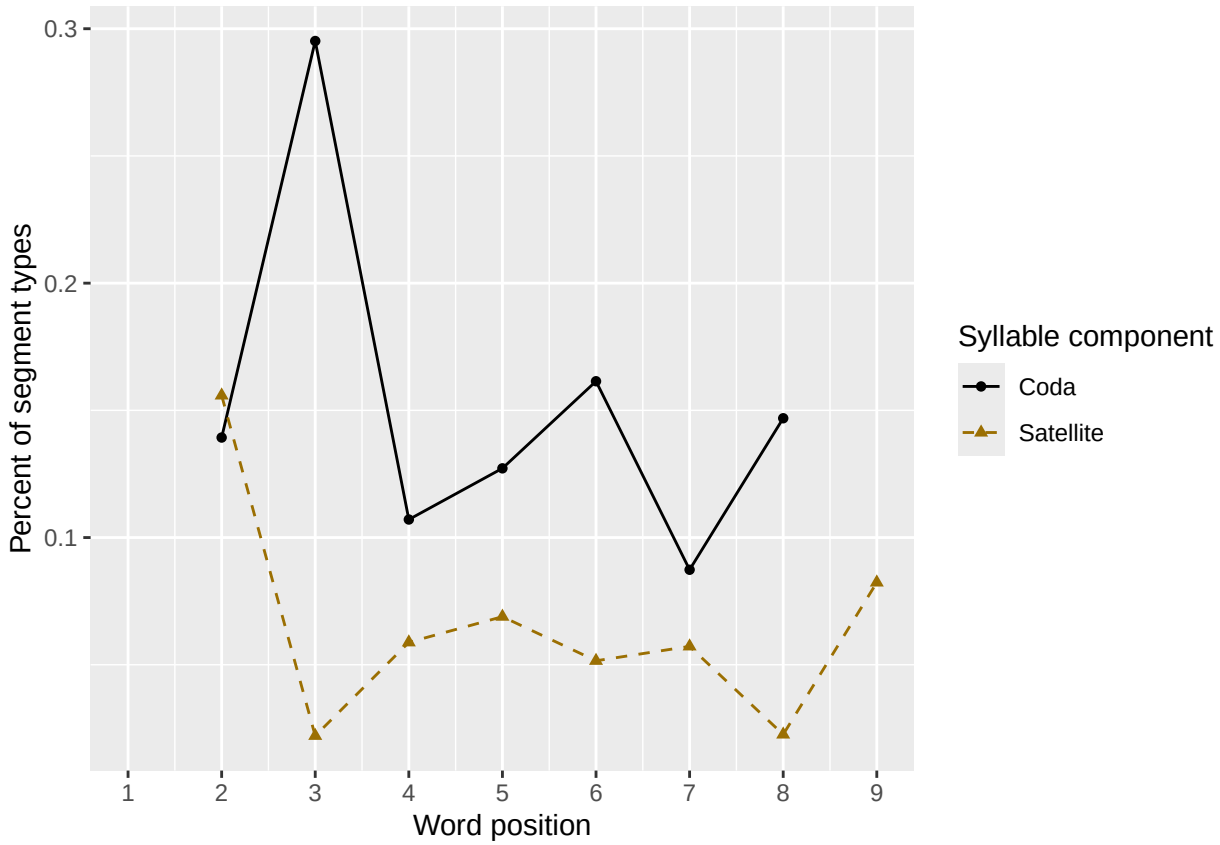
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.952 0.984 0.935 NA     NA     NA     NA     NA     NA
## 2       5 0.979 0.974 0.985  0.948 NA     NA     NA     NA     NA
## 3       6 0.967 0.958 0.925  0.938  0.908 NA     NA     NA     NA
## 4       7 0.982 0.996 0.939  0.925  0.917  0.851 NA     NA     NA
## 5       8 0.974 0.981 0.970  0.932  0.920  0.928  0.909 NA     NA
## 6       9 0.951 0.964 0.911  0.864  0.808  0.855  0.815  0.788 NA
## 7      10 0.988 0.965 0.975  0.910  0.904  0.833  0.835  0.857  0.816
```

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

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

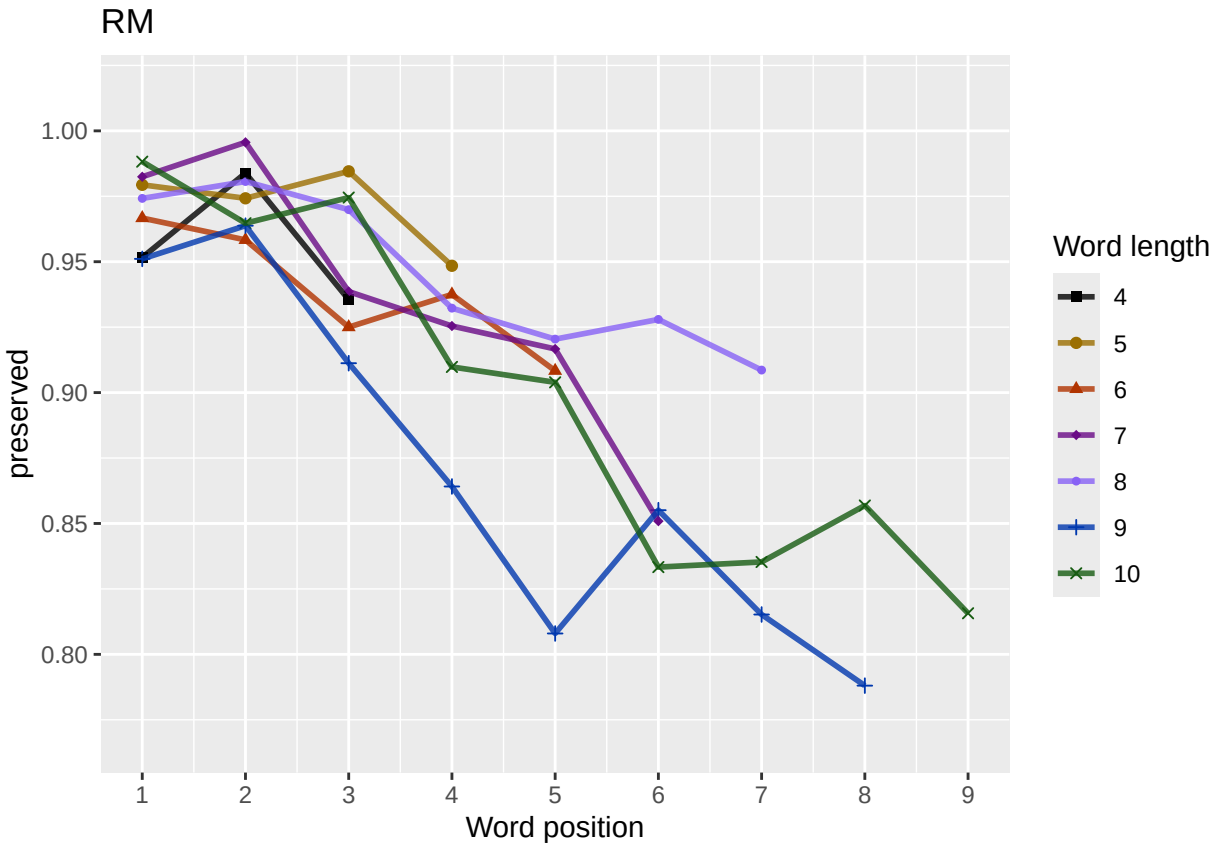
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1       4    62    62    62    NA    NA    NA    NA    NA    NA
## 2       5    97    97    97    97    NA    NA    NA    NA    NA
## 3       6   120   120   120   120   120    NA    NA    NA    NA
## 4       7   114   114   114   114   114   114    NA    NA    NA
## 5       8   155   155   155   155   155   155   155    NA    NA
## 6       9    92    92    92    92    92    92    92    92    NA
## 7      10    85    85    85    85    85    85    85    85    85
```

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

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

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



Length and position

```
# length and position

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

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

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

```
## **************************
## model index:  7
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.87168     -0.07884      0.02940     -0.55294
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2173  AIC: 2334
## log likelihood:  -1086.575
## Nagelkerke R2:  0.07390166
## % pres/err predicted correctly:  -593.243
## % of predictable range [ (model-null)/(1-null) ]:  0.03269973
## **************************
## 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)
##         3.232425          0.142975         -0.020771          0.114550          0.006994
##      stimlen:pos
##        -0.090284
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4438 Residual
## Null Deviance:       2308
## Residual Deviance: 2171  AIC: 2335
## log likelihood:  -1085.427
## Nagelkerke R2:  0.07513828
## % pres/err predicted correctly:  -592.7674
## % of predictable range [ (model-null)/(1-null) ]:  0.03347385
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.30094      0.02557     -0.54392
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 2176  AIC: 2336
## log likelihood:  -1088.245
## Nagelkerke R2:  0.0721011
## % pres/err predicted correctly:  -593.9704
## % of predictable range [ (model-null)/(1-null) ]:  0.03151565
## **************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.17324     -0.05977     -0.27004
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 2179  AIC: 2340
## log likelihood:  -1089.612
## Nagelkerke R2:  0.07062668
## % pres/err predicted correctly:  -593.4623
## % of predictable range [ (model-null)/(1-null) ]:  0.03234274
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.7978      -0.2918
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2181  AIC: 2340
## log likelihood:  -1090.607
## Nagelkerke R2:  0.06955337
## % pres/err predicted correctly:  -593.9712
## % of predictable range [ (model-null)/(1-null) ]:  0.03151442
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     4.53120     -0.10209     -0.36423      0.01079
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2179  AIC: 2341
## log likelihood:  -1089.445
## Nagelkerke R2:  0.07080708
## % pres/err predicted correctly:  -593.553
## % of predictable range [ (model-null)/(1-null) ]:  0.0321951
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##       4.276       -0.223
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2268  AIC: 2428
## log likelihood:  -1134.181
## Nagelkerke R2:  0.02204708
## % pres/err predicted correctly:  -607.5032
## % of predictable range [ (model-null)/(1-null) ]:  0.009487168
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.511
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4443 Residual
## Null Deviance:       2308
## Residual Deviance: 2308  AIC: 2469
## log likelihood:  -1154.117
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -613.3315
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	2333.710	0.000000	1.0000000	0.5391381	0.0739041	7871677	-0.0788433	-0.5529448	NA	0.0294013	NA
preserved ~ stimlen * (I(pos^2) + pos)	2335.275	1.564996	0.4572624	0.2465275	0.0751383	3.2324250	0.1429748	1.1145499	-0.0902839	-0.0207705	0.0069944

