AM - repetition - Serial position analysis (v5)

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

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

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

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

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

```
}
PosDat<-read.csv(ModelDatFilename)
```

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	518	32	124	NA	NA	674
2	62	NA	411	94	107	674
3	294	NA	164	201	15	674
4	290	NA	228	67	35	620
5	223	NA	203	69	33	528
6	192	1	126	73	21	413
7	167	NA	95	28	19	309
8	89	NA	50	21	4	164
9	67	NA	2	NA	7	76

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7685460	0.0474777	0.1839763	NA	NA	674
2	0.0919881	NA	0.6097923	0.1394659	0.1587537	674
3	0.4362018	NA	0.2433234	0.2982196	0.0222552	674
4	0.4677419	NA	0.3677419	0.1080645	0.0564516	620
5	0.4223485	NA	0.3844697	0.1306818	0.0625000	528
6	0.4648910	0.0024213	0.3050847	0.1767554	0.0508475	413

pos_factor	O	P	V	1	S	total
7	0.5404531	NA	0.3074434	0.0906149	0.0614887	309
8	0.5426829	NA	0.3048780	0.1280488	0.0243902	164
9	0.8815789	NA	0.0263158	NA	0.0921053	76

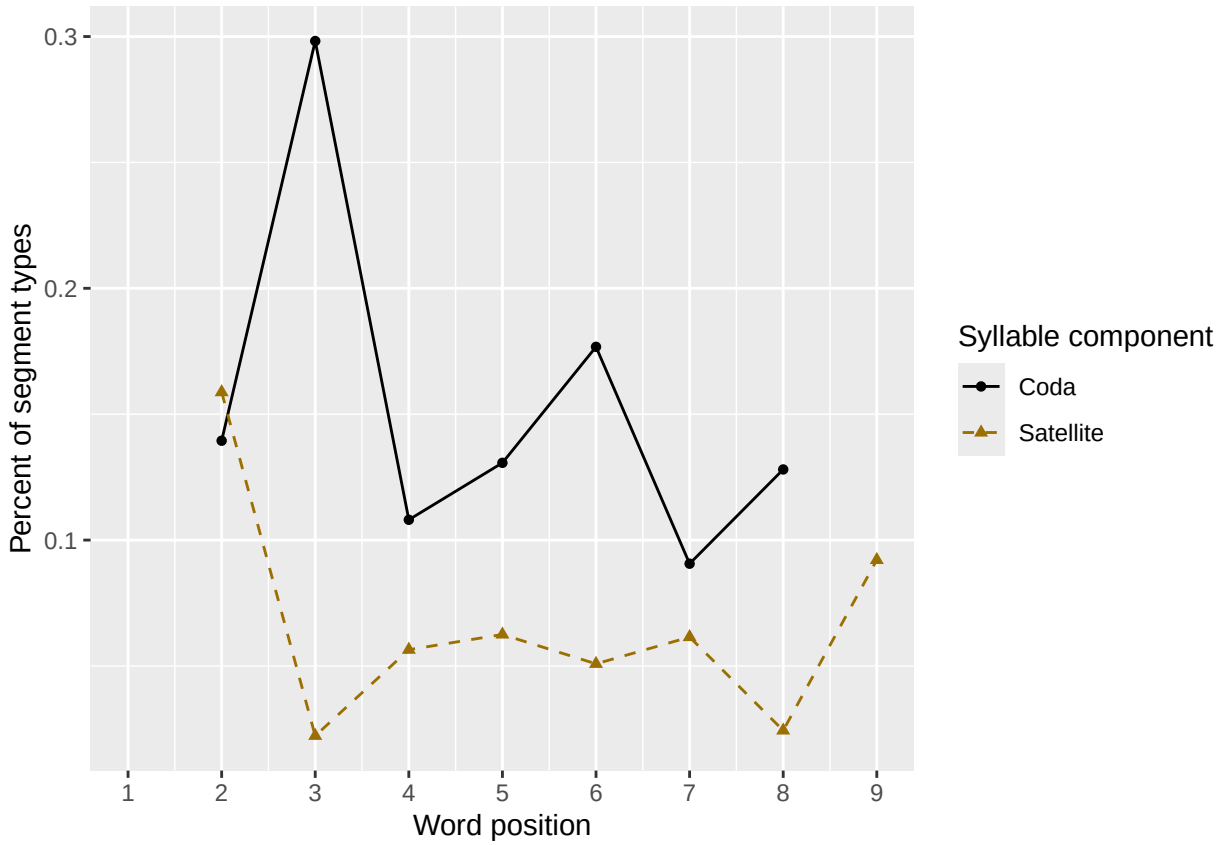
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.870 0.889 0.833 NA     NA     NA     NA     NA     NA
## 2       5 0.859 0.924 0.853  0.864 NA     NA     NA     NA     NA
## 3       6 0.904 0.896 0.809  0.839  0.770 NA     NA     NA     NA
## 4       7 0.808 0.875 0.837  0.817  0.894  0.865 NA     NA     NA
## 5       8 0.821 0.875 0.862  0.828  0.833  0.855  0.814 NA     NA
## 6       9 0.872 0.855 0.784  0.881  0.781  0.847  0.830  0.770 NA
## 7      10 0.771 0.904 0.847  0.776  0.822  0.743  0.842  0.847  0.75
```

```
# 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.
```

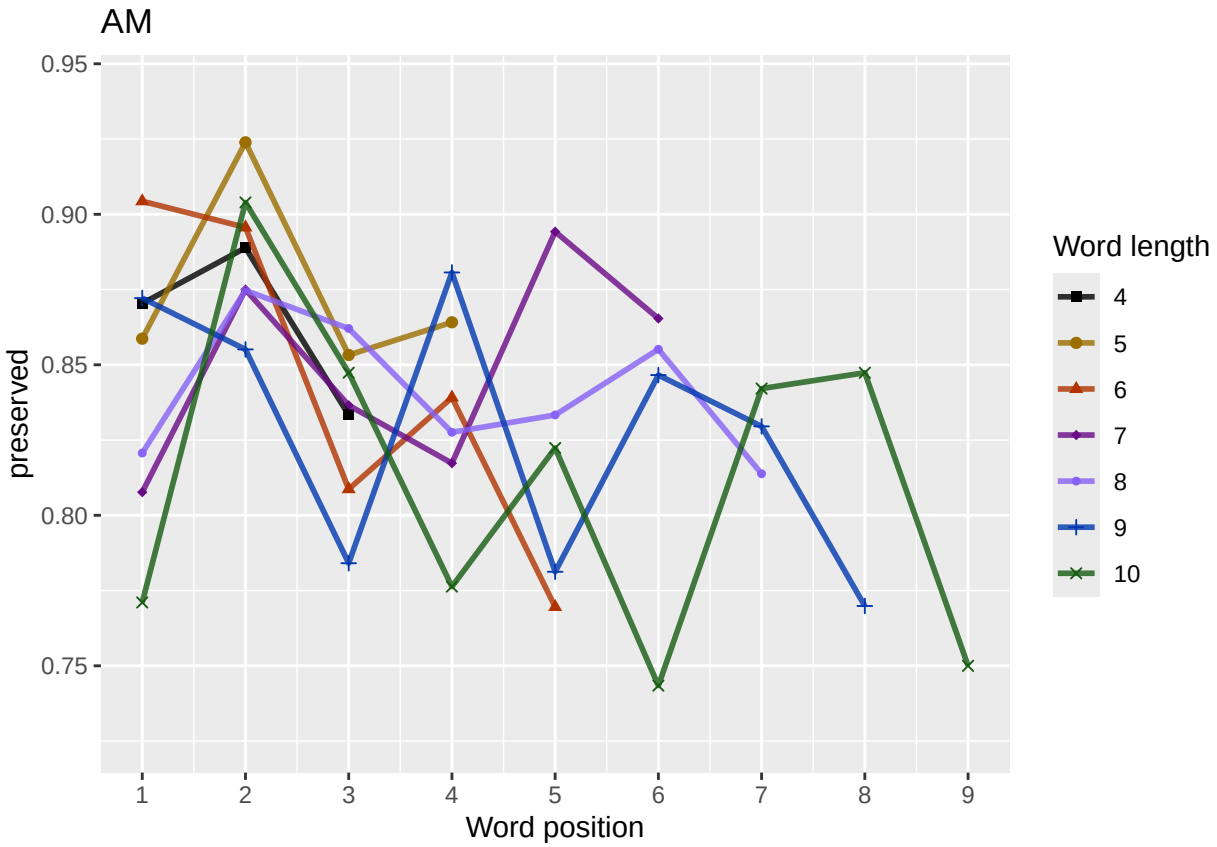
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    54    54    54    NA    NA    NA    NA    NA    NA
## 2       5    92    92    92    92    NA    NA    NA    NA    NA
## 3       6   115   115   115   115   115    NA    NA    NA    NA
## 4       7   104   104   104   104   104   104    NA    NA    NA
## 5       8   145   145   145   145   145   145   145    NA    NA
## 6       9    88    88    88    88    88    88    88    88    NA
## 7      10    76    76    76    76    76    76    76    76    76
```

```
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:  4
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.20997     -0.05062     -0.04235
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3553  AIC: 3718
## log likelihood:  -1776.381
## Nagelkerke R2:  0.005053071
## % pres/err predicted correctly:  -1100.907
## % of predictable range [ (model-null)/(1-null) ]:  0.003225429
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    2.376179    -0.070760    -0.095042     0.006181
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3553  AIC: 3719
## log likelihood:  -1776.277
## Nagelkerke R2:  0.00513938
## % pres/err predicted correctly:  -1100.873
## % of predictable range [ (model-null)/(1-null) ]:  0.003256033
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.175351    -0.049562    -0.001914    -0.025831
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3553  AIC: 3720
## log likelihood:  -1776.357
## Nagelkerke R2:  0.005072981
## % pres/err predicted correctly:  -1100.902
## % of predictable range [ (model-null)/(1-null) ]:  0.003229658
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen
##     2.21219     -0.07239
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3557  AIC: 3720
## log likelihood:  -1778.363
## Nagelkerke R2:  0.003397076
## % pres/err predicted correctly:  -1101.958
## % of predictable range [ (model-null)/(1-null) ]:  0.002274296
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.8817      -0.0583
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3720
## log likelihood:  -1778.027
## Nagelkerke R2:  0.003678173
## % pres/err predicted correctly:  -1101.974
## % of predictable range [ (model-null)/(1-null) ]:  0.002260435
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.815654    -0.004566    -0.018119
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3722
## log likelihood:  -1777.887
## Nagelkerke R2:  0.003794809
## % pres/err predicted correctly:  -1101.912
## % of predictable range [ (model-null)/(1-null) ]:  0.002316437
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)              pos  stimlen:I(pos^2)
```

```
##         2.842500         -0.136677          0.033767         -0.385984         -0.004538
##      stimlen:pos
##         0.046291
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4126 Residual
## Null Deviance:       3565
## Residual Deviance: 3552  AIC: 3723
## log likelihood:  -1775.828
## Nagelkerke R2:  0.005514248
## % pres/err predicted correctly:  -1100.659
## % of predictable range [ (model-null)/(1-null) ]:  0.003449756
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.654
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4131 Residual
## Null Deviance:       3565
## Residual Deviance: 3565  AIC: 3728
## log likelihood:  -1782.424
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1104.473
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	3717.61	0.000000	1.000000	0.6384397	0.0050532	1.209973	-0.050623	-0.042351	NA	NA	NA
preserved ~ stimlen * pos	3719.37	1.761320	0.414509	0.2645933	0.0051324	1.376179	-0.070759	-0.095041	0.0061811	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	3719.765	0.153776	0.340654	0.130946	0.0050732	2.20175351	-0.04956241	-0.04258310	NA	0.0019144	NA
preserved ~ stimlen	3719.812	0.199902	0.333034	0.128017	0.0033972	1.21212190	-0.07238750	NA	NA	NA	NA
preserved ~ pos	3719.822	0.210068	0.331199	0.127312	0.0036782	1.88172281	NA	-0.05829620	NA	NA	NA
preserved ~ I(pos^2) + pos	3721.974	2.365276	0.112748	0.043338	0.0037948	1.15654815	NA	-0.01811890	NA	0.0045663	NA
preserved ~ stimlen * (I(pos^2) + pos)	3723.155	3.538592	0.062706	0.024104	0.0055142	0.42842500	-0.13667663	-0.03859840	0.0462910	0.0337672	-0.0045381
preserved ~ 1	3727.645	10.033488	0.006626	0.002547	0.0000000	0.10653530	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

