PM - repetition - Serial position analysis (v5)

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

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

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

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

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

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	440	26	108	NA	NA	574
2	56	NA	353	78	87	574
3	255	NA	141	165	13	574
4	247	NA	192	51	28	518
5	182	NA	161	58	34	435
6	162	NA	99	52	17	330
7	123	NA	71	23	16	233
8	69	NA	28	15	3	115
9	41	NA	1	NA	6	48

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7665505	0.0452962	0.1881533	NA	NA	574
2	0.0975610	NA	0.6149826	0.1358885	0.1515679	574
3	0.4442509	NA	0.2456446	0.2874564	0.0226481	574
4	0.4768340	NA	0.3706564	0.0984556	0.0540541	518
5	0.4183908	NA	0.3701149	0.1333333	0.0781609	435
6	0.4909091	NA	0.3000000	0.1575758	0.0515152	330

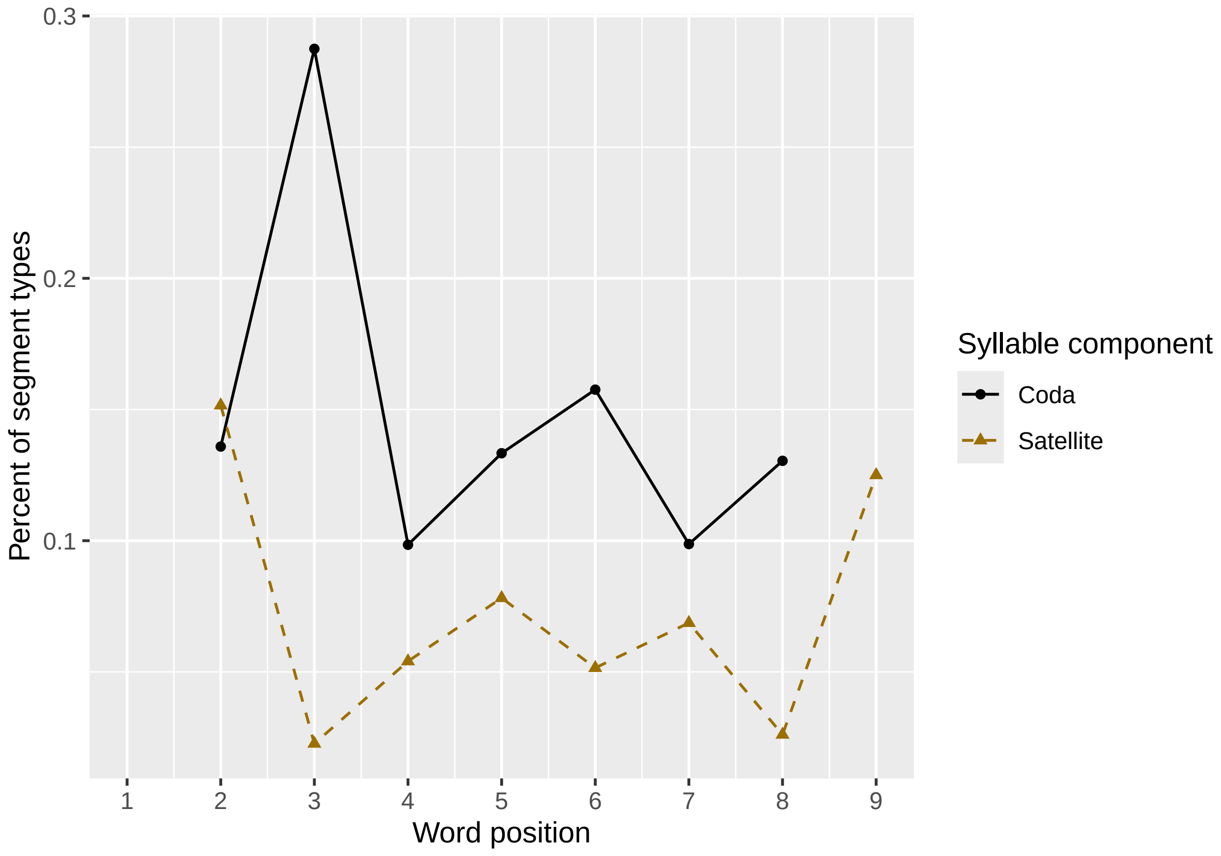
pos_factor	O	P	V	1	S	total
7	0.5278970	NA	0.3047210	0.0987124	0.0686695	233
8	0.6000000	NA	0.2434783	0.1304348	0.0260870	115
9	0.8541667	NA	0.0208333	NA	0.1250000	48

```r
# 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()`).
```

```r
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 0.821 0.929 0.857 NA     NA     NA     NA     NA     NA
## 2       5 0.944 0.825 0.859  0.884 NA     NA     NA     NA     NA
## 3       6 0.910 0.852 0.793  0.893  0.795 NA     NA     NA     NA
## 4       7 0.948 0.918 0.861  0.752  0.802  0.774 NA     NA     NA
## 5       8 0.921 0.896 0.832  0.778  0.783  0.774  0.715 NA     NA
## 6       9 0.821 0.861 0.739  0.649  0.710  0.703  0.642  0.693 NA
## 7      10 0.667 0.861 0.764  0.722  0.726  0.816  0.712  0.755  0.674
```

```
# 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.
```

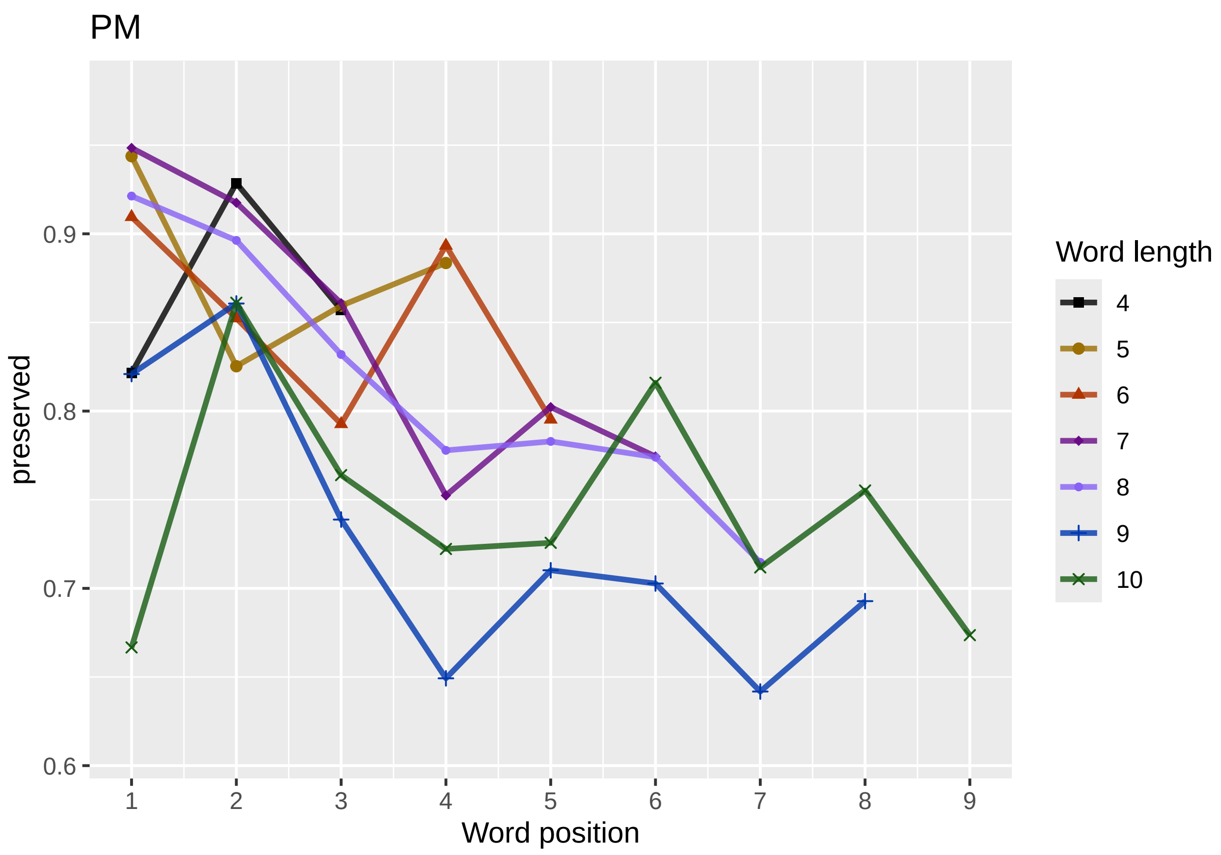
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    56    56    56    NA    NA    NA    NA    NA    NA
## 2       5    83    83    83    83    NA    NA    NA    NA    NA
## 3       6   105   105   105   105   105    NA    NA    NA    NA
## 4       7    97    97    97    97    97    97    NA    NA    NA
## 5       8   118   118   118   118   118   118   118    NA    NA
## 6       9    67    67    67    67    67    67    67    67    NA
## 7      10    48    48    48    48    48    48    48    48    48
```

```
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
##     3.29782     -0.12876      0.02153     -0.32714
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2942  AIC: 3386
## log likelihood:  -1470.969
## Nagelkerke R2:  0.04590758
## % pres/err predicted correctly:  -971.0897
## % of predictable range [ (model-null)/(1-null) ]:  0.03569902
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         3.241967         -0.127208          0.001764         -0.231037          0.001942
##      stimlen:pos
##        -0.008311
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3395 Residual
## Null Deviance:       3035
## Residual Deviance: 2942  AIC: 3390
## log likelihood:  -1470.866
## Nagelkerke R2:  0.0460073
## % pres/err predicted correctly:  -971.0016
## % of predictable range [ (model-null)/(1-null) ]:  0.03578636
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.56803     -0.19912     -0.34381      0.02421
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2944  AIC: 3391
## log likelihood:  -1472.157
## Nagelkerke R2:  0.0447556
## % pres/err predicted correctly:  -972.1658
## % of predictable range [ (model-null)/(1-null) ]:  0.0346315
## ***************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.8838      -0.1156      -0.1393
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2947  AIC: 3394
## log likelihood:  -1473.546
## Nagelkerke R2:  0.04340774
## % pres/err predicted correctly:  -973.0329
## % of predictable range [ (model-null)/(1-null) ]:  0.03377131
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.37560      0.01415     -0.30415
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2959  AIC: 3407
## log likelihood:  -1479.697
## Nagelkerke R2:  0.03742697
## % pres/err predicted correctly:  -978.168
## % of predictable range [ (model-null)/(1-null) ]:  0.0286774
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1582      -0.1779
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2962  AIC: 3411
## log likelihood:  -1480.85
## Nagelkerke R2:  0.03630383
## % pres/err predicted correctly:  -978.8792
## % of predictable range [ (model-null)/(1-null) ]:  0.02797191
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      2.8949      -0.1894
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2986  AIC: 3438
## log likelihood:  -1492.796
## Nagelkerke R2:  0.02461889
## % pres/err predicted correctly:  -987.407
## % of predictable range [ (model-null)/(1-null) ]:  0.01951243
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.454
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3400 Residual
## Null Deviance:       3035
## Residual Deviance: 3035  AIC: 3498
## log likelihood:  -1517.692
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1007.077
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	3386.139	0.000000	1.0000000	0.7654569	0.0459076	3.6297815	-0.1287596	-0.3271412	NA	0.0215319	NA
preserved ~ stimlen * (I(pos^2) + pos)	3389.579	3.440119	0.1790555	0.1370593	0.0460073	3.3241967	-0.1272078	-0.2310367	-0.0083106	0.0017640	0.0019418