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	2336.182	1.467048	0.2912604	0.1570307	0.07072104	1.1300942	NA	-0.5439159	NA	0.0255734	NA
preserved ~ stimlen + pos	2339.766	5.051957	0.0485103	0.0261538	0.07062647	1.7173244	-0.0597683	-0.2700387	NA	NA	NA
preserved ~ pos	2340.436	5.721575	0.0347079	0.0187124	0.06955347	0.7797772	NA	-0.2918152	NA	NA	NA
preserved ~ stimlen * pos	2341.257	7.538662	0.0230675	0.0124366	0.07080741	1.1531201	-0.1020888	-0.3642256	0.0107899	NA	NA
preserved ~ stimlen	2427.526	93.812054	0.0000000	0.0000000	0.00220471	1.1276488	-0.2229972	NA	NA	NA	NA
preserved ~ 1	2468.541	134.828954	0.0000000	0.0000000	0.00000000	0.2510577	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
##     4.87168     -0.07884      0.02940     -0.55294
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2173  AIC: 2334
```

```
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
```

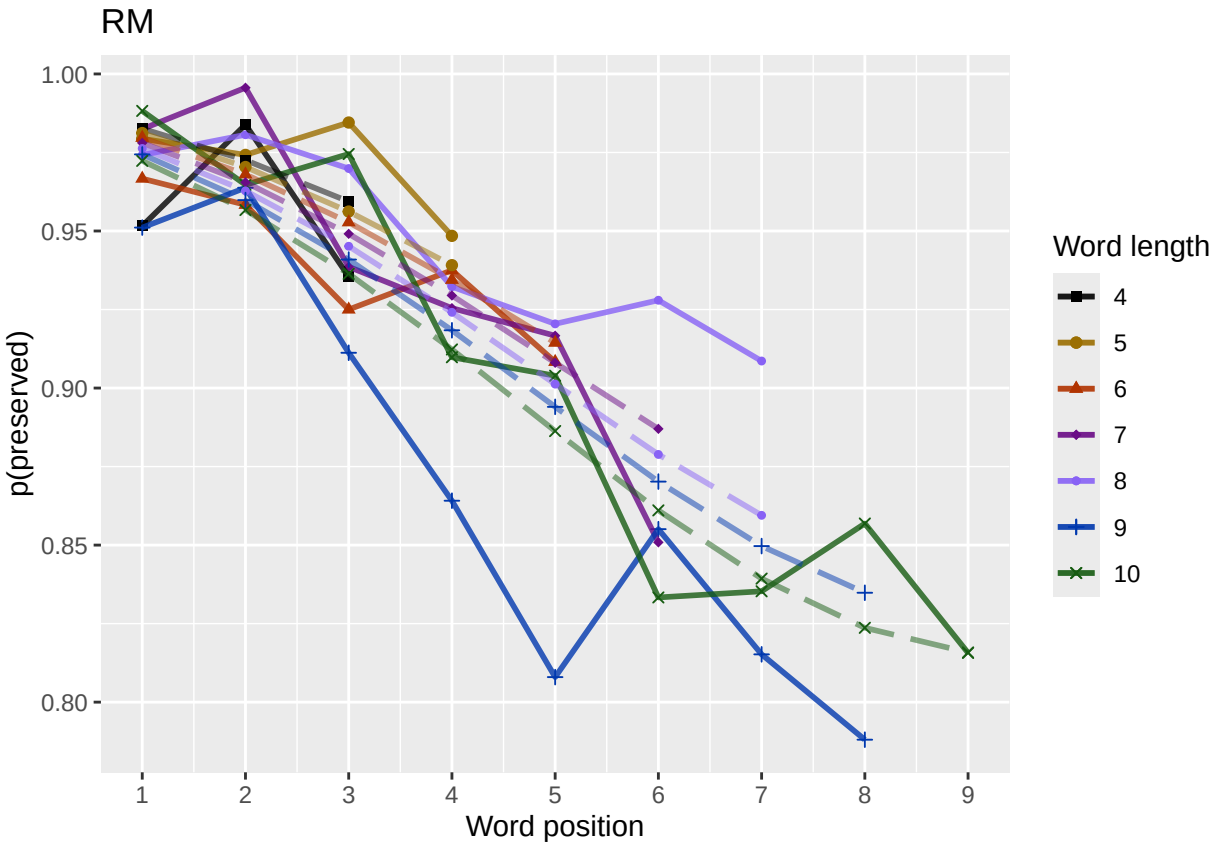
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.983 0.973 0.959 NA     NA     NA     NA     NA     NA
## 2       5 0.981 0.970 0.956  0.939 NA     NA     NA     NA     NA
## 3       6 0.980 0.968 0.953  0.934  0.914 NA     NA     NA     NA
## 4       7 0.978 0.965 0.949  0.929  0.908  0.887 NA     NA     NA
## 5       8 0.976 0.963 0.945  0.924  0.901  0.879  0.859 NA     NA
## 6       9 0.974 0.960 0.941  0.918  0.894  0.870  0.850  0.835 NA
```

```
## 7      10 0.972 0.957 0.936  0.912  0.886  0.861  0.839  0.824  0.816
```

```
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
```

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       65   725
```

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

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

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

```
## [1] "The number of responses with fragments was 65 / 725 = 8.97 percent"
```

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

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

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

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

```
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
```

```
##          1.93939           0.32601          -0.02412           0.84782           0.01383
##      stimlen:pos
##         -0.19996
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4262 Residual
## Null Deviance:       1359
## Residual Deviance: 1329  AIC: 1452
## log likelihood:  -664.6981
## Nagelkerke R2:  0.02528408
## % pres/err predicted correctly:  -312.8908
## % of predictable range [ (model-null)/(1-null) ]:  0.009178532
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.33126     -0.14486      0.06244     -0.56227
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4264 Residual
## Null Deviance:       1359
## Residual Deviance: 1339  AIC: 1458
## log likelihood:  -669.5574
## Nagelkerke R2:  0.01698173
## % pres/err predicted correctly:  -313.9271
## % of predictable range [ (model-null)/(1-null) ]:  0.005907083
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.25466      0.05479     -0.53841
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4265 Residual
## Null Deviance:       1359
## Residual Deviance: 1347  AIC: 1466
## log likelihood:  -673.3676
## Nagelkerke R2:  0.01045853
## % pres/err predicted correctly:  -314.7869
## % of predictable range [ (model-null)/(1-null) ]:  0.003193171
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.232       -0.131
```

```
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4266 Residual
## Null Deviance:       1359
## Residual Deviance: 1351  AIC: 1469
## log likelihood:  -675.7196
## Nagelkerke R2:  0.006425968
## % pres/err predicted correctly:  -315.1226
## % of predictable range [ (model-null)/(1-null) ]:  0.002133429
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##       4.235       -0.117       -0.029
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4265 Residual
## Null Deviance:       1359
## Residual Deviance: 1351  AIC: 1470
## log likelihood:  -675.4445
## Nagelkerke R2:  0.0068979
## % pres/err predicted correctly:  -315.0786
## % of predictable range [ (model-null)/(1-null) ]:  0.002272558
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.91405     -0.07871      0.07401     -0.01190
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4264 Residual
## Null Deviance:       1359
## Residual Deviance: 1351  AIC: 1472
## log likelihood:  -675.3372
## Nagelkerke R2:  0.007081972
## % pres/err predicted correctly:  -315.0616
## % of predictable range [ (model-null)/(1-null) ]:  0.002326143
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     3.45938     -0.06417
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4266 Residual
## Null Deviance:       1359
```