```
## [1] "preserved ~ stimlen + pos"
```

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.20997     -0.05062     -0.04235
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3553  AIC: 3718
```

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

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

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

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

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

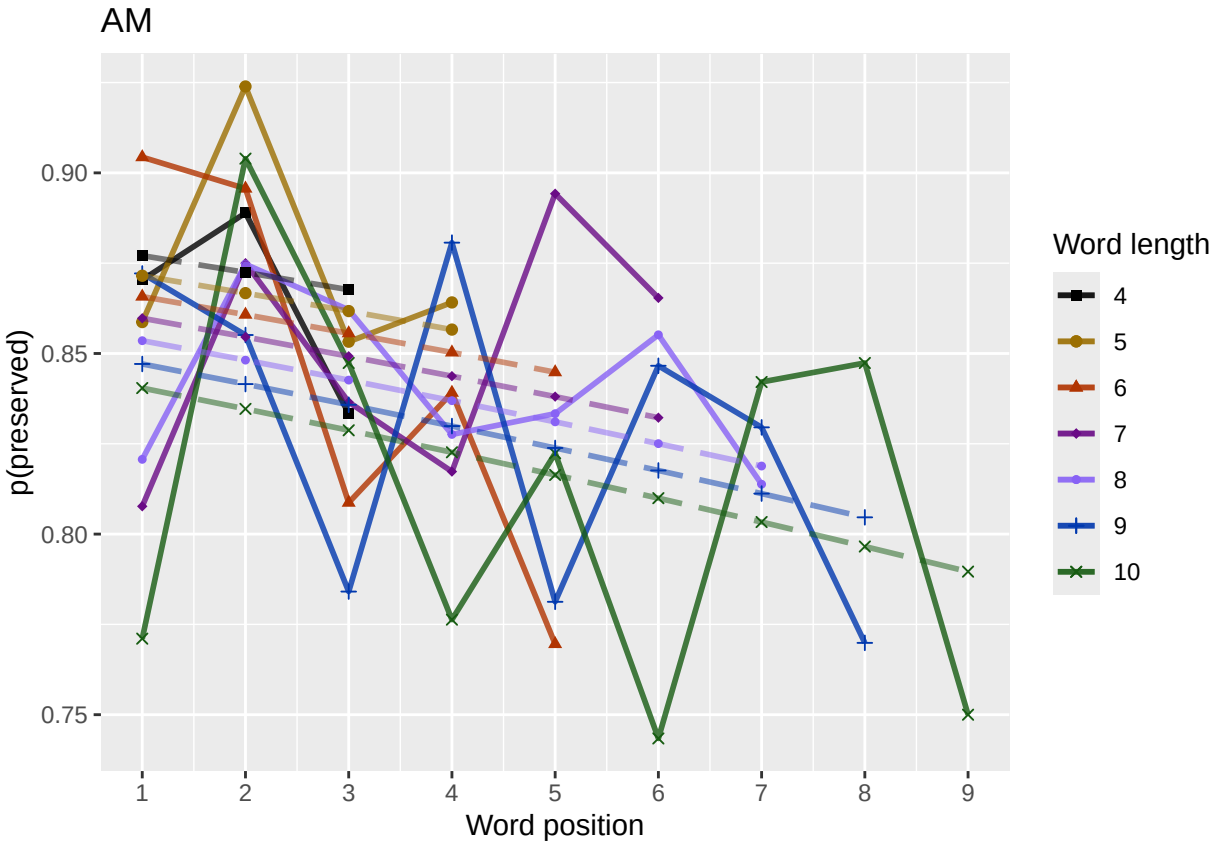
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.877 0.872 0.868 NA     NA     NA     NA     NA     NA
## 2       5 0.872 0.867 0.862  0.857 NA     NA     NA     NA     NA
## 3       6 0.866 0.861 0.856  0.850  0.845 NA     NA     NA     NA
## 4       7 0.860 0.855 0.849  0.844  0.838  0.832 NA     NA     NA
```

```
## 5       8 0.854 0.848 0.843  0.837  0.831  0.825  0.819 NA     NA
## 6       9 0.847 0.842 0.836  0.830  0.824  0.818  0.811  0.805 NA
## 7      10 0.840 0.835 0.829  0.823  0.816  0.810  0.803  0.797  0.790
```

```
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

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

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

```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       42   674
```

```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 42 / 674 = 6.23 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:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.17350     -0.06198      0.04336
##
## Degrees of Freedom: 4000 Total (i.e. Null);  3998 Residual
## Null Deviance:       3106
## Residual Deviance: 3101  AIC: 3233
## log likelihood:  -1550.291
## Nagelkerke R2:  0.002627924
## % pres/err predicted correctly:  -924.9626
## % of predictable range [ (model-null)/(1-null) ]:  0.001678084
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.54225     -0.10690     -0.08110      0.01473
##
## Degrees of Freedom: 4000 Total (i.e. Null);  3997 Residual
## Null Deviance:       3106
## Residual Deviance: 3100  AIC: 3234
## log likelihood:  -1549.832
## Nagelkerke R2:  0.003052287
## % pres/err predicted correctly:  -924.7164
## % of predictable range [ (model-null)/(1-null) ]:  0.001943531
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.17967     -0.04174
##
## Degrees of Freedom: 4000 Total (i.e. Null);  3999 Residual
## Null Deviance:       3106
## Residual Deviance: 3104  AIC: 3234
## log likelihood:  -1551.992
## Nagelkerke R2:  0.001054418
## % pres/err predicted correctly:  -925.8653
## % of predictable range [ (model-null)/(1-null) ]:  0.0007049086
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
```

```
##    2.219246    -0.063317     0.002813     0.020005
##
## Degrees of Freedom: 4000 Total (i.e. Null);  3997 Residual
## Null Deviance:       3106
## Residual Deviance: 3101  AIC: 3235
## log likelihood:  -1550.252
## Nagelkerke R2:  0.002664095
## % pres/err predicted correctly:  -924.9217
## % of predictable range [ (model-null)/(1-null) ]:  0.001722156
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.859
##
## Degrees of Freedom: 4000 Total (i.e. Null);  4000 Residual
## Null Deviance:       3106
## Residual Deviance: 3106  AIC: 3235
## log likelihood:  -1553.132
## Nagelkerke R2:  4.112487e-16
## % pres/err predicted correctly:  -926.5191
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.76544      0.02524
##
## Degrees of Freedom: 4000 Total (i.e. Null);  3999 Residual
## Null Deviance:       3106
## Residual Deviance: 3105  AIC: 3236
## log likelihood:  -1552.476
## Nagelkerke R2:  0.0006067582
## % pres/err predicted correctly:  -926.2137
## % of predictable range [ (model-null)/(1-null) ]:  0.0003292653
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.571e+00       -1.167e-01       -3.508e-03       -8.200e-02       -7.666e-05
##      stimlen:pos
##        1.892e-02
```

```
##
## Degrees of Freedom: 4000 Total (i.e. Null);  3995 Residual
## Null Deviance:       3106
## Residual Deviance: 3100  AIC: 3238
## log likelihood:  -1549.775
## Nagelkerke R2:  0.00310514
## % pres/err predicted correctly:  -924.7123
## % of predictable range [ (model-null)/(1-null) ]:  0.001947981
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##   1.7574206   -0.0006128    0.0304021
##
## Degrees of Freedom: 4000 Total (i.e. Null);  3998 Residual
## Null Deviance:       3106
## Residual Deviance: 3105  AIC: 3239
## log likelihood:  -1552.474
## Nagelkerke R2:  0.0006085175
## % pres/err predicted correctly:  -926.2167
## % of predictable range [ (model-null)/(1-null) ]:  0.0003260098
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	3233.21	0.0000000	1.0000000	0.3089946	0.0026272	1.9173503	- 0.0619803	0.0433579	NA	NA	NA
preserved ~ stimlen * pos	3234.20	0.9932407	0.6085828	0.1880183	0.0030523	1.7542254	- 0.1069007	0.0147261	NA	0.0811023	NA
preserved ~ stimlen	3234.47	1.2576765	0.5332109	0.1647326	0.0010524	1.4179668	- 0.0417403	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	3234.88	0.6670962	0.4345049	0.1342379	0.0026642	1.1219246	- 0.0633175	0.0200050	NA	0.0028131	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ 1	3235.482	2.2734958	0.3208608	0.0991282	0.0000000	1.0859386	NA	NA	NA	NA	NA
preserved ~ pos	3236.465	3.2564836	0.1962744	0.0606379	0.0006061	1.8765445	NA	0.0252359	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	3238.451	5.2379100	0.0728790	0.0225156	0.0031052	2.1571285	-0.1166726	-0.0820045	0.0189208	-0.0035079	-7.67e-05
preserved ~ I(pos^2) + pos	3238.515	5.3038854	0.0705140	0.0217849	0.0006081	1.5757421	NA	0.0304021	NA	-0.0006128	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
```

