MS - repetition - Serial position analysis (v5)

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

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

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

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

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

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

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

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

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	531	29	128	NA	NA	688
2	61	NA	424	96	107	688
3	305	NA	165	204	14	688
4	295	NA	231	64	37	627
5	221	NA	208	71	35	535
6	203	1	129	69	21	423
7	169	NA	100	26	18	313
8	86	NA	52	25	4	167
9	72	NA	2	NA	4	78

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7718023	0.0421512	0.1860465	NA	NA	688
2	0.0886628	NA	0.6162791	0.1395349	0.1555233	688
3	0.4433140	NA	0.2398256	0.2965116	0.0203488	688
4	0.4704944	NA	0.3684211	0.1020734	0.0590112	627
5	0.4130841	NA	0.3887850	0.1327103	0.0654206	535
6	0.4799054	0.0023641	0.3049645	0.1631206	0.0496454	423

pos_factor	O	P	V	1	S	total
7	0.5399361	NA	0.3194888	0.0830671	0.0575080	313
8	0.5149701	NA	0.3113772	0.1497006	0.0239521	167
9	0.9230769	NA	0.0256410	NA	0.0512821	78

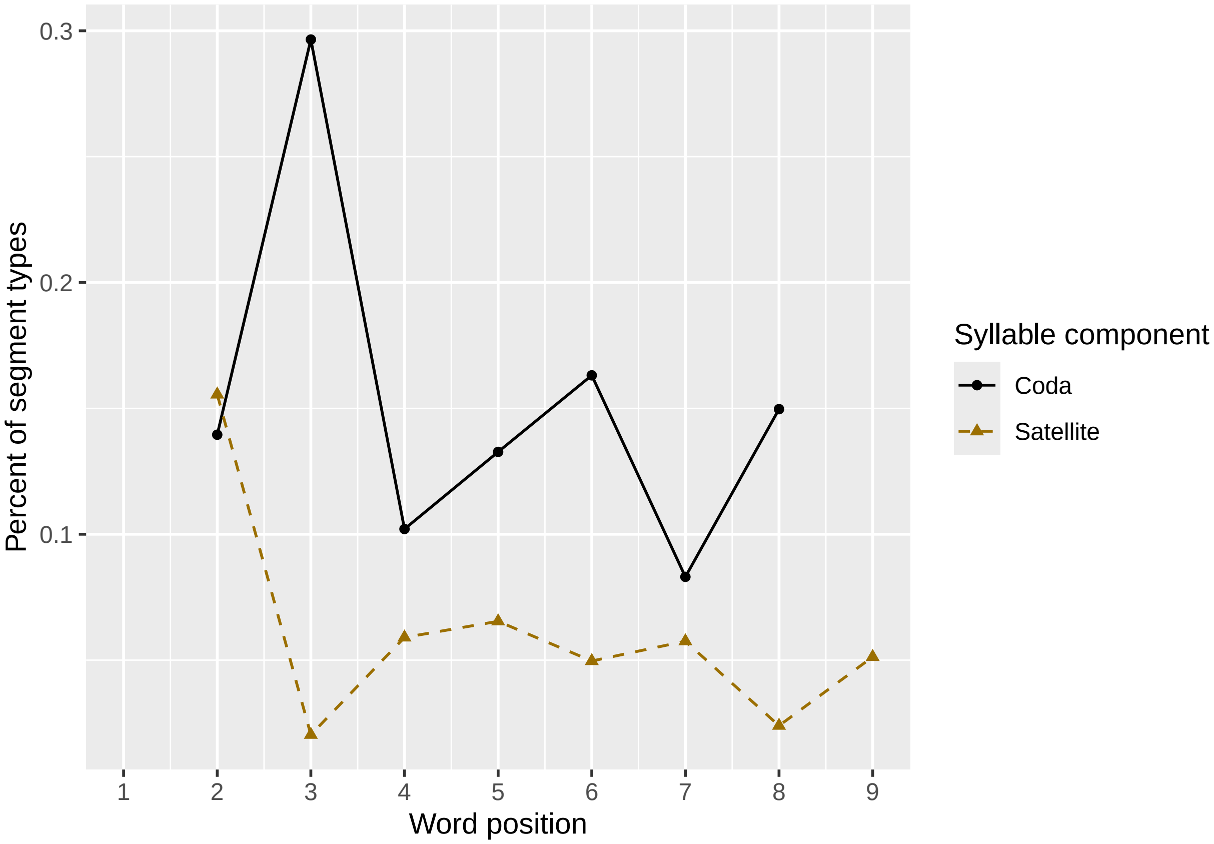
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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