```
## Residual Deviance: 1356  AIC: 1474
## log likelihood:  -677.9663
## Nagelkerke R2:  0.002569838
## % pres/err predicted correctly:  -315.5614
## % of predictable range [ (model-null)/(1-null) ]:  0.0007485468
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       3.211
##
## Degrees of Freedom: 4267 Total (i.e. Null);  4267 Residual
## Null Deviance:       1359
## Residual Deviance: 1359  AIC: 1476
## log likelihood:  -679.4623
## Nagelkerke R2:  8.142836e-16
## % pres/err predicted correctly:  -315.7985
## % 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)	1451.884	0.000000	1.000000	0.9625579	0.0025284	19393910	0.3260121	1.8478177	-0.1999570	-0.0241203	0.0138276
preserved ~ stimlen + I(pos^2) + pos	1458.450	6.566379	0.037508	0.0361040	0.0016985	17331257	-0.1448560	-1.5622679	NA	0.0624375	NA
preserved ~ I(pos^2) + pos	1465.647	13.763452	0.001026	0.0009879	0.0010458	15254662	NA	-0.5384093	NA	0.0547924	NA
preserved ~ stimlen	1468.931	17.047758	0.000198	0.0001910	0.0006426	14232494	-0.1309712	NA	NA	NA	NA
preserved ~ stimlen + pos	1470.226	18.342644	0.000104	0.0001001	0.0006897	14235429	-0.1169747	-1.0289968	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	1472.022	0.138249	0.0000424	0.0000408	0.0070823	0.914049	-0.0787110	0.0740121	-0.0118996	NA	NA
preserved ~ pos	1474.358	2.474062	0.0000132	0.0000127	0.0025693	0.8459378	NA	-0.0641651	NA	NA	NA
preserved ~ 1	1476.087	4.203921	0.0000055	0.0000053	0.0000000	0.211429	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.
```

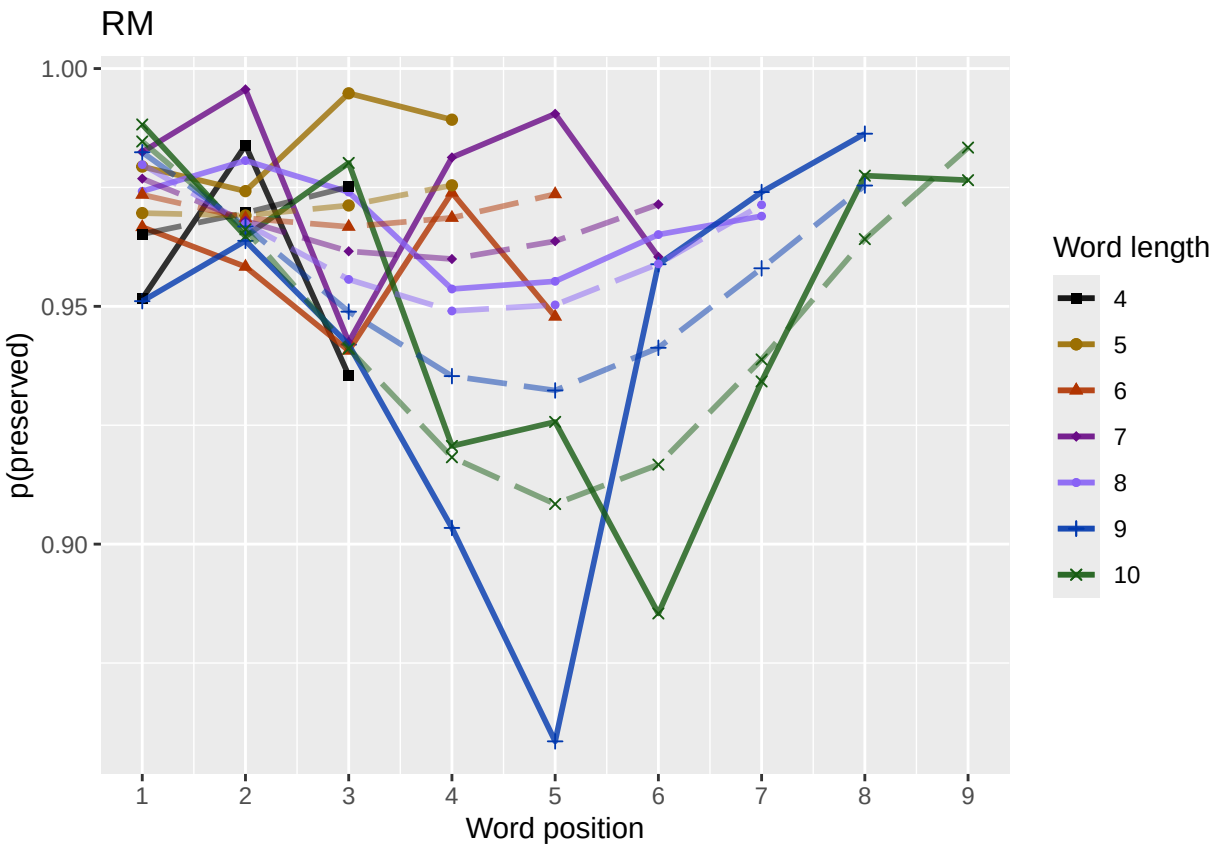
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.965 0.970 0.975 NA     NA     NA     NA     NA     NA
## 2       5 0.970 0.969 0.971  0.975 NA     NA     NA     NA     NA
## 3       6 0.973 0.969 0.967  0.969  0.974 NA     NA     NA     NA
## 4       7 0.977 0.968 0.962  0.960  0.964  0.971 NA     NA     NA
## 5       8 0.980 0.967 0.956  0.949  0.950  0.959  0.971 NA     NA
## 6       9 0.982 0.967 0.949  0.935  0.932  0.941  0.958  0.975 NA
## 7      10 0.985 0.966 0.941  0.918  0.908  0.917  0.939  0.964  0.983
```

```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: 0.77 - 1.02"
```

```
# 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] -0.004405768
```

```
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] -0.01742281
```

```
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:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     4.46546     -0.02039      0.02825     -0.54420      0.19309
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4439 Residual
## Null Deviance:       2308
## Residual Deviance: 2136  AIC: 2297
## log likelihood:  -1068.031
## Nagelkerke R2:  0.09380182
## % pres/err predicted correctly:  -586.2681
## % of predictable range [ (model-null)/(1-null) ]:  0.04405332
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         4.460315         -0.018152          0.122527          0.028569         -0.547529
## stimlen:log_freq
##         0.008652
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4438 Residual
## Null Deviance:       2308
## Residual Deviance: 2136  AIC: 2299
```

```
## log likelihood:  -1067.939
## Nagelkerke R2:  0.09390012
## % pres/err predicted correctly:  -586.0439
## % of predictable range [ (model-null)/(1-null) ]:  0.04441823
## ***************************
## 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
##          4.333442           0.029523          -0.557288           0.206709           0.002393
##      pos:log_freq
##         -0.016042
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4438 Residual
## Null Deviance:       2308
## Residual Deviance: 2136  AIC: 2299
## log likelihood:  -1067.943
## Nagelkerke R2:  0.09389613
## % pres/err predicted correctly:  -586.0315
## % of predictable range [ (model-null)/(1-null) ]:  0.04443842
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          4.510634          -0.024432           0.030798          -0.560854           0.198020
## I(pos^2):log_freq       pos:log_freq
##          0.002448          -0.015377
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4437 Residual
## Null Deviance:       2308
## Residual Deviance: 2136  AIC: 2300
## log likelihood:  -1067.797
## Nagelkerke R2:  0.09405186
## % pres/err predicted correctly:  -585.8747
## % of predictable range [ (model-null)/(1-null) ]:  0.0446937
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      3.7840      -0.2735       0.1949
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
```

```
## Residual Deviance: 2142  AIC: 2300
## log likelihood:  -1070.792
## Nagelkerke R2:  0.09084972
## % pres/err predicted correctly:  -586.3949
## % of predictable range [ (model-null)/(1-null) ]:  0.04384694
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##         4.506213          -0.023323           0.177580           0.030727          -0.561055
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         0.002785           0.002314          -0.015046
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4436 Residual
## Null Deviance:       2308
## Residual Deviance: 2136  AIC: 2302
## log likelihood:  -1067.79
## Nagelkerke R2:  0.09405956
## % pres/err predicted correctly:  -585.8459
## % of predictable range [ (model-null)/(1-null) ]:  0.04474058
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     3.770907     -0.270511      0.173295      0.004398
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2141  AIC: 2302
## log likelihood:  -1070.743
## Nagelkerke R2:  0.09090207
## % pres/err predicted correctly:  -586.1917
## % of predictable range [ (model-null)/(1-null) ]:  0.04417775
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##    3.792801    -0.001381    -0.273065     0.194644
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2142  AIC: 2302
```

```
## log likelihood:  -1070.791
## Nagelkerke R2:  0.09085025
## % pres/err predicted correctly:  -586.39
## % of predictable range [ (model-null)/(1-null) ]:  0.04385496
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     3.791538     -0.003299     -0.269317      0.172076      0.004538
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4439 Residual
## Null Deviance:       2308
## Residual Deviance: 2141  AIC: 2304
## log likelihood:  -1070.74
## Nagelkerke R2:  0.09090502
## % pres/err predicted correctly:  -586.1732
## % of predictable range [ (model-null)/(1-null) ]:  0.0442079
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##        3.7846505         0.0001888         0.1483277        -0.2731973         0.0056889
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4439 Residual
## Null Deviance:       2308
## Residual Deviance: 2142  AIC: 2304
## log likelihood:  -1070.75
## Nagelkerke R2:  0.09089433
## % pres/err predicted correctly:  -586.2372
## % of predictable range [ (model-null)/(1-null) ]:  0.04410362
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         3.787196         -0.001894          0.150565         -0.270365          0.003374
##     log_freq:pos
##         0.003340
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4438 Residual
## Null Deviance:       2308
## Residual Deviance: 2141  AIC: 2306
## log likelihood:  -1070.729
```

```
## Nagelkerke R2:  0.09091672
## % pres/err predicted correctly:  -586.1396
## % of predictable range [ (model-null)/(1-null) ]:  0.04426258
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.87168     -0.07884      0.02940     -0.55294
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2173  AIC: 2334
## log likelihood:  -1086.575
## Nagelkerke R2:  0.07390166
## % pres/err predicted correctly:  -593.243
## % of predictable range [ (model-null)/(1-null) ]:  0.03269973
## ***************************
## 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)
##         3.232425          0.142975         -0.020771          0.114550          0.006994
##      stimlen:pos
##        -0.090284
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4438 Residual
## Null Deviance:       2308
## Residual Deviance: 2171  AIC: 2335
## log likelihood:  -1085.427
## Nagelkerke R2:  0.07513828
## % pres/err predicted correctly:  -592.7674
## % of predictable range [ (model-null)/(1-null) ]:  0.03347385
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.30094      0.02557     -0.54392
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 2176  AIC: 2336
## log likelihood:  -1088.245
## Nagelkerke R2:  0.0721011
```

```
## % pres/err predicted correctly:  -593.9704
## % of predictable range [ (model-null)/(1-null) ]:  0.03151565
## **************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.17324     -0.05977     -0.27004
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 2179  AIC: 2340
## log likelihood:  -1089.612
## Nagelkerke R2:  0.07062668
## % pres/err predicted correctly:  -593.4623
## % of predictable range [ (model-null)/(1-null) ]:  0.03234274
## **************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.7978      -0.2918
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2181  AIC: 2340
## log likelihood:  -1090.607
## Nagelkerke R2:  0.06955337
## % pres/err predicted correctly:  -593.9712
## % of predictable range [ (model-null)/(1-null) ]:  0.03151442
## **************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     4.53120     -0.10209     -0.36423      0.01079
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2179  AIC: 2341
## log likelihood:  -1089.445
## Nagelkerke R2:  0.07080708
## % pres/err predicted correctly:  -593.553
## % of predictable range [ (model-null)/(1-null) ]:  0.0321951
## **************************
```

```
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      3.9038      -0.1675       0.1896
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 2232  AIC: 2391
## log likelihood:  -1115.846
## Nagelkerke R2:  0.04214971
## % pres/err predicted correctly:  -601.7592
## % of predictable range [ (model-null)/(1-null) ]:  0.01883718
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         3.897512         -0.166402          0.155369          0.004191
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 2232  AIC: 2393
## log likelihood:  -1115.823
## Nagelkerke R2:  0.04217467
## % pres/err predicted correctly:  -601.6742
## % of predictable range [ (model-null)/(1-null) ]:  0.01897554
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.276       -0.223
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2268  AIC: 2428
## log likelihood:  -1134.181
## Nagelkerke R2:  0.02204708
## % pres/err predicted correctly:  -607.5032
## % of predictable range [ (model-null)/(1-null) ]:  0.009487168
## **************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.511
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4443 Residual
## Null Deviance:       2308
## Residual Deviance: 2308  AIC: 2469
## log likelihood:  -1154.117
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -613.3315
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

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

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

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

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

Model	AIC DeltaAICAICexpAICwtNagR2(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	I(pos^2):log_freq	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos + log_freq	2296.8570000000040042991280465461	- 0.0203929	0.1930NA14	- 0.5442016	NA	NA	0.0282463 NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	2298.6277005412701366852339461315	- 0.0181521	0.122652708652 0	- 0.5475286	NA	NA	0.0285694 NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	2298.6781590470334883070934933442 NA		0.2067NA86	- 0.557298460424	-	NA	0.0295230239231 NA		NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	2300.3620710320101787133940510634	- 0.0244315	0.1980NA15	- 0.560854553770	-	NA	0.030798024476 NA		NA	NA

Model	AIC	DeltaAIC	expAIC	w	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	[illegible]	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ pos + log_freq	2300.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1948[illegible]	NA	-0.2735256	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	2302.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0233232	0.177[illegible]	[illegible]	-0.5610555	NA	-0.0150464	0.0307269	NA	0.0023141	NA	NA
preserved ~ pos * log_freq	2302.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.173[illegible]	NA	-0.2705112	0.0043976	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	2302.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0013812	0.1946[illegible]	NA	-0.2730652	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	2304.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0032992	0.1720[illegible]	NA	-0.2693168	0.0045[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	2304.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2731973	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	2306.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0018936	0.150[illegible]	[illegible]	-0.2703653	NA	0.0033401	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	2333.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0788433	NA	NA	-0.5529448	NA	NA	0.0294013	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	2335.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	0.1145[illegible]	NA	NA	-0.0207705	NA	NA	-0.0902839	0.0069944
preserved ~ I(pos^2) + pos	2336.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.5439159	NA	NA	0.0255734	NA	NA	NA	NA
preserved ~ stimlen + pos	2339.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0597683	NA	NA	-0.2700387	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	2340.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.2918152	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	2341.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1020888	NA	NA	-0.3642256	NA	NA	NA	NA	NA	0.0107890	NA
preserved ~ stimlen + log_freq	2391.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1675375	0.1895[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	2392.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1664021	0.155[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	2427.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2229972	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:pos	pos	log_freq:pos	log_freq:stimlen	I(pos^2)	log_freq:I(pos^2)	log_freq:stimlen:pos	stimlen:I(pos^2)
preserved ~ 1	2468.571	173.6859	0	0.0000000	0	2.510577	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     4.46546     -0.02039      0.02825     -0.54420      0.19309
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4439 Residual
## Null Deviance:       2308
## Residual Deviance: 2136  AIC: 2297
```

