MC - 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	522	33	126	NA	NA	681
2	65	NA	416	94	106	681
3	297	NA	166	202	16	681
4	293	NA	233	65	35	626
5	226	NA	210	69	37	542
6	202	1	137	70	22	432
7	175	NA	102	28	18	323
8	90	NA	55	25	4	174
9	75	NA	2	NA	7	84

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7665198	0.0484581	0.1850220	NA	NA	681
2	0.0954479	NA	0.6108664	0.1380323	0.1556535	681
3	0.4361233	NA	0.2437592	0.2966226	0.0234949	681
4	0.4680511	NA	0.3722045	0.1038339	0.0559105	626
5	0.4169742	NA	0.3874539	0.1273063	0.0682657	542
6	0.4675926	0.0023148	0.3171296	0.1620370	0.0509259	432

pos_factor	O	P	V	1	S	total
7	0.5417957	NA	0.3157895	0.0866873	0.0557276	323
8	0.5172414	NA	0.3160920	0.1436782	0.0229885	174
9	0.8928571	NA	0.0238095	NA	0.0833333	84

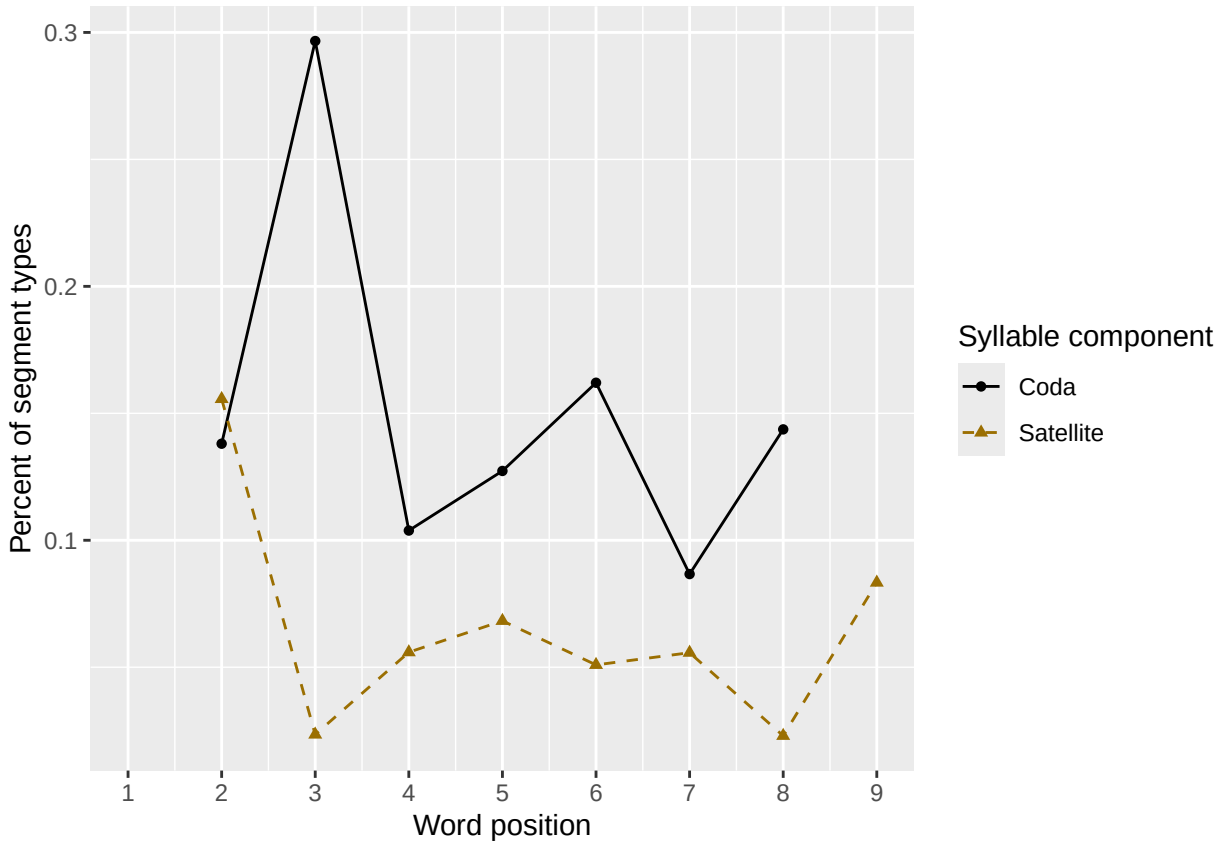
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 0.764 0.827 0.645 NA     NA     NA     NA     NA     NA
## 2       5 0.732 0.796 0.700  0.581 NA     NA     NA     NA     NA
## 3       6 0.605 0.788 0.665  0.716  0.592 NA     NA     NA     NA
## 4       7 0.744 0.787 0.685  0.676  0.775  0.614 NA     NA     NA
## 5       8 0.787 0.794 0.698  0.685  0.616  0.727  0.648 NA     NA
## 6       9 0.739 0.767 0.694  0.677  0.657  0.631  0.729  0.620 NA
## 7      10 0.731 0.848 0.691  0.679  0.710  0.551  0.662  0.750  0.576
```

```
# 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.
```

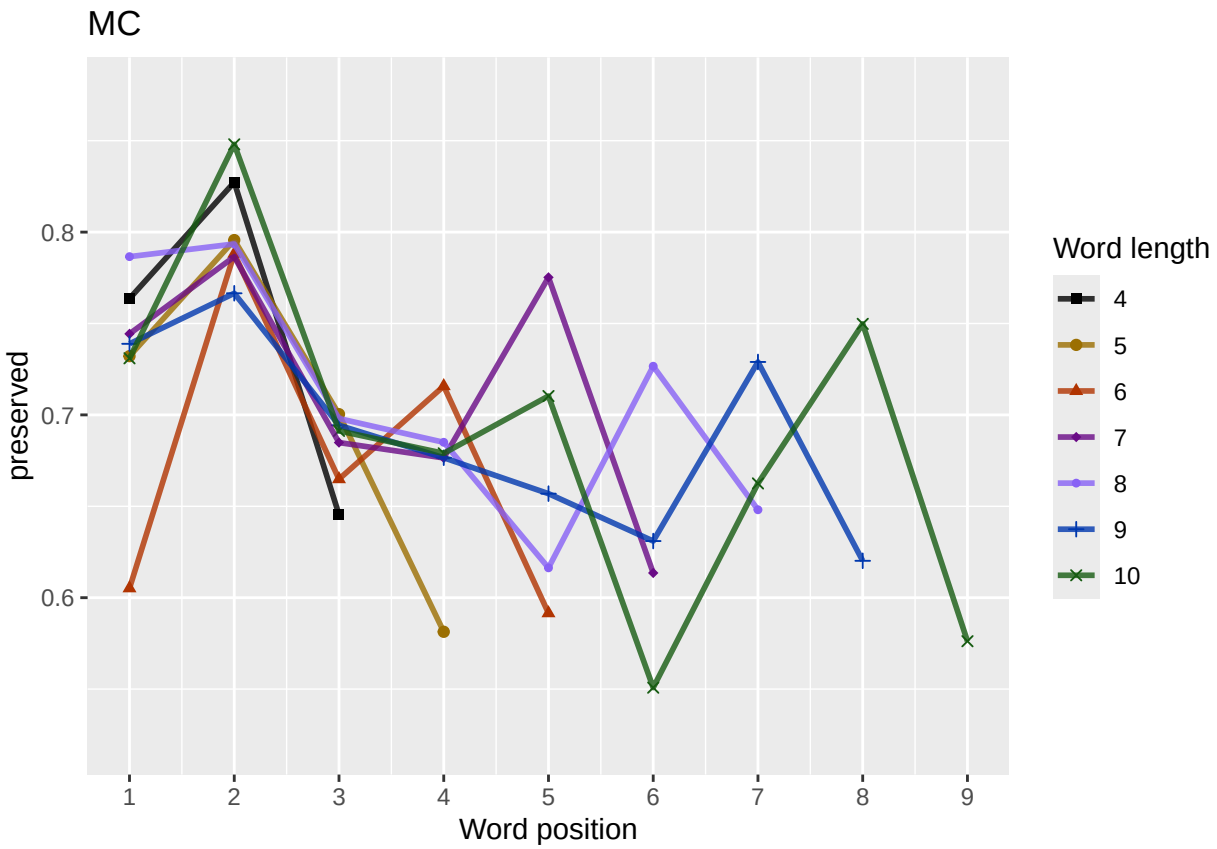
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    55    55    55    NA    NA    NA    NA    NA    NA
## 2       5    84    84    84    84    NA    NA    NA    NA    NA
## 3       6   110   110   110   110   110    NA    NA    NA    NA
## 4       7   109   109   109   109   109   109    NA    NA    NA
## 5       8   149   149   149   149   149   149   149    NA    NA
## 6       9    90    90    90    90    90    90    90    90    NA
## 7      10    84    84    84    84    84    84    84    84    84
```

```
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
##    1.101748     0.028071     0.008357    -0.164173
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4725  AIC: 5304
## log likelihood:  -2362.453
## Nagelkerke R2:  0.01095696
## % pres/err predicted correctly:  -1670.888
## % of predictable range [ (model-null)/(1-null) ]:  0.008932169
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     0.95149      0.03256     -0.09212
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5304
## log likelihood:  -2363.163
## Nagelkerke R2:  0.01046339
## % pres/err predicted correctly:  -1672.106
## % of predictable range [ (model-null)/(1-null) ]:  0.008209711
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.304963     0.009837    -0.168274
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5305
## log likelihood:  -2363.24
## Nagelkerke R2:  0.01040948
## % pres/err predicted correctly:  -1671.477
## % of predictable range [ (model-null)/(1-null) ]:  0.008582886
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    1.184911     0.004122    -0.166290     0.008734
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5305
## log likelihood:  -2362.826
## Nagelkerke R2:  0.01069774
## % pres/err predicted correctly:  -1671.62
## % of predictable range [ (model-null)/(1-null) ]:  0.008498014
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.16224     -0.08178
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4729  AIC: 5306
## log likelihood:  -2364.253
## Nagelkerke R2:  0.009704784
## % pres/err predicted correctly:  -1673.087
## % of predictable range [ (model-null)/(1-null) ]:  0.007628175
## ***************************
## 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)
##        1.0578883         0.0321301         0.0007167        -0.1203786         0.0007897
##      stimlen:pos
##       -0.0043903
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4218 Residual
## Null Deviance:       4756
## Residual Deviance: 4725  AIC: 5308
## log likelihood:  -2362.435
## Nagelkerke R2:  0.01096959
## % pres/err predicted correctly:  -1670.941
## % of predictable range [ (model-null)/(1-null) ]:  0.008900174
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##      0.8406
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4223 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5334
## log likelihood:  -2378.147
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1685.956
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      0.9581      -0.0152
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5336
## log likelihood:  -2377.858
## Nagelkerke R2:  0.0002026833
## % pres/err predicted correctly:  -1685.674
## % of predictable range [ (model-null)/(1-null) ]:  0.0001668072
## ***************************
```

```
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	5303.617	0.0000000	1.0000000	0.6314709	0.0010957	0.1017482	-0.0280708	-0.1641735	NA	0.0083573	NA
preserved ~ stimlen + pos	5304.144	0.5276378	0.7681126	0.4851733	0.0010468	0.0951491	-0.0325567	-0.0921184	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	5304.736	0.1196598	0.8857130	0.2179795	0.5010409	1.3049631	NA	-0.1682743	NA	0.0098370	NA
preserved ~ stimlen * pos	5305.136	0.5167334	0.4468430	0.1474196	0.5010697	1.1849109	-0.0041216	-0.1662897	0.0087341	NA	NA
preserved ~ pos	5306.402	1.7850700	0.2484447	0.0781879	0.5009704	1.1622445	NA	-0.0817768	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	5307.837	3.2199831	0.1212390	0.0381551	0.5010969	1.0578880	-0.0321301	-0.1203786	0.0043903	0.0007167	0.0007897
preserved ~ 1	5333.778	29.1598982	0.0000000	0.0000000	0.0000000	0.8406460	NA	NA	NA	NA	NA
preserved ~ stimlen	5336.086	31.4698939	0.0000000	0.0000000	0.0002027	0.9580859	-0.0151990	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
##    1.101748     0.028071     0.008357    -0.164173
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4725  AIC: 5304
```

```
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
```