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	3390.58	6.446835	0.108238	0.0828520	0.044755	3.568030	-0.1991223	-0.3438085	0.0242143	NA	NA
preserved ~ stimlen + pos	3394.05	7.917512	0.0190868	0.0146102	0.043407	2.883832	-0.1156165	-0.1392558	NA	NA	NA
preserved ~ I(pos^2) + pos	3407.45	21.320491	0.0000235	0.0000180	0.037427	2.0375599	NA	-0.3041456	NA	0.0141547	NA
preserved ~ pos	3410.59	24.459236	0.0000049	0.0000037	0.036303	3.158226	NA	-0.1778709	NA	NA	NA
preserved ~ stimlen	3438.22	52.090658	0.0000000	0.0000000	0.024618	2.894937	-0.1894312	NA	NA	NA	NA
preserved ~ 1	3497.81	111.676238	0.0000000	0.0000000	0.0000000	1.454378	NA	NA	NA	NA	NA

```r
print(BestLPModelFormula)
```

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

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.29782     -0.12876      0.02153     -0.32714
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2942  AIC: 3386
```

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

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

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

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

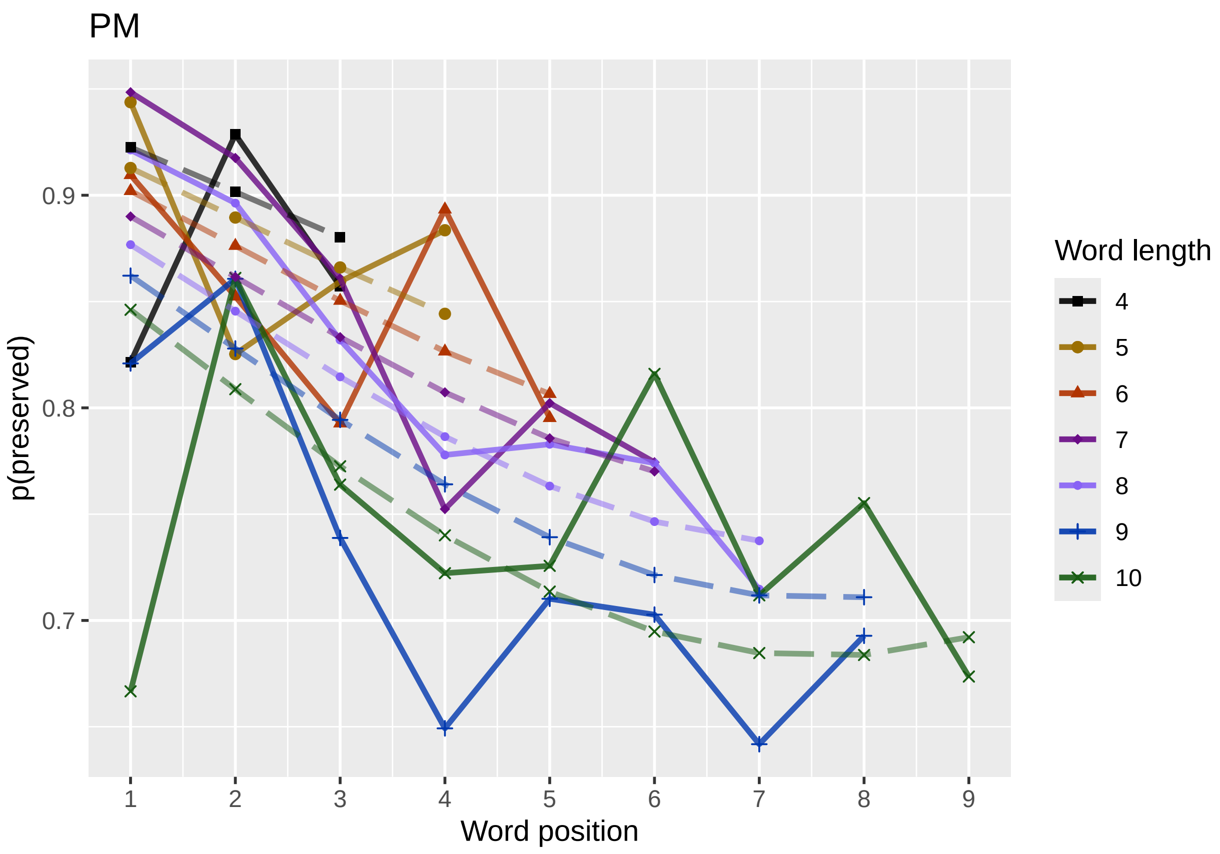
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.923 0.902 0.880 NA     NA     NA     NA     NA     NA
## 2       5 0.913 0.890 0.866  0.844 NA     NA     NA     NA     NA
## 3       6 0.902 0.876 0.850  0.827  0.807 NA     NA     NA     NA
## 4       7 0.890 0.862 0.833  0.807  0.786  0.770 NA     NA     NA
## 5       8 0.877 0.845 0.815  0.786  0.763  0.747  0.737 NA     NA
## 6       9 0.862 0.828 0.794  0.764  0.739  0.721  0.712  0.711 NA
## 7      10 0.846 0.809 0.773  0.740  0.714  0.695  0.685  0.684  0.692
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       10   574
```

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

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

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

```
## [1] "The number of responses with fragments was 10 / 574 = 1.74 percent"
```

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

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

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

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

```
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.34058     -0.12936      0.02705     -0.35905
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3379 Residual
## Null Deviance:       2978
## Residual Deviance: 2897  AIC: 3342
## log likelihood:  -1448.392
## Nagelkerke R2:  0.04049037
## % pres/err predicted correctly:  -953.6774
## % of predictable range [ (model-null)/(1-null) ]:  0.0316524
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         3.176235         -0.114289         -0.001085         -0.195970          0.002931
##      stimlen:pos
##        -0.016431
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3377 Residual
## Null Deviance:       2978
## Residual Deviance: 2897  AIC: 3345
## log likelihood:  -1448.256
## Nagelkerke R2:  0.04062446
## % pres/err predicted correctly:  -953.5083
## % of predictable range [ (model-null)/(1-null) ]:  0.03182387
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.62533     -0.21052     -0.36441      0.02842
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3379 Residual
## Null Deviance:       2978
## Residual Deviance: 2901  AIC: 3349
## log likelihood:  -1450.409
## Nagelkerke R2:  0.03850074
## % pres/err predicted correctly:  -955.4038
## % of predictable range [ (model-null)/(1-null) ]:  0.02990126
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.8312      -0.1134      -0.1249
```

```
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3380 Residual
## Null Deviance:       2978
## Residual Deviance: 2905  AIC: 3354
## log likelihood:  -1452.264
## Nagelkerke R2:  0.03666811
## % pres/err predicted correctly:  -956.5669
## % of predictable range [ (model-null)/(1-null) ]:  0.02872151
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.41325      0.01956     -0.33535
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3380 Residual
## Null Deviance:       2978
## Residual Deviance: 2914  AIC: 3363
## log likelihood:  -1457.155
## Nagelkerke R2:  0.03182738
## % pres/err predicted correctly:  -960.7747
## % of predictable range [ (model-null)/(1-null) ]:  0.02445344
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1183      -0.1624
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3381 Residual
## Null Deviance:       2978
## Residual Deviance: 2918  AIC: 3369
## log likelihood:  -1459.247
## Nagelkerke R2:  0.02975229
## % pres/err predicted correctly:  -962.1357
## % of predictable range [ (model-null)/(1-null) ]:  0.02307293
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8341      -0.1782
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3381 Residual
## Null Deviance:       2978
## Residual Deviance: 2935  AIC: 3389
## log likelihood:  -1467.332
## Nagelkerke R2:  0.02170899
## % pres/err predicted correctly:  -967.7863
## % of predictable range [ (model-null)/(1-null) ]:  0.01734148
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.482
##
## Degrees of Freedom: 3382 Total (i.e. Null);  3382 Residual
## Null Deviance:       2978
## Residual Deviance: 2978  AIC: 3441
## log likelihood:  -1488.964
## Nagelkerke R2:  -3.793521e-16
## % pres/err predicted correctly:  -984.8829
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	3342.019	0.000000	1.0000000	0.8079610	0.0404903	1.4340577	-0.1293585	-0.3590491	NA	0.0270510	NA
preserved ~ stimlen * (I(pos^2) + pos)	3345.243	3.224782	0.1994102	0.1611158	0.0406243	1.5176235	-0.1142893	-0.1959703	-0.0164306	-0.0010851	0.0029314
preserved ~ stimlen * pos	3348.716	6.691622	0.0352310	0.0284658	0.0385003	1.7625328	-0.2105223	-0.3644130	0.0284248	NA	NA
preserved ~ stimlen + pos	3353.627	11.607601	0.0030161	0.0024369	0.0366682	1.1831212	-0.1134033	-0.1248867	NA	NA	NA
preserved ~ I(pos^2) + pos	3363.354	21.334255	0.0000233	0.0000188	0.0318272	1.4413246	NA	-0.3353490	NA	0.0195586	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ pos	3369.382	27.364056	0.0000011	0.0000009	0.0297523	2.118296	NA	-0.1624178	NA	NA	NA
preserved ~ stimlen	3388.864	46.843867	0.0000000	0.0000000	0.0217092	2.834105	-0.1781982	NA	NA	NA	NA
preserved ~ 1	3441.162	99.142813	0.0000000	0.0000000	0.0000000	1.481750	NA	NA	NA	NA	NA