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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.984 0.984 0.984 NA     NA     NA     NA     NA     NA
## 2       5 0.913 0.957 0.940  0.940 NA     NA     NA     NA     NA
## 3       6 0.982 0.964 0.942  0.955  0.920 NA     NA     NA     NA
## 4       7 0.964 0.932 0.909  0.927  0.932  0.927 NA     NA     NA
## 5       8 0.947 0.901 0.914  0.817  0.844  0.828  0.823 NA     NA
## 6       9 0.938 0.843 0.831  0.770  0.824  0.777  0.760  0.751 NA
## 7      10 0.957 0.915 0.923  0.880  0.850  0.821  0.806  0.838  0.829
```

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

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

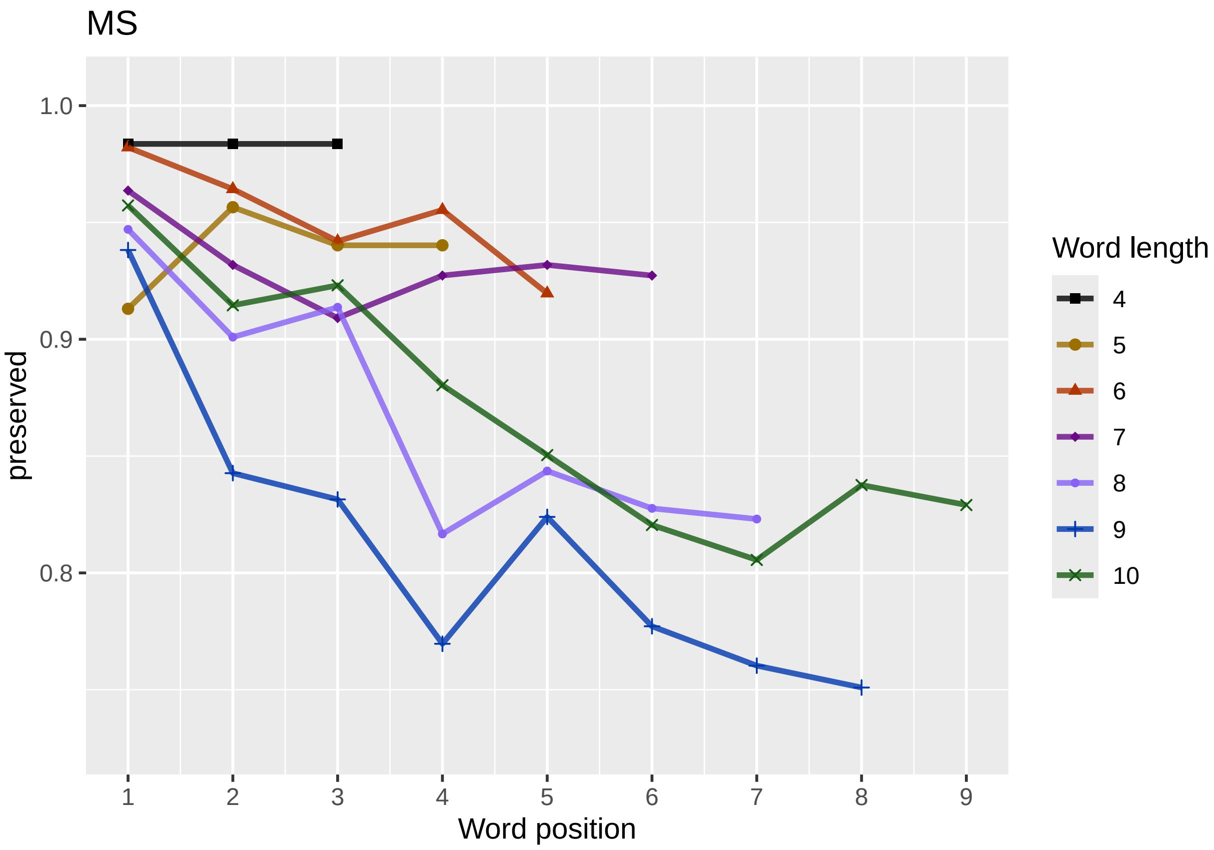
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1       4    61    61    61    NA    NA    NA    NA    NA    NA
## 2       5    92    92    92    92    NA    NA    NA    NA    NA
## 3       6   112   112   112   112   112    NA    NA    NA    NA
## 4       7   110   110   110   110   110   110    NA    NA    NA
## 5       8   146   146   146   146   146   146   146    NA    NA
## 6       9    89    89    89    89    89    89    89    89    NA
## 7      10    78    78    78    78    78    78    78    78    78
```