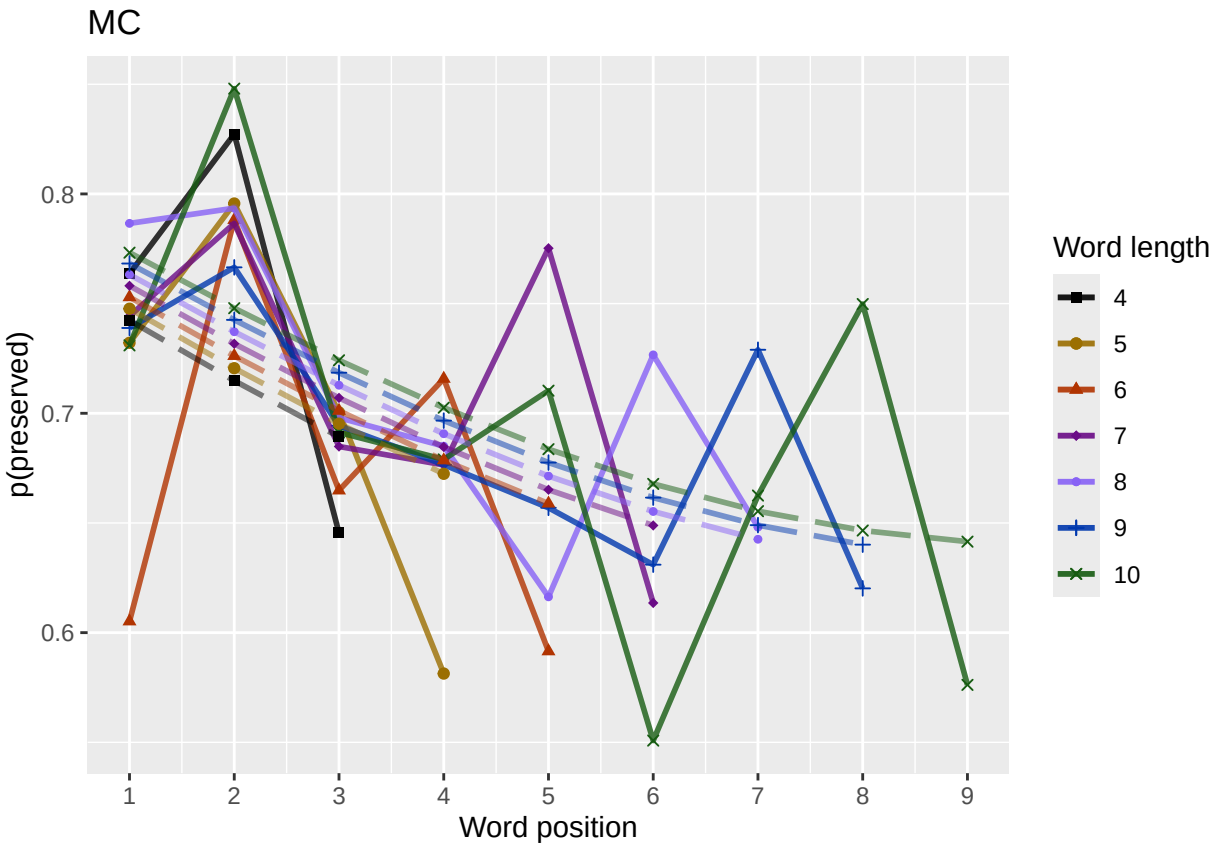
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.742 0.715 0.689 NA     NA     NA     NA     NA     NA
## 2       5 0.748 0.721 0.695  0.672 NA     NA     NA     NA     NA
## 3       6 0.753 0.726 0.701  0.679  0.659 NA     NA     NA     NA
## 4       7 0.758 0.732 0.707  0.685  0.665  0.649 NA     NA     NA
## 5       8 0.763 0.737 0.713  0.691  0.671  0.655  0.643 NA     NA
```

```
## 6       9 0.768 0.743 0.719  0.697  0.678  0.662  0.649  0.640 NA
## 7      10 0.773 0.748 0.724  0.703  0.684  0.668  0.655  0.647  0.641
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       33   681
```

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

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

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

```
## [1] "The number of responses with fragments was 33 / 681 = 4.85 percent"
```

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

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

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

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

```
## ****************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    1.091533     0.022447     0.007952    -0.131119
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       4542
## Residual Deviance: 4528  AIC: 5109
## log likelihood:  -2263.986
## Nagelkerke R2:  0.005200003
## % pres/err predicted correctly:  -1594.547
## % of predictable range [ (model-null)/(1-null) ]:  0.004742017
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.254135     0.009134    -0.134375
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       4542
## Residual Deviance: 4529  AIC: 5110
## log likelihood:  -2264.475
## Nagelkerke R2:  0.004846245
## % pres/err predicted correctly:  -1594.888
## % of predictable range [ (model-null)/(1-null) ]:  0.004529386
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     0.95185      0.02659     -0.06312
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       4542
## Residual Deviance: 4529  AIC: 5110
## log likelihood:  -2264.592
## Nagelkerke R2:  0.004761788
## % pres/err predicted correctly:  -1595.658
## % of predictable range [ (model-null)/(1-null) ]:  0.004048718
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
```

```
##     1.12458     -0.05476
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       4542
## Residual Deviance: 4531  AIC: 5111
## log likelihood:  -2265.298
## Nagelkerke R2:  0.004251093
## % pres/err predicted correctly:  -1596.277
## % of predictable range [ (model-null)/(1-null) ]:  0.003662453
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.05557      0.01395     -0.09666      0.00395
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       4542
## Residual Deviance: 4529  AIC: 5111
## log likelihood:  -2264.526
## Nagelkerke R2:  0.004809142
## % pres/err predicted correctly:  -1595.492
## % of predictable range [ (model-null)/(1-null) ]:  0.004152096
## ***************************
## 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)
##         1.149888          0.019720          0.026737         -0.221557         -0.001850
##      stimlen:pos
##         0.007979
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4126 Residual
## Null Deviance:       4542
## Residual Deviance: 4528  AIC: 5113
## log likelihood:  -2263.842
## Nagelkerke R2:  0.005304033
## % pres/err predicted correctly:  -1594.341
## % of predictable range [ (model-null)/(1-null) ]:  0.004870393
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9127
```

```
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4131 Residual
## Null Deviance:       4542
## Residual Deviance: 4542  AIC: 5122
## log likelihood:  -2271.163
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1602.149
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    0.953869    -0.005338
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       4542
## Residual Deviance: 4542  AIC: 5124
## log likelihood:  -2271.129
## Nagelkerke R2:  2.479915e-05
## % pres/err predicted correctly:  -1602.101
## % of predictable range [ (model-null)/(1-null) ]:  2.984239e-05
## ***************************
```