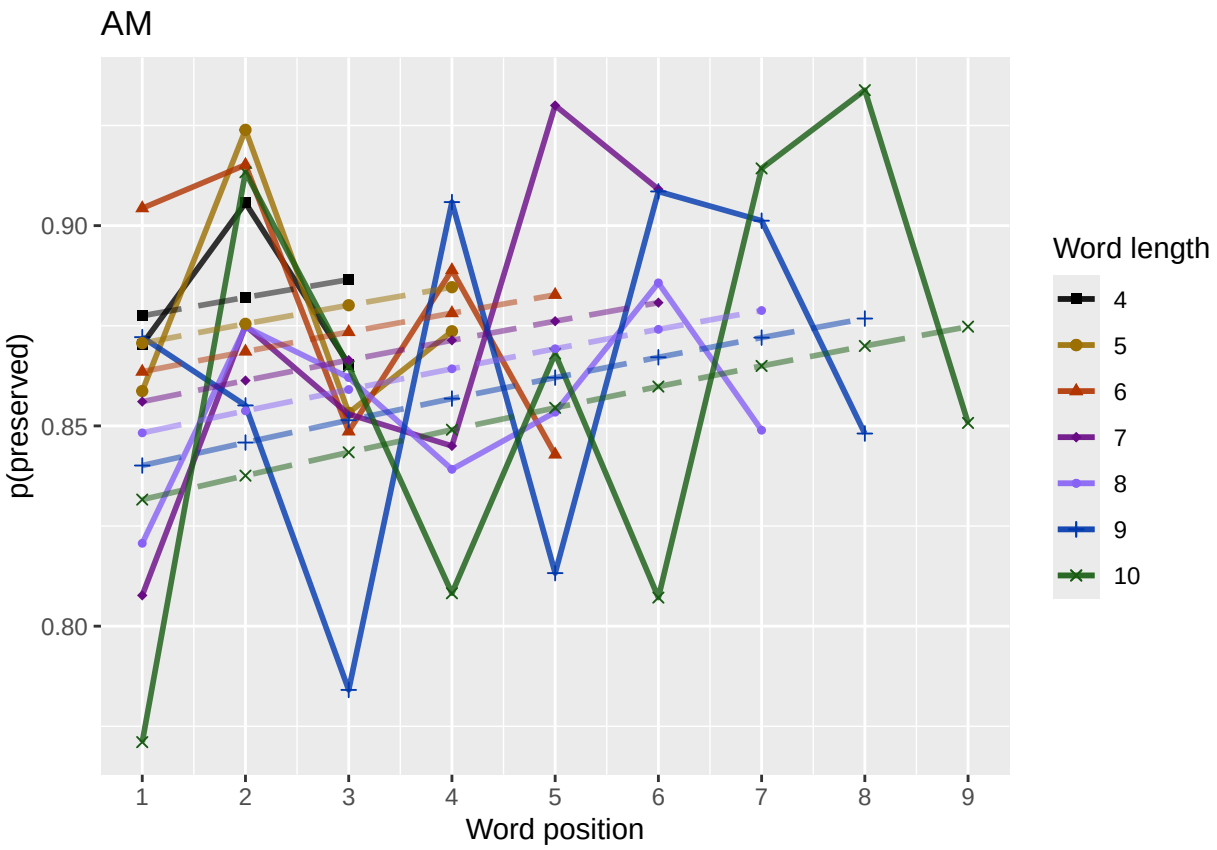
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.877 0.882 0.887 NA     NA     NA     NA     NA     NA
## 2       5 0.871 0.875 0.880  0.885 NA     NA     NA     NA     NA
## 3       6 0.864 0.869 0.873  0.878  0.883 NA     NA     NA     NA
## 4       7 0.856 0.861 0.866  0.871  0.876  0.881 NA     NA     NA
## 5       8 0.848 0.854 0.859  0.864  0.869  0.874  0.879 NA     NA
## 6       9 0.840 0.846 0.851  0.857  0.862  0.867  0.872  0.877 NA
## 7      10 0.832 0.838 0.843  0.849  0.855  0.860  0.865  0.870  0.875
```

```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: 0.73 - 0.94"
```

```
# 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
```

```r
# in case min is close to the end or we are not using a min (for non-U-shaped)
# functions, we need to get differences _first_ and then average those (e.g.
# if there is an effect of length and we just average position data,
# later positions) will have a lower average (because of the length effect)
# even if all position functions go upward

table_pos_diffs <- t(diff(t(as.matrix(table_to_use))))
ncol_pos_diff_matrix <- ncol(table_pos_diffs)
first_col_mean <- mean(as.numeric(unlist(table_pos_diffs[,1])))
potential_u_shape <- 0
if(first_col_mean < 0){ # downward, so potential U-shape
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs >= 0] <- NA
  potential_u_shape <- 1
}else{
  # upward - use the positive values
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs <= 0] <- NA
}
average_pos_diffs <- apply(filtered_table_pos_diffs,2,mean,na.rm=TRUE)
OA_mean_pos_diff <- mean(average_pos_diffs,na.rm=TRUE)

table_len_diffs <- diff(as.matrix(table_to_use))
average_len_diffs <- apply(table_len_diffs,1,mean,na.rm=TRUE)
OA_mean_len_diff <- mean(average_len_diffs,na.rm=TRUE)
CurrentLabel<-"mean change in probability for each additional length: "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional length: "
```

```r
print(OA_mean_len_diff)
```

```
## [1] -0.006544043
```

```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] -0.00602086
```

```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] "No U-shape in this participant"
```

```
if(potential_u_shape){
  n_rows<-nrow(table_to_use)
  downward_dist <- numeric(n_rows)
  upward_dist <- numeric(n_rows)
  for(i in seq(1,n_rows)){
    current_row <- as.numeric(unlist(table_to_use[i,1:9]))
    current_row <- current_row[!is.na(current_row)]
    current_row_len <- length(current_row)
    row_min <- min(current_row,na.rm=TRUE)
    min_pos <- which(current_row == row_min)
    left_max <- max(current_row[1:min_pos],na.rm=TRUE)
    right_max <- max(current_row[min_pos:current_row_len])
    left_diff <- left_max - row_min
    right_diff <- right_max - row_min
    downward_dist[i]<-left_diff
    upward_dist[i]<-right_diff
  }
  print("differences from left max to min for each row: ")
  print(downward_dist)
  print("differences from min to right max for each row: ")
  print(upward_dist)

  # Use the max return upward (min of upward_dist) and then the
  # downward distance from the left for the same row

  print(" ")
  biggest_return_upward_row <- which(upward_dist == max(upward_dist))
  CurrentLabel <- "row with biggest return upward"
  print(CurrentLabel)
  print(biggest_return_upward_row)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, biggest_return_upward_row)
```

```
  print(" ")
  CurrentLabel<-"downward distance for row with the largest upward value"
  print(CurrentLabel)
  print(downward_dist[biggest_return_upward_row])
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, downward_dist[biggest_return_upwa

  CurrentLabel<-"return upward value"
  print(upward_dist[biggest_return_upward_row])
   results_report_DF <- AddReportLine(results_report_DF,
                                      CurrentLabel,
                                      upward_dist[biggest_return_upward_row])

  print(" ")
  # percentage return upward
  percentage_return_upward <- upward_dist[biggest_return_upward_row]/downward_dist[biggest_return_upwar
  CurrentLabel <- "Return upward as a proportion of the downward distance:"
  print(CurrentLabel)
  print(percentage_return_upward)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,
                         percentage_return_upward)
}else{
  print("no U-shape in this participant")
}
```

```
## [1] "no U-shape in this participant"
```

```
FLPModelEquations<-c(
  # models with log frequency, but no three-way interactions (due to difficulty interpreting and small
            "preserved ~ stimlen*log_freq",
            "preserved ~ stimlen+log_freq",
            "preserved ~ pos*log_freq",
            "preserved ~ pos+log_freq",
            "preserved ~ stimlen*log_freq + pos*log_freq",
            "preserved ~ stimlen*log_freq + pos",
            "preserved ~ stimlen + pos*log_freq",
            "preserved ~ stimlen + pos + log_freq",
            "preserved ~ (I(pos^2)+pos)*log_freq",
            "preserved ~ stimlen*log_freq + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen*log_freq + I(pos^2) + pos",
            "preserved ~ stimlen + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen + I(pos^2) + pos + log_freq",

            # models without frequency
            "preserved ~ 1",
            "preserved ~ stimlen",
            "preserved ~ pos",
            "preserved ~ stimlen + pos",
            "preserved ~ stimlen*pos",
            "preserved ~ I(pos^2)+pos",
            "preserved ~ stimlen + I(pos^2) + pos",
            "preserved ~ stimlen * (I(pos^2) + pos)"
)