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

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

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



Length and position

```
# length and position

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

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

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

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

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.01285     -0.21958      0.03131     -0.44263
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 2673  AIC: 2858
## log likelihood:  -1336.702
## Nagelkerke R2:  0.06385472
## % pres/err predicted correctly:  -782.3133
## % of predictable range [ (model-null)/(1-null) ]:  0.03333209
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          3.01491           0.01179          -0.12464           0.80620           0.01733
##      stimlen:pos
##         -0.14131
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4201 Residual
## Null Deviance:       2806
## Residual Deviance: 2669  AIC: 2858
## log likelihood:  -1334.405
## Nagelkerke R2:  0.06602675
## % pres/err predicted correctly:  -781.1732
## % of predictable range [ (model-null)/(1-null) ]:  0.0347391
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     5.18426     -0.29899     -0.39068      0.02698
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 2680  AIC: 2866
## log likelihood:  -1340.082
## Nagelkerke R2:  0.06065364
## % pres/err predicted correctly:  -783.5009
## % of predictable range [ (model-null)/(1-null) ]:  0.03186653
## ***************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      4.3317      -0.1996      -0.1534
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 2683  AIC: 2868
## log likelihood:  -1341.346
## Nagelkerke R2:  0.05945542
## % pres/err predicted correctly:  -783.6142
## % of predictable range [ (model-null)/(1-null) ]:  0.03172665
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.36526      0.02025     -0.40714
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 2710  AIC: 2898
## log likelihood:  -1355.226
## Nagelkerke R2:  0.04624884
## % pres/err predicted correctly:  -789.881
## % of predictable range [ (model-null)/(1-null) ]:  0.02399298
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.0116      -0.2164
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2714  AIC: 2902
## log likelihood:  -1357.219
## Nagelkerke R2:  0.044345
## % pres/err predicted correctly:  -790.0866
## % of predictable range [ (model-null)/(1-null) ]:  0.02373923
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      4.3544      -0.2837
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2724  AIC: 2904
## log likelihood:  -1361.802
## Nagelkerke R2:  0.03996133
## % pres/err predicted correctly:  -792.8356
## % of predictable range [ (model-null)/(1-null) ]:  0.02034676
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##         2.1
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4206 Residual
## Null Deviance:       2806
## Residual Deviance: 2806  AIC: 2990
## log likelihood:  -1403.123
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -809.323
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	2857.904	0.0000000	1.0000000	0.5017882	0.0638544	4.7012852	-0.2195777	-0.4426336	NA	0.0313124	NA
preserved ~ stimlen * (I(pos^2) + pos)	2857.964	0.0601862	0.9703552	0.4869127	0.0660263	3.7014907	0.0117930	0.8061966	-0.1413132	-0.1246396	0.0173297

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	2866.368	8.464260	0.0145210	0.0072867	0.0606553	5.184263	-0.2989929	-0.3906788	0.0269816	NA	NA
preserved ~ stimlen + pos	2867.561	9.657567	0.0079962	0.0040124	0.0594554	4.331697	-0.1996427	-0.1534431	NA	NA	NA
preserved ~ I(pos^2) + pos	2898.465	40.561145	0.0000000	0.0000000	0.0462488	3.365260	NA	-0.4071437	NA	0.0202477	NA
preserved ~ pos	2901.708	43.804198	0.0000000	0.0000000	0.0443450	3.011649	NA	-0.2164271	NA	NA	NA
preserved ~ stimlen	2903.792	45.888634	0.0000000	0.0000000	0.0399641	3.354417	-0.2836826	NA	NA	NA	NA
preserved ~ 1	2989.971	132.067567	0.0000000	0.0000000	0.0000000	2.099764	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.01285     -0.21958      0.03131     -0.44263
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 2673  AIC: 2858
```

```
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.
```