```r
# plot no fragment data

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

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

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

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

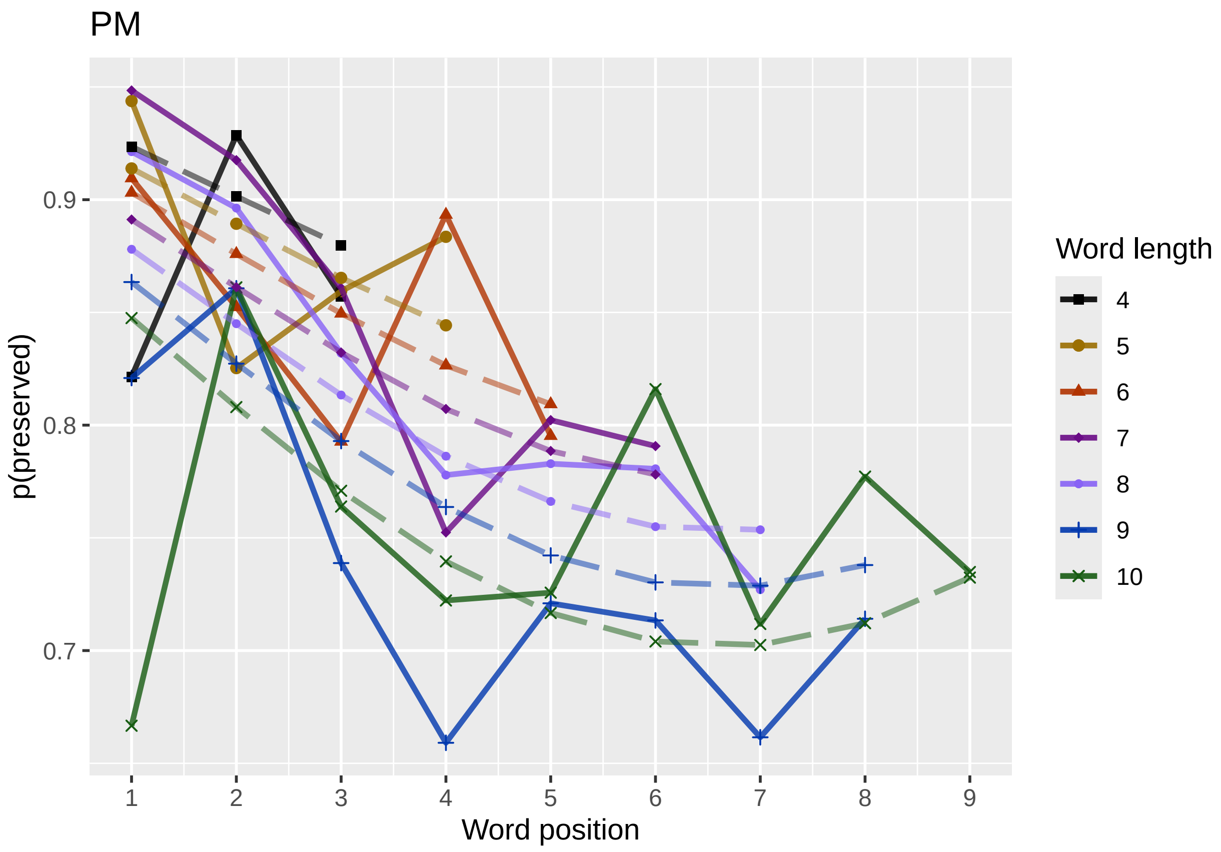
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.924 0.901 0.880 NA     NA     NA     NA     NA     NA
## 2       5 0.914 0.889 0.865  0.844 NA     NA     NA     NA     NA
## 3       6 0.903 0.876 0.850  0.827  0.809 NA     NA     NA     NA
## 4       7 0.891 0.861 0.832  0.807  0.789  0.778 NA     NA     NA
## 5       8 0.878 0.845 0.813  0.786  0.766  0.755  0.754 NA     NA
## 6       9 0.863 0.827 0.793  0.764  0.742  0.730  0.729  0.738 NA
## 7      10 0.847 0.808 0.771  0.740  0.717  0.704  0.702  0.712  0.732
```

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

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

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

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



back to full data

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

```
## [1] "Min/max preserved range: 0.61 - 0.98"
```

```
# 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.
```

```r
# 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: "
```

```r
print(OA_mean_len_diff)
```

```
## [1] -0.01785127
```

```r
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): "
```

```r
print(OA_mean_pos_diff)
```

```
## [1] -0.01932699
```

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

```r
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] "Average upward change after U minimum"
## [1] 0.008350749
```

```
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] "differences from left max to min for each row: "
## [1] 0.04223973 0.06858839 0.09543426 0.11992822 0.13930254 0.15123594 0.16237069
## [1] "differences from min to right max for each row: "
## [1] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.008350749
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.1623707
## [1] 0.008350749
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.05143015
```

```
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:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           3.28242           -0.12586            0.02258           -0.33158           -0.08514
## I(pos^2):log_freq       pos:log_freq
##          -0.00237            0.04117
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3394 Residual
## Null Deviance:       3035
## Residual Deviance: 2936  AIC: 3386
## log likelihood:  -1468.229
## Nagelkerke R2:  0.04856004
## % pres/err predicted correctly:  -969.116
## % of predictable range [ (model-null)/(1-null) ]:  0.03765682
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.29782     -0.12876      0.02153     -0.32714
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2942  AIC: 3386
## log likelihood:  -1470.969
## Nagelkerke R2:  0.04590758
## % pres/err predicted correctly:  -971.0897
## % of predictable range [ (model-null)/(1-null) ]:  0.03569902
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          3.249940          -0.123577           0.051981           0.022617          -0.330207
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.018734          -0.001425           0.038467
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3393 Residual
## Null Deviance:       3035
## Residual Deviance: 2935  AIC: 3386
## log likelihood:  -1467.615
## Nagelkerke R2:  0.04915429
## % pres/err predicted correctly:  -968.4926
## % of predictable range [ (model-null)/(1-null) ]:  0.03827529
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     3.23193     -0.12010      0.02132     -0.32531      0.03656
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3396 Residual
## Null Deviance:       3035
## Residual Deviance: 2940  AIC: 3386
## log likelihood:  -1469.851
## Nagelkerke R2:  0.04699019
## % pres/err predicted correctly:  -970.1498
## % of predictable range [ (model-null)/(1-null) ]:  0.03663131
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         3.219024         -0.119148          0.081132          0.021166         -0.323973
## stimlen:log_freq
##        -0.005679
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3395 Residual
## Null Deviance:       3035
## Residual Deviance: 2940  AIC: 3388
## log likelihood:  -1469.782
## Nagelkerke R2:  0.04705762
## % pres/err predicted correctly:  -970.0764
## % of predictable range [ (model-null)/(1-null) ]:  0.03670413
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         3.241967         -0.127208          0.001764         -0.231037          0.001942
##       stimlen:pos
##        -0.008311
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3395 Residual
## Null Deviance:       3035
## Residual Deviance: 2942  AIC: 3390
## log likelihood:  -1470.866
## Nagelkerke R2:  0.0460073
## % pres/err predicted correctly:  -971.0016
## % of predictable range [ (model-null)/(1-null) ]:  0.03578636
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.56803     -0.19912     -0.34381      0.02421
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2944  AIC: 3391
## log likelihood:  -1472.157
## Nagelkerke R2:  0.0447556
## % pres/err predicted correctly:  -972.1658
## % of predictable range [ (model-null)/(1-null) ]:  0.0346315
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.80851          -0.10901           0.09731          -0.13307          -0.02002
##     log_freq:pos
##          0.02296
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3395 Residual
## Null Deviance:       3035
## Residual Deviance: 2941  AIC: 3393
## log likelihood:  -1470.515
## Nagelkerke R2:  0.04634745
## % pres/err predicted correctly:  -970.5625
## % of predictable range [ (model-null)/(1-null) ]:  0.03622198
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.84086      -0.11128      -0.13450      -0.03334       0.01675
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3396 Residual
## Null Deviance:       3035
## Residual Deviance: 2942  AIC: 3393
## log likelihood:  -1471.239
## Nagelkerke R2:  0.04564521
## % pres/err predicted correctly:  -971.1698
## % of predictable range [ (model-null)/(1-null) ]:  0.03561956
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.82062     -0.10686     -0.13935      0.03745
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2945  AIC: 3394
## log likelihood:  -1472.373
## Nagelkerke R2:  0.04454604
## % pres/err predicted correctly:  -972.0464
## % of predictable range [ (model-null)/(1-null) ]:  0.0347499
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.8838      -0.1156      -0.1393
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2947  AIC: 3394
## log likelihood:  -1473.546
## Nagelkerke R2:  0.04340774
## % pres/err predicted correctly:  -973.0329
## % of predictable range [ (model-null)/(1-null) ]:  0.03377131
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         log_freq              pos  stimlen:log_freq
##        2.808164        -0.105787         0.092692        -0.139319         -0.007047
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3396 Residual
## Null Deviance:       3035
## Residual Deviance: 2945  AIC: 3396
## log likelihood:  -1472.264
## Nagelkerke R2:  0.04465185
## % pres/err predicted correctly:  -971.9665
## % of predictable range [ (model-null)/(1-null) ]:  0.03482916
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##         2.373792           0.015116          -0.306197          -0.051037          -0.002756
##     pos:log_freq
##         0.039494
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3395 Residual
## Null Deviance:       3035
## Residual Deviance: 2952  AIC: 3406
## log likelihood:  -1476.153
## Nagelkerke R2:  0.04087615
## % pres/err predicted correctly:  -975.3493
## % of predictable range [ (model-null)/(1-null) ]:  0.03147347
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.37560      0.01415     -0.30415
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2959  AIC: 3407
## log likelihood:  -1479.697
## Nagelkerke R2:  0.03742697
## % pres/err predicted correctly:  -978.168
## % of predictable range [ (model-null)/(1-null) ]:  0.0286774
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.14509     -0.17356      0.05347
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2957  AIC: 3408
## log likelihood:  -1478.367
## Nagelkerke R2:  0.03872186
## % pres/err predicted correctly:  -976.7258
## % of predictable range [ (model-null)/(1-null) ]:  0.03010806
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     2.138591     -0.170772     -0.001047      0.013005
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2955  AIC: 3408
## log likelihood:  -1477.687
## Nagelkerke R2:  0.03938415
## % pres/err predicted correctly:  -976.2271
## % of predictable range [ (model-null)/(1-null) ]:  0.03060273
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1582      -0.1779
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2962  AIC: 3411
## log likelihood:  -1480.85
## Nagelkerke R2:  0.03630383
## % pres/err predicted correctly:  -978.8792
## % of predictable range [ (model-null)/(1-null) ]:  0.02797191
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.83264     -0.18085      0.03699
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2983  AIC: 3438
## log likelihood:  -1491.636
## Nagelkerke R2:  0.02575708
## % pres/err predicted correctly:  -986.6357
## % of predictable range [ (model-null)/(1-null) ]:  0.02027754
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8949      -0.1894
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2986  AIC: 3438
## log likelihood:  -1492.796
## Nagelkerke R2:  0.02461889
## % pres/err predicted correctly:  -987.407
## % of predictable range [ (model-null)/(1-null) ]:  0.01951243
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         2.820021         -0.179746          0.093139         -0.007153
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2983  AIC: 3440
## log likelihood:  -1491.522
## Nagelkerke R2:  0.02586865
## % pres/err predicted correctly:  -986.5682
## % of predictable range [ (model-null)/(1-null) ]:  0.0203445
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.454
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3400 Residual
## Null Deviance:       3035
## Residual Deviance: 3035  AIC: 3498
## log likelihood:  -1517.692
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1007.077
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	I(pos^2)	stimlen:pos	log_freq:pos	log_freq:I(pos^2)	log_freq:stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	3386.008	0	1	0.2220	0.0498	3.6282415	-0.1258608	0.0251424	NA	-0.3315814	0.0411653	NA	0.0225813	-0.0023701	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	3386.1[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1287596	NA	NA	-0.3271412	NA	NA	0.0215319	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	3386.1[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1235773	0.0519814	0.0187343	-0.3302070	NA	0.0384[illegible]	[illegible]	NA	-0.0014246	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	3386.2[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1201012	0.0365601	NA	-0.3253111	NA	NA	0.0213155	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	3388.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1191483	0.0811322	-0.005678[illegible]	-0.3239727	NA	NA	0.0211663	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	3389.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1272078	NA	NA	-0.2310367	NA	NA	0.0017643	NA	NA	-0.0083106	0.0019418
preserved ~ stimlen * pos	3390.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1991223	NA	NA	-0.3438085	NA	NA	NA	NA	NA	0.0242143	NA
preserved ~ stimlen * log_freq + pos * log_freq	3392.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1090113	0.0973051	-0.0200[illegible]	-0.[illegible]330749	NA	0.0229576	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	3393.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.111[illegible]	-0.0733367	NA	-0.1345029	0.0167510	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	3394.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1068632	0.0374505	NA	-0.1393527	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	3394.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1156165	NA	NA	-0.1392558	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	3395.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1057866	0.0926923	-0.00704[illegible]	-0.[illegible]393191	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	3405.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0510366	NA	-0.3061972	0.0394942	NA	0.0151164	-0.0027558	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.3041456	NA	NA	0.0141547	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.0534742	NA	-0.1735598	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0010474	NA	-0.1707718	0.0130053	NA	NA	NA	NA	NA	NA
preserved ~ pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.1778709	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1808540	0.0369911	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1894312	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1797461	0.0931391	-0.0071526	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           3.28242           -0.12586            0.02258           -0.33158           -0.08514
## I(pos^2):log_freq       pos:log_freq
##          -0.00237            0.04117
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3394 Residual
## Null Deviance:       3035
## Residual Deviance: 2936  AIC: 3386
```