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

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

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

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

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

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

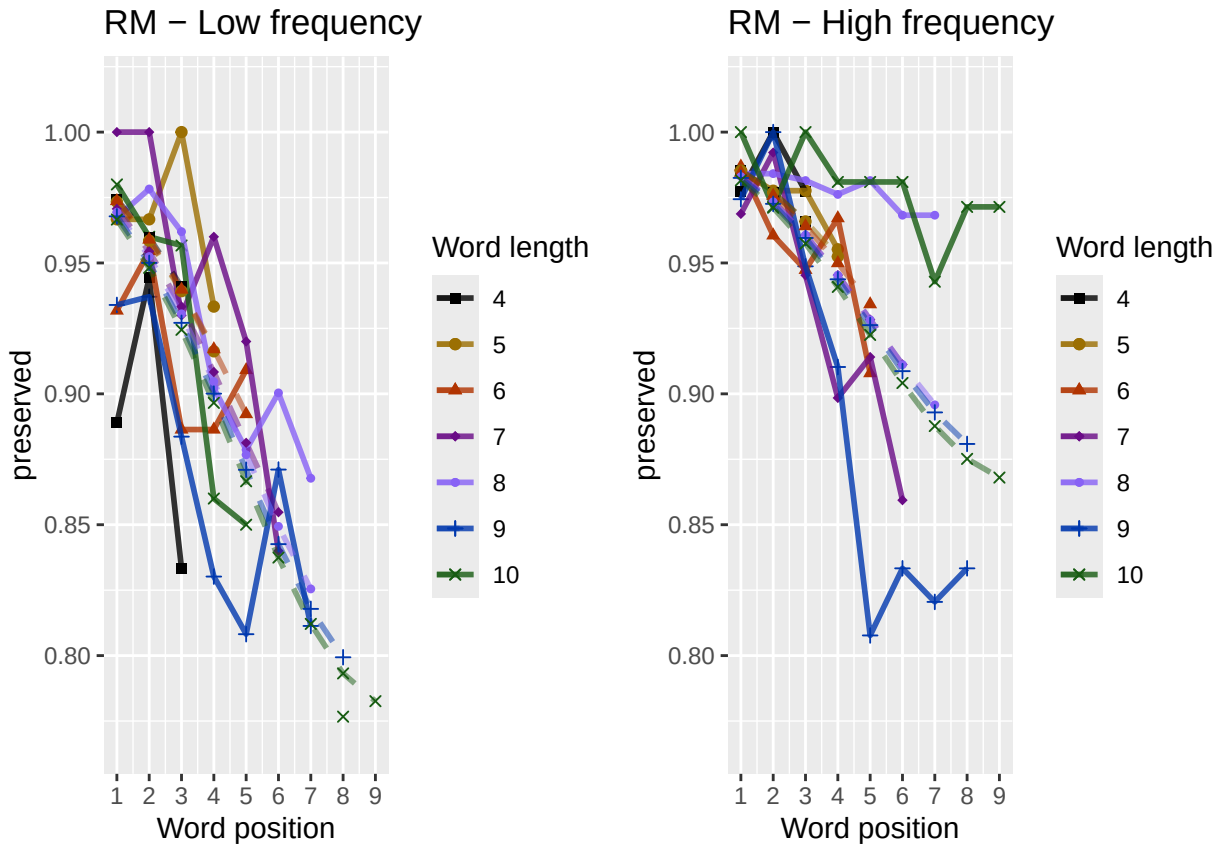
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

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

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




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

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

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

```
## Coefficients:
## (Intercept)       CumErr
##       3.336       -2.163
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1542
## log likelihood:  -733.3668
## Nagelkerke R2:  0.4258151
## % pres/err predicted correctly:  -367.193
## % of predictable range [ (model-null)/(1-null) ]:  0.4006607
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.30094      0.02557     -0.54392
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 2176  AIC: 2336
## log likelihood:  -1088.245
## Nagelkerke R2:  0.0721011
## % pres/err predicted correctly:  -593.9704
## % of predictable range [ (model-null)/(1-null) ]:  0.03151565
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.7978      -0.2918
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2181  AIC: 2340
## log likelihood:  -1090.607
## Nagelkerke R2:  0.06955337
## % pres/err predicted correctly:  -593.9712
## % of predictable range [ (model-null)/(1-null) ]:  0.03151442
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.276       -0.223
```

```
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2268  AIC: 2428
## log likelihood:  -1134.181
## Nagelkerke R2:  0.02204708
## % pres/err predicted correctly:  -607.5032
## % of predictable range [ (model-null)/(1-null) ]:  0.009487168
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.511
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4443 Residual
## Null Deviance:       2308
## Residual Deviance: 2308  AIC: 2469
## log likelihood:  -1154.117
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -613.3315
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.4602       0.0191
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2308  AIC: 2470
## log likelihood:  -1153.881
## Nagelkerke R2:  0.0002619491
## % pres/err predicted correctly:  -613.2503
## % of predictable range [ (model-null)/(1-null) ]:  0.0001320471
## ***************************
```

```
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	1542.340	0.0000	1	1	0.4258153	3.335869	NA	-2.162532	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2336.181	793.8404	0	0	0.0721011	4.300942	NA	NA	0.0255734	-0.5439159	NA
preserved ~ pos	2340.435	798.0949	0	0	0.0695533	4.797772	NA	NA	NA	-0.2918152	NA
preserved ~ stimlen	2427.526	885.1854	0	0	0.0220471	4.276488	NA	NA	NA	NA	-0.2229972
preserved ~ 1	2468.543	926.2023	0	0	0.0000000	2.510577	NA	NA	NA	NA	NA
preserved ~ CumPres	2469.794	927.4538	0	0	0.0002612	2.460203	0.019098	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	0.9216524	585
O	0.9158939	2049
P	0.9459459	37
S	0.9096154	260
V	0.9404054	1513

```
#  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
##       3.341       -2.330
##
## Degrees of Freedom: 4146 Total (i.e. Null);  4145 Residual
## Null Deviance:       2138
## Residual Deviance: 1342  AIC: 1421
## log likelihood:  -670.9129
## Nagelkerke R2:  0.4335399
## % pres/err predicted correctly:  -333.665
## % of predictable range [ (model-null)/(1-null) ]:  0.4106894
```

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.30847      0.02759     -0.55338
##
## Degrees of Freedom: 4146 Total (i.e. Null);  4144 Residual
## Null Deviance:       2138
## Residual Deviance: 2023  AIC: 2178
## log likelihood:  -1011.595
## Nagelkerke R2:  0.06761848
## % pres/err predicted correctly:  -550.3905
## % of predictable range [ (model-null)/(1-null) ]:  0.02905815
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.7673      -0.2819
##
## Degrees of Freedom: 4146 Total (i.e. Null);  4145 Residual
## Null Deviance:       2138
## Residual Deviance: 2028  AIC: 2182
## log likelihood:  -1014.119
## Nagelkerke R2:  0.0646769
## % pres/err predicted correctly:  -550.4809
## % of predictable range [ (model-null)/(1-null) ]:  0.02889912
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.280       -0.222
##
## Degrees of Freedom: 4146 Total (i.e. Null);  4145 Residual
## Null Deviance:       2138
## Residual Deviance: 2101  AIC: 2255
## log likelihood:  -1050.57
## Nagelkerke R2:  0.02179458
## % pres/err predicted correctly:  -561.6171
## % of predictable range [ (model-null)/(1-null) ]:  0.009289415
## ***************************
## model index:  1
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.36778      0.06443
##
## Degrees of Freedom: 4146 Total (i.e. Null);  4145 Residual
## Null Deviance:       2138
## Residual Deviance: 2133  AIC: 2289
## log likelihood:  -1066.639
## Nagelkerke R2:  0.002648258
## % pres/err predicted correctly:  -566.2959
## % of predictable range [ (model-null)/(1-null) ]:  0.001050489
## ****************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.522
##
## Degrees of Freedom: 4146 Total (i.e. Null);  4146 Residual
## Null Deviance:       2138
## Residual Deviance: 2138  AIC: 2292
## log likelihood:  -1068.852
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -566.8925
## % 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	1420.758	0.0000	1	1	0.4335399	3.341164	NA	-2.329693	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2177.953	757.1949	0	0	0.0676185	4.308471	NA	NA	0.0275873	-0.5533818	NA
preserved ~ pos	2182.342	761.5839	0	0	0.0646769	3.767269	NA	NA	NA	-0.2819277	NA
preserved ~ stimlen	2254.560	833.8012	0	0	0.0217946	4.280210	NA	NA	NA	NA	-0.221962
preserved ~ CumPres	2288.797	868.0386	0	0	0.0026483	2.367785	0.0644311	NA	NA	NA	NA
preserved ~ 1	2291.986	871.2280	0	0	0.0000000	2.521699	NA	NA	NA	NA	NA

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