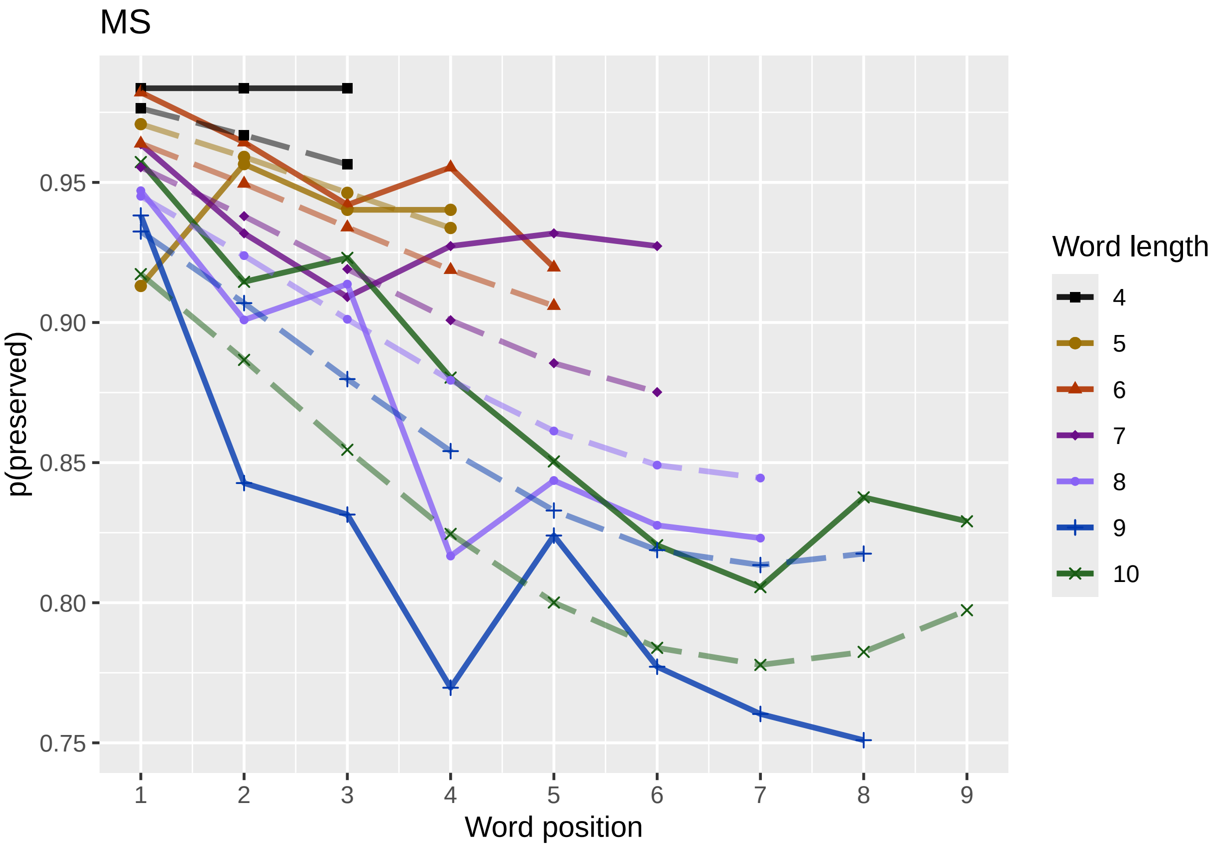
```
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # 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 0.976 0.967 0.956 NA     NA     NA     NA     NA     NA
## 2       5 0.971 0.959 0.946  0.934 NA     NA     NA     NA     NA
## 3       6 0.964 0.950 0.934  0.919  0.906 NA     NA     NA     NA
## 4       7 0.955 0.938 0.919  0.901  0.885  0.875 NA     NA     NA
## 5       8 0.945 0.924 0.901  0.879  0.861  0.849  0.845 NA     NA
## 6       9 0.932 0.907 0.880  0.854  0.833  0.819  0.813  0.818 NA
## 7      10 0.917 0.887 0.855  0.825  0.800  0.784  0.778  0.782  0.797
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       66   688
```

```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 66 / 688 = 9.59 percent"
```