```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.
```

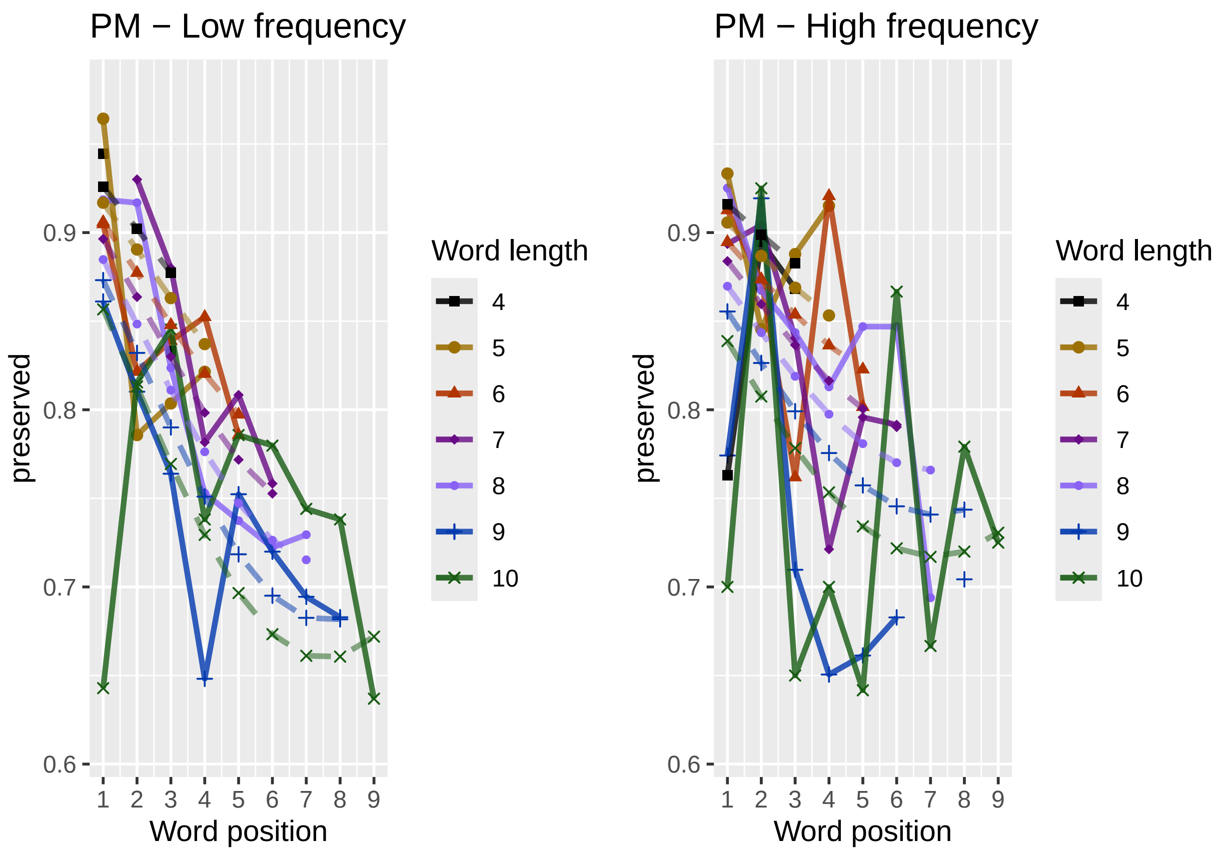
```
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

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

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

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

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

## Warning: Removed 1 row 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
##      1.7749      -0.5448
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2887  AIC: 3309
## log likelihood:  -1443.693
## Nagelkerke R2:  0.07212943
## % pres/err predicted correctly:  -944.7192
## % of predictable range [ (model-null)/(1-null) ]:  0.06185814
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.37560      0.01415     -0.30415
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2959  AIC: 3407
## log likelihood:  -1479.697
## Nagelkerke R2:  0.03742697
## % pres/err predicted correctly:  -978.168
## % of predictable range [ (model-null)/(1-null) ]:  0.0286774
## ***************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1582      -0.1779
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2962  AIC: 3411
## log likelihood:  -1480.85
## Nagelkerke R2:  0.03630383
## % pres/err predicted correctly:  -978.8792
## % of predictable range [ (model-null)/(1-null) ]:  0.02797191
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8949      -0.1894
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2986  AIC: 3438
## log likelihood:  -1492.796
## Nagelkerke R2:  0.02461889
## % pres/err predicted correctly:  -987.407
## % of predictable range [ (model-null)/(1-null) ]:  0.01951243
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.66424     -0.08984
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 3022  AIC: 3485
## log likelihood:  -1510.86
## Nagelkerke R2:  0.006792219
## % pres/err predicted correctly:  -1002.423
## % of predictable range [ (model-null)/(1-null) ]:  0.004617044
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##       1.454
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3400 Residual
## Null Deviance:       3035
## Residual Deviance: 3035  AIC: 3498
## log likelihood:  -1517.692
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1007.077
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	3308.611	0.00000	1	1	0.0721294	1.774949	NA	-0.5448407	NA	NA	NA
preserved ~ (I(pos^2) + pos)	3407.459	98.84848	0	0	0.0374270	2.375599	NA	NA	0.0141547	-0.3041456	NA
preserved ~ pos	3410.598	101.98722	0	0	0.0363033	2.158226	NA	NA	NA	-0.1778709	NA
preserved ~ stimlen	3438.229	129.61865	0	0	0.0246189	2.894937	NA	NA	NA	NA	-0.1894312
preserved ~ CumPres	3485.036	176.42501	0	0	0.0067922	1.664243	-0.0898369	NA	NA	NA	NA
preserved ~ 1	3497.815	189.20423	0	0	0.0000000	1.454378	NA	NA	NA	NA	NA

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

    RndModelAIC<-numeric(length=RandomSamples)
    for(rindex in seq(1,RandomSamples)){
```

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

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

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

syll_component	MeanPres	N
1	0.8161765	442
O	0.8033333	1575
P	0.8076923	26
S	0.8978758	204
V	0.8032289	1154

```
#  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
##      1.7195      -0.5379
##
## Degrees of Freedom: 3170 Total (i.e. Null);  3169 Residual
## Null Deviance:       2868
## Residual Deviance: 2740  AIC: 3149
## log likelihood:  -1369.864
## Nagelkerke R2:  0.06660244
## % pres/err predicted correctly:  -902.2661
## % of predictable range [ (model-null)/(1-null) ]:  0.05815175
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.38870      0.01742     -0.33248
##
## Degrees of Freedom: 3170 Total (i.e. Null);  3168 Residual
## Null Deviance:       2868
## Residual Deviance: 2795  AIC: 3229
## log likelihood:  -1397.579
## Nagelkerke R2:  0.0381515
## % pres/err predicted correctly:  -929.6743
## % of predictable range [ (model-null)/(1-null) ]:  0.0295729
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1217      -0.1774
##
## Degrees of Freedom: 3170 Total (i.e. Null);  3169 Residual
## Null Deviance:       2868
## Residual Deviance: 2798  AIC: 3234
## log likelihood:  -1399.228
## Nagelkerke R2:  0.03644317
## % pres/err predicted correctly:  -930.6576
## % of predictable range [ (model-null)/(1-null) ]:  0.02854754
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       2.854       -0.189
##
## Degrees of Freedom: 3170 Total (i.e. Null);  3169 Residual
## Null Deviance:       2868
## Residual Deviance: 2821  AIC: 3260
## log likelihood:  -1410.332
## Nagelkerke R2:  0.02489204
## % pres/err predicted correctly:  -938.6488
## % of predictable range [ (model-null)/(1-null) ]:  0.02021506
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.59025     -0.08006
##
## Degrees of Freedom: 3170 Total (i.e. Null);  3169 Residual
## Null Deviance:       2868
## Residual Deviance: 2859  AIC: 3310
## log likelihood:  -1429.331
## Nagelkerke R2:  0.004938624
## % pres/err predicted correctly:  -954.7685
## % of predictable range [ (model-null)/(1-null) ]:  0.003406819
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.418
##
## Degrees of Freedom: 3170 Total (i.e. Null);  3170 Residual
## Null Deviance:       2868
## Residual Deviance: 2868  AIC: 3317
## log likelihood:  -1433.999
## Nagelkerke R2:  5.595563e-16
## % pres/err predicted correctly:  -958.0357
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	3149.357	0.00000	1	1	0.0666024	1.719522	NA	-0.5379156	NA	NA	NA
preserved ~ (I(pos^2) + pos)	3229.260	79.90282	0	0	0.0381511	2.388698	NA	NA	0.0174228	-0.3324768	NA
preserved ~ pos	3233.748	84.39059	0	0	0.0364433	2.121720	NA	NA	NA	-0.1773803	NA
preserved ~ stimlen	3259.727	110.36963	0	0	0.0248920	2.854388	NA	NA	NA	NA	-0.1889576
preserved ~ CumPres	3309.578	160.22078	0	0	0.0049386	1.590253	-0.080058	NA	NA	NA	NA
preserved ~ 1	3317.343	167.98581	0	0	0.0000000	1.418388	NA	NA	NA	NA	NA