```
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	5109.422	0.000000	1.000000	0.6258100	0.0100520	0.0915330	-0.0224468	-0.1311190	NA	0.0079517	NA
preserved ~ I(pos^2) + pos	5109.579	0.156939	0.924529	0.5786210	0.0048462	0.2541347	NA	-0.1343748	NA	0.0091337	NA
preserved ~ stimlen + pos	5109.719	0.297080	0.861965	0.5392478	0.0047608	0.9518510	-0.0265928	-0.0631207	NA	NA	NA
preserved ~ pos	5110.762	1.340522	0.511570	0.3203750	0.0042511	1.1245803	NA	-0.0547618	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	5111.395	0	1	0.6372898	0.0962455	0.0004809	0.1055571	0.0139542	- 0.0966600	0.0039504	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	5112.629	1.207084	0.5201182	0.3405192	0.0530534	0.1498881	0.0197201	- 0.2215565	0.0079787	0.0267375	- 0.0018502
preserved ~ 1	5122.130	10.708280	0.0017395	0.0004449	0.0000000	0.0000000	0.0126921	NA	NA	NA	NA	NA
preserved ~ stimlen	5124.341	12.919378	0.0005758	0.0001486	0.0000248	0.0953868	- 0.0053382	NA	NA	NA	NA	NA

```
# plot no fragment data

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

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

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

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

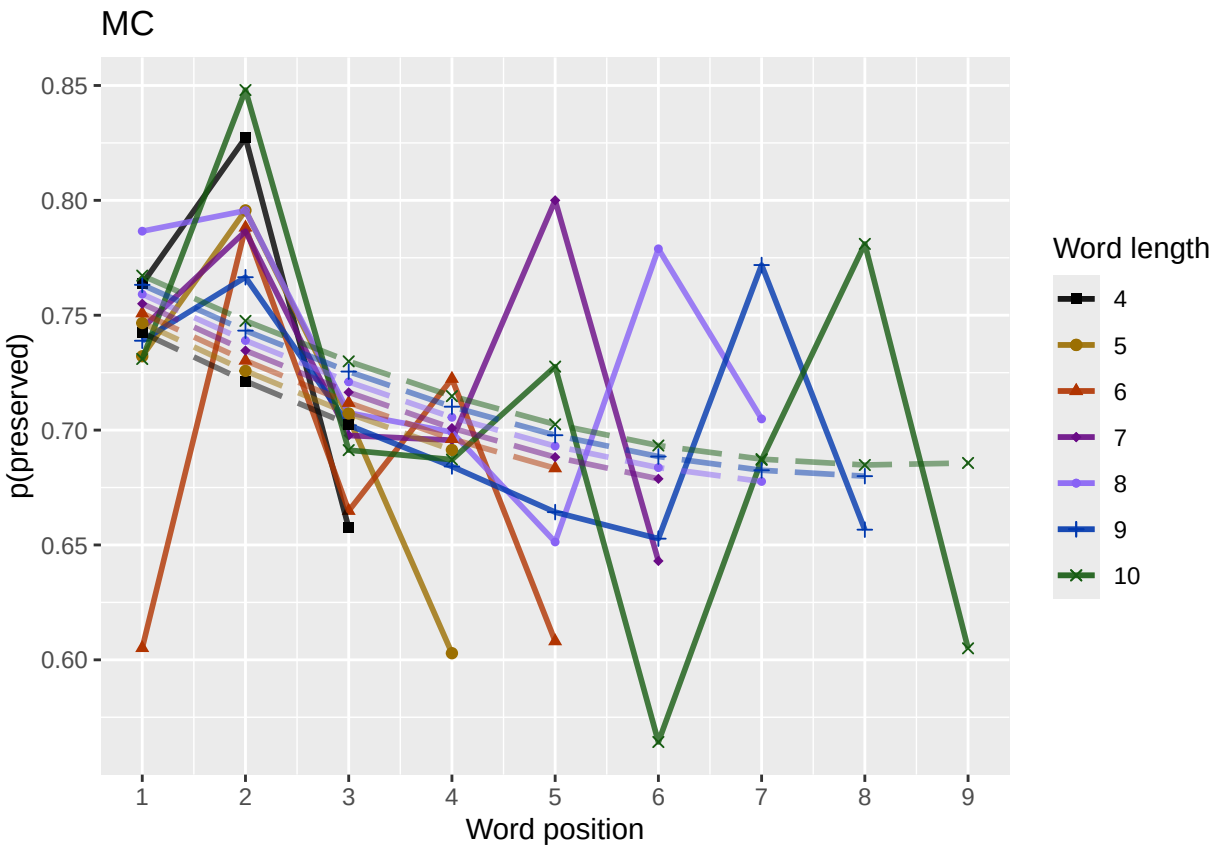
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_fi
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.742 0.721 0.703 NA     NA     NA     NA     NA     NA
## 2       5 0.747 0.726 0.707  0.691 NA     NA     NA     NA     NA
## 3       6 0.751 0.730 0.712  0.696  0.683 NA     NA     NA     NA
## 4       7 0.755 0.735 0.716  0.701  0.688  0.679 NA     NA     NA
## 5       8 0.759 0.739 0.721  0.706  0.693  0.684  0.678 NA     NA
## 6       9 0.763 0.743 0.725  0.710  0.698  0.688  0.683  0.680 NA
## 7      10 0.767 0.748 0.730  0.715  0.702  0.693  0.687  0.685  0.686
```

```
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: 0.52 - 0.88"
```

```
# 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] 0.005792851
```

```
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] -0.01689432
```

```
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:  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
##         1.3340955          0.0113128         -0.1810575         -0.0739798          0.0007354
##      pos:log_freq
##         0.0091979
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4218 Residual
## Null Deviance:       4756
## Residual Deviance: 4721  AIC: 5302
## log likelihood:  -2360.63
## Nagelkerke R2:  0.01222456
## % pres/err predicted correctly:  -1668.872
## % of predictable range [ (model-null)/(1-null) ]:  0.01012707
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      1.01923       0.02326      -0.08935      -0.07028       0.01301
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4219 Residual
## Null Deviance:       4756
```

```
## Residual Deviance: 4723  AIC: 5303
## log likelihood:  -2361.424
## Nagelkerke R2:  0.01167264
## % pres/err predicted correctly:  -1670.444
## % of predictable range [ (model-null)/(1-null) ]:  0.009194842
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)               pos           log_freq
##         1.2085043          0.0170878          0.0103688         -0.1779746         -0.0687540
## I(pos^2):log_freq       pos:log_freq
##         0.0006547          0.0091546
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4217 Residual
## Null Deviance:       4756
## Residual Deviance: 4721  AIC: 5303
## log likelihood:  -2360.361
## Nagelkerke R2:  0.01241138
## % pres/err predicted correctly:  -1668.726
## % of predictable range [ (model-null)/(1-null) ]:  0.01021345
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      1.17176      -0.08233      -0.07825       0.01383
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4724  AIC: 5303
## log likelihood:  -2361.94
## Nagelkerke R2:  0.01131421
## % pres/err predicted correctly:  -1670.886
## % of predictable range [ (model-null)/(1-null) ]:  0.008932987
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    1.101748     0.028071     0.008357    -0.164173
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4725  AIC: 5304
```

```
## log likelihood:  -2362.453
## Nagelkerke R2:  0.01095696
## % pres/err predicted correctly:  -1670.888
## % of predictable range [ (model-null)/(1-null) ]:  0.008932169
## **************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    1.147780     0.022269     0.008426    -0.164785    -0.018274
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4219 Residual
## Null Deviance:       4756
## Residual Deviance: 4724  AIC: 5304
## log likelihood:  -2361.958
## Nagelkerke R2:  0.01130121
## % pres/err predicted correctly:  -1670.392
## % of predictable range [ (model-null)/(1-null) ]:  0.009225797
## **************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     0.95149      0.03256     -0.09212
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5304
## log likelihood:  -2363.163
## Nagelkerke R2:  0.01046339
## % pres/err predicted correctly:  -1672.106
## % of predictable range [ (model-null)/(1-null) ]:  0.008209711
## **************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)
##         1.185352          0.019140         -0.089338          0.008790
##              pos  stimlen:log_freq
##        -0.167930          0.009176
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4218 Residual
## Null Deviance:       4756
## Residual Deviance: 4723  AIC: 5305
## log likelihood:  -2361.575
```

```
## Nagelkerke R2:  0.01156754
## % pres/err predicted correctly:  -1670.02
## % of predictable range [ (model-null)/(1-null) ]:  0.009446182
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     0.99575      0.02686     -0.09214     -0.01806
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4725  AIC: 5305
## log likelihood:  -2362.679
## Nagelkerke R2:  0.01079972
## % pres/err predicted correctly:  -1671.653
## % of predictable range [ (model-null)/(1-null) ]:  0.00847835
## **************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.304963     0.009837    -0.168274
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5305
## log likelihood:  -2363.24
## Nagelkerke R2:  0.01040948
## % pres/err predicted correctly:  -1671.477
## % of predictable range [ (model-null)/(1-null) ]:  0.008582886
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         1.023963          0.022856         -0.080920         -0.089443          0.001664
##     log_freq:pos
##         0.012456
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4218 Residual
## Null Deviance:       4756
## Residual Deviance: 4723  AIC: 5305
## log likelihood:  -2361.414
## Nagelkerke R2:  0.01167985
```

```
## % pres/err predicted correctly:  -1670.434
## % of predictable range [ (model-null)/(1-null) ]:  0.00920092
## **************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##         1.2124611          0.0167538         -0.0787108          0.0103606         -0.1780266
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         0.0013981          0.0005845          0.0093024
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4216 Residual
## Null Deviance:       4756
## Residual Deviance: 4721  AIC: 5305
## log likelihood:  -2360.354
## Nagelkerke R2:  0.01241634
## % pres/err predicted correctly:  -1668.729
## % of predictable range [ (model-null)/(1-null) ]:  0.01021152
## **************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    1.184911     0.004122    -0.166290     0.008734
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5305
## log likelihood:  -2362.826
## Nagelkerke R2:  0.01069774
## % pres/err predicted correctly:  -1671.62
## % of predictable range [ (model-null)/(1-null) ]:  0.008498014
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         1.024034          0.024177         -0.082932         -0.092122          0.008381
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4219 Residual
## Null Deviance:       4756
## Residual Deviance: 4725  AIC: 5305
## log likelihood:  -2362.358
## Nagelkerke R2:  0.01102358
## % pres/err predicted correctly:  -1671.346
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.00866053
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     1.17179     -0.08414     -0.02355
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4727  AIC: 5305
## log likelihood:  -2363.374
## Nagelkerke R2:  0.01031663
## % pres/err predicted correctly:  -1672.248
## % of predictable range [ (model-null)/(1-null) ]:  0.008125883
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.16224     -0.08178
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4729  AIC: 5306
## log likelihood:  -2364.253
## Nagelkerke R2:  0.009704784
## % pres/err predicted correctly:  -1673.087
## % of predictable range [ (model-null)/(1-null) ]:  0.007628175
## ***************************
## 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)
##        1.0578883         0.0321301         0.0007167        -0.1203786         0.0007897
##      stimlen:pos
##       -0.0043903
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4218 Residual
## Null Deviance:       4756
## Residual Deviance: 4725  AIC: 5308
## log likelihood:  -2362.435
## Nagelkerke R2:  0.01096959
## % pres/err predicted correctly:  -1670.941
## % of predictable range [ (model-null)/(1-null) ]:  0.008900174
```

```
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.8406
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4223 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5334
## log likelihood:  -2378.147
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1685.956
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      0.9581      -0.0152
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5336
## log likelihood:  -2377.858
## Nagelkerke R2:  0.0002026833
## % pres/err predicted correctly:  -1685.674
## % of predictable range [ (model-null)/(1-null) ]:  0.0001668072
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     1.00202     -0.02086     -0.01794
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4755  AIC: 5336
## log likelihood:  -2377.378
## Nagelkerke R2:  0.000539247
## % pres/err predicted correctly:  -1685.206
## % of predictable range [ (model-null)/(1-null) ]:  0.0004445839
## ***************************
## model index:  1
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         log_freq  stimlen:log_freq
##        1.030346        -0.023542        -0.082911          0.008388
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4754  AIC: 5337
## log likelihood:  -2377.053
## Nagelkerke R2:  0.0007664223
## % pres/err predicted correctly:  -1684.919
## % of predictable range [ (model-null)/(1-null) ]:  0.0006142491
## ***************************
```