FLPRes<-TestModels(FLPModelEquations,PosDat)
```

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

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      1.87964      -0.05777       0.16049      -0.02172
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3541  AIC: 3708
## log likelihood:  -1770.454
## Nagelkerke R2:  0.009994521
## % pres/err predicted correctly:  -1098.004
## % of predictable range [ (model-null)/(1-null) ]:  0.005851881
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.01849      -0.02131      -0.05126       0.15216      -0.02079
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4127 Residual
## Null Deviance:       3565
## Residual Deviance: 3540  AIC: 3709
## log likelihood:  -1770.19
```

```
## Nagelkerke R2:  0.01021459
## % pres/err predicted correctly:  -1097.794
## % of predictable range [ (model-null)/(1-null) ]:  0.006041614
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.008185         -0.021234          0.199798         -0.050163         -0.007182
##     log_freq:pos
##        -0.018643
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4126 Residual
## Null Deviance:       3565
## Residual Deviance: 3540  AIC: 3711
## log likelihood:  -1770.075
## Nagelkerke R2:  0.01031012
## % pres/err predicted correctly:  -1097.739
## % of predictable range [ (model-null)/(1-null) ]:  0.006091036
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     1.85859     -0.05052      0.07431
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3545  AIC: 3711
## log likelihood:  -1772.618
## Nagelkerke R2:  0.008192282
## % pres/err predicted correctly:  -1099.134
## % of predictable range [ (model-null)/(1-null) ]:  0.004829197
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.04284     -0.02805     -0.04245      0.06820
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3544  AIC: 3712
## log likelihood:  -1772.152
## Nagelkerke R2:  0.008579959
```

```
## % pres/err predicted correctly:  -1098.792
## % of predictable range [ (model-null)/(1-null) ]:  0.005138427
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.01057          -0.02572           0.20052          -0.04244          -0.01688
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4127 Residual
## Null Deviance:       3565
## Residual Deviance: 3543  AIC: 3712
## log likelihood:  -1771.4
## Nagelkerke R2:  0.009206699
## % pres/err predicted correctly:  -1098.403
## % of predictable range [ (model-null)/(1-null) ]:  0.0054907
## **************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##         1.8062763         -0.0050865         -0.0131139          0.1648526          0.0002412
##      pos:log_freq
##        -0.0241921
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4126 Residual
## Null Deviance:       3565
## Residual Deviance: 3541  AIC: 3712
## log likelihood:  -1770.276
## Nagelkerke R2:  0.01014245
## % pres/err predicted correctly:  -1097.933
## % of predictable range [ (model-null)/(1-null) ]:  0.005915996
## **************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##         1.9438872         -0.0188163         -0.0040363         -0.0165467          0.1585063
## I(pos^2):log_freq       pos:log_freq
##         0.0003229         -0.0240206
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4125 Residual
## Null Deviance:       3565
## Residual Deviance: 3540  AIC: 3714
## log likelihood:  -1770.078
```

```
## Nagelkerke R2:  0.01030788
## % pres/err predicted correctly:  -1097.761
## % of predictable range [ (model-null)/(1-null) ]:  0.00607109
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    2.002446    -0.026787    -0.002225    -0.023258     0.068276
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4127 Residual
## Null Deviance:       3565
## Residual Deviance: 3544  AIC: 3714
## log likelihood:  -1772.12
## Nagelkerke R2:  0.008606693
## % pres/err predicted correctly:  -1098.785
## % of predictable range [ (model-null)/(1-null) ]:  0.005144864
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.04525     -0.04989      0.06813
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3548  AIC: 3714
## log likelihood:  -1774.139
## Nagelkerke R2:  0.00692346
## % pres/err predicted correctly:  -1099.778
## % of predictable range [ (model-null)/(1-null) ]:  0.004246377
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         1.958186         -0.024060          0.202511         -0.002857         -0.017813
## stimlen:log_freq
##        -0.017119
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4126 Residual
## Null Deviance:       3565
## Residual Deviance: 3543  AIC: 3714
## log likelihood:  -1771.347
## Nagelkerke R2:  0.009250549
```

```
## % pres/err predicted correctly:  -1098.387
## % of predictable range [ (model-null)/(1-null) ]:  0.005505139
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          2.01300          -0.04756           0.20043          -0.01687
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3547  AIC: 3714
## log likelihood:  -1773.387
## Nagelkerke R2:  0.007551047
## % pres/err predicted correctly:  -1099.404
## % of predictable range [ (model-null)/(1-null) ]:  0.004584699
## **************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##         1.9348962         -0.0188058          0.2153259         -0.0039172         -0.0163070
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##        -0.0076438          0.0007496         -0.0254860
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4124 Residual
## Null Deviance:       3565
## Residual Deviance: 3540  AIC: 3715
## log likelihood:  -1769.952
## Nagelkerke R2:  0.01041244
## % pres/err predicted correctly:  -1097.704
## % of predictable range [ (model-null)/(1-null) ]:  0.006122636
## **************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.20997     -0.05062     -0.04235
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3553  AIC: 3718
## log likelihood:  -1776.381
## Nagelkerke R2:  0.005053071
## % pres/err predicted correctly:  -1100.907
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.003225429
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    2.376179    -0.070760    -0.095042     0.006181
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3553  AIC: 3719
## log likelihood:  -1776.277
## Nagelkerke R2:  0.00513938
## % pres/err predicted correctly:  -1100.873
## % of predictable range [ (model-null)/(1-null) ]:  0.003256033
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.175351    -0.049562    -0.001914    -0.025831
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3553  AIC: 3720
## log likelihood:  -1776.357
## Nagelkerke R2:  0.005072981
## % pres/err predicted correctly:  -1100.902
## % of predictable range [ (model-null)/(1-null) ]:  0.003229658
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.21219     -0.07239
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3557  AIC: 3720
## log likelihood:  -1778.363
## Nagelkerke R2:  0.003397076
## % pres/err predicted correctly:  -1101.958
## % of predictable range [ (model-null)/(1-null) ]:  0.002274296
## ***************************
## model index:  16
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.8817      -0.0583
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3720
## log likelihood:  -1778.027
## Nagelkerke R2:  0.003678173
## % pres/err predicted correctly:  -1101.974
## % of predictable range [ (model-null)/(1-null) ]:  0.002260435
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.815654    -0.004566    -0.018119
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3722
## log likelihood:  -1777.887
## Nagelkerke R2:  0.003794809
## % pres/err predicted correctly:  -1101.912
## % of predictable range [ (model-null)/(1-null) ]:  0.002316437
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.842500         -0.136677          0.033767         -0.385984         -0.004538
##      stimlen:pos
##         0.046291
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4126 Residual
## Null Deviance:       3565
## Residual Deviance: 3552  AIC: 3723
## log likelihood:  -1775.828
## Nagelkerke R2:  0.005514248
## % pres/err predicted correctly:  -1100.659
## % of predictable range [ (model-null)/(1-null) ]:  0.003449756
## ***************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.654
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4131 Residual
## Null Deviance:       3565
## Residual Deviance: 3565  AIC: 3728
## log likelihood:  -1782.424
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1104.473
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1604906	NA	-0.0577012	-0.0717177	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0213147	0.1521555	NA	-0.0512061	-0.0207873	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0212337	0.1997983	0.0071822	-0.0501633	NA	-0.0186426	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.0743094	NA	-0.0505250	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0280456	0.0682016	NA	-0.0424508	NA	NA	NA	NA	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 ~ stimlen * log_freq + pos	3712.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0257224	0.2005242	-0.0168766	-0.0424446	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	3712.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1648526	NA	-0.01301[illegible]	-0.0[illegible]41921	NA	-0.0050865	0.0002412	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	3713.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0188163	0.1585063	NA	-0.0165[illegible]	-0.0[illegible]40206	NA	-0.0040363	0.0003229	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	3713.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0267869	0.0682764	NA	-0.0232582	NA	NA	-0.0022250	NA	NA	NA	NA
preserved ~ stimlen + log_freq	3713.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0498852	0.0681318	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	3714.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0240602	0.2025108	-0.0171187	-0.0178133	NA	NA	-0.0028566	NA	NA	NA	NA
preserved ~ stimlen * log_freq	3714.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0475609	0.2004336	-0.0168726	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	3715.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0188058	0.2153259	-0.0076438	-0.0163070	NA	-0.0254[illegible]	-0.0039172	NA	0.0007496	NA	NA
preserved ~ stimlen + pos	3717.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0506237	NA	NA	-0.0423512	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	3719.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0707598	NA	NA	-0.0950415	NA	NA	NA	NA	NA	0.0061811	NA
preserved ~ stimlen + I(pos^2) + pos	3719.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0495624	NA	NA	-0.0258310	NA	NA	-0.0019144	NA	NA	NA	NA
preserved ~ stimlen	3719.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0723875	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	3719.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.0582962	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	3721.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.0181189	NA	NA	-0.0045663	NA	NA	NA	NA

Model	AIC DeltaAIC expAIC AICw NagR2 (Intercept)	stimlen	log_freq	stimlen:pos	log_freq:pos	logfreq:pos	I(pos^2)	pos^2	log_freq:stimlen	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	3723.1458877023585002055184250 -0.1366766	NA	NA	-0.3859840	NA	NA	0.0337672 NA	NA	0.0462914	-0.0045381
preserved ~ 1	3727.6945382619006180021701065353NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

```
## [1] "preserved ~ pos * log_freq"
```

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)          pos      log_freq  pos:log_freq
##      1.87964     -0.05777       0.16049      -0.02172
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3541  AIC: 3708
```

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

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

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

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

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

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

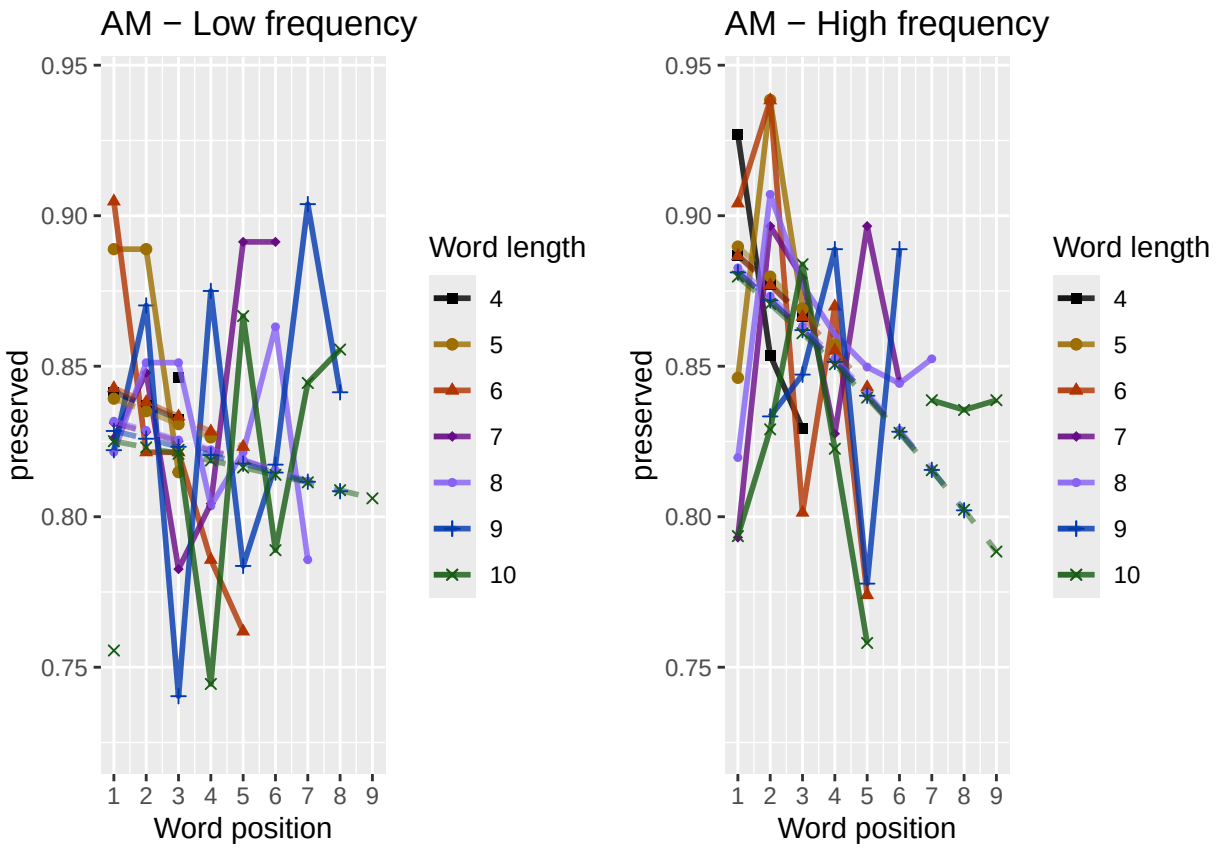
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

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

## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_point()`).