```r
#  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
##       3.256       -2.541
##
## Degrees of Freedom: 3561 Total (i.e. Null);  3560 Residual
## Null Deviance:       1828
## Residual Deviance: 1218  AIC: 1286
## log likelihood:  -609.1599
## Nagelkerke R2:  0.3920596
## % pres/err predicted correctly:  -303.2331
## % of predictable range [ (model-null)/(1-null) ]:  0.3714529
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.20933      0.02662     -0.52642
##
## Degrees of Freedom: 3561 Total (i.e. Null);  3559 Residual
## Null Deviance:       1828
## Residual Deviance: 1736  AIC: 1857
## log likelihood:  -867.8843
## Nagelkerke R2:  0.06392635
## % pres/err predicted correctly:  -469.8701
## % of predictable range [ (model-null)/(1-null) ]:  0.02717985
```

```
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.7052      -0.2668
##
## Degrees of Freedom: 3561 Total (i.e. Null);  3560 Residual
## Null Deviance:       1828
## Residual Deviance: 1740  AIC: 1860
## log likelihood:  -870.0878
## Nagelkerke R2:  0.060922
## % pres/err predicted correctly:  -469.9254
## % of predictable range [ (model-null)/(1-null) ]:  0.02706561
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.124       -0.202
##
## Degrees of Freedom: 3561 Total (i.e. Null);  3560 Residual
## Null Deviance:       1828
## Residual Deviance: 1802  AIC: 1922
## log likelihood:  -900.9745
## Nagelkerke R2:  0.01841538
## % pres/err predicted correctly:  -479.131
## % of predictable range [ (model-null)/(1-null) ]:  0.008046726
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.531
##
## Degrees of Freedom: 3561 Total (i.e. Null);  3561 Residual
## Null Deviance:       1828
## Residual Deviance: 1828  AIC: 1950
## log likelihood:  -914.1912
## Nagelkerke R2:  2.765298e-16
## % pres/err predicted correctly:  -483.0258
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.47638      0.02657
##
## Degrees of Freedom: 3561 Total (i.e. Null);  3560 Residual
## Null Deviance:       1828
## Residual Deviance: 1828  AIC: 1951
## log likelihood:  -913.9392
## Nagelkerke R2:  0.0003524388
## % pres/err predicted correctly:  -482.9575
## % of predictable range [ (model-null)/(1-null) ]:  0.0001411536
## ***************************
```

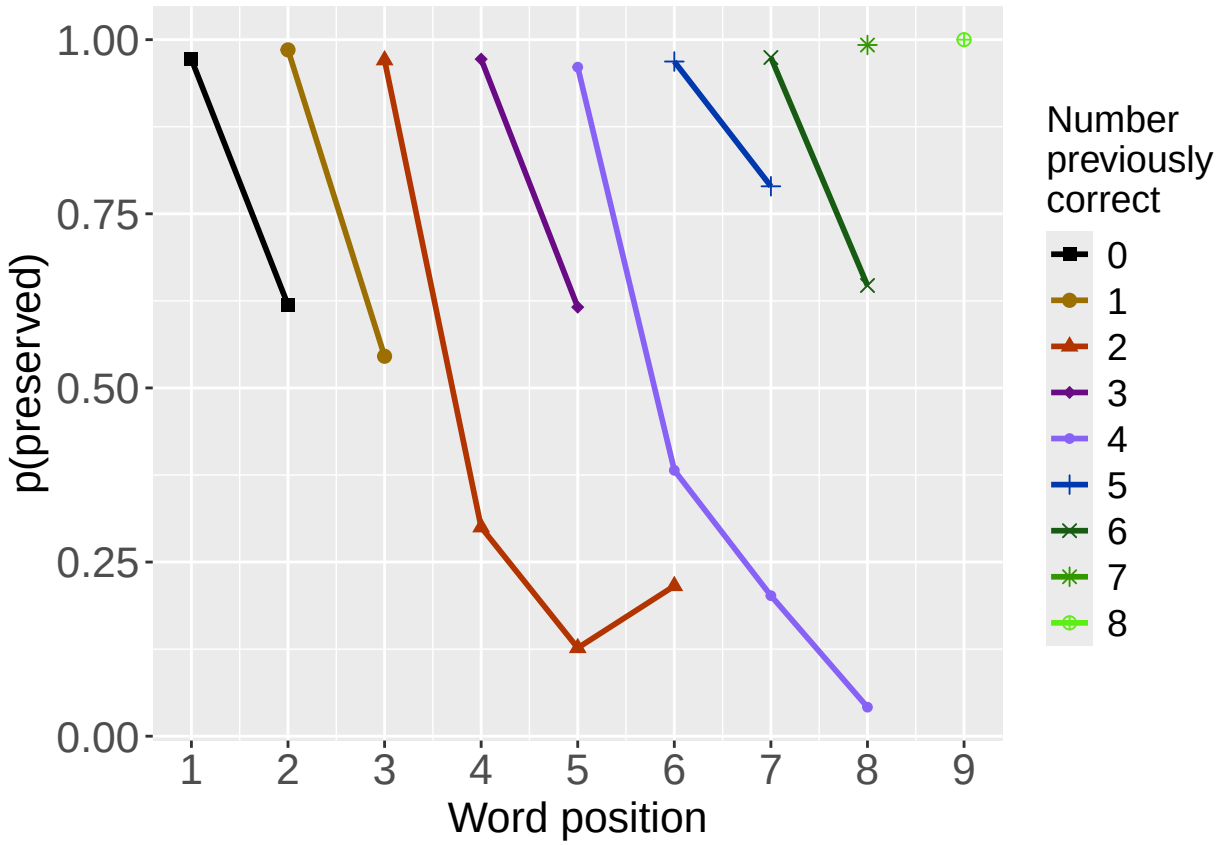
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1286.271	0.0000	1	1	0.392059	6.256300	NA	-2.540936	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1856.649	570.3778	0	0	0.063926	4.209333	NA	NA	0.0266247	-0.5264242	NA
preserved ~ pos	1860.335	574.0632	0	0	0.060922	3.705190	NA	NA	NA	-0.2668386	NA
preserved ~ stimlen	1922.019	635.7475	0	0	0.018415	4.124080	NA	NA	NA	NA	-0.201975
preserved ~ 1	1949.677	663.4061	0	0	0.000000	2.531275	NA	NA	NA	NA	NA
preserved ~ CumPres	1951.103	664.8317	0	0	0.000352	2.476376	0.0265678	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.
```

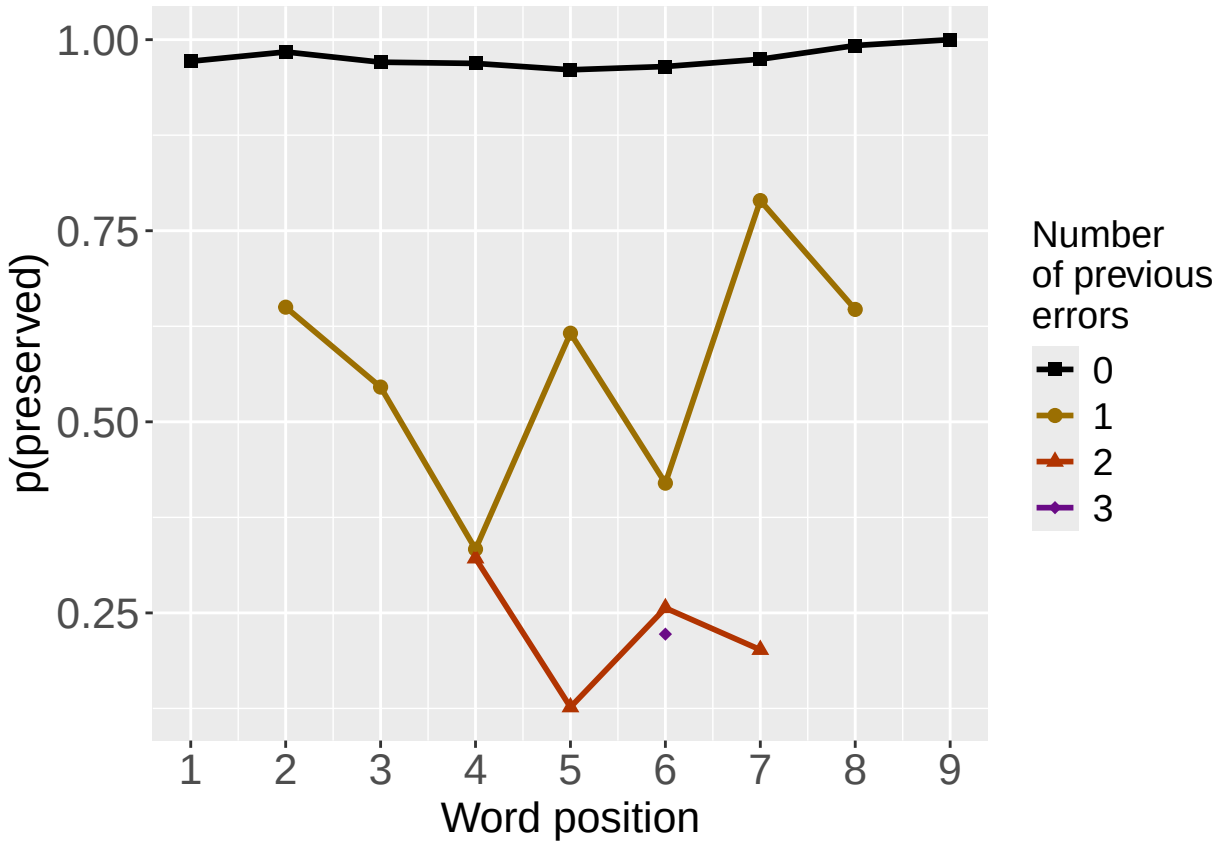
```
print(PrevCorPlot)
```




```
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
```