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

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

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

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

Model	AIC DeltaAIC AICexp AICwt NagR2 (Intercept) [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ (I(pos^2) + pos) * log_freq	5302.[illegible]	-0.0739798	NA	-0.1810575	0.0091979	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen + pos * log_freq	5303.[illegible]	-0.0702786	NA	-0.0893475	0.0130134	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	5303.[illegible]	-0.0687540	NA	-0.1779746	0.0091546	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ pos * log_freq	5303.[illegible]	-0.0782456	NA	-0.0823342	0.0138279	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	5303.[illegible]	[illegible]	NA	-0.1641735	NA	NA	[illegible]	NA	NA	NA	NA

Model	AIC	Delta_AIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0182738	NA	-0.1647853	NA	NA	0.0084257	NA	NA	NA	NA
preserved ~ stimlen + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0921184	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0893382	0.0091761	-0.1679303	NA	NA	0.0087902	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0180556	NA	-0.0921375	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	-0.1682743	NA	NA	0.0098870	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0809196	0.0016638	-0.0894429	NA	0.0124561	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0787108	0.0013981	-0.1780266	NA	[illegible]	[illegible]	NA	0.0005845	NA	NA
preserved ~ stimlen * pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.1662897	NA	NA	NA	NA	NA	0.0087841	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0829317	0.0083812	-0.0921219	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0235471	NA	-0.0841424	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	-0.0817768	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	-0.1203786	NA	NA	0.0007167	NA	NA	-0.0043903	0.0007897
preserved ~ 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0151990	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0208641	-0.0179355	NA	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	wAIC	NagR2	(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 * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0083882	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##         1.3340955          0.0113128         -0.1810575         -0.0739798          0.0007354
##      pos:log_freq
##         0.0091979
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4218 Residual
## Null Deviance:      4756
## Residual Deviance: 4721  AIC: 5302
```

```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.
```

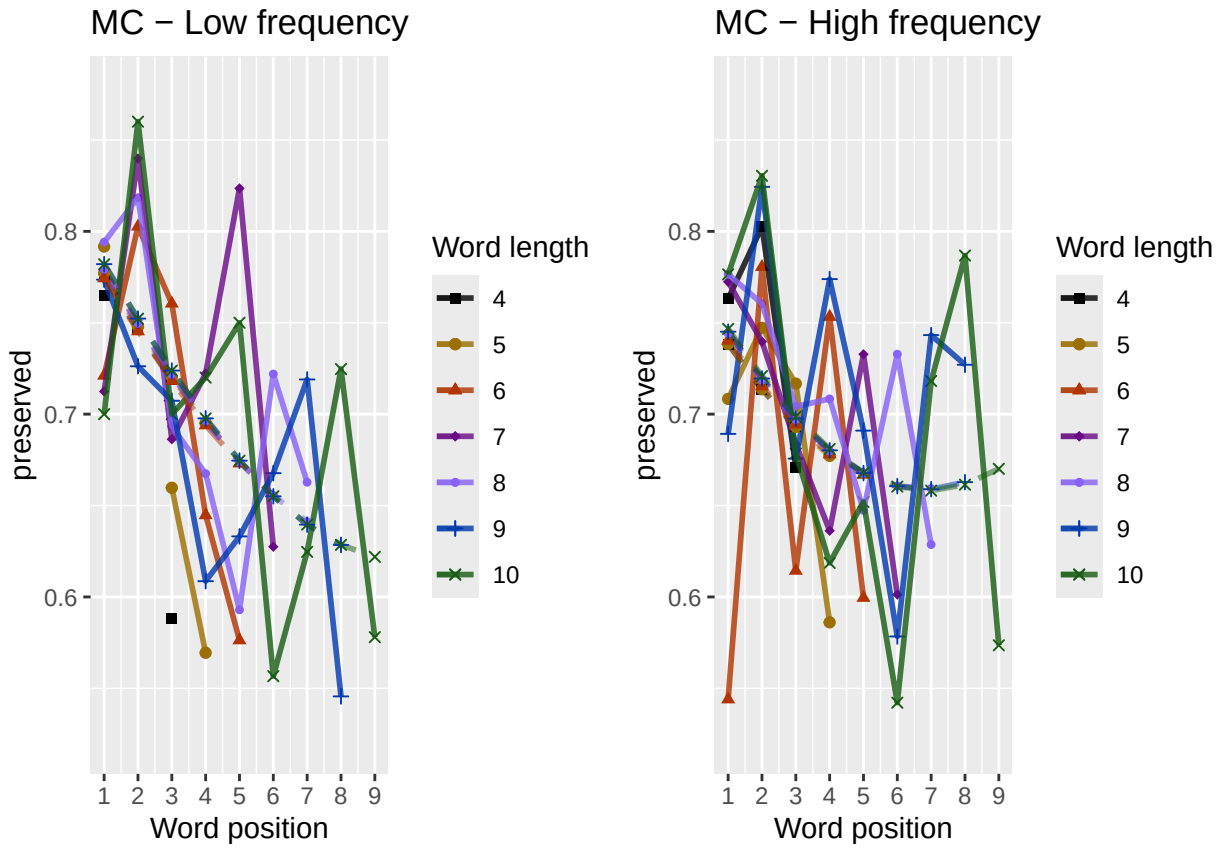
```r
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

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

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




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

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

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

```
## Coefficients:
## (Intercept)       CumErr
##      1.2118      -0.4273
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4556  AIC: 5092
## log likelihood:  -2278.152
## Nagelkerke R2:  0.06843933
## % pres/err predicted correctly:  -1588.963
## % of predictable range [ (model-null)/(1-null) ]:  0.05749597
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.304963     0.009837    -0.168274
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5305
## log likelihood:  -2363.24
## Nagelkerke R2:  0.01040948
## % pres/err predicted correctly:  -1671.477
## % of predictable range [ (model-null)/(1-null) ]:  0.008582886
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.16224     -0.08178
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4729  AIC: 5306
## log likelihood:  -2364.253
## Nagelkerke R2:  0.009704784
## % pres/err predicted correctly:  -1673.087
## % of predictable range [ (model-null)/(1-null) ]:  0.007628175
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.7440       0.0474
```

```
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4750  AIC: 5326
## log likelihood:  -2375.147
## Nagelkerke R2:  0.002101338
## % pres/err predicted correctly:  -1682.244
## % of predictable range [ (model-null)/(1-null) ]:  0.002200455
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.8406
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4223 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5334
## log likelihood:  -2378.147
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1685.956
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      0.9581      -0.0152
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5336
## log likelihood:  -2377.858
## Nagelkerke R2:  0.0002026833
## % pres/err predicted correctly:  -1685.674
## % of predictable range [ (model-null)/(1-null) ]:  0.0001668072
## ***************************
```

```
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	5091.563	0.0000	1	1	0.0684393	3.2118220	NA	-0.4273043	NA	NA	NA
preserved ~ (I(pos^2) + pos)	5304.736	213.1733	0	0	0.0104095	1.3049631	NA	NA	0.009837	-0.1682743	NA
preserved ~ pos	5306.402	214.8387	0	0	0.0097048	1.1622445	NA	NA	NA	-0.0817768	NA
preserved ~ CumPres	5326.011	234.4480	0	0	0.0021018	0.7439771	0.0473979	NA	NA	NA	NA
preserved ~ 1	5333.776	242.2135	0	0	0.0000000	0.8406460	NA	NA	NA	NA	NA
preserved ~ stimlen	5336.086	244.5235	0	0	0.0002027	0.9580859	NA	NA	NA	NA	-0.015199

```
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	0.6415511	553
O	0.5996022	1945
P	0.5588235	34
S	0.6537755	245
V	0.8643488	1447

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       1.225       -0.471
##
## Degrees of Freedom: 3944 Total (i.e. Null);  3943 Residual
## Null Deviance:       4410
## Residual Deviance: 4215  AIC: 4725
## log likelihood:  -2107.588
## Nagelkerke R2:  0.07149691
## % pres/err predicted correctly:  -1467.955
## % of predictable range [ (model-null)/(1-null) ]:  0.0600982
```

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.37985      0.01147     -0.18999
##
## Degrees of Freedom: 3944 Total (i.e. Null);  3942 Residual
## Null Deviance:       4410
## Residual Deviance: 4377  AIC: 4928
## log likelihood:  -2188.335
## Nagelkerke R2:  0.01239567
## % pres/err predicted correctly:  -1545.973
## % of predictable range [ (model-null)/(1-null) ]:  0.01017894
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.21202     -0.08884
##
## Degrees of Freedom: 3944 Total (i.e. Null);  3943 Residual
## Null Deviance:       4410
## Residual Deviance: 4379  AIC: 4930
## log likelihood:  -2189.622
## Nagelkerke R2:  0.01143398
## % pres/err predicted correctly:  -1547.903
## % of predictable range [ (model-null)/(1-null) ]:  0.008944499
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     0.78264      0.03959
##
## Degrees of Freedom: 3944 Total (i.e. Null);  3943 Residual
## Null Deviance:       4410
## Residual Deviance: 4406  AIC: 4956
## log likelihood:  -2203.021
## Nagelkerke R2:  0.001383968
## % pres/err predicted correctly:  -1559.432
## % of predictable range [ (model-null)/(1-null) ]:  0.001567161
## ***************************
## model index:  6
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.8597
##
## Degrees of Freedom: 3944 Total (i.e. Null);  3944 Residual
## Null Deviance:       4410
## Residual Deviance: 4410  AIC: 4960
## log likelihood:  -2204.859
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1561.882
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     0.90207     -0.00549
##
## Degrees of Freedom: 3944 Total (i.e. Null);  3943 Residual
## Null Deviance:       4410
## Residual Deviance: 4410  AIC: 4962
## log likelihood:  -2204.824
## Nagelkerke R2:  2.648017e-05
## % pres/err predicted correctly:  -1561.838
## % of predictable range [ (model-null)/(1-null) ]:  2.790784e-05
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	4724.683	0.0000	1	1	0.0714969	1.2250920	NA	-0.4710294	NA	NA	NA
preserved ~ (I(pos^2) + pos)	4927.562	202.8787	0	0	0.0123957	1.3798465	NA	NA	0.0114736	-0.1899915	NA
preserved ~ pos	4930.119	205.4357	0	0	0.0114340	1.2120238	NA	NA	NA	-0.0888440	NA
preserved ~ CumPres	4955.855	231.1714	0	0	0.0013840	0.7826386	0.0395856	NA	NA	NA	NA
preserved ~ 1	4960.251	235.5677	0	0	0.0000000	0.8596608	NA	NA	NA	NA	NA
preserved ~ stimlen	4962.484	237.8006	0	0	0.0000265	0.9020726	NA	NA	NA	NA	-0.0054901