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

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




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

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0571      -0.6846
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3312  AIC: 3457
## log likelihood:  -1655.979
## Nagelkerke R2:  0.1027126
## % pres/err predicted correctly:  -1012.913
## % of predictable range [ (model-null)/(1-null) ]:  0.08282366
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.4159       0.1057
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3543  AIC: 3707
## log likelihood:  -1771.574
## Nagelkerke R2:  0.009061769
## % pres/err predicted correctly:  -1098.674
## % of predictable range [ (model-null)/(1-null) ]:  0.005245299
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.21219     -0.07239
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3557  AIC: 3720
## log likelihood:  -1778.363
## Nagelkerke R2:  0.003397076
## % pres/err predicted correctly:  -1101.958
## % of predictable range [ (model-null)/(1-null) ]:  0.002274296
## ***************************
```

```
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      1.8817      -0.0583
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3720
## log likelihood:  -1778.027
## Nagelkerke R2:  0.003678173
## % pres/err predicted correctly:  -1101.974
## % of predictable range [ (model-null)/(1-null) ]:  0.002260435
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.815654    -0.004566    -0.018119
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3722
## log likelihood:  -1777.887
## Nagelkerke R2:  0.003794809
## % pres/err predicted correctly:  -1101.912
## % of predictable range [ (model-null)/(1-null) ]:  0.002316437
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.654
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4131 Residual
## Null Deviance:       3565
## Residual Deviance: 3565  AIC: 3728
## log likelihood:  -1782.424
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1104.473
## % 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	3456.847	0.0000	1	1	0.1027120	2.057133	NA	-0.6846013	NA	NA	NA
preserved ~ CumPres	3707.150	250.3031	0	0	0.0090618	8.415940	0.1056967	NA	NA	NA	NA
preserved ~ stimlen	3719.811	262.9633	0	0	0.0033972	1.212190	NA	NA	NA	NA	-0.0723875
preserved ~ pos	3719.822	262.9743	0	0	0.0036782	1.881722	NA	NA	NA	-0.0582962	NA
preserved ~ (I(pos^2) + pos)	3721.977	265.1295	0	0	0.0037948	1.815654	NA	NA	-0.0045663	-0.0181189	NA
preserved ~ 1	3727.645	270.7978	0	0	0.0000000	0.653530	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	0.8551537	553
O	0.7984227	1902
P	0.9090909	33
S	0.6929461	241
V	0.9121644	1403