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.6629      -0.5616
##
## Degrees of Freedom: 2728 Total (i.e. Null);  2727 Residual
## Null Deviance:       2497
## Residual Deviance: 2407  AIC: 2729
## log likelihood:  -1203.391
## Nagelkerke R2:  0.05427294
## % pres/err predicted correctly:  -792.0184
## % of predictable range [ (model-null)/(1-null) ]:  0.04830216
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.43629      0.01953     -0.35949
##
## Degrees of Freedom: 2728 Total (i.e. Null);  2726 Residual
## Null Deviance:       2497
## Residual Deviance: 2423  AIC: 2757
## log likelihood:  -1211.491
## Nagelkerke R2:  0.04466488
## % pres/err predicted correctly:  -802.9512
## % of predictable range [ (model-null)/(1-null) ]:  0.03518172
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1440      -0.1863
##
## Degrees of Freedom: 2728 Total (i.e. Null);  2727 Residual
## Null Deviance:       2497
## Residual Deviance: 2427  AIC: 2762
## log likelihood:  -1213.422
## Nagelkerke R2:  0.04236545
## % pres/err predicted correctly:  -804.104
## % of predictable range [ (model-null)/(1-null) ]:  0.03379824
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       2.847       -0.190
##
## Degrees of Freedom: 2728 Total (i.e. Null);  2727 Residual
## Null Deviance:       2497
## Residual Deviance: 2455  AIC: 2799
## log likelihood:  -1227.531
## Nagelkerke R2:  0.02546814
## % pres/err predicted correctly:  -815.0984
## % of predictable range [ (model-null)/(1-null) ]:  0.02060398
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.6814      -0.1487
##
## Degrees of Freedom: 2728 Total (i.e. Null);  2727 Residual
## Null Deviance:       2497
## Residual Deviance: 2475  AIC: 2827
## log likelihood:  -1237.689
## Nagelkerke R2:  0.01319488
## % pres/err predicted correctly:  -824.3547
## % of predictable range [ (model-null)/(1-null) ]:  0.009495489
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.407
##
## Degrees of Freedom: 2728 Total (i.e. Null);  2728 Residual
## Null Deviance:       2497
## Residual Deviance: 2497  AIC: 2849
## log likelihood:  -1248.525
## Nagelkerke R2:  3.703919e-16
## % pres/err predicted correctly:  -832.267
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

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

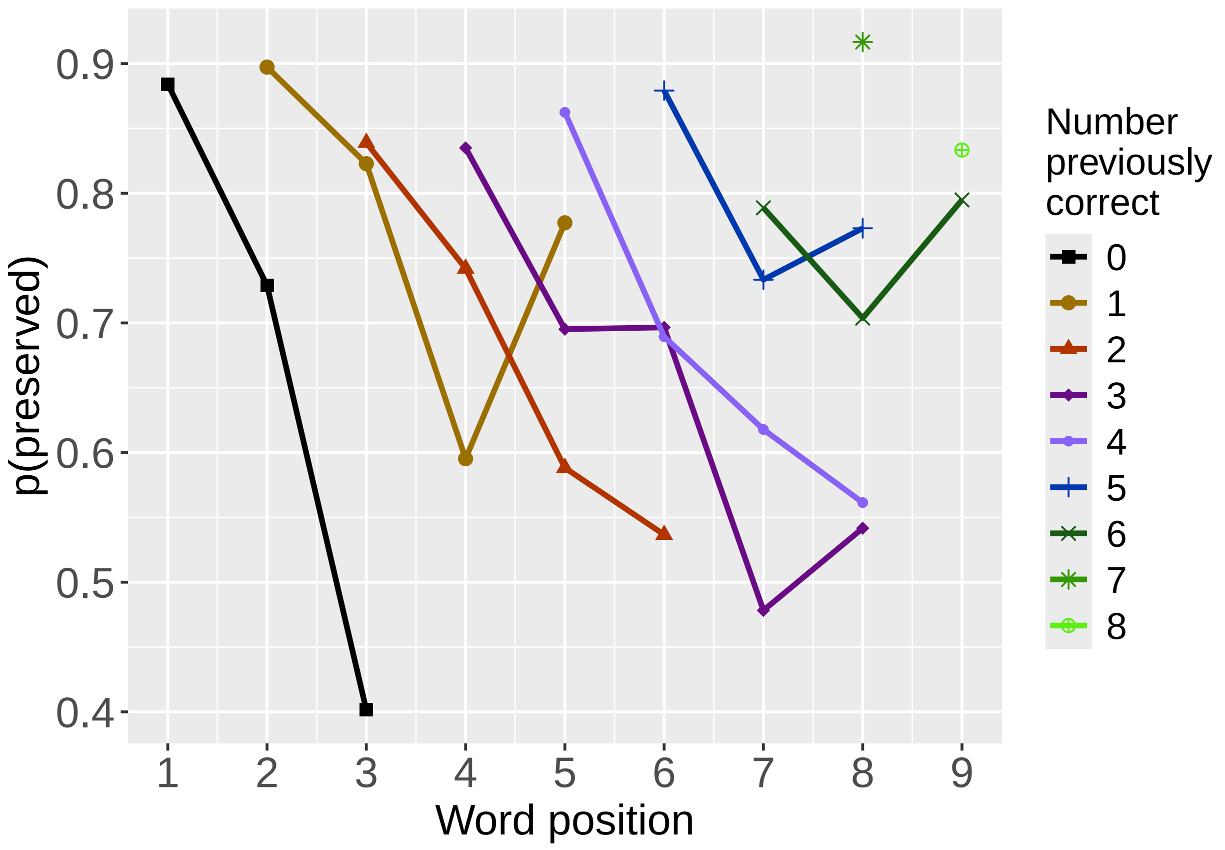
Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2729.071	0.00000	1e+00	0.9999991	0.0542729	1.662863	NA	-0.5615978	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2757.139	28.06745	8e-07	0.0000008	0.0446642	2.436291	NA	NA	0.0195276	-0.3594933	NA
preserved ~ pos	2761.980	32.90839	1e-07	0.0000001	0.0423652	2.144044	NA	NA	NA	-0.1862709	NA
preserved ~ stimlen	2799.073	70.00187	0e+00	0.0000000	0.0254681	1.846566	NA	NA	NA	NA	-0.1899566
preserved ~ CumPres	2827.419	98.34793	0e+00	0.0000000	0.0131949	1.681374	-0.1487219	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ 1	2849.318	120.2464	0e+00	0.0000000	0.0000000	0.406980	NA	NA	NA	NA	NA

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

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

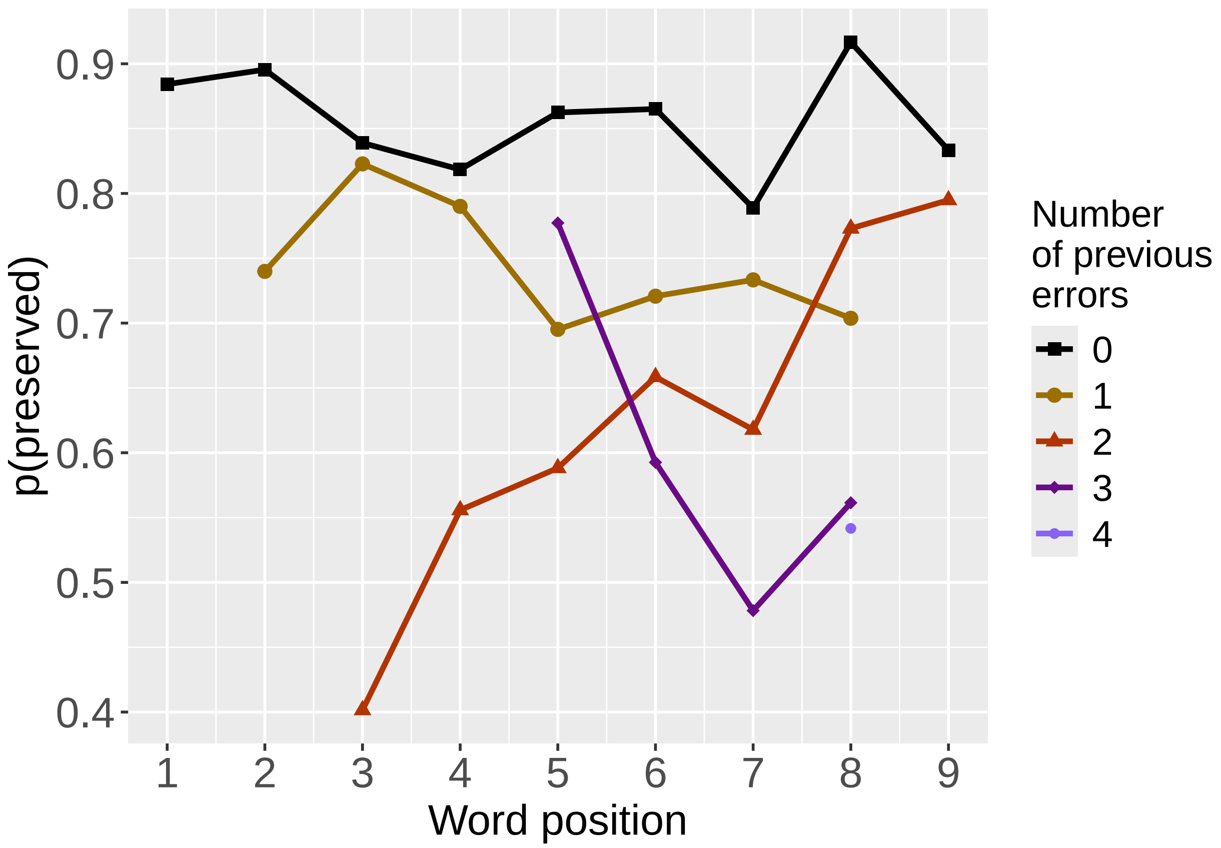
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