```
#  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
##      1.1875      -0.4473
##
## Degrees of Freedom: 3391 Total (i.e. Null);  3390 Residual
## Null Deviance:       3740
## Residual Deviance: 3636  AIC: 4072
## log likelihood:  -1817.801
## Nagelkerke R2:  0.04557161
## % pres/err predicted correctly:  -1265.391
## % of predictable range [ (model-null)/(1-null) ]:  0.04034065
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.27801     -0.09349
##
## Degrees of Freedom: 3391 Total (i.e. Null);  3390 Residual
## Null Deviance:       3740
## Residual Deviance: 3710  AIC: 4173
## log likelihood:  -1855.131
## Nagelkerke R2:  0.0132726
## % pres/err predicted correctly:  -1304.782
## % of predictable range [ (model-null)/(1-null) ]:  0.01049106
```

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.358390     0.005668    -0.143267
##
## Degrees of Freedom: 3391 Total (i.e. Null);  3389 Residual
## Null Deviance:       3740
## Residual Deviance: 3710  AIC: 4174
## log likelihood:  -1854.843
## Nagelkerke R2:  0.01352387
## % pres/err predicted correctly:  -1304.167
## % of predictable range [ (model-null)/(1-null) ]:  0.01095707
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9078
##
## Degrees of Freedom: 3391 Total (i.e. Null);  3391 Residual
## Null Deviance:       3740
## Residual Deviance: 3740  AIC: 4204
## log likelihood:  -1870.236
## Nagelkerke R2:  -3.323835e-16
## % pres/err predicted correctly:  -1318.626
## % 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
##    0.900376     0.004414
##
## Degrees of Freedom: 3391 Total (i.e. Null);  3390 Residual
## Null Deviance:       3740
## Residual Deviance: 3740  AIC: 4206
## log likelihood:  -1870.221
## Nagelkerke R2:  1.3257e-05
## % pres/err predicted correctly:  -1318.557
## % of predictable range [ (model-null)/(1-null) ]:  5.217921e-05
## ***************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##    0.926147    -0.002387
##
## Degrees of Freedom: 3391 Total (i.e. Null);  3390 Residual
## Null Deviance:       3740
## Residual Deviance: 3740  AIC: 4206
## log likelihood:  -1870.23
## Nagelkerke R2:  5.010255e-06
## % pres/err predicted correctly:  -1318.608
## % of predictable range [ (model-null)/(1-null) ]:  1.371328e-05
## ***************************
```

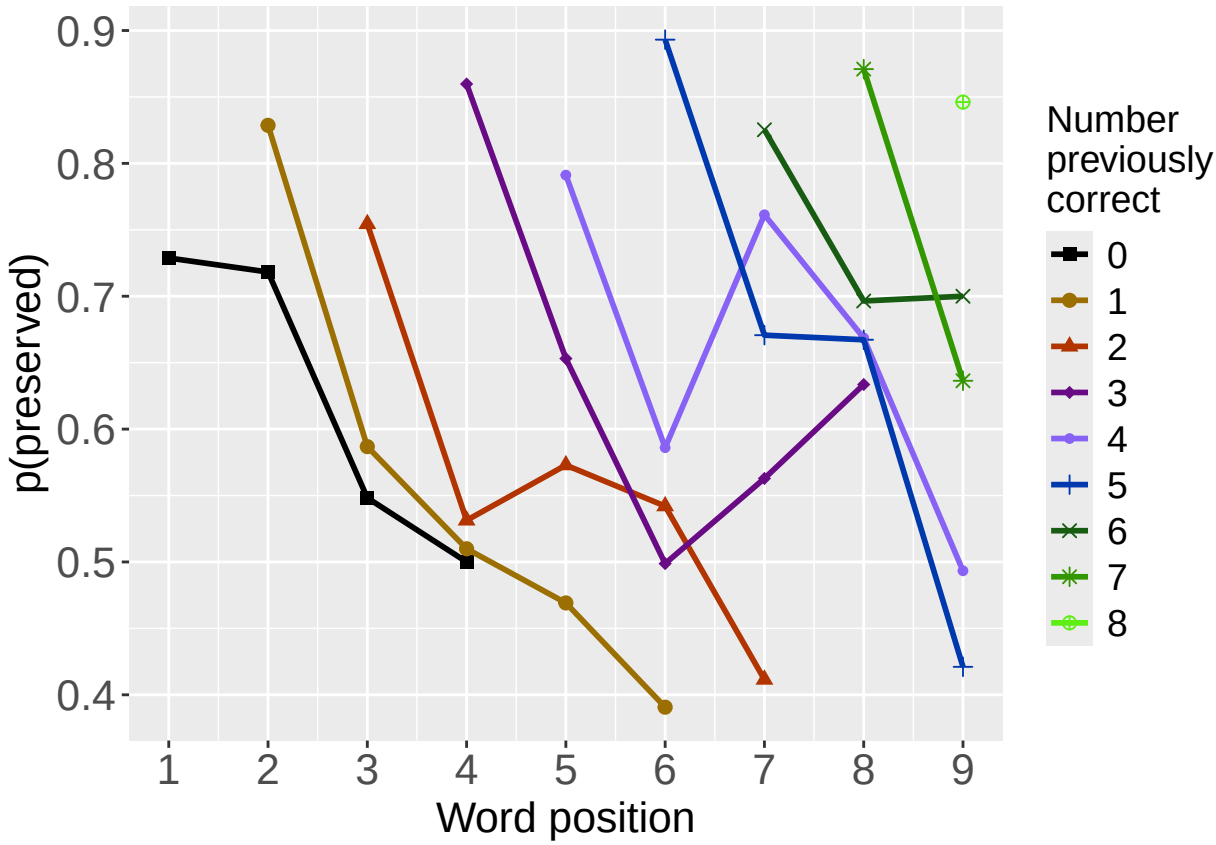
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	4072.161	0.0000	1	1	0.0455716	1.1875153	NA	-0.4472977	NA	NA	NA
preserved ~ pos	4173.189	101.0275	0	0	0.0132726	1.2780114	NA	NA	NA	-0.0934851	NA
preserved ~ (I(pos^2) + pos)	4173.895	101.7336	0	0	0.0135239	1.3583902	NA	NA	0.0056678	-0.1432673	NA
preserved ~ 1	4203.878	131.7165	0	0	0.0000000	0.9077569	NA	NA	NA	NA	NA
preserved ~ CumPres	4205.668	133.5070	0	0	0.0000130	0.9003765	0.0044139	NA	NA	NA	NA
preserved ~ stimlen	4205.935	133.7737	0	0	0.0000050	0.9261466	NA	NA	NA	NA	-0.0023867

```
# 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.
```

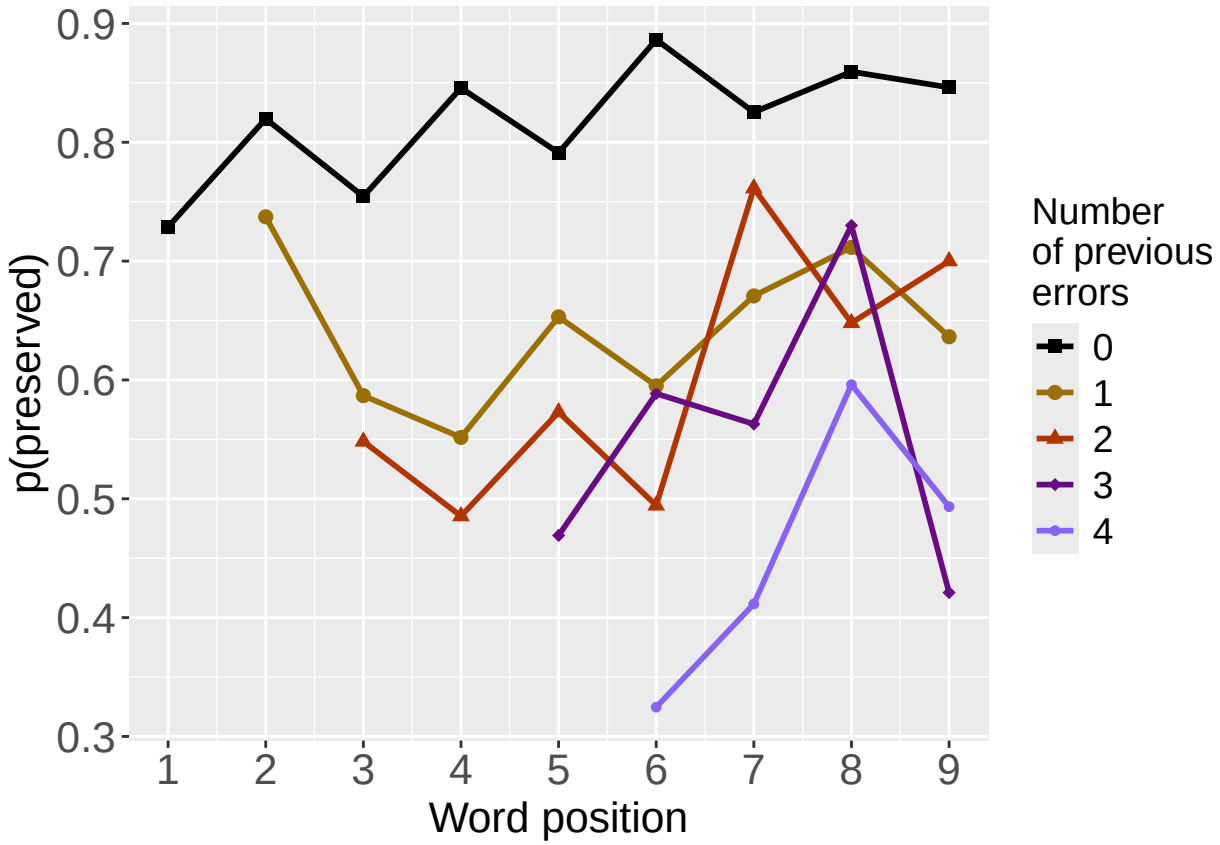
```
print(PrevCorPlot)
```




```
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
```