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

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

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

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

```
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.1205      -0.7734
```

```
##
## Degrees of Freedom: 3857 Total (i.e. Null);  3856 Residual
## Null Deviance:       3218
## Residual Deviance: 2972  AIC: 3106
## log likelihood:  -1485.925
## Nagelkerke R2:  0.1091194
## % pres/err predicted correctly:  -897.1309
## % of predictable range [ (model-null)/(1-null) ]:  0.09033264
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.4792       0.1129
##
## Degrees of Freedom: 3857 Total (i.e. Null);  3856 Residual
## Null Deviance:       3218
## Residual Deviance: 3197  AIC: 3352
## log likelihood:  -1598.585
## Nagelkerke R2:  0.009366691
## % pres/err predicted correctly:  -981.0752
## % of predictable range [ (model-null)/(1-null) ]:  0.005310096
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.94886     -0.05849
##
## Degrees of Freedom: 3857 Total (i.e. Null);  3856 Residual
## Null Deviance:       3218
## Residual Deviance: 3210  AIC: 3364
## log likelihood:  -1604.879
## Nagelkerke R2:  0.003620708
## % pres/err predicted correctly:  -984.1284
## % of predictable range [ (model-null)/(1-null) ]:  0.002217764
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.76926     -0.01240      0.05077
##
## Degrees of Freedom: 3857 Total (i.e. Null);  3855 Residual
## Null Deviance:       3218
```

```
## Residual Deviance: 3208  AIC: 3366
## log likelihood:  -1603.935
## Nagelkerke R2:  0.004483209
## % pres/err predicted correctly:  -983.6378
## % of predictable range [ (model-null)/(1-null) ]:  0.002714584
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.19872     -0.06229
##
## Degrees of Freedom: 3857 Total (i.e. Null);  3856 Residual
## Null Deviance:       3218
## Residual Deviance: 3212  AIC: 3366
## log likelihood:  -1606.135
## Nagelkerke R2:  0.002471552
## % pres/err predicted correctly:  -984.6497
## % of predictable range [ (model-null)/(1-null) ]:  0.001689743
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.718
##
## Degrees of Freedom: 3857 Total (i.e. Null);  3857 Residual
## Null Deviance:       3218
## Residual Deviance: 3218  AIC: 3371
## log likelihood:  -1608.834
## Nagelkerke R2:  1.962557e-16
## % pres/err predicted correctly:  -986.318
## % 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	3106.099	0.0000	1	1	0.1091192	4.120481	NA	-0.7734088	NA	NA	NA
preserved ~ CumPres	3352.318	246.2189	0	0	0.0093667	1.479169	0.1129268	NA	NA	NA	NA
preserved ~ pos	3364.355	258.2563	0	0	0.0036207	1.948859	NA	NA	NA	-0.0584882	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ (I(pos^2) + pos)	3365.536	259.4370	0	0	0.0044832	2.769264	NA	NA	-0.0123954	0.0507734	NA
preserved ~ stimlen	3366.441	260.3424	0	0	0.0024716	2.198724	NA	NA	NA	NA	-0.0622909
preserved ~ 1	3371.323	265.2241	0	0	0.0000000	0.718362	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
##      2.0700      -0.8238
##
## Degrees of Freedom: 3304 Total (i.e. Null);  3303 Residual
## Null Deviance:       2771
## Residual Deviance: 2588  AIC: 2701
## log likelihood:  -1293.776
## Nagelkerke R2:  0.09509647
## % pres/err predicted correctly:  -782.6431
## % of predictable range [ (model-null)/(1-null) ]:  0.07957135
## ****************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)          pos
##     1.96936     -0.06615
##
## Degrees of Freedom: 3304 Total (i.e. Null);  3303 Residual
## Null Deviance:       2771
## Residual Deviance: 2762  AIC: 2891
## log likelihood:  -1380.816
## Nagelkerke R2:  0.004954376
## % pres/err predicted correctly:  -847.7824
## % of predictable range [ (model-null)/(1-null) ]:  0.00306193
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.77534     -0.01376      0.05494
##
## Degrees of Freedom: 3304 Total (i.e. Null);  3302 Residual
## Null Deviance:       2771
## Residual Deviance: 2759  AIC: 2892
## log likelihood:  -1379.73
## Nagelkerke R2:  0.006109002
## % pres/err predicted correctly:  -847.1779
## % of predictable range [ (model-null)/(1-null) ]:  0.003771922
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.54763      0.08976
##
## Degrees of Freedom: 3304 Total (i.e. Null);  3303 Residual
## Null Deviance:       2771
## Residual Deviance: 2762  AIC: 2893
## log likelihood:  -1381.172
## Nagelkerke R2:  0.004575848
## % pres/err predicted correctly:  -848.0942
## % of predictable range [ (model-null)/(1-null) ]:  0.002695622
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.25861     -0.07134
##
```

```
## Degrees of Freedom: 3304 Total (i.e. Null);  3303 Residual
## Null Deviance:       2771
## Residual Deviance: 2765  AIC: 2894
## log likelihood:  -1382.384
## Nagelkerke R2:  0.003286984
## % pres/err predicted correctly:  -848.4662
## % of predictable range [ (model-null)/(1-null) ]:  0.002258696
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.709
##
## Degrees of Freedom: 3304 Total (i.e. Null);  3304 Residual
## Null Deviance:       2771
## Residual Deviance: 2771  AIC: 2900
## log likelihood:  -1385.47
## Nagelkerke R2:  1.955989e-16
## % pres/err predicted correctly:  -850.3893
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

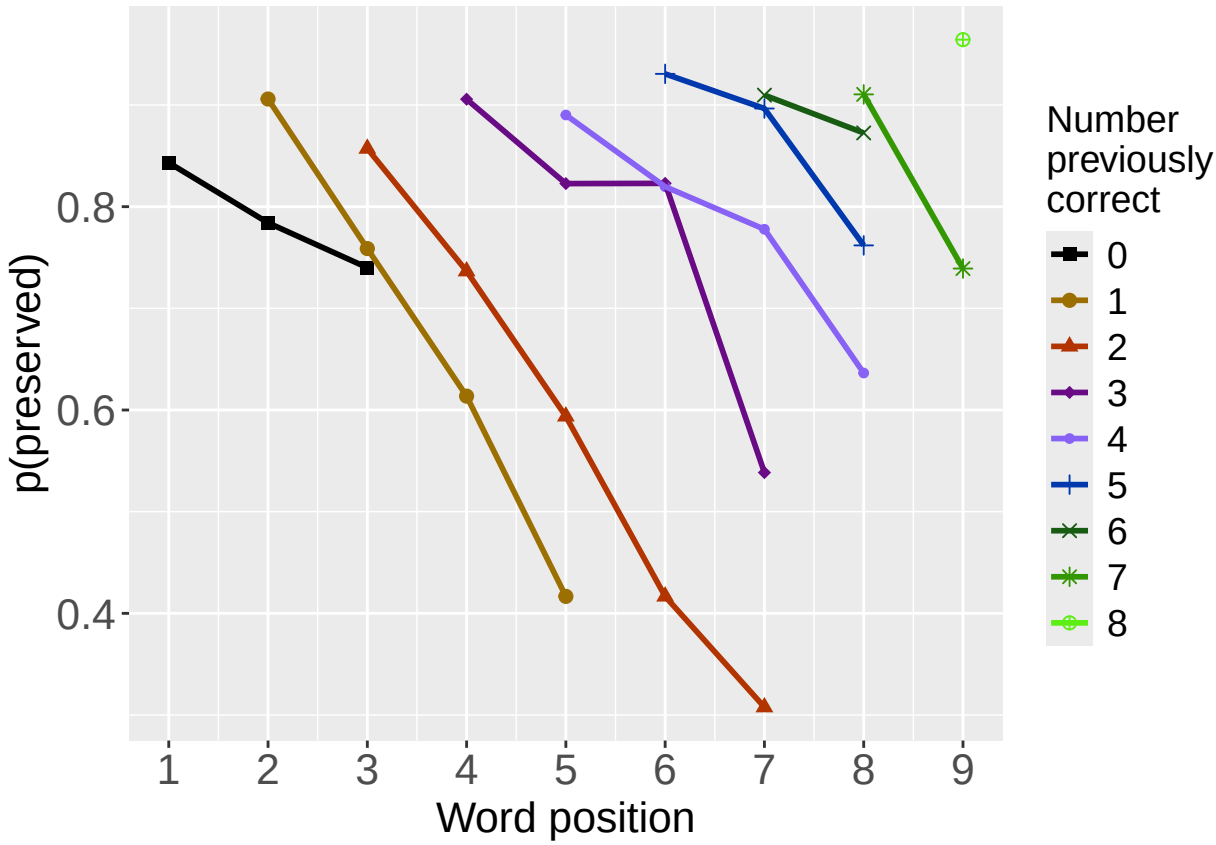
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2700.683	0.0000	1	1	0.0950963	2.069953	NA	-0.8238325	NA	NA	NA
preserved ~ pos	2891.241	190.5579	0	0	0.0049544	1.969362	NA	NA	NA	-0.0661491	NA
preserved ~ (I(pos^2) + pos)	2892.098	191.4144	0	0	0.0061090	1.775343	NA	NA	-0.013762	0.0549415	NA
preserved ~ CumPres	2893.002	192.3188	0	0	0.0045758	1.547633	0.0897591	NA	NA	NA	NA
preserved ~ stimlen	2893.948	193.2651	0	0	0.0032870	2.258606	NA	NA	NA	NA	-0.0713388
preserved ~ 1	2899.780	199.0964	0	0	0.0000000	1.709004	NA	NA	NA	NA	NA

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

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

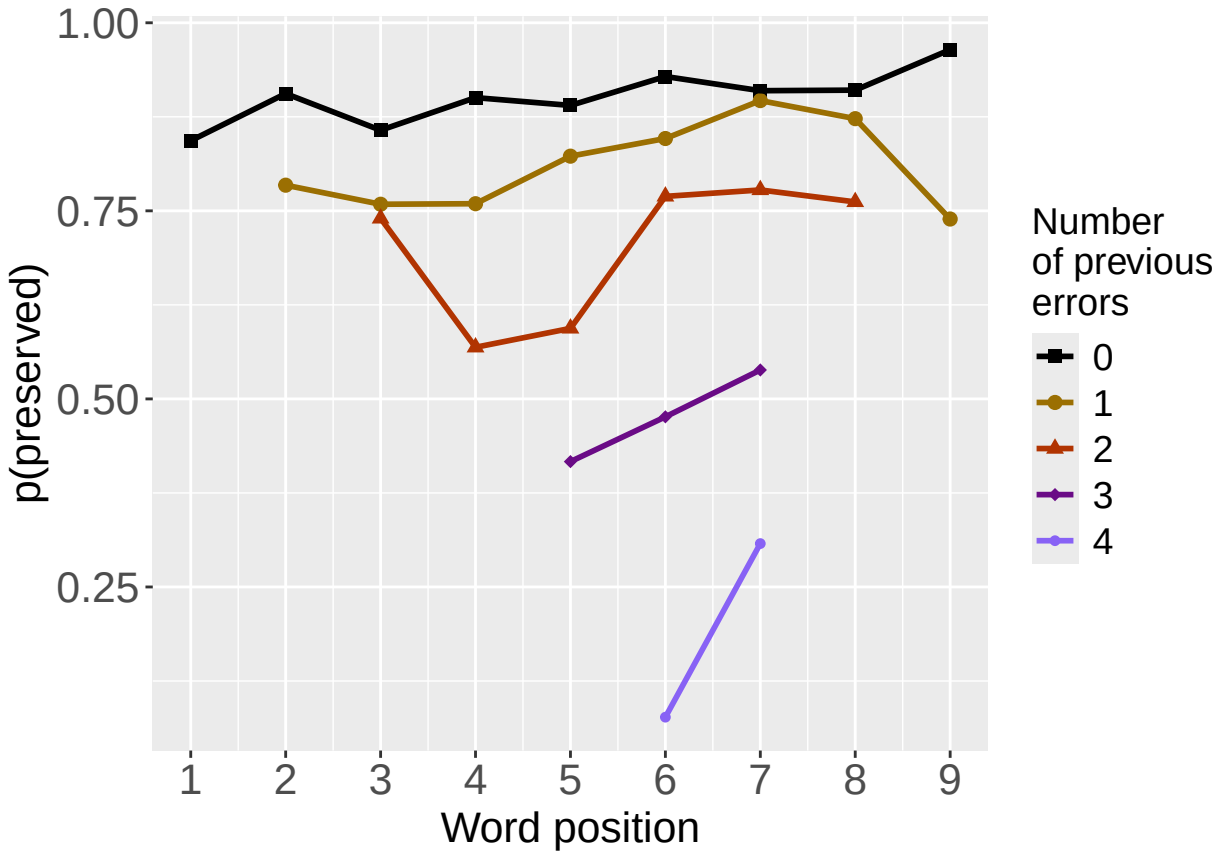
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

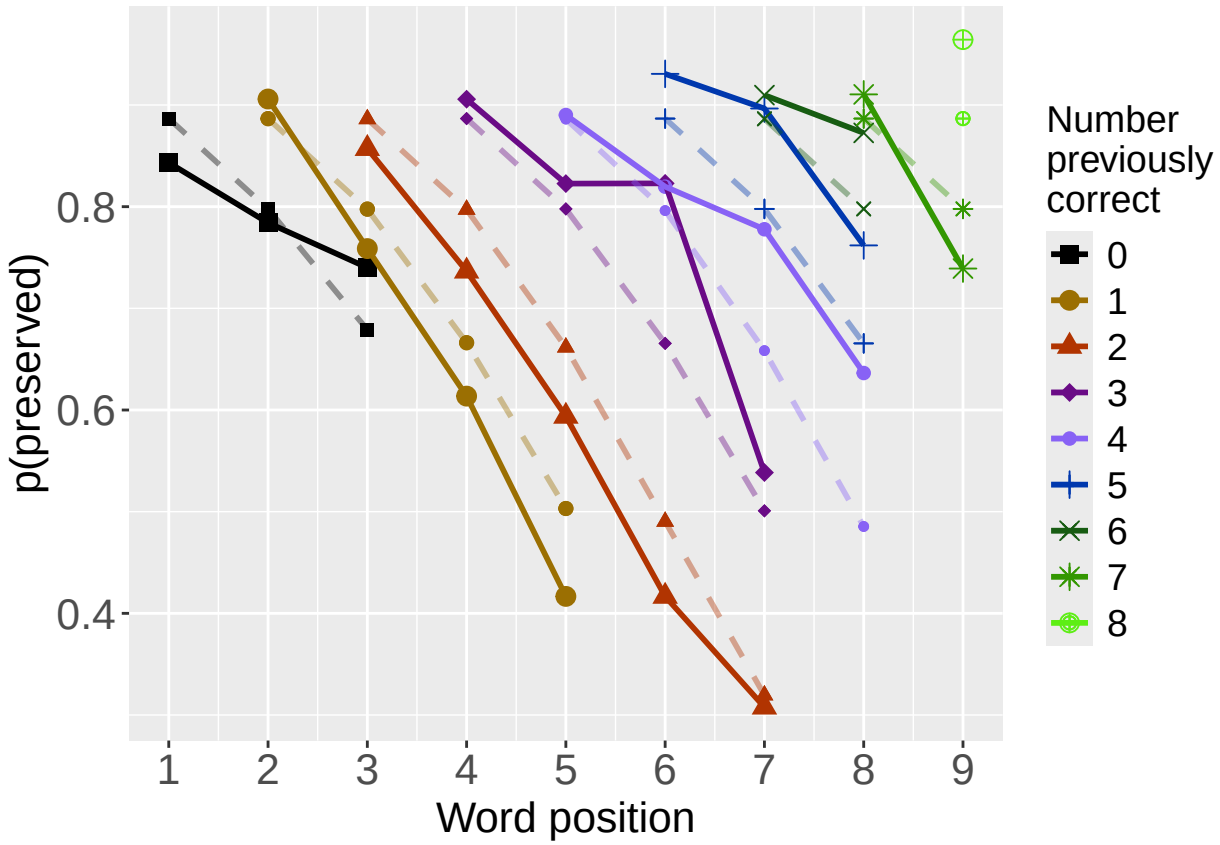
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

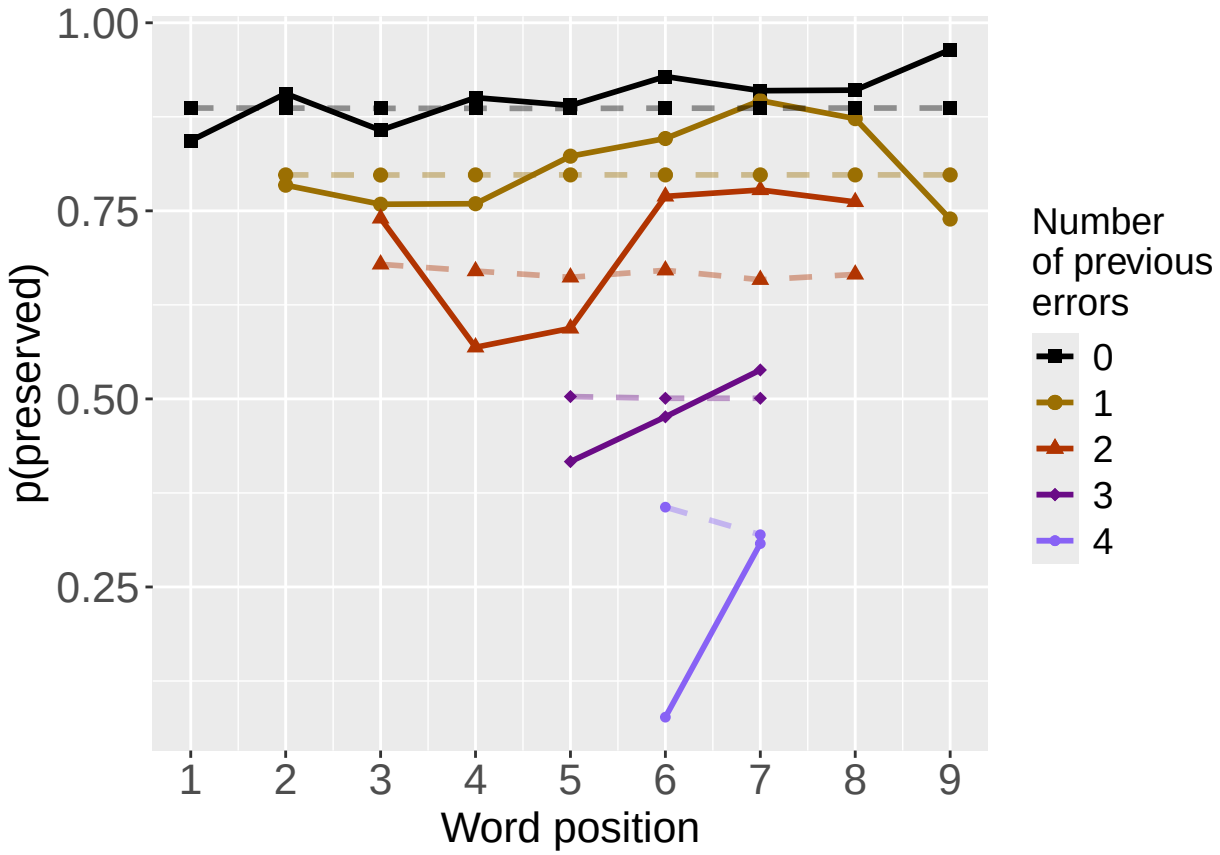
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##    1.688700    -0.824540     0.002684     0.103529
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3285  AIC: 3432
## log likelihood:  -1642.426
## Nagelkerke R2:  0.1133531
## % pres/err predicted correctly:  -1004.676
## % of predictable range [ (model-null)/(1-null) ]:  0.09027493
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0571      -0.6846
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3312  AIC: 3457
## log likelihood:  -1655.979
## Nagelkerke R2:  0.1027126
## % pres/err predicted correctly:  -1012.913
## % of predictable range [ (model-null)/(1-null) ]:  0.08282366
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    1.815654    -0.004566    -0.018119
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3722
## log likelihood:  -1777.887
## Nagelkerke R2:  0.003794809
## % pres/err predicted correctly:  -1101.912
## % of predictable range [ (model-null)/(1-null) ]:  0.002316437
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	3432.063	0.00000	1.0e+00	0.9999958	0.1133531	1.688700	-0.8245399	0.0026836	0.1035294

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	3456.847	24.78413	4.2e-06	0.0000042	0.1027126	2.057133	-0.6846013	NA	NA
preserved ~ I(posˆ2) + pos	3721.977	289.91367	0.0e+00	0.0000000	0.0037948	1.815654	NA	-0.0045663	-0.0181189

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0571      -0.6846
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3312  AIC: 3457
## log likelihood:  -1655.979
## Nagelkerke R2:  0.1027126
## % pres/err predicted correctly:  -1012.913
## % of predictable range [ (model-null)/(1-null) ]:  0.08282366
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     1.89002     -0.69377      0.02244
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3311  AIC: 3458
## log likelihood:  -1655.641
## Nagelkerke R2:  0.1029786
## % pres/err predicted correctly:  -1012.649
## % of predictable range [ (model-null)/(1-null) ]:  0.083063
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.21219     -0.07239
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3557  AIC: 3720
## log likelihood:  -1778.363
## Nagelkerke R2:  0.003397076
## % pres/err predicted correctly:  -1101.958
## % of predictable range [ (model-null)/(1-null) ]:  0.002274296
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	3456.847	0.000000	1.0000000	0.6923005	0.1027126	2.057133	-0.6846013	NA
preserved ~ CumErr + stimlen	3458.469	1.621793	0.4444594	0.3076995	0.1029786	1.890017	-0.6937653	0.0224373
preserved ~ stimlen	3719.811	262.963286	0.0000000	0.0000000	0.0033971	2.212190	NA	-0.0723875