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

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

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

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

```
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.28213     -0.26175      0.04795     -0.35071
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3966 Residual
## Null Deviance:       1699
## Residual Deviance: 1660  AIC: 1804
## log likelihood:  -829.9947
## Nagelkerke R2:  0.02840694
## % pres/err predicted correctly:  -427.3655
## % of predictable range [ (model-null)/(1-null) ]:  0.01105076
## ***************************
## 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)
##          2.97838           0.02243          -0.10298           0.99393           0.01798
##      stimlen:pos
##         -0.16280
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3964 Residual
## Null Deviance:       1699
## Residual Deviance: 1656  AIC: 1805
## log likelihood:  -828.204
## Nagelkerke R2:  0.03097061
## % pres/err predicted correctly:  -427.0995
## % of predictable range [ (model-null)/(1-null) ]:  0.01166493
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     4.48023     -0.24015      0.05056
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3967 Residual
## Null Deviance:       1699
## Residual Deviance: 1670  AIC: 1814
## log likelihood:  -834.921
## Nagelkerke R2:  0.02134196
## % pres/err predicted correctly:  -428.6053
## % of predictable range [ (model-null)/(1-null) ]:  0.008188477
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.4932      -0.2177
```

```
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3968 Residual
## Null Deviance:       1699
## Residual Deviance: 1672  AIC: 1815
## log likelihood:  -836.1103
## Nagelkerke R2:  0.01963374
## % pres/err predicted correctly:  -428.8732
## % of predictable range [ (model-null)/(1-null) ]:  0.007570072
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     4.83260     -0.28140     -0.06532      0.01327
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3966 Residual
## Null Deviance:       1699
## Residual Deviance: 1670  AIC: 1815
## log likelihood:  -834.7607
## Nagelkerke R2:  0.02157214
## % pres/err predicted correctly:  -428.566
## % of predictable range [ (model-null)/(1-null) ]:  0.008279166
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      3.2864       0.0339      -0.2991
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3967 Residual
## Null Deviance:       1699
## Residual Deviance: 1694  AIC: 1840
## log likelihood:  -847.1558
## Nagelkerke R2:  0.003719732
## % pres/err predicted correctly:  -431.654
## % of predictable range [ (model-null)/(1-null) ]:  0.00115008
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.782
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3969 Residual
## Null Deviance:       1699
## Residual Deviance: 1699  AIC: 1842
## log likelihood:  -849.7288
## Nagelkerke R2:  6.376242e-16
## % pres/err predicted correctly:  -432.1522
## % 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
##     2.83096     -0.01323
##
## Degrees of Freedom: 3969 Total (i.e. Null);  3968 Residual
## Null Deviance:       1699
## Residual Deviance: 1699  AIC: 1844
## log likelihood:  -849.6441
## Nagelkerke R2:  0.0001225123
## % pres/err predicted correctly:  -432.1201
## % of predictable range [ (model-null)/(1-null) ]:  7.390891e-05
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	1803.928	0.000000	1.0000000	0.6183398	0.0284065	5.282128	-0.2617528	-0.3507092	NA	0.0479517	NA
preserved ~ stimlen * (I(pos^2) + pos)	1804.941	1.013072	0.6025792	0.3725987	0.0309702	6.978379	0.0224342	0.9939326	-0.1628010	-0.1029819	0.0179843
preserved ~ stimlen + pos	1813.961	10.032360	0.0066298	0.0040995	0.0213442	4.480226	-0.2401457	0.0505583	NA	NA	NA
preserved ~ stimlen	1814.585	10.657588	0.0048499	0.0029989	0.0196347	4.493168	-0.2176988	NA	NA	NA	NA
preserved ~ stimlen * pos	1815.433	11.505056	0.0031747	0.0019631	0.0215741	4.832605	-0.2814008	-0.0653203	0.0132688	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	1839.636	5.708103	0.0000000	0.0000000	0.0037193	7.286421	NA	-0.2991051	NA	0.0338970	NA
preserved ~ 1	1842.153	8.223755	0.0000000	0.0000000	0.0000000	2.781554	NA	NA	NA	NA	NA
preserved ~ pos	1843.979	10.051961	0.0000000	0.0000000	0.0001225	2.830964	NA	-0.0132285	NA	NA	NA