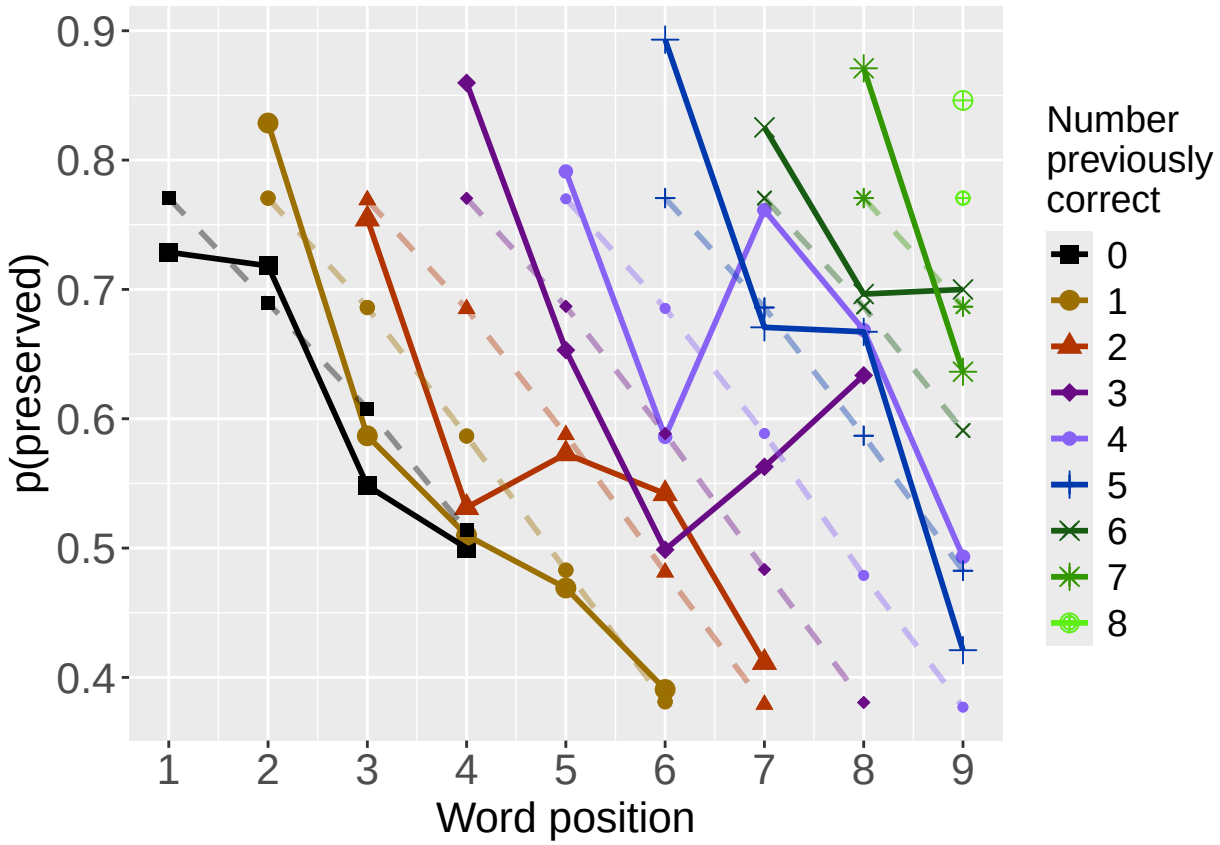
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

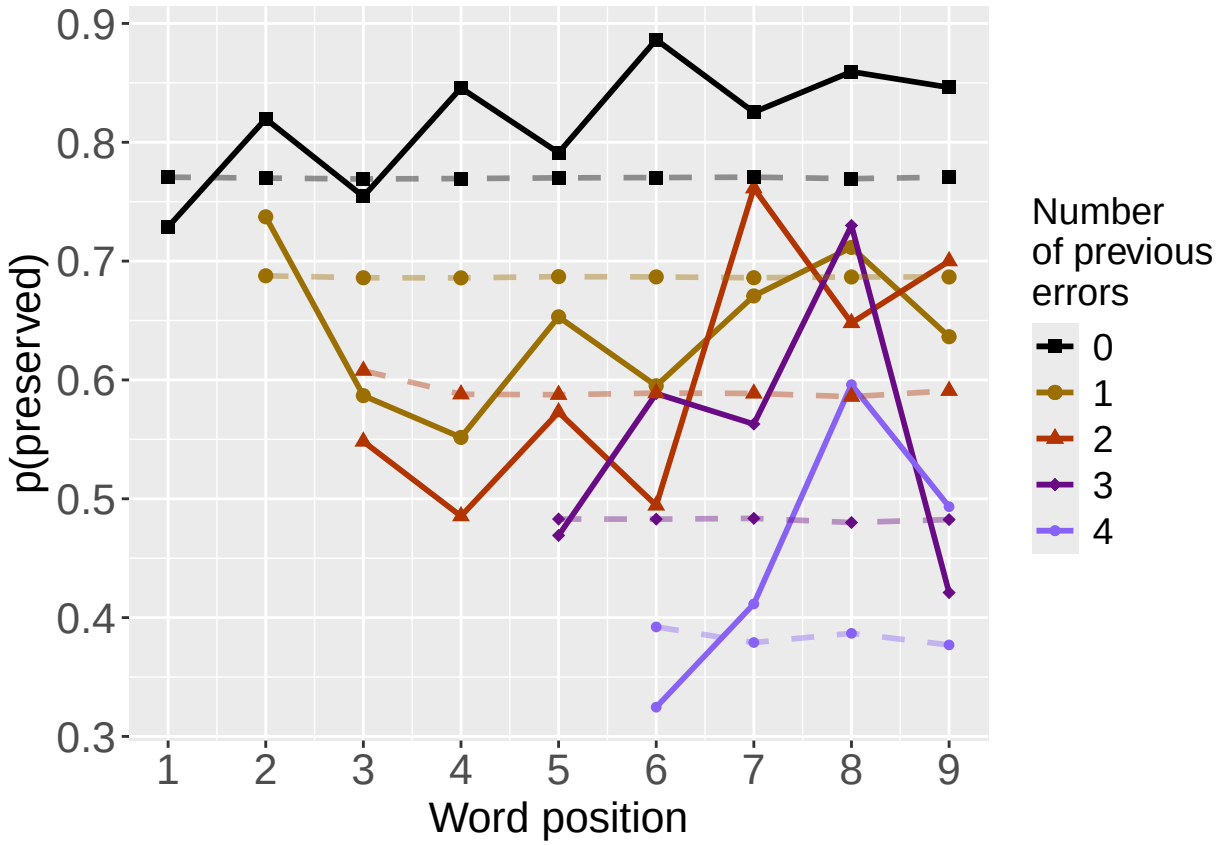
```
print(PrevCorPlot)
```




```
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
##     1.17116     -0.53186      0.01437     -0.03924
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4535  AIC: 5063
## log likelihood:  -2267.553
## Nagelkerke R2:  0.0755056
## % pres/err predicted correctly:  -1577.68
## % of predictable range [ (model-null)/(1-null) ]:  0.06418395
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.2118      -0.4273
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4556  AIC: 5092
## log likelihood:  -2278.152
## Nagelkerke R2:  0.06843933
## % pres/err predicted correctly:  -1588.963
## % of predictable range [ (model-null)/(1-null) ]:  0.05749597
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.304963     0.009837    -0.168274
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4726  AIC: 5305
## log likelihood:  -2363.24
## Nagelkerke R2:  0.01040948
## % pres/err predicted correctly:  -1671.477
## % of predictable range [ (model-null)/(1-null) ]:  0.008582886
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	5063.333	0.00000	1e+00	0.9999993	0.0755056	1.171160	-0.5318562	0.0143675	-0.0392385

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	5091.563	28.22963	7e-07	0.0000007	0.0684393	1.211822	-0.4273043	NA	NA
preserved ~ I(pos^2) + pos	5304.736	241.40290	0e+00	0.0000000	0.0104095	1.304963	NA	0.0098370	-0.1682743

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     0.79644     -0.44789      0.05618
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4549  AIC: 5081
## log likelihood:  -2274.625
## Nagelkerke R2:  0.07079448
## % pres/err predicted correctly:  -1585.122
## % of predictable range [ (model-null)/(1-null) ]:  0.05977287
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.2118      -0.4273
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4556  AIC: 5092
## log likelihood:  -2278.152
## Nagelkerke R2:  0.06843933
## % pres/err predicted correctly:  -1588.963
## % of predictable range [ (model-null)/(1-null) ]:  0.05749597
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      0.9581      -0.0152
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5336
## log likelihood:  -2377.858
## Nagelkerke R2:  0.0002026833
## % pres/err predicted correctly:  -1685.674
## % of predictable range [ (model-null)/(1-null) ]:  0.0001668072
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	5081.252	0.00000	1.0000000	0.9942663	0.0707945	0.7964395	-0.4478863	0.0561779
preserved ~ CumErr	5091.563	10.31129	0.0057668	0.0057337	0.0684393	1.2118220	-0.4273043	NA
preserved ~ stimlen	5336.086	254.83480	0.0000000	0.0000000	0.0002027	0.9580859	NA	-0.0151990

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     1.05399     -0.44415      0.08478
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
```

```
## Null Deviance:       4756
## Residual Deviance: 4539  AIC: 5068
## log likelihood:  -2269.564
## Nagelkerke R2:  0.07416737
## % pres/err predicted correctly:  -1580.571
## % of predictable range [ (model-null)/(1-null) ]:  0.06247005
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.2118      -0.4273
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4556  AIC: 5092
## log likelihood:  -2278.152
## Nagelkerke R2:  0.06843933
## % pres/err predicted correctly:  -1588.963
## % of predictable range [ (model-null)/(1-null) ]:  0.05749597
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.7440       0.0474
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4750  AIC: 5326
## log likelihood:  -2375.147
## Nagelkerke R2:  0.002101338
## % pres/err predicted correctly:  -1682.244
## % of predictable range [ (model-null)/(1-null) ]:  0.002200455
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	5067.996	0.00000	1.0e+00	0.9999924	0.0741674	1.0539878	-0.4441492	0.0847773
preserved ~ CumErr	5091.563	23.56713	7.6e-06	0.0000076	0.0684393	1.2118220	-0.4273043	NA
preserved ~ CumPres	5326.011	258.01518	0.0e+00	0.0000000	0.0021013	0.7439771	NA	0.0473979

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     0.96921     -0.52893      0.08478
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4539  AIC: 5068
## log likelihood:  -2269.564
## Nagelkerke R2:  0.07416737
## % pres/err predicted correctly:  -1580.571
## % of predictable range [ (model-null)/(1-null) ]:  0.06247005
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.2118      -0.4273
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4556  AIC: 5092
## log likelihood:  -2278.152
## Nagelkerke R2:  0.06843933
## % pres/err predicted correctly:  -1588.963
## % of predictable range [ (model-null)/(1-null) ]:  0.05749597
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     1.16224     -0.08178
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4729  AIC: 5306
## log likelihood:  -2364.253
## Nagelkerke R2:  0.009704784
## % pres/err predicted correctly:  -1673.087
## % of predictable range [ (model-null)/(1-null) ]:  0.007628175
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	5067.996	0.00000	1.0e+00	0.9999924	0.0741674	0.9692104	-0.5289266	0.0847773
preserved ~ CumErr	5091.563	23.56713	7.6e-06	0.0000076	0.0684393	1.2118220	-0.4273043	NA
preserved ~ pos	5306.402	238.40582	0.0e+00	0.0000000	0.0097048	1.1622445	NA	-0.0817768