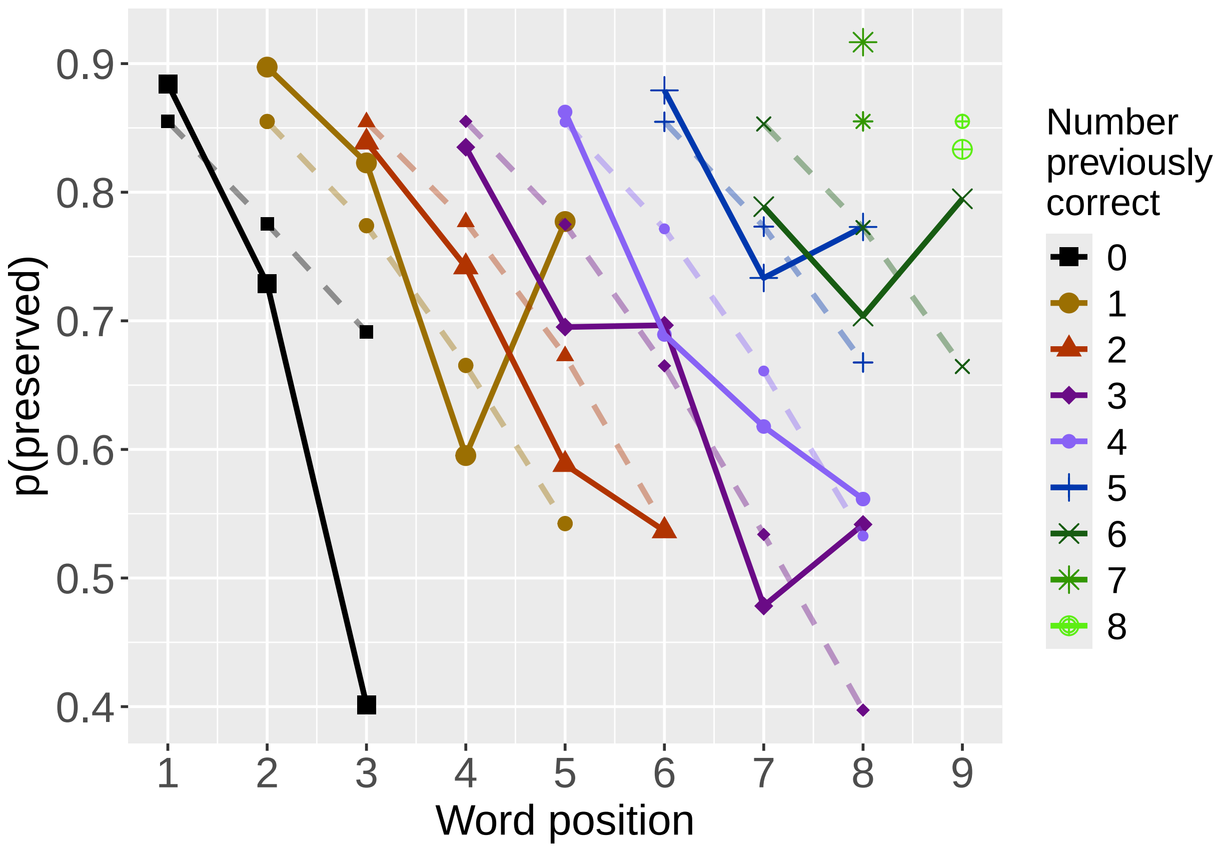
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

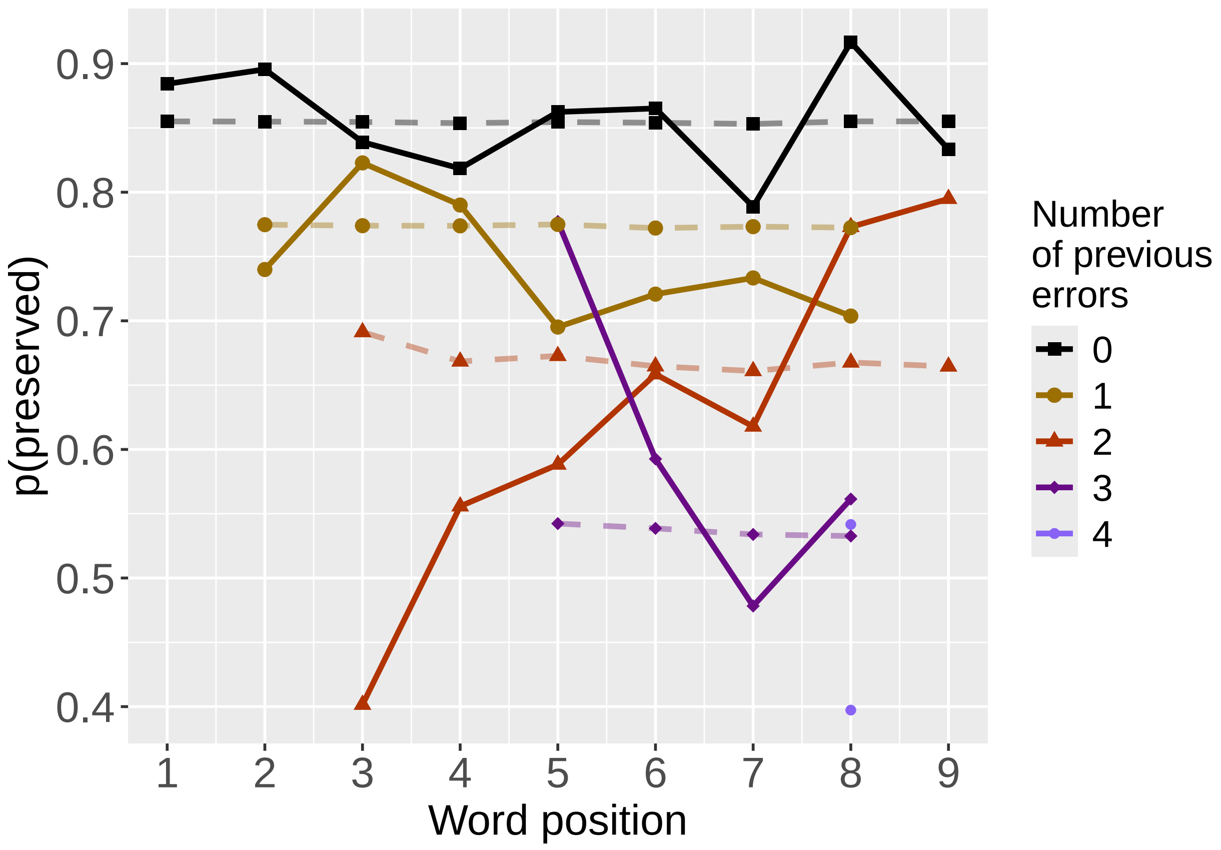
```
print(PrevCorPlot)
```



```r
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.
```

```r
print(PrevErrPlot)
```



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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.35655     -0.49507      0.02707     -0.29110
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2874  AIC: 3294
## log likelihood:  -1437.241
## Nagelkerke R2:  0.07827055
## % pres/err predicted correctly:  -940.1669
## % of predictable range [ (model-null)/(1-null) ]:  0.06637406
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.7749      -0.5448
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2887  AIC: 3309
## log likelihood:  -1443.693
## Nagelkerke R2:  0.07212943
## % pres/err predicted correctly:  -944.7192
## % of predictable range [ (model-null)/(1-null) ]:  0.06185814
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.37560      0.01415     -0.30415
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2959  AIC: 3407
## log likelihood:  -1479.697
## Nagelkerke R2:  0.03742697
## % pres/err predicted correctly:  -978.168
## % of predictable range [ (model-null)/(1-null) ]:  0.0286774
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	3293.750	0.00000	1.0000000	0.9994075	0.0782705	2.356555	-0.4950672	0.0270687	-0.2910986

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	3308.611	14.86111	0.0005929	0.0005925	0.0721294	1.774949	-0.5448407	NA	NA
preserved ~ I(pos^2) + pos	3407.459	113.70959	0.0000000	0.0000000	0.0374270	2.375599	NA	0.0141547	-0.3041456

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     2.46983     -0.49120     -0.09572
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2876  AIC: 3298
## log likelihood:  -1438.198
## Nagelkerke R2:  0.07736153
## % pres/err predicted correctly:  -940.3957
## % of predictable range [ (model-null)/(1-null) ]:  0.06614708
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.7749      -0.5448
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2887  AIC: 3309
## log likelihood:  -1443.693
## Nagelkerke R2:  0.07212943
## % pres/err predicted correctly:  -944.7192
## % of predictable range [ (model-null)/(1-null) ]:  0.06185814
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8949      -0.1894
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2986  AIC: 3438
## log likelihood:  -1492.796
## Nagelkerke R2:  0.02461889
## % pres/err predicted correctly:  -987.407
## % of predictable range [ (model-null)/(1-null) ]:  0.01951243
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	3297.718	0.0000	1.0000000	0.9957073	0.0773615	2.469827	-0.4911976	-0.0957200
preserved ~ CumErr	3308.611	10.8931	0.0043112	0.0042927	0.0721294	1.774949	-0.5448407	NA
preserved ~ stimlen	3438.229	140.5117	0.0000000	0.0000000	0.0246189	2.894937	NA	-0.1894312

```r
########
# 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
##     1.90222     -0.53194     -0.05737
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2882  AIC: 3306
## log likelihood:  -1441.183
## Nagelkerke R2:  0.07452152
## % pres/err predicted correctly:  -943.1982
## % of predictable range [ (model-null)/(1-null) ]:  0.06336702
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.7749      -0.5448
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2887  AIC: 3309
## log likelihood:  -1443.693
## Nagelkerke R2:  0.07212943
## % pres/err predicted correctly:  -944.7192
## % of predictable range [ (model-null)/(1-null) ]:  0.06185814
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.66424     -0.08984
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 3022  AIC: 3485
## log likelihood:  -1510.86
## Nagelkerke R2:  0.006792219
## % pres/err predicted correctly:  -1002.423
## % of predictable range [ (model-null)/(1-null) ]:  0.004617044
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	3306.017	0.000000	1.0000000	0.7853062	0.0745215	1.902224	-0.5319362	-0.0573670
preserved ~ CumErr	3308.611	2.593721	0.2733887	0.2146938	0.0721294	1.774949	-0.5448407	NA
preserved ~ CumPres	3485.036	179.018735	0.0000000	0.0000000	0.0067922	1.664243	NA	-0.0898369

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