```r
# plot no fragment data

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

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

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

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

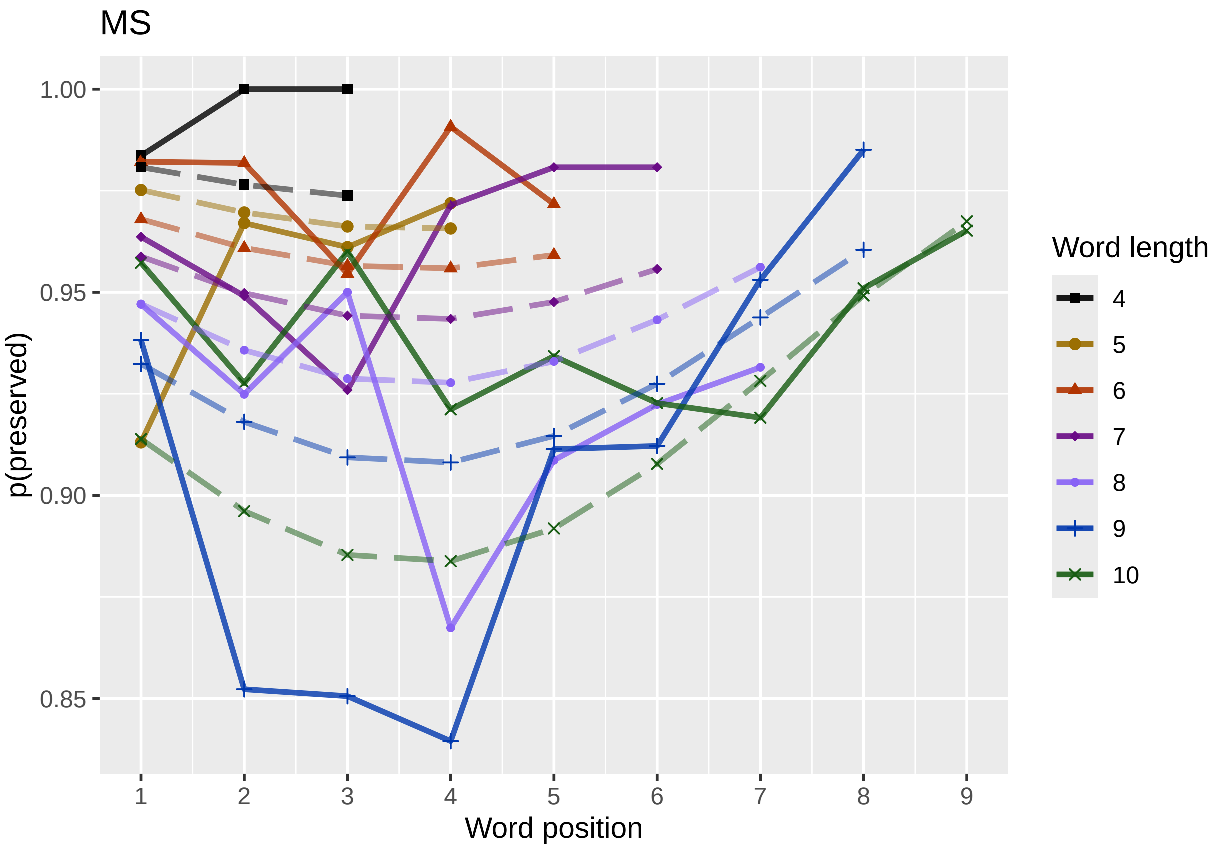
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.981 0.976 0.974 NA     NA     NA     NA     NA     NA
## 2       5 0.975 0.970 0.966  0.966 NA     NA     NA     NA     NA
## 3       6 0.968 0.961 0.957  0.956  0.959 NA     NA     NA     NA
## 4       7 0.959 0.950 0.944  0.943  0.948  0.956 NA     NA     NA
## 5       8 0.947 0.936 0.929  0.928  0.933  0.943  0.956 NA     NA
## 6       9 0.932 0.918 0.909  0.908  0.915  0.927  0.944  0.960 NA
## 7      10 0.914 0.896 0.885  0.884  0.892  0.908  0.928  0.949  0.967
```

```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 - 1.01"
```

```
# find the average difference between lengths and the average difference
# between positions taking the other factor into account