```
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	5063.330	0.00000	1.0000000	0.9999999	0.0755056	1.1711599	-0.5318562	0.0143675	-0.0392385	NA	NA
preserved ~ CumErr + pos	5067.996	0.00000	1.0000000	0.9999924	0.0741674	0.9692104	-0.5289266	NA	0.0847773	NA	NA
preserved ~ CumErr + CumPres	5067.996	0.00000	1.0000000	0.9999924	0.0741674	0.0539878	-0.4441492	NA	NA	NA	0.0847773
preserved ~ CumErr + stimlen	5081.252	0.00000	1.0000000	0.9994266	0.0707945	7.7964395	-0.4478863	NA	NA	0.0561779	NA
preserved ~ CumErr	5091.563	28.22963	0.0000007	0.0000007	0.0684393	1.2118220	-0.4273043	NA	NA	NA	NA
preserved ~ CumErr	5091.563	0.31129	0.0057668	0.0057337	0.0684393	1.2118220	-0.4273043	NA	NA	NA	NA
preserved ~ CumErr	5091.563	23.56713	0.0000076	0.0000076	0.0684393	1.2118220	-0.4273043	NA	NA	NA	NA
preserved ~ CumErr	5091.563	23.56713	0.0000076	0.0000076	0.0684393	1.2118220	-0.4273043	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	5304.730	241.40299	0.0000000	0.0000000	0.0104095	1.3049631	NA	0.0098370	-0.1682743	NA	NA
preserved ~ pos	5306.402	238.40582	0.0000000	0.0000000	0.0097048	1.1622445	NA	NA	-0.0817768	NA	NA
preserved ~ CumPres	5326.012	258.01518	0.0000000	0.0000000	0.0021013	7.7439771	NA	NA	NA	NA	0.0473979
preserved ~ stimlen	5336.082	268.34830	0.0000000	0.0000000	0.0002027	7.9580859	NA	NA	NA	-0.0151990	NA

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     1.00584     -0.53106      0.01314     -0.03596      0.02283
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4219 Residual
## Null Deviance:       4756
## Residual Deviance: 4534  AIC: 5063
## log likelihood:  -2267.049
## Nagelkerke R2:  0.07584095
## % pres/err predicted correctly:  -1577.061
## % of predictable range [ (model-null)/(1-null) ]:  0.06455103
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     1.17116     -0.53186      0.01437     -0.03924
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4535  AIC: 5063
## log likelihood:  -2267.553
## Nagelkerke R2:  0.0755056
## % pres/err predicted correctly:  -1577.68
## % of predictable range [ (model-null)/(1-null) ]:  0.06418395
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     1.17378     -0.53061      0.01420     -0.03931     -0.01119
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4219 Residual
## Null Deviance:       4756
## Residual Deviance: 4535  AIC: 5064
## log likelihood:  -2267.364
## Nagelkerke R2:  0.07563123
## % pres/err predicted correctly:  -1577.515
## % of predictable range [ (model-null)/(1-null) ]:  0.06428173
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     1.02357     -0.53036      0.01315     -0.03633      0.02062     -0.00701
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4218 Residual
## Null Deviance:       4756
## Residual Deviance: 4534  AIC: 5065
## log likelihood:  -2266.979
## Nagelkerke R2:  0.07588705
## % pres/err predicted correctly:  -1577.013
## % of predictable range [ (model-null)/(1-null) ]:  0.06457971
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##      0.8406
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4223 Residual
## Null Deviance:       4756
## Residual Deviance: 4756  AIC: 5334
## log likelihood:  -2378.147
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1685.956
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen	5063.13	1.000000	1.000000	0.3504370	0.0758410	0.005844	- 0.5310629	0.0131363	- 0.0359578	NA	0.0228347
preserved ~ CumErr + I(pos^2) + pos	5063.33	1.202042	1.703916	0.3167648	0.0755056	0.171160	- 0.5318562	0.0143675	- 0.0392385	NA	NA
preserved ~ CumErr + I(pos^2) + pos + log_freq	5064.45	2.327050	3.515032	0.1804865	0.0756312	0.173779	- 0.5306102	0.0141997	- 0.0393142	- 0.0111921	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	5064.79	1.666503	7.434636	0.1523107	0.0758871	1.023566	- 0.5303564	0.0131508	- 0.0363283	- 0.0070100	0.0206156
preserved ~ 1	5333.77	270.645	1.84e-60	0.0000000	0.0000000	0.840646	NA	NA	NA	NA	NA

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

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

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

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

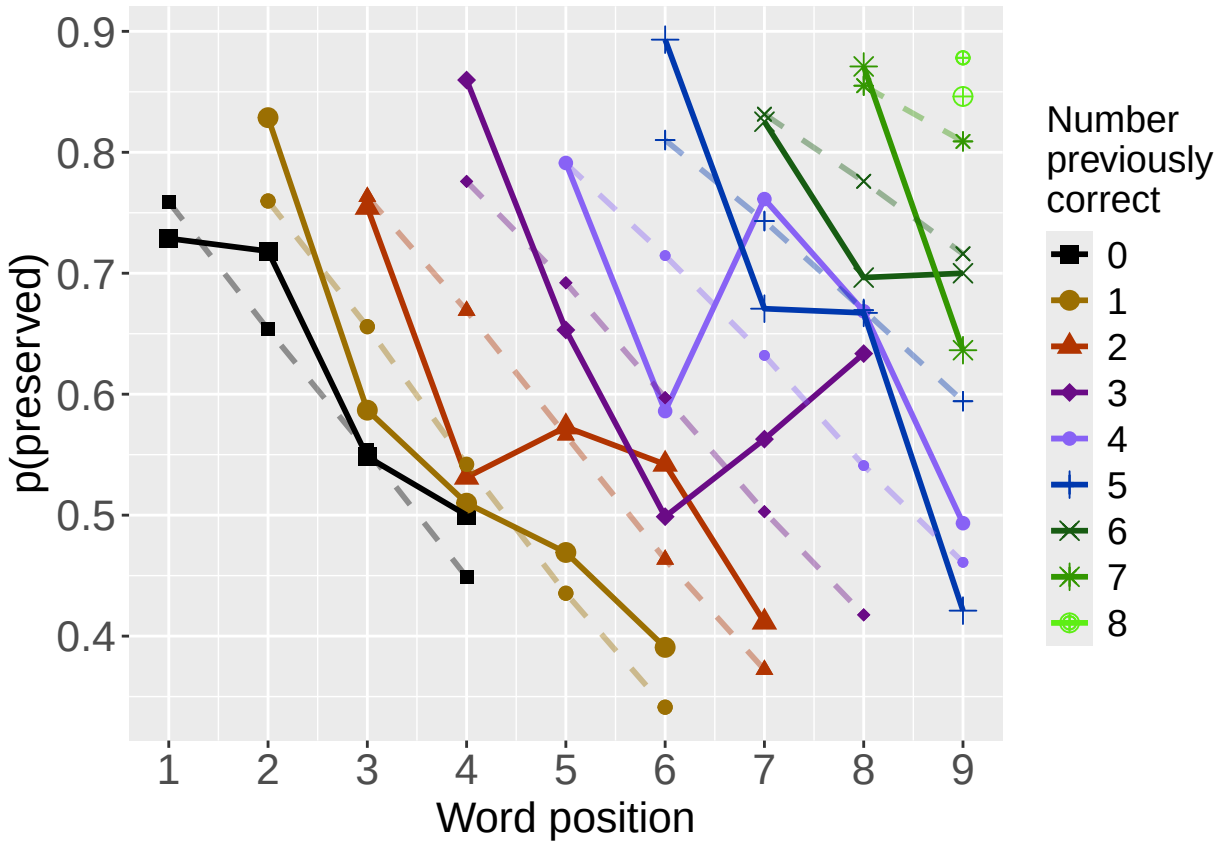
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen
##          Df Deviance    AIC
## CumErr    1   4724.9 5251.9
## I(pos^2)  1   4537.4 5064.4
## <none>        4534.1 5063.1
## stimlen   1   4535.1 5062.1
## pos       1   4534.4 5061.4
```

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

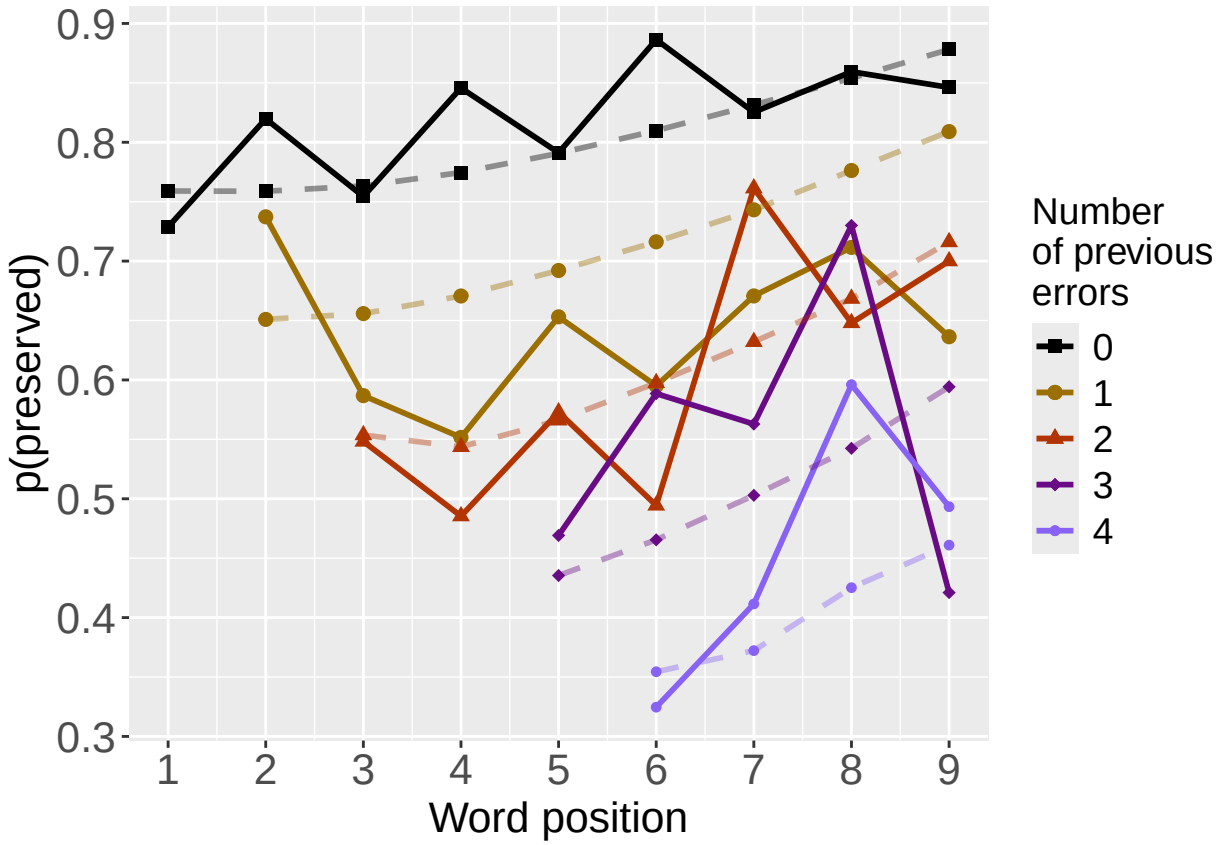
```
print(PrevCorPlot)
```