```r
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
##     1.95959     -0.47457     -0.05737
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2882  AIC: 3306
## log likelihood:  -1441.183
## Nagelkerke R2:  0.07452152
## % pres/err predicted correctly:  -943.1982
## % of predictable range [ (model-null)/(1-null) ]:  0.06336702
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.7749      -0.5448
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2887  AIC: 3309
## log likelihood:  -1443.693
## Nagelkerke R2:  0.07212943
## % pres/err predicted correctly:  -944.7192
## % of predictable range [ (model-null)/(1-null) ]:  0.06185814
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.1582      -0.1779
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2962  AIC: 3411
## log likelihood:  -1480.85
## Nagelkerke R2:  0.03630383
## % pres/err predicted correctly:  -978.8792
## % of predictable range [ (model-null)/(1-null) ]:  0.02797191
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	3306.017	0.000000	1.0000000	0.7853062	0.0745215	1.959591	-0.4745692	-0.0573670
preserved ~ CumErr	3308.611	2.593721	0.2733887	0.2146938	0.0721294	1.774949	-0.5448407	NA
preserved ~ pos	3410.598	104.580945	0.0000000	0.0000000	0.0363038	2.158226	NA	-0.1778709

```
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	3293.750	0.000000	1.0000000	0.9994075	0.0782705	2.356555	-0.4950672	0.0270687	-0.2910986	NA	NA
preserved ~ CumErr + stimlen	3297.718	0.000000	1.0000000	0.9957073	0.0773615	2.469827	-0.4911976	NA	NA	-0.0957200	NA
preserved ~ CumErr + CumPres	3306.017	0.000000	1.0000000	0.7853062	0.0745215	1.902224	-0.5319362	NA	NA	NA	-0.0573670
preserved ~ CumErr + pos	3306.017	0.000000	1.0000000	0.7853062	0.0745215	1.959591	-0.4745692	NA	-0.0573670	NA	NA
preserved ~ CumErr	3308.611	14.861108	0.0005929	0.0005925	0.0721294	1.774949	-0.5448407	NA	NA	NA	NA
preserved ~ CumErr	3308.611	10.893098	0.0043110	0.0042927	0.0721294	1.774949	-0.5448407	NA	NA	NA	NA
preserved ~ CumErr	3308.611	2.593721	0.2733887	0.2146938	0.0721294	1.774949	-0.5448407	NA	NA	NA	NA
preserved ~ CumErr	3308.611	2.593721	0.2733887	0.2146938	0.0721294	1.774949	-0.5448407	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	3407.459	113.709587	0.0000000	0.0000000	0.0374270	2.375599	NA	0.0141547	-0.3041456	NA	NA
preserved ~ pos	3410.598	104.580945	0.0000000	0.0000000	0.0363038	2.158226	NA	NA	-0.1778709	NA	NA
preserved ~ stimlen	3438.229	140.511704	0.0000000	0.0000000	0.0246189	2.894937	NA	NA	NA	-0.1894312	NA
preserved ~ CumPres	3485.036	179.018735	0.0000000	0.0000000	0.0067922	1.664243	NA	NA	NA	NA	-0.0898369

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     2.99956     -0.47823      0.03192     -0.30492     -0.09113      0.04103
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3395 Residual
## Null Deviance:       3035
## Residual Deviance: 2861  AIC: 3281
## log likelihood:  -1430.589
## Nagelkerke R2:  0.08457873
## % pres/err predicted correctly:  -934.8115
## % of predictable range [ (model-null)/(1-null) ]:  0.07168648
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     3.07343     -0.47614      0.03213     -0.30731     -0.10076
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3396 Residual
## Null Deviance:       3035
## Residual Deviance: 2864  AIC: 3282
## log likelihood:  -1431.952
## Nagelkerke R2:  0.08328829
## % pres/err predicted correctly:  -935.8153
## % of predictable range [ (model-null)/(1-null) ]:  0.07069074
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     2.34826     -0.49539      0.02742     -0.28974      0.05452
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3396 Residual
## Null Deviance:       3035
## Residual Deviance: 2869  AIC: 3291
## log likelihood:  -1434.747
## Nagelkerke R2:  0.08063883
## % pres/err predicted correctly:  -938.1528
## % of predictable range [ (model-null)/(1-null) ]:  0.06837198
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.35655     -0.49507      0.02707     -0.29110
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2874  AIC: 3294
## log likelihood:  -1437.241
## Nagelkerke R2:  0.07827055
## % pres/err predicted correctly:  -940.1669
## % of predictable range [ (model-null)/(1-null) ]:  0.06637406
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       1.454
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3400 Residual
## Null Deviance:       3035
## Residual Deviance: 3035  AIC: 3498
## log likelihood:  -1517.692
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1007.077
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	3281.265	0.0000000	1.0000000	0.5549988	0.0845728	2.7999559	-0.4782344	0.0319194	-0.3049155	0.0410307	-0.0911334
preserved ~ CumErr + I(pos^2) + pos + stimlen	3281.734	0.4695569	0.7907466	0.4388633	0.1083288	3.3073427	-0.4761435	0.0321343	-0.3073123	NA	-0.1007571
preserved ~ CumErr + I(pos^2) + pos + log_freq	3290.661	9.3958097	0.0091144	0.0050585	0.0806328	2.8348262	-0.4953902	0.0274238	-0.2897373	0.0545219	NA
preserved ~ CumErr + I(pos^2) + pos	3293.750	12.4847496	0.0019452	0.0010796	0.0782720	2.5356555	-0.4950672	0.0270687	-0.2910986	NA	NA
preserved ~ 1	3497.815	216.5500837	0.0000000	0.0000000	0.0000000	1.454378	NA	NA	NA	NA	NA

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

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

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

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

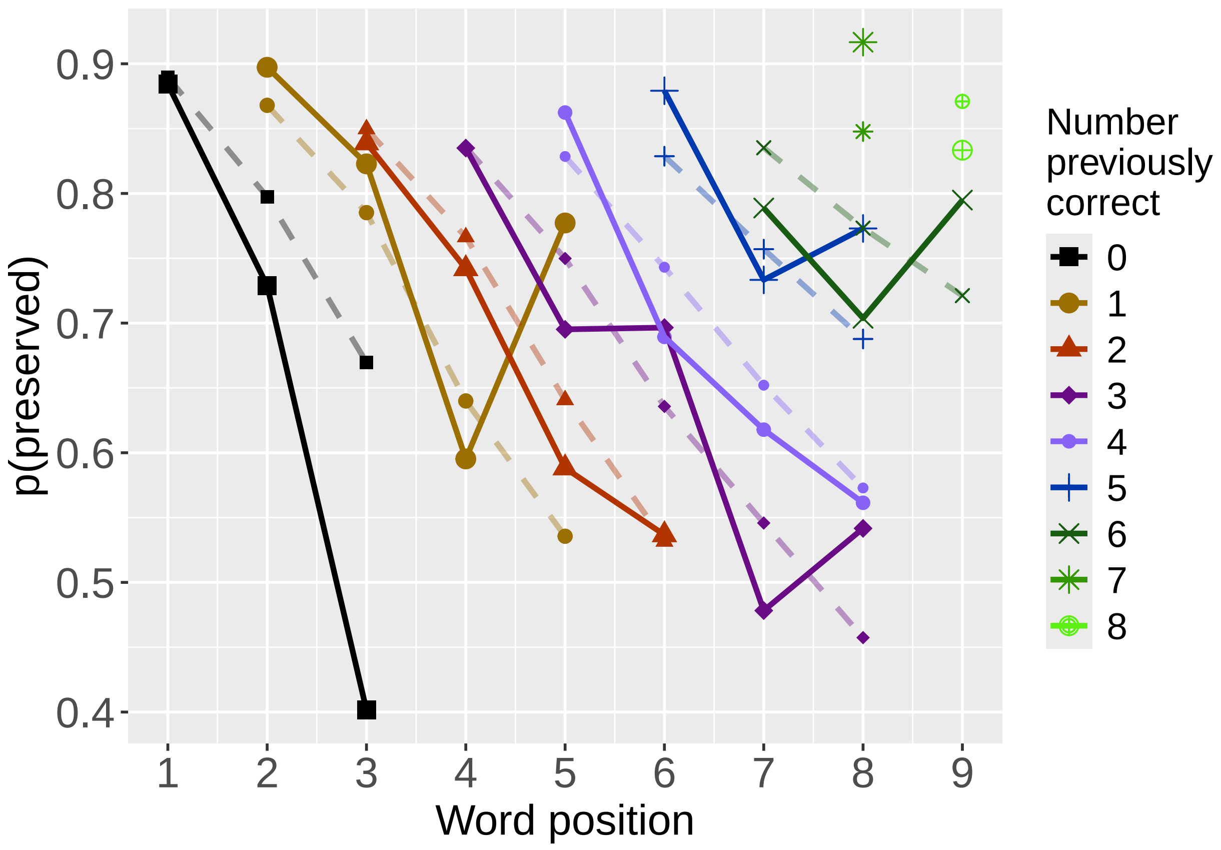
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   2939.7 3357.8
## pos       1   2873.4 3291.5
## I(pos^2)  1   2871.9 3290.0
## stimlen   1   2869.5 3287.6
## log_freq  1   2863.9 3282.0
## <none>        2861.2 3281.3
```