# choose fitted or raw -- we use fitted values because they are smoothed averages
# that take into account the influence of the other variable
table_to_use <- fitted_pos_len_table
# take away column with length information
table_to_use <- table_to_use[,2:ncol(table_to_use)]

# take averages for positions
# don't want downward estimates influenced by return upward of U
# therefore, for downward influence, use only the values before the min
# take the difference between each value (differences between position proportion correct) **NOTE** pro
# average the difference in probabilities

# in case min is close to the end or we are not using a min (for non-U-shaped)
# functions, we need to get differences _first_ and then average those (e.g.
```

```r
# if there is an effect of length and we just average position data,
# later positions) will have a lower average (because of the length effect)
# even if all position functions go upward

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.01745796
```

```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] -0.01601546
```

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

```r
if(potential_u_shape){ # downward, so potential U-shape
  # take only positive values from the table
  filtered_pos_upward_u<-table_pos_diffs
  filtered_pos_upward_u[table_pos_diffs <= 0] <- NA
  average_pos_u_diffs <- apply(filtered_pos_upward_u,
                               2,mean,na.rm=TRUE)
  OA_mean_pos_u_diff <- mean(average_pos_u_diffs,na.rm=TRUE)
  if(is.nan(OA_mean_pos_u_diff) | (OA_mean_pos_u_diff <= 0)){
    print("No U-shape in this participant")
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
    potential_u_shape <- FALSE
  }else{
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 1)

    CurrentLabel<-"Average upward change after U minimum"
    print(CurrentLabel)
    print(OA_mean_pos_u_diff)
    results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, OA_mean_pos_u_diff)

    CurrentLabel<-"Proportion of average downward change"
    prop_ave_down <- abs(OA_mean_pos_u_diff/OA_mean_pos_diff)
    results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, prop_ave_down)
  }
}else{
  print("No U-shape in this participant")
  results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
}
```

```
## [1] "Average upward change after U minimum"
## [1] 0.009616732
```

```r
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_upwar

  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_upward
  CurrentLabel <- "Return upward as a proportion of the downward distance:"
  print(CurrentLabel)
  print(percentage_return_upward)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,
                        percentage_return_upward)
}else{
  print("no U-shape in this participant")
}
```

```
## [1] "differences from left max to min for each row: "
## [1] 0.01999982 0.03706263 0.05793915 0.08025712 0.10055452 0.11903339 0.13945546
## [1] "differences from min to right max for each row: "
## [1] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.004070529 0.019518221
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.1394555
## [1] 0.01951822
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.1399603
```

```
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(posˆ2)+pos",
            "preserved ~ stimlen + I(posˆ2) + pos",
            "preserved ~ stimlen ∗ (I(posˆ2) + pos)"
)