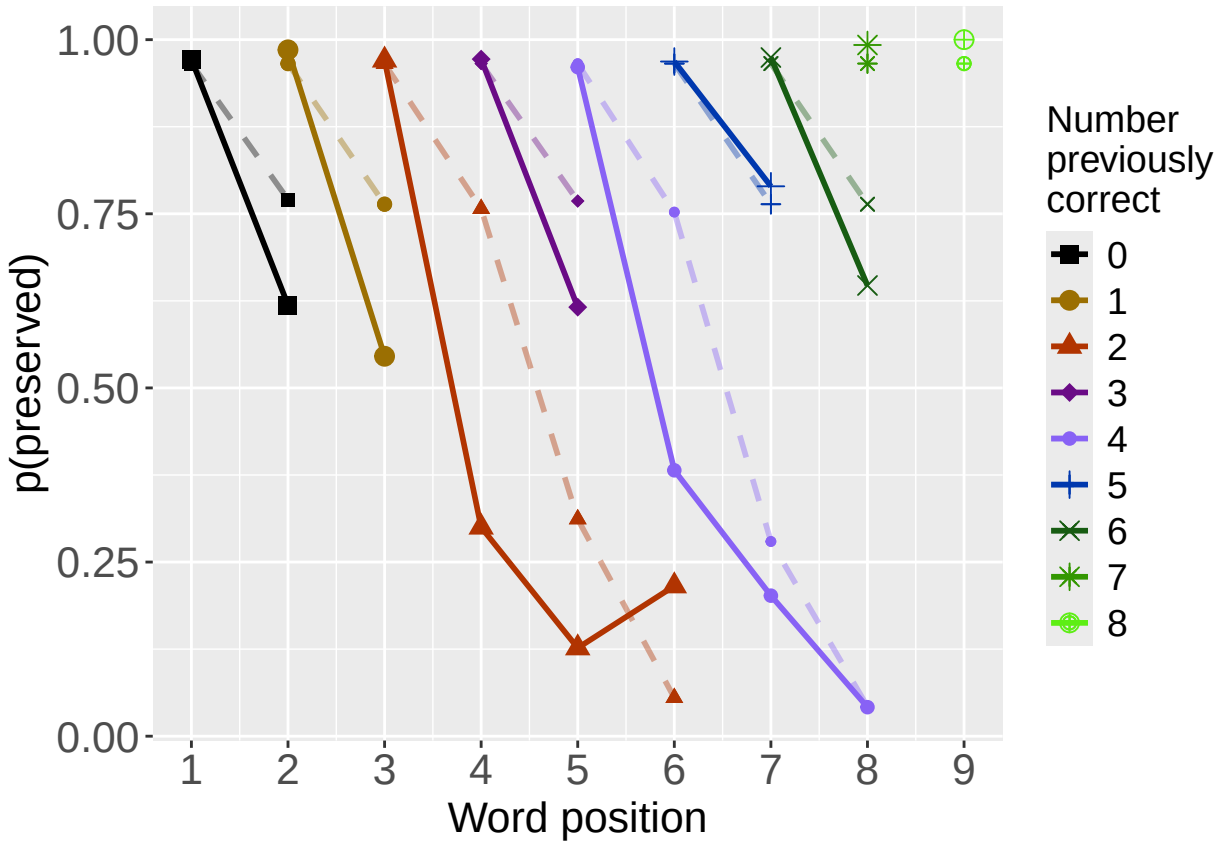
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

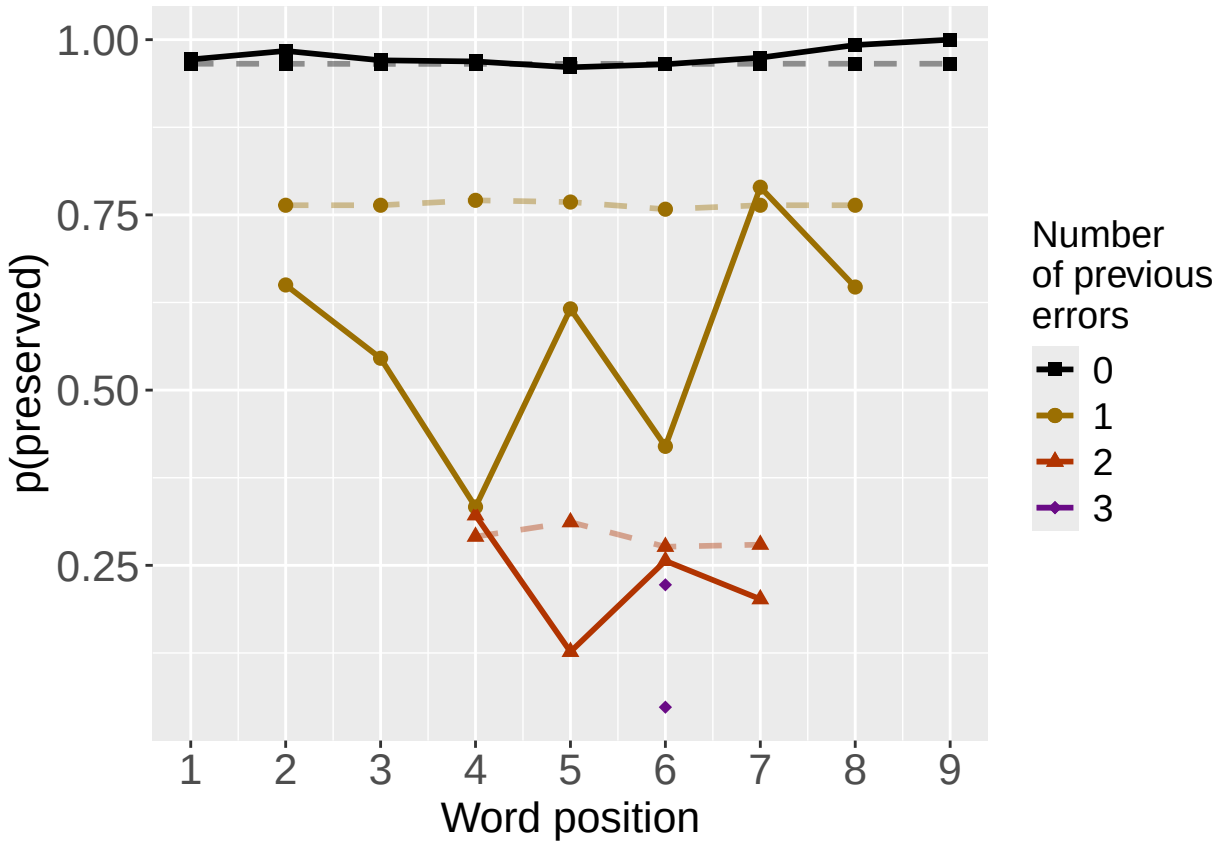
```
print(PrevCorPlot)
```




```
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
##     4.25517     -2.15434      0.05513     -0.50774
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 1454  AIC: 1531
## log likelihood:  -727.2334
## Nagelkerke R2:  0.4314454
## % pres/err predicted correctly:  -364.0445
## % of predictable range [ (model-null)/(1-null) ]:  0.4057858
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.336       -2.163
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1542
## log likelihood:  -733.3668
## Nagelkerke R2:  0.4258151
## % pres/err predicted correctly:  -367.193
## % of predictable range [ (model-null)/(1-null) ]:  0.4006607
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.30094      0.02557     -0.54392
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 2176  AIC: 2336
## log likelihood:  -1088.245
## Nagelkerke R2:  0.0721011
## % pres/err predicted correctly:  -593.9704
## % of predictable range [ (model-null)/(1-null) ]:  0.03151565
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1531.300	0.00000	1.0000000	0.9960104	0.4314454	4.255170	-2.154335	0.0551327	-0.5077400

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1542.340	11.04014	0.0040056	0.0039896	0.4258151	3.335869	-2.162532	NA	NA
preserved ~ I(posˆ2) + pos	2336.181	804.88053	0.0000000	0.0000000	0.0721011	4.300942	NA	0.0255734	-0.5439159

```
########
# 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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.336       -2.163
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1542
## log likelihood:  -733.3668
## Nagelkerke R2:  0.4258151
## % pres/err predicted correctly:  -367.193
## % of predictable range [ (model-null)/(1-null) ]:  0.4006607
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     3.71021     -2.14464     -0.04874
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 1466  AIC: 1543
## log likelihood:  -732.7539
## Nagelkerke R2:  0.4263784
## % pres/err predicted correctly:  -367.3288
## % of predictable range [ (model-null)/(1-null) ]:  0.4004396
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.276       -0.223
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2268  AIC: 2428
## log likelihood:  -1134.181
## Nagelkerke R2:  0.02204708
## % pres/err predicted correctly:  -607.5032
## % of predictable range [ (model-null)/(1-null) ]:  0.009487168
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	1542.340	0.0000000	1.0000000	0.5769487	0.4258151	3.335869	-2.162532	NA
preserved ~ CumErr + stimlen	1542.961	0.6205201	0.7332563	0.4230513	0.4263784	3.710208	-2.144644	-0.0487376
preserved ~ stimlen	2427.526	885.18539570	0.0000000	0.0000000	0.0220471	4.276488	NA	-0.2229972

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.336       -2.163
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
```

```
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1542
## log likelihood:  -733.3668
## Nagelkerke R2:  0.4258151
## % pres/err predicted correctly:  -367.193
## % of predictable range [ (model-null)/(1-null) ]:  0.4006607
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     3.37324     -2.16153     -0.01369
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1544
## log likelihood:  -733.2949
## Nagelkerke R2:  0.4258812
## % pres/err predicted correctly:  -367.2562
## % of predictable range [ (model-null)/(1-null) ]:  0.4005578
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.4602       0.0191
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2308  AIC: 2470
## log likelihood:  -1153.881
## Nagelkerke R2:  0.0002619491
## % pres/err predicted correctly:  -613.2503
## % of predictable range [ (model-null)/(1-null) ]:  0.0001320471
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr	1542.340	0.000000	1.0000000	0.746337	0.4258151	3.335869	-2.162532	NA
preserved ~ CumErr + CumPres	1544.499	2.158342	0.3398772	0.253663	0.4258812	3.373238	-2.161526	-0.0136876
preserved ~ CumPres	2469.794	927.453759	0.0000000	0.000000	0.0002619	2.460203	NA	0.0190980

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

```
BestMEFactor<-BestMEModelFormula
OtherFactor<-"pos"

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

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