```
########
# 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
##      1.7788      -0.6980       0.1264
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
```

```
## Null Deviance:       3565
## Residual Deviance: 3285  AIC: 3430
## log likelihood:  -1642.466
## Nagelkerke R2:  0.1133215
## % pres/err predicted correctly:  -1004.783
## % of predictable range [ (model-null)/(1-null) ]:  0.09017845
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0571      -0.6846
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3312  AIC: 3457
## log likelihood:  -1655.979
## Nagelkerke R2:  0.1027126
## % pres/err predicted correctly:  -1012.913
## % of predictable range [ (model-null)/(1-null) ]:  0.08282366
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.4159       0.1057
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3543  AIC: 3707
## log likelihood:  -1771.574
## Nagelkerke R2:  0.009061769
## % pres/err predicted correctly:  -1098.674
## % of predictable range [ (model-null)/(1-null) ]:  0.005245299
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	3430.375	0.00000	1.0e+00	0.9999982	0.1133215	1.778823	-0.6979535	0.1263621
preserved ~ CumErr	3456.847	26.47199	1.8e-06	0.0000018	0.1027126	2.057133	-0.6846013	NA
preserved ~ CumPres	3707.150	276.77506	0.0e+00	0.0000000	0.0090618	1.415940	NA	0.1056967

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      1.6525      -0.8243       0.1264
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3285  AIC: 3430
## log likelihood:  -1642.466
## Nagelkerke R2:  0.1133215
## % pres/err predicted correctly:  -1004.783
## % of predictable range [ (model-null)/(1-null) ]:  0.09017845
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0571      -0.6846
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3312  AIC: 3457
## log likelihood:  -1655.979
## Nagelkerke R2:  0.1027126
## % pres/err predicted correctly:  -1012.913
## % of predictable range [ (model-null)/(1-null) ]:  0.08282366
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      1.8817      -0.0583
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3556  AIC: 3720
## log likelihood:  -1778.027
## Nagelkerke R2:  0.003678173
## % pres/err predicted correctly:  -1101.974
## % of predictable range [ (model-null)/(1-null) ]:  0.002260435
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	3430.375	0.00000	1.0e+00	0.9999982	0.1133215	1.652461	-0.8243157	0.1263621
preserved ~ CumErr	3456.847	26.47199	1.8e-06	0.0000018	0.1027126	2.057133	-0.6846013	NA
preserved ~ pos	3719.822	289.44633	0.0e+00	0.0000000	0.0036782	1.881722	NA	-0.0582962

```
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 + pos	3430.375	0.0000001	1.000000	0.9999982	0.1133215	1.652461	-0.8243157	NA	0.1263621	NA	NA
preserved ~ CumErr + CumPres	3430.375	0.0000001	1.000000	0.9999982	0.1133215	1.5778823	-0.6979535	NA	NA	NA	0.1263621
preserved ~ CumErr + I(pos^2) + pos	3432.068	0.0000001	1.000000	0.9999995	0.1133353	1.1688700	-0.8245399	0.0026836	0.1035294	NA	NA
preserved ~ CumErr	3456.847	24.784133	0.0000042	0.0000042	0.1027126	2.057133	-0.6846013	NA	NA	NA	NA
preserved ~ CumErr	3456.847	0.0000001	1.000000	0.0692300	0.1027126	2.057133	-0.6846013	NA	NA	NA	NA
preserved ~ CumErr	3456.847	26.471994	0.0000018	0.0000018	0.1027126	2.057133	-0.6846013	NA	NA	NA	NA
preserved ~ CumErr	3456.847	26.471994	0.0000018	0.0000018	0.1027126	2.057133	-0.6846013	NA	NA	NA	NA
preserved ~ CumErr + stimlen	3458.469	1.6217930	0.4444594	0.4307699	0.1029786	1.6890017	-0.6937653	NA	NA	0.0224373	NA
preserved ~ CumPres	3707.152	276.775062	0.0000000	0.0000000	0.0009061	1.8415940	NA	NA	NA	NA	0.1056967
preserved ~ stimlen	3719.812	62.963286	0.0000000	0.0000000	0.0003397	1.2121900	NA	NA	NA	-0.0723875	NA
preserved ~ pos	3719.822	289.446328	0.0000000	0.0000000	0.0003678	1.2881722	NA	NA	-0.0582962	NA	NA
preserved ~ I(pos^2) + pos	3721.977	289.913674	0.0000000	0.0000000	0.0003794	1.8815654	NA	-0.0045663	-0.0181189	NA	NA

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq
##     1.64084     -0.81507      0.12895      0.04372
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3282  AIC: 3429
## log likelihood:  -1640.752
## Nagelkerke R2:  0.1146628
## % pres/err predicted correctly:  -1004.058
## % of predictable range [ (model-null)/(1-null) ]:  0.09083366
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      1.6525      -0.8243       0.1264
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3285  AIC: 3430
## log likelihood:  -1642.466
## Nagelkerke R2:  0.1133215
## % pres/err predicted correctly:  -1004.783
## % of predictable range [ (model-null)/(1-null) ]:  0.09017845
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen     log_freq
##     1.74366     -0.81442      0.13309     -0.01554      0.04027
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4127 Residual
## Null Deviance:       3565
## Residual Deviance: 3281  AIC: 3431
## log likelihood:  -1640.616
## Nagelkerke R2:  0.1147689
## % pres/err predicted correctly:  -1003.956
## % of predictable range [ (model-null)/(1-null) ]:  0.09092627
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen
##     1.84044     -0.82180      0.13446     -0.02871
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3284  AIC: 3431
## log likelihood:  -1641.965
## Nagelkerke R2:  0.1137138
## % pres/err predicted correctly:  -1004.482
## % of predictable range [ (model-null)/(1-null) ]:  0.09045071
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       1.654
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4131 Residual
## Null Deviance:       3565
## Residual Deviance: 3565  AIC: 3728
## log likelihood:  -1782.424
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1104.473
## % 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	pos	log_freq	stimlen
preserved ~ CumErr + pos + log_freq	3429.284	0.000000	1.000000	0.3976805	0.1146628	1.640838	-0.8150712	0.1289505	0.0437155	NA
preserved ~ CumErr + pos	3430.375	1.091424	0.579429	0.2304310	0.1133215	1.652461	-0.8243157	0.1263621	NA	NA
preserved ~ CumErr + pos + stimlen + log_freq	3430.724	1.439976	0.486758	0.1935770	0.1147689	1.743659	-0.8144220	0.1330943	0.0402661	-0.0155425
preserved ~ CumErr + pos + stimlen	3430.888	1.604335	0.448356	0.1783052	0.1137138	1.840442	-0.8217967	0.1344607	NA	-0.0287115
preserved ~ 1	3727.645	298.361172	0.000000	0.0000000	0.0000000	1.653530	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!
```