```
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!
```

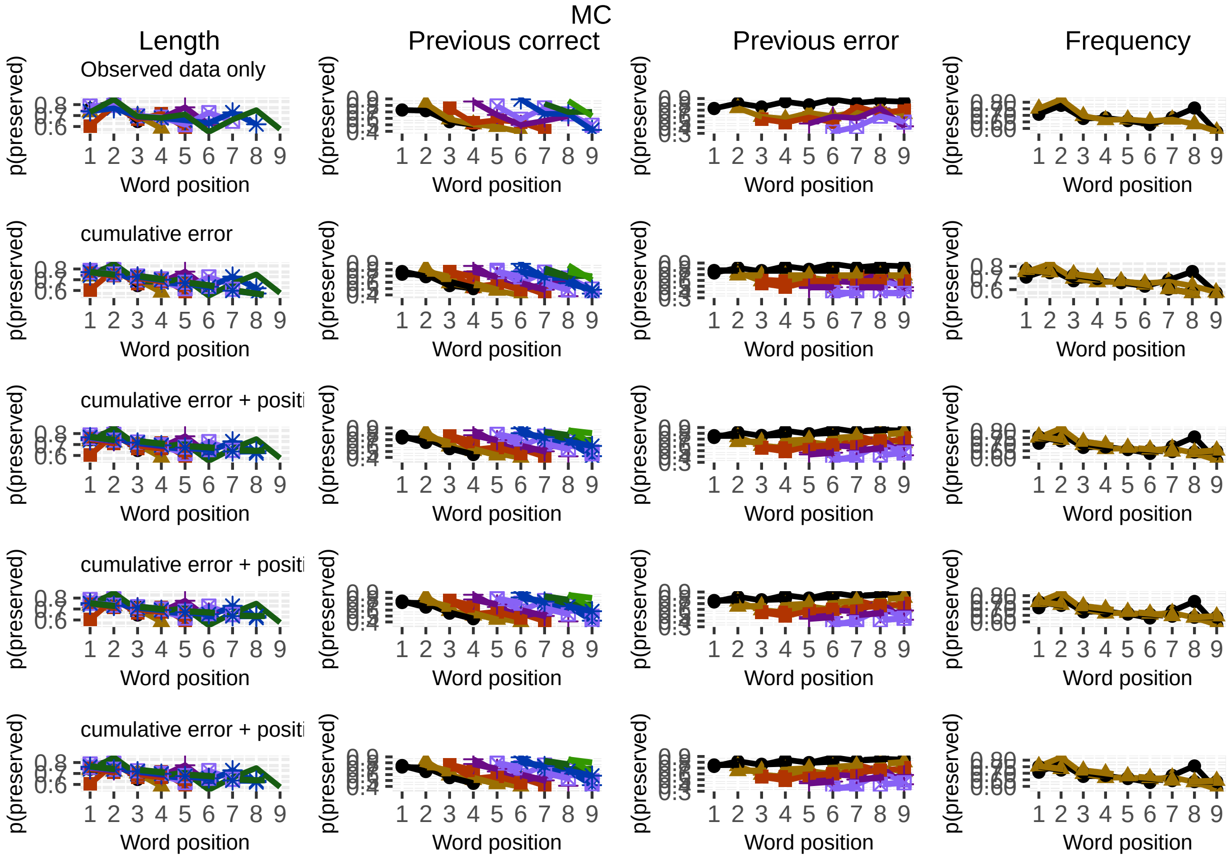
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.2118      -0.4273
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4222 Residual
## Null Deviance:       4756
## Residual Deviance: 4556  AIC: 5092
## log likelihood:  -2278.152
```

```
## Nagelkerke R2:  0.06843933
## % pres/err predicted correctly:  -1588.963
## % of predictable range [ (model-null)/(1-null) ]:  0.05749597
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)      stimlen          pos
##     1.00584     -0.53106      0.01314      0.02283     -0.03596
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4219 Residual
## Null Deviance:       4756
## Residual Deviance: 4534  AIC: 5063
## log likelihood:  -2267.049
## Nagelkerke R2:  0.07584095
## % pres/err predicted correctly:  -1577.061
## % of predictable range [ (model-null)/(1-null) ]:  0.06455103
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)
##     1.10240     -0.53576      0.01028
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4221 Residual
## Null Deviance:       4756
## Residual Deviance: 4535  AIC: 5062
## log likelihood:  -2267.733
## Nagelkerke R2:  0.07538572
## % pres/err predicted correctly:  -1578.131
## % of predictable range [ (model-null)/(1-null) ]:  0.06391702
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)      stimlen
##    0.938540    -0.534608     0.009368     0.023447
##
## Degrees of Freedom: 4223 Total (i.e. Null);  4220 Residual
## Null Deviance:       4756
## Residual Deviance: 4534  AIC: 5062
## log likelihood:  -2267.199
## Nagelkerke R2:  0.07574059
## % pres/err predicted correctly:  -1577.42
## % of predictable range [ (model-null)/(1-null) ]:  0.06433808
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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




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

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

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

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

	CumErr	I(pos^2)	pos	stimlen
McFadden	0.0460337	0.0027526	0.0029272	0.0005477
SquaredCorrelation	0.0561706	0.0033773	0.0036303	0.0006602
Nagelkerke	0.0561706	0.0033773	0.0036303	0.0006602
Estrella	0.0576904	0.0034533	0.0036809	0.0006845

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

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

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

```
##                                                          model deviance
## CumErr + I(pos^2) + stimlen + pos CumErr + I(pos^2) + stimlen + pos 4534.097
## CumErr + I(pos^2) + stimlen             CumErr + I(pos^2) + stimlen 4534.399
## CumErr + I(pos^2)                                 CumErr + I(pos^2) 4535.466
## CumErr                                                       CumErr 4556.304
## null                                                           null 4756.295
##                                   deviance_explained percent_explained
## CumErr + I(pos^2) + stimlen + pos           222.1978          4.671658
## CumErr + I(pos^2) + stimlen                 221.8959          4.665310
## CumErr + I(pos^2)                           220.8286          4.642871
## CumErr                                      199.9906          4.204757
## null                                          0.0000          0.000000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + I(pos^2) + stimlen + pos                     100.00000              0.1358743
## CumErr + I(pos^2) + stimlen                            99.86413              0.4803355
## CumErr + I(pos^2)                                      99.38379              9.3781238
## CumErr                                                 90.00567             90.0056663
## null                                                         NA              0.0000000
```

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

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

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

	deviance	deviance_explained
CumErr + I(pos^2) + stimlen + pos	4534.097	222.1978
CumErr + I(pos^2) + stimlen	4534.399	221.8959
CumErr + I(pos^2)	4535.466	220.8286
CumErr	4556.304	199.9906
null	4756.295	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + I(pos^2) + stimlen + pos	4.671658	100.00000	0.1358743
CumErr + I(pos^2) + stimlen	4.665310	99.86413	0.4803355
CumErr + I(pos^2)	4.642871	99.38379	9.3781238
CumErr	4.204757	90.00567	90.0056663
null	0.000000	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   0.87988849
## I(pos^2) 0.05290381
## pos      0.05686639
## stimlen  0.01034130
```

```
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	0.6587968	4556.304
preserved ~ CumErr+I(pos^2)	0.6951672	4535.466
preserved ~ CumErr+I(pos^2)+stimlen	0.6966657	4534.399
preserved ~ CumErr+I(pos^2)+stimlen+pos	0.6973127	4534.097

```
## 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       0.6587968       4556.304  0.00000000
## 2            preserved ~ CumErr+I(pos^2)       0.6951672       4535.466  0.03637045
## 3    preserved ~ CumErr+I(pos^2)+stimlen       0.6966657       4534.399  0.03786894
## 4 preserved ~ CumErr+I(pos^2)+stimlen+pos       0.6973127       4534.097  0.03851591
##   diff_CumErr+I(pos^2) diff_CumErr+I(pos^2)+stimlen diff_CumErr+I(pos^2)+stimlen+pos
## 1         -0.036370450                -0.0378689392                    -0.0385159128
## 2          0.000000000                -0.0014984890                    -0.0021454626
## 3          0.001498489                 0.0000000000                    -0.0006469736
## 4          0.002145463                 0.0006469736                     0.0000000000
```

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

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

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

model	diff_CumErr	diff_CumErr+I(pos^2)	diff_CumErr+I(pos^2)+stimlen
preserved ~ CumErr	0.0000000	-0.0363705	-0.0378689
preserved ~ CumErr+I(pos^2)	0.0363705	0.0000000	-0.0014985
preserved ~ CumErr+I(pos^2)+stimlen	0.0378689	0.0014985	0.0000000
preserved ~ CumErr+I(pos^2)+stimlen+pos	0.0385159	0.0021455	0.0006470