```
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.336       -2.163
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1542
## log likelihood:  -733.3668
## Nagelkerke R2:  0.4258151
## % pres/err predicted correctly:  -367.193
## % of predictable range [ (model-null)/(1-null) ]:  0.4006607
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     3.38693     -2.14784     -0.01369
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1544
## log likelihood:  -733.2949
## Nagelkerke R2:  0.4258812
## % pres/err predicted correctly:  -367.2562
## % of predictable range [ (model-null)/(1-null) ]:  0.4005578
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      3.7978      -0.2918
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 2181  AIC: 2340
## log likelihood:  -1090.607
## Nagelkerke R2:  0.06955337
## % pres/err predicted correctly:  -593.9712
## % of predictable range [ (model-null)/(1-null) ]:  0.03151442
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr	1542.340	0.000000	1.0000000	0.746337	0.4258151	3.335869	-2.162532	NA
preserved ~ CumErr + pos	1544.499	2.158342	0.3398772	0.253663	0.4258812	3.386926	-2.147838	-0.0136876
preserved ~ pos	2340.435	798.094917	0.0000000	0.000000	0.0695534	3.797772	NA	-0.2918152

```
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	1531.300	0.0000000	1.0000000	0.9960104	0.4431445	4.255170	-2.154335	0.0551327	-0.5077400	NA	NA
preserved ~ CumErr	1542.340	11.040143	0.0040035	0.0039896	0.4258153	3.335869	-2.162532	NA	NA	NA	NA
preserved ~ CumErr	1542.340	0.0000000	1.0000000	0.5769487	0.4258153	3.335869	-2.162532	NA	NA	NA	NA
preserved ~ CumErr	1542.340	0.0000000	1.0000000	0.7463370	0.4258153	3.335869	-2.162532	NA	NA	NA	NA
preserved ~ CumErr	1542.340	0.0000000	1.0000000	0.7463370	0.4258153	3.335869	-2.162532	NA	NA	NA	NA
preserved ~ CumErr + stimlen	1542.960	0.6205201	0.7332563	0.5473051	0.4263783	4.710208	-2.144644	NA	NA	-0.0487376	NA
preserved ~ CumErr + CumPres	1544.499	2.1583417	0.3398772	0.2536630	0.4258881	2.373238	-2.161526	NA	NA	NA	-0.0136876
preserved ~ CumErr + pos	1544.499	2.1583417	0.3398772	0.2536630	0.4258881	2.386926	-2.147838	NA	-0.0136876	NA	NA
preserved ~ I(pos^2) + pos	2336.188	804.8805338	0.0000000	0.0000000	0.0721014	1.300942	NA	0.0255734	-0.5439159	NA	NA
preserved ~ pos	2340.435	798.0949176	0.0000000	0.0000000	0.0695534	3.797772	NA	NA	-0.2918152	NA	NA
preserved ~ stimlen	2427.528	885.1853957	0.0000000	0.0000000	0.0022047	1.276488	NA	NA	NA	-0.2229972	NA
preserved ~ CumPres	2469.794	927.4537587	0.0000000	0.0000000	0.0000261	2.460203	NA	NA	NA	NA	0.0190980

```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:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     4.26469     -2.12914      0.05731     -0.50959      0.16044
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4439 Residual
## Null Deviance:       2308
## Residual Deviance: 1437  AIC: 1515
## log likelihood:  -718.718
## Nagelkerke R2:  0.4392366
## % pres/err predicted correctly:  -363.2442
## % of predictable range [ (model-null)/(1-null) ]:  0.4070884
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     4.52409     -2.13025      0.05917     -0.51531     -0.03547      0.15371
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4438 Residual
## Null Deviance:       2308
## Residual Deviance: 1437  AIC: 1516
## log likelihood:  -718.478
## Nagelkerke R2:  0.4394558
## % pres/err predicted correctly:  -363.1541
## % of predictable range [ (model-null)/(1-null) ]:  0.4072351
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     4.85159     -2.15420      0.05961     -0.51997     -0.08201
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4439 Residual
## Null Deviance:       2308
## Residual Deviance: 1452  AIC: 1529
## log likelihood:  -725.8499
## Nagelkerke R2:  0.4327133
## % pres/err predicted correctly:  -363.8471
## % of predictable range [ (model-null)/(1-null) ]:  0.406107
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     4.25517     -2.15434      0.05513     -0.50774
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 1454  AIC: 1531
## log likelihood:  -727.2334
## Nagelkerke R2:  0.4314454
## % pres/err predicted correctly:  -364.0445
## % of predictable range [ (model-null)/(1-null) ]:  0.4057858
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       2.511
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4443 Residual
## Null Deviance:       2308
## Residual Deviance: 2308  AIC: 2469
## log likelihood:  -1154.117
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -613.3315
## % 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 + log_freq	1515.415	0.000000	1.0000000	0.6307001	0.4392366	4.264689	-2.129142	0.0573061	-0.5095908	0.1604350	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	1516.490	1.074748	0.5842807	0.3685069	0.4394558	4.524090	-2.130253	0.0591706	-0.5153060	0.1537104	-0.0354705
preserved ~ CumErr + I(pos^2) + pos + stimlen	1529.442	14.027482	0.0008994	0.0005673	0.4327143	3.851586	-2.154201	0.0596131	-0.5199695	NA	-0.0820084
preserved ~ CumErr + I(pos^2) + pos	1531.300	15.885551	0.0003552	0.0002240	0.4314454	4.255170	-2.154335	0.0551327	-0.5077400	NA	NA
preserved ~ 1	2468.543	953.127990	0.0000000	0.0000000	0.0000000	2.510577	NA	NA	NA	NA	NA

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

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

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

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

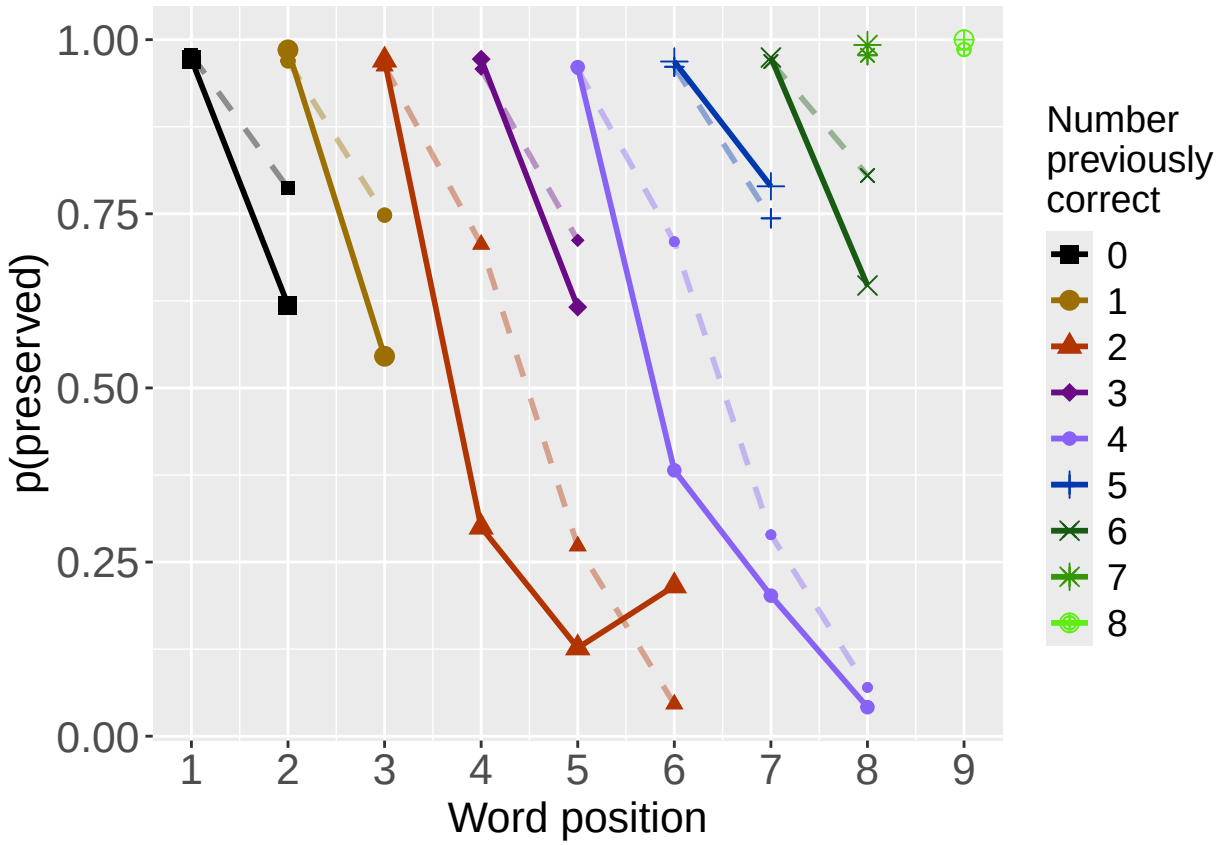
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + log_freq
##          Df Deviance    AIC
## CumErr    1   2136.3 2212.2
## log_freq  1   1454.5 1530.5
## I(pos^2)  1   1450.6 1526.5
## pos       1   1449.5 1525.5
## <none>        1437.4 1515.4
```

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

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

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

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

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

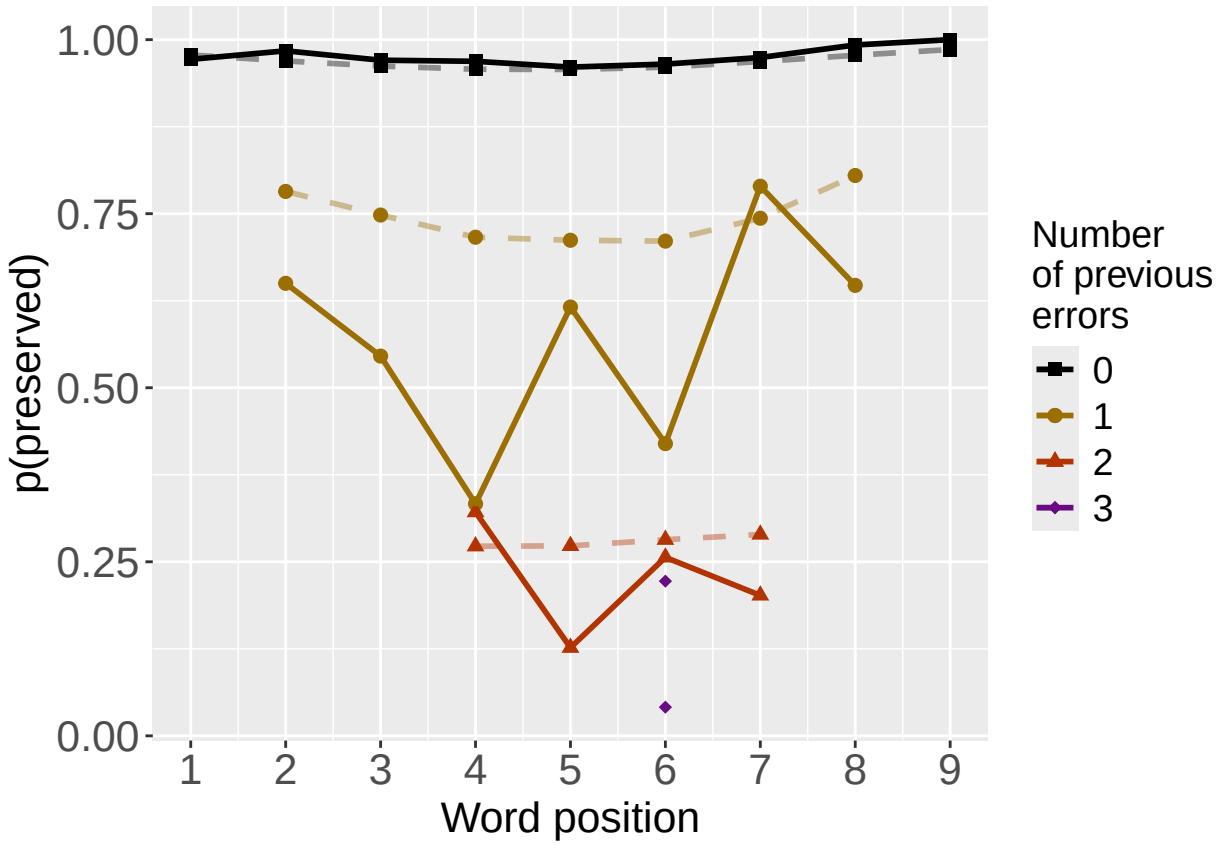
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

  RndModelAIC<-numeric(length=RandomSamples)
  for(rindex in seq(1,RandomSamples)){
  # Shuffle cumulative values
    PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
    PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
    BestModelRnd <- glm(as.formula(BestModelFormulaL3Rnd),
                        family="binomial",data=PosDat)
    RndModelAIC[rindex] <- BestModelRnd$aic
  }
  ModelNames<-c(paste0("***",BestModelFormulaL3),
```

```
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random average"),
                                     AIC=c(mean(RndModelAIC))))
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random SD"),
                                     AIC=c(sd(RndModelAIC))))
  write.csv(BestModelRndDF,paste0(TablesDir,CurPat,"_",CurTask,
                                  "_best_model_with_random_cum_term.csv"),
            row.names = FALSE)
}
```