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

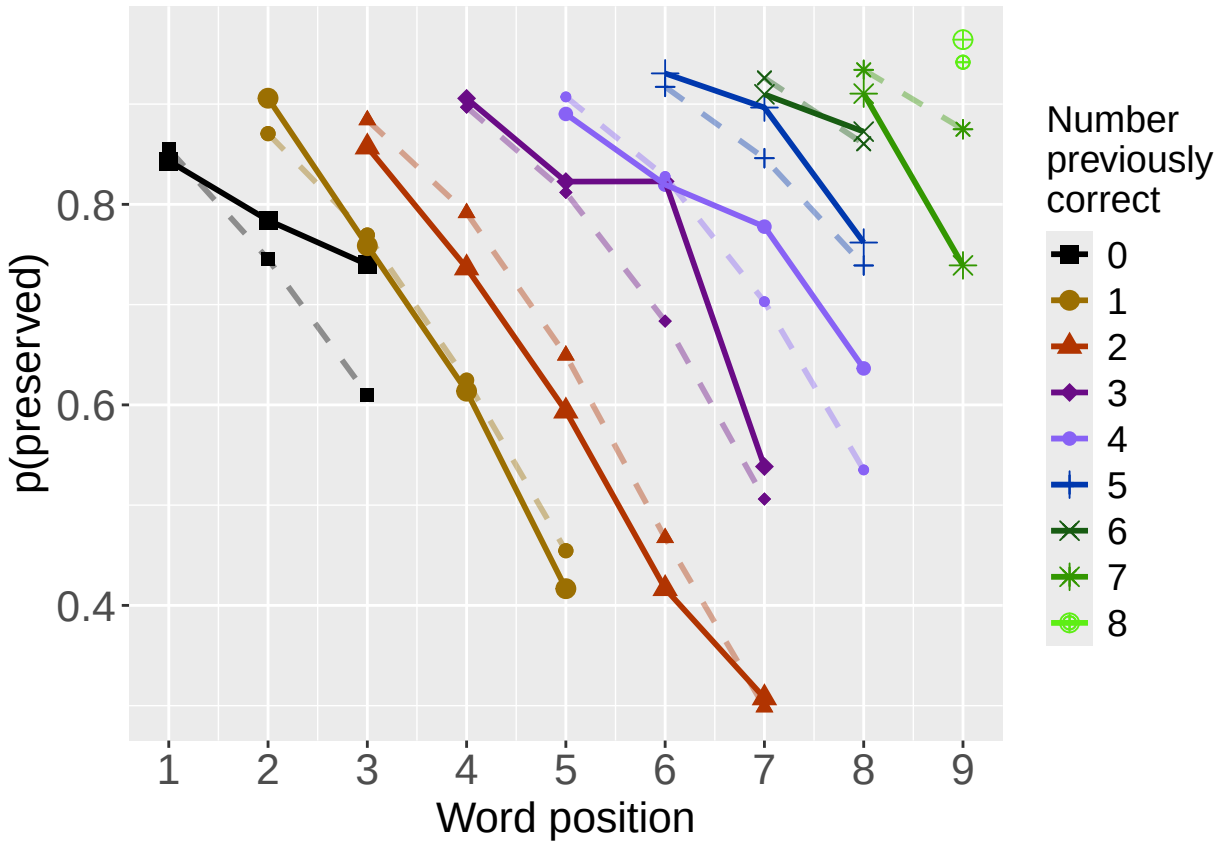
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + pos + log_freq
##          Df Deviance    AIC
## CumErr    1   3545.2 3691.0
## pos       1   3309.6 3455.4
## log_freq  1   3284.9 3430.7
## <none>        3281.5 3429.3
```

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

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

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

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

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

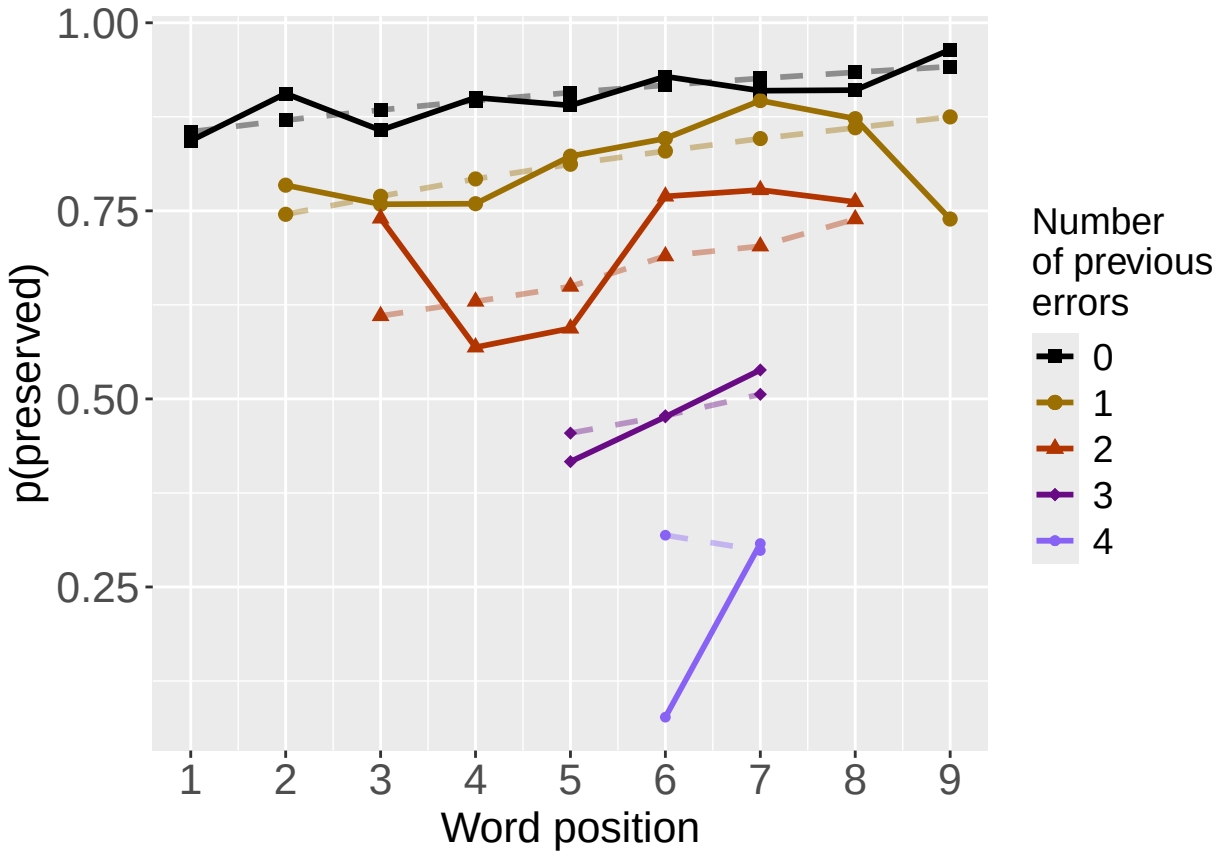
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

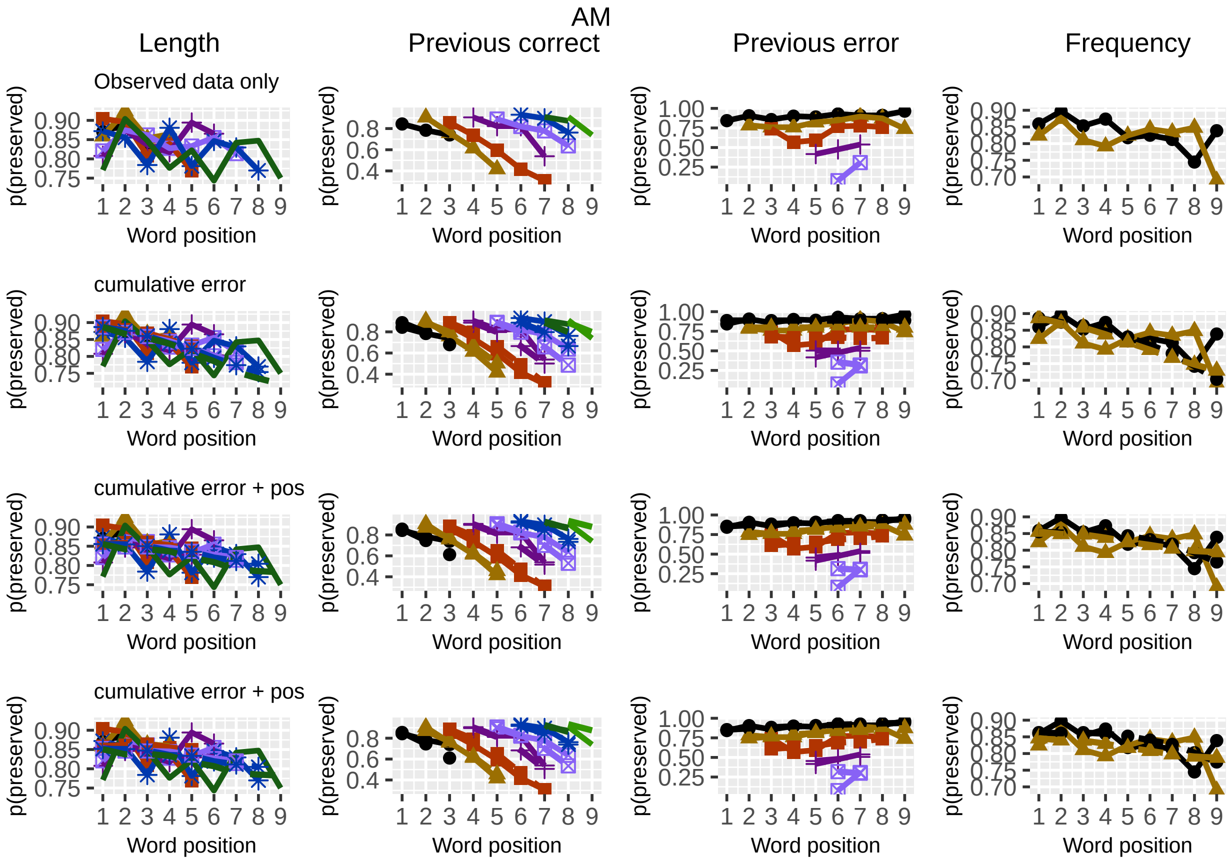
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0571      -0.6846
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4130 Residual
## Null Deviance:       3565
## Residual Deviance: 3312  AIC: 3457
## log likelihood:  -1655.979
## Nagelkerke R2:  0.1027126
```

```
## % pres/err predicted correctly:  -1012.913
## % of predictable range [ (model-null)/(1-null) ]:  0.08282366
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      1.6525      -0.8243       0.1264
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4129 Residual
## Null Deviance:       3565
## Residual Deviance: 3285  AIC: 3430
## log likelihood:  -1642.466
## Nagelkerke R2:  0.1133215
## % pres/err predicted correctly:  -1004.783
## % of predictable range [ (model-null)/(1-null) ]:  0.09017845
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq
##     1.64084     -0.81507      0.12895      0.04372
##
## Degrees of Freedom: 4131 Total (i.e. Null);  4128 Residual
## Null Deviance:       3565
## Residual Deviance: 3282  AIC: 3429
## log likelihood:  -1640.752
## Nagelkerke R2:  0.1146628
## % pres/err predicted correctly:  -1004.058
## % of predictable range [ (model-null)/(1-null) ]:  0.09083366
## ***************************
## `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()`).
```

```
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!
```

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

```
kable(DAContributionAverage)
```

	CumErr	pos	log_freq
McFadden	0.0745598	0.0051283	0.0020054
SquaredCorrelation	0.0648465	0.0043834	0.0017822
Nagelkerke	0.0648465	0.0043834	0.0017822
Estrella	0.0675011	0.0046523	0.0018113

	deviance	deviance_explained
CumErr + pos + log_freq	3281.504	283.3435
CumErr + pos	3284.933	279.9142
CumErr	3311.957	252.8899
null	3564.847	0.0000

```
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!
```

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

```
##                                         model deviance deviance_explained percent_explained
## CumErr + pos + log_freq CumErr + pos + log_freq 3281.504          283.3435          7.948265
## CumErr + pos                       CumErr + pos 3284.933          279.9142          7.852067
## CumErr                                 CumErr 3311.957          252.8899          7.093991
## null                                     null 3564.847            0.0000          0.000000
##                         percent_of_explained_deviance increment_in_explained
## CumErr + pos + log_freq                     100.00000              1.210308
## CumErr + pos                                 98.78969              9.537623
## CumErr                                       89.25207             89.252069
## null                                               NA              0.000000
```

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

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

```
NagContributions <- t(DAContributionAverage[3,])
NagTotal<-sum(NagContributions)
NagPercents <- NagContributions / NagTotal
NagResultTable <- data.frame(NagContributions = NagContributions, NagPercents = NagPercents)
names(NagResultTable)<-c("NagContributions","NagPercents")
write.csv(NagResultTable,paste0(TablesDir,CurPat,"_",CurTask,"_nagelkerke_contributions_table.csv"),row.names = TRUE)
```

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + log_freq	7.948265	100.00000	1.210308
CumErr + pos	7.852067	98.78969	9.537623
CumErr	7.093991	89.25207	89.252069
null	0.000000	NA	0.000000

```
NagPercents
```

```
##          Nagelkerke
## CumErr   0.91317566
## pos      0.06172767
## log_freq 0.02509667
```

```
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
```

```
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 diff_CumErr+pos
## 1              preserved ~ CumErr       0.8373284       3311.957  0.00000000   -0.0510543062
## 2          preserved ~ CumErr+pos       0.8883827       3284.933  0.05105431    0.0000000000
## 3 preserved ~ CumErr+pos+log_freq       0.8893418       3281.504  0.05201342    0.0009591175
##   diff_CumErr+pos+log_freq
## 1            -0.0520134237
## 2            -0.0009591175
## 3             0.0000000000
```

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.8373284	3311.957
preserved ~ CumErr+pos	0.8883827	3284.933
preserved ~ CumErr+pos+log_freq	0.8893418	3281.504

model	diff_CumErr	diff_CumErr+pos	diff_CumErr+pos+log_freq
preserved ~ CumErr	0.0000000	-0.0510543	-0.0520134
preserved ~ CumErr+pos	0.0510543	0.0000000	-0.0009591
preserved ~ CumErr+pos+log_freq	0.0520134	0.0009591	0.0000000

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

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