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

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

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

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

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

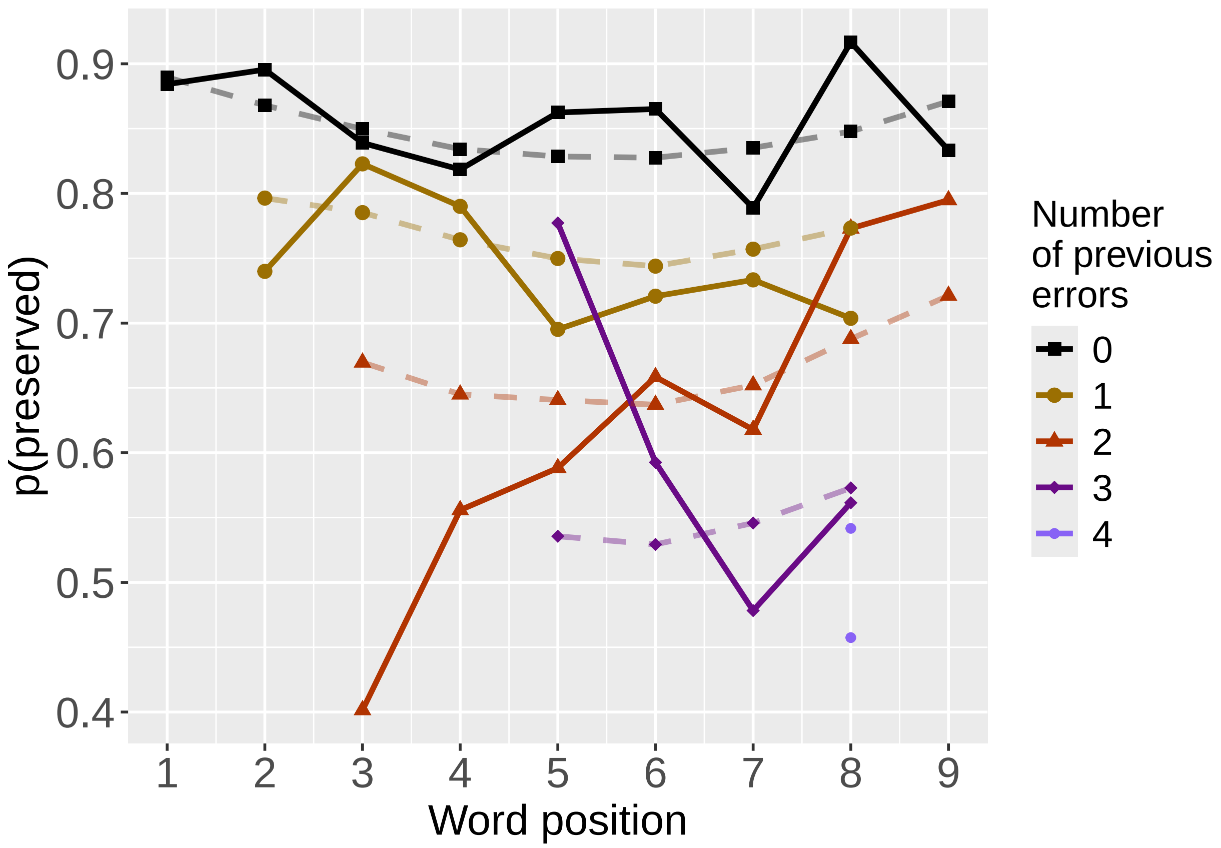
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

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

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

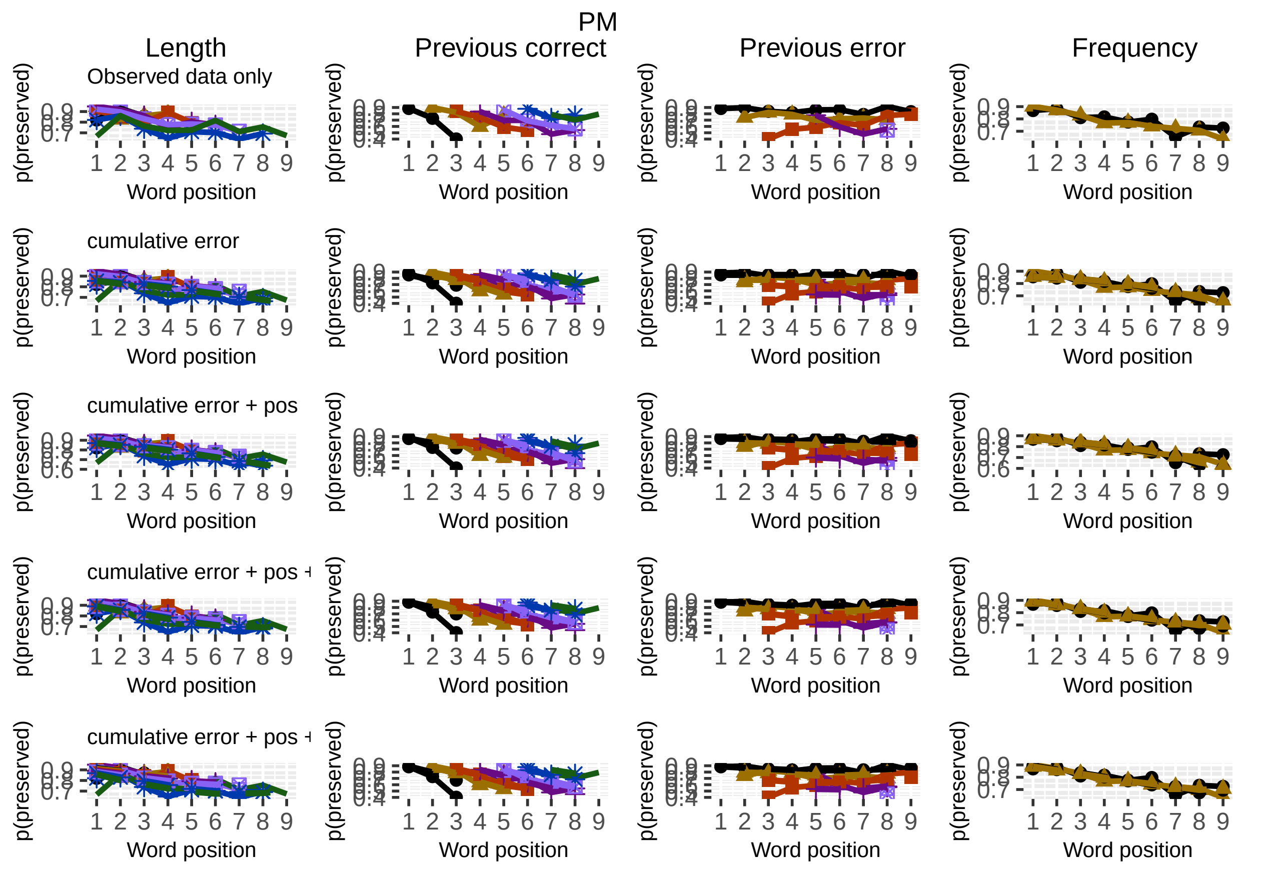
```r
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random 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)
}
```

```r
# 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
##      1.7749      -0.5448
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3399 Residual
## Null Deviance:       3035
## Residual Deviance: 2887  AIC: 3309
## log likelihood:  -1443.693
## Nagelkerke R2:  0.07212943
## % pres/err predicted correctly:  -944.7192
## % of predictable range [ (model-null)/(1-null) ]:  0.06185814
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     1.95959     -0.47457     -0.05737
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3398 Residual
## Null Deviance:       3035
## Residual Deviance: 2882  AIC: 3306
## log likelihood:  -1441.183
## Nagelkerke R2:  0.07452152
## % pres/err predicted correctly:  -943.1982
## % of predictable range [ (model-null)/(1-null) ]:  0.06336702
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)
##     2.35655     -0.49507     -0.29110      0.02707
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3397 Residual
## Null Deviance:       3035
## Residual Deviance: 2874  AIC: 3294
## log likelihood:  -1437.241
## Nagelkerke R2:  0.07827055
## % pres/err predicted correctly:  -940.1669
## % of predictable range [ (model-null)/(1-null) ]:  0.06637406
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)      stimlen
##     3.07343     -0.47614     -0.30731      0.03213     -0.10076
##
## Degrees of Freedom: 3400 Total (i.e. Null);  3396 Residual
## Null Deviance:       3035
## Residual Deviance: 2864  AIC: 3282
## log likelihood:  -1431.952
## Nagelkerke R2:  0.08328829
## % pres/err predicted correctly:  -935.8153
## % of predictable range [ (model-null)/(1-null) ]:  0.07069074
## ***************************
## `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 5 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 rows containing missing values or values outside the scale range (`geom_point()`).
```

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



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

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

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

	CumErr	I(pos^2)	pos	stimlen	log_freq
McFadden	0.0377382	0.0080172	0.0104619	0.0072585	0.0013303
SquaredCorrelation	0.0373023	0.0080457	0.0104842	0.0072818	0.0013287
Nagelkerke	0.0373023	0.0080457	0.0104842	0.0072818	0.0013287
Estrella	0.0387481	0.0082352	0.0107460	0.0074558	0.0013663

```r
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!
```

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

```
##                                                               model deviance
## CumErr + pos + I(pos^2) + stimlen CumErr + pos + I(pos^2) + stimlen 2863.904
## CumErr + pos + I(pos^2)                     CumErr + pos + I(pos^2) 2874.483
## CumErr + pos                                           CumErr + pos 2882.366
## CumErr                                                       CumErr 2887.386
## null                                                           null 3035.385
##                                   deviance_explained percent_explained
## CumErr + pos + I(pos^2) + stimlen           171.4812          5.649407
## CumErr + pos + I(pos^2)                     160.9019          5.300872
## CumErr + pos                                153.0189          5.041169
## CumErr                                      147.9986          4.875777
## null                                          0.0000          0.000000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + pos + I(pos^2) + stimlen                     100.00000               6.169409
## CumErr + pos + I(pos^2)                                93.83059               4.597002
## CumErr + pos                                           89.23359               2.927592
## CumErr                                                 86.30600              86.305997
## null                                                         NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + I(pos^2) + stimlen	2863.904	171.4812
CumErr + pos + I(pos^2)	2874.483	160.9019
CumErr + pos	2882.366	153.0189
CumErr	2887.386	147.9986
null	3035.385	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + I(pos^2) + stimlen	5.649407	100.00000	6.169409
CumErr + pos + I(pos^2)	5.300872	93.83059	4.597002
CumErr + pos	5.041169	89.23359	2.927592
CumErr	4.875777	86.30600	86.305997
null	0.000000	NA	0.000000

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

```
##           Nagelkerke
## CumErr    0.57884500
## I(pos^2) 0.12485068
## pos       0.16268979
## stimlen   0.11299617
## log_freq 0.02061836
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr+pos	0.6559124	2882.366
preserved ~ CumErr	0.6911616	2887.386
preserved ~ CumErr+pos+I(pos^2)	0.7039597	2874.483
preserved ~ CumErr+pos+I(pos^2)+stimlen	0.7326523	2863.904

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

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

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

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

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

```
##                                       model p_accounted_for model_deviance diff_CumErr+pos
## 1                    preserved ~ CumErr+pos       0.6559124       2882.366      0.00000000
## 2                        preserved ~ CumErr       0.6911616       2887.386      0.03524922
## 3           preserved ~ CumErr+pos+I(pos^2)       0.7039597       2874.483      0.04804739
## 4 preserved ~ CumErr+pos+I(pos^2)+stimlen       0.7326523       2863.904      0.07673996
##   diff_CumErr diff_CumErr+pos+I(pos^2) diff_CumErr+pos+I(pos^2)+stimlen
## 1 -0.03524922              -0.04804739                      -0.07673996
## 2  0.00000000              -0.01279817                      -0.04149073
## 3  0.01279817               0.00000000                      -0.02869256
## 4  0.04149073               0.02869256                       0.00000000
```

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

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

model	diff_CumErr+pos	diff_CumErr	diff_CumErr+pos+I(pos^2)
preserved ~ CumErr+pos	0.0000000	-0.0352492	-0.0480474
preserved ~ CumErr	0.0352492	0.0000000	-0.0127982
preserved ~ CumErr+pos+I(pos^2)	0.0480474	0.0127982	0.0000000
preserved ~ CumErr+pos+I(pos^2)+stimlen	0.0767400	0.0414907	0.0286926

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