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

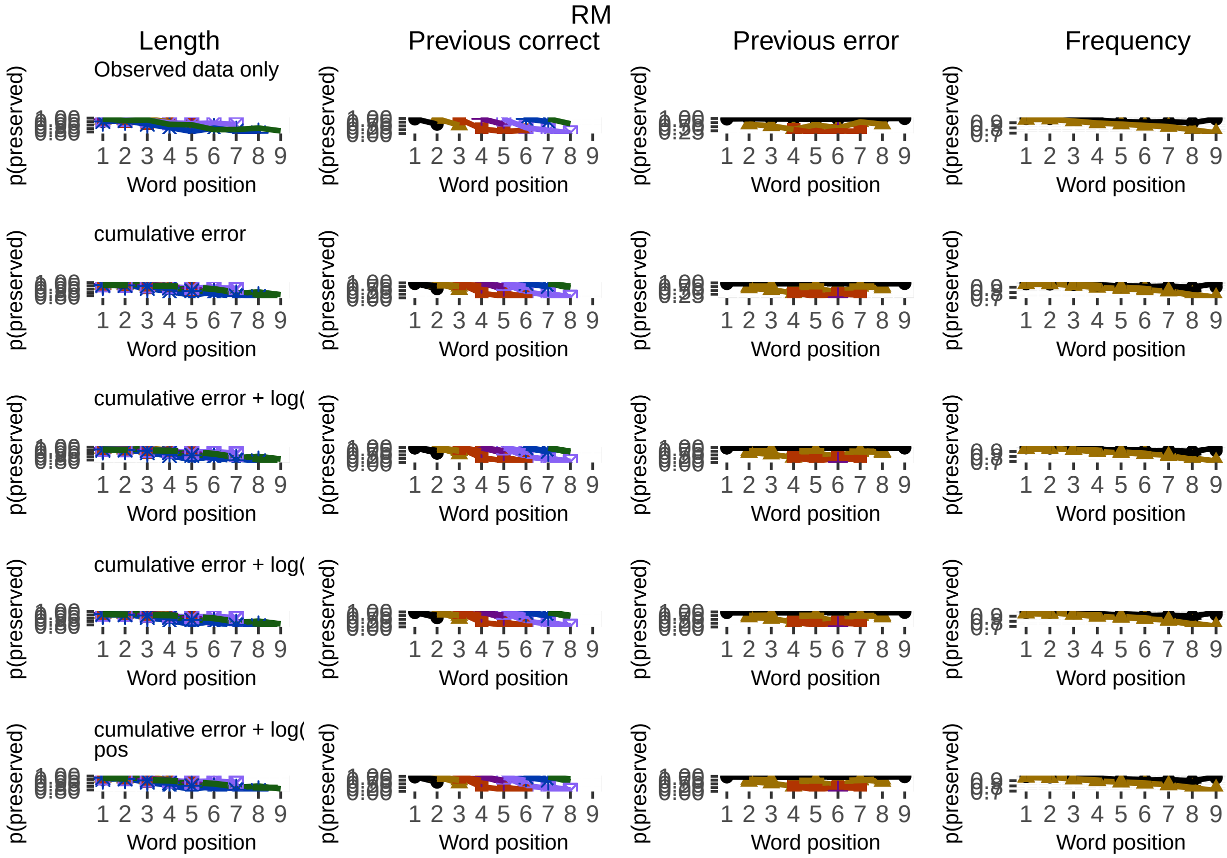
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.336       -2.163
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4442 Residual
## Null Deviance:       2308
## Residual Deviance: 1467  AIC: 1542
## log likelihood:  -733.3668
```

```
## Nagelkerke R2:  0.4258151
## % pres/err predicted correctly:  -367.193
## % of predictable range [ (model-null)/(1-null) ]:  0.4006607
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq
##       3.376       -2.118        0.155
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4441 Residual
## Null Deviance:       2308
## Residual Deviance: 1451  AIC: 1528
## log likelihood:  -725.2843
## Nagelkerke R2:  0.4332314
## % pres/err predicted correctly:  -366.7652
## % of predictable range [ (model-null)/(1-null) ]:  0.401357
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq     I(pos^2)
##    3.306020    -2.156862     0.159769     0.004055
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4440 Residual
## Null Deviance:       2308
## Residual Deviance: 1450  AIC: 1528
## log likelihood:  -724.7643
## Nagelkerke R2:  0.4337076
## % pres/err predicted correctly:  -366.3035
## % of predictable range [ (model-null)/(1-null) ]:  0.4021086
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq     I(pos^2)          pos
##     4.26469     -2.12914      0.16044      0.05731     -0.50959
##
## Degrees of Freedom: 4443 Total (i.e. Null);  4439 Residual
## Null Deviance:       2308
## Residual Deviance: 1437  AIC: 1515
## log likelihood:  -718.718
## Nagelkerke R2:  0.4392366
## % pres/err predicted correctly:  -363.2442
## % of predictable range [ (model-null)/(1-null) ]:  0.4070884
```

```
## ***************************
## `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 4 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 4 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 4 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 4 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 4 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 4 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 4 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 4 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 4 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!
```

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

	CumErr	I(pos^2)	pos	log_freq
McFadden	0.3379615	0.0171625	0.0206449	0.0138971
SquaredCorrelation	0.1670339	0.0091546	0.0110602	0.0072379
Nagelkerke	0.1670339	0.0091546	0.0110602	0.0072379
Estrella	0.2096182	0.0099372	0.0118788	0.0082674

```
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 + log_freq + I(pos^2) + pos CumErr + log_freq + I(pos^2) + pos 1437.436
## CumErr + log_freq + I(pos^2)             CumErr + log_freq + I(pos^2) 1449.529
## CumErr + log_freq                                 CumErr + log_freq 1450.569
## CumErr                                                       CumErr 1466.734
## null                                                           null 2308.233
##                                    deviance_explained percent_explained
## CumErr + log_freq + I(pos^2) + pos           870.7974          37.72571
## CumErr + log_freq + I(pos^2)                 858.7048          37.20182
## CumErr + log_freq                            857.6648          37.15676
## CumErr                                       841.4999          36.45645
## null                                           0.0000           0.00000
##                                    percent_of_explained_deviance increment_in_explained
## CumErr + log_freq + I(pos^2) + pos                     100.00000              1.3886772
## CumErr + log_freq + I(pos^2)                            98.61132              0.1194315
## CumErr + log_freq                                       98.49189              1.8563305
## CumErr                                                  96.63556             96.6355609
## 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 + log_freq + I(pos^2) + pos	1437.436	870.7974
CumErr + log_freq + I(pos^2)	1449.529	858.7048
CumErr + log_freq	1450.569	857.6648
CumErr	1466.734	841.4999
null	2308.233	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + log_freq + I(pos^2) + pos	37.72571	100.00000	1.3886772
CumErr + log_freq + I(pos^2)	37.20182	98.61132	0.1194315
CumErr + log_freq	37.15676	98.49189	1.8563305
CumErr	36.45645	96.63556	96.6355609
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   0.85884520
## I(pos^2) 0.04707081
## pos      0.05686880
## log_freq 0.03721518
```

```
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.
## `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.
```

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.9636295	1466.734
preserved ~ CumErr+log_freq	0.9640058	1450.569
preserved ~ CumErr+log_freq+I(pos^2)	0.9676138	1449.529
preserved ~ CumErr+log_freq+I(pos^2)+pos	0.9711990	1437.436

```
## 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       0.9636295       1466.734 0.0000000000
## 2             preserved ~ CumErr+log_freq       0.9640058       1450.569 0.0003762478
## 3    preserved ~ CumErr+log_freq+I(pos^2)       0.9676138       1449.529 0.0039842181
## 4 preserved ~ CumErr+log_freq+I(pos^2)+pos       0.9711990       1437.436 0.0075694918
##   diff_CumErr+log_freq diff_CumErr+log_freq+I(pos^2) diff_CumErr+log_freq+I(pos^2)+pos
## 1        -0.0003762478                  -0.003984218                       -0.007569492
## 2         0.0000000000                  -0.003607970                       -0.007193244
## 3         0.0036079702                   0.000000000                       -0.003585274
## 4         0.0071932440                   0.003585274                        0.000000000
```

```
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)
```

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

model	diff_CumErr	diff_CumErr+log_freq	diff_CumErr+log_freq+I(pos^2)
preserved ~ CumErr	0.0000000	-0.0003762	-0.0039842
preserved ~ CumErr+log_freq	0.0003762	0.0000000	-0.0036080
preserved ~ CumErr+log_freq+I(pos^2)	0.0039842	0.0036080	0.0000000
preserved ~ CumErr+log_freq+I(pos^2)+pos	0.0075695	0.0071932	0.0035853