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

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

```
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          4.78526          -0.19476           0.51175           0.02970          -0.42766
## stimlen:log_freq
##         -0.05058
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4201 Residual
## Null Deviance:       2806
## Residual Deviance: 2654  AIC: 2842
## log likelihood:  -1326.922
## Nagelkerke R2:  0.07308758
## % pres/err predicted correctly:  -778.3903
## % of predictable range [ (model-null)/(1-null) ]:  0.03817331
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          4.752292          -0.198252           0.362637           0.027558          -0.401710
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.055566          -0.006735           0.078950
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4199 Residual
## Null Deviance:       2806
## Residual Deviance: 2651  AIC: 2844
## log likelihood:  -1325.474
## Nagelkerke R2:  0.07445135
## % pres/err predicted correctly:  -777.6119
## % of predictable range [ (model-null)/(1-null) ]:  0.03913396
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     4.80094     -0.19086      0.03095     -0.43959      0.09411
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4202 Residual
## Null Deviance:       2806
## Residual Deviance: 2662  AIC: 2848
## log likelihood:  -1330.788
## Nagelkerke R2:  0.06944316
## % pres/err predicted correctly:  -779.934
## % of predictable range [ (model-null)/(1-null) ]:  0.03626828
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          4.739078          -0.190185           0.026931          -0.404818          -0.069168
## I(pos^2):log_freq       pos:log_freq
##         -0.009184           0.086900
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4200 Residual
## Null Deviance:       2806
## Residual Deviance: 2659  AIC: 2849
## log likelihood:  -1329.372
## Nagelkerke R2:  0.07077931
## % pres/err predicted correctly:  -779.1722
## % of predictable range [ (model-null)/(1-null) ]:  0.03720843
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          4.13534          -0.17525           0.52522          -0.15377          -0.05214
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4202 Residual
## Null Deviance:       2806
## Residual Deviance: 2662  AIC: 2851
## log likelihood:  -1331.065
## Nagelkerke R2:  0.06918198
## % pres/err predicted correctly:  -779.6932
## % of predictable range [ (model-null)/(1-null) ]:  0.03656552
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          4.13613          -0.17923           0.53015          -0.14669          -0.06043
##     log_freq:pos
##          0.01378
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4201 Residual
## Null Deviance:       2806
## Residual Deviance: 2661  AIC: 2852
## log likelihood:  -1330.517
## Nagelkerke R2:  0.06969894
## % pres/err predicted correctly:  -779.4081
## % of predictable range [ (model-null)/(1-null) ]:  0.03691726
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     4.12542     -0.17072     -0.15399      0.09513
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 2671  AIC: 2858
## log likelihood:  -1335.301
## Nagelkerke R2:  0.06518008
## % pres/err predicted correctly:  -781.2842
## % of predictable range [ (model-null)/(1-null) ]:  0.03460205
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.01285     -0.21958      0.03131     -0.44263
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 2673  AIC: 2858
## log likelihood:  -1336.702
## Nagelkerke R2:  0.06385472
## % pres/err predicted correctly:  -782.3133
## % of predictable range [ (model-null)/(1-null) ]:  0.03333209
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          3.01491           0.01179          -0.12464           0.80620           0.01733
##      stimlen:pos
##         -0.14131
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4201 Residual
## Null Deviance:       2806
## Residual Deviance: 2669  AIC: 2858
## log likelihood:  -1334.405
## Nagelkerke R2:  0.06602675
## % pres/err predicted correctly:  -781.1732
## % of predictable range [ (model-null)/(1-null) ]:  0.0347391
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     4.123494     -0.169724     -0.155544      0.107363     -0.002658
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4202 Residual
## Null Deviance:       2806
## Residual Deviance: 2671  AIC: 2860
## log likelihood:  -1335.276
## Nagelkerke R2:  0.06520349
## % pres/err predicted correctly:  -781.2929
## % of predictable range [ (model-null)/(1-null) ]:  0.03459136
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     5.18426     -0.29899     -0.39068      0.02698
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 2680  AIC: 2866
## log likelihood:  -1340.082
## Nagelkerke R2:  0.06065364
## % pres/err predicted correctly:  -783.5009
## % of predictable range [ (model-null)/(1-null) ]:  0.03186653
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      4.3317      -0.1996      -0.1534
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 2683  AIC: 2868
## log likelihood:  -1341.346
## Nagelkerke R2:  0.05945542
## % pres/err predicted correctly:  -783.6142
## % of predictable range [ (model-null)/(1-null) ]:  0.03172665
## ***************************
## 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
##          3.306980           0.016936          -0.369829          -0.004650          -0.009644
##      pos:log_freq
##          0.082404
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4201 Residual
## Null Deviance:       2806
## Residual Deviance: 2684  AIC: 2877
## log likelihood:  -1342.071
## Nagelkerke R2:  0.05876772
## % pres/err predicted correctly:  -783.9855
## % of predictable range [ (model-null)/(1-null) ]:  0.03126845
## ***************************
## model index:  4
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)          pos     log_freq  
##      2.9840      -0.2036       0.1250  
## 
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806 
## Residual Deviance: 2692  AIC: 2881
## log likelihood:  -1346.126
## Nagelkerke R2:  0.05491705
## % pres/err predicted correctly:  -785.383
## % of predictable range [ (model-null)/(1-null) ]:  0.02954382
## ***************************
## model index:  3
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq  
##     3.002197     -0.208409      0.169118     -0.009713  
## 
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806 
## Residual Deviance: 2692  AIC: 2882
## log likelihood:  -1345.799
## Nagelkerke R2:  0.05522789
## % pres/err predicted correctly:  -785.3109
## % of predictable range [ (model-null)/(1-null) ]:  0.02963284
## ***************************
## model index:  1
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq  
##          4.16092          -0.25984           0.52373          -0.05204  
## 
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806 
## Residual Deviance: 2703  AIC: 2887
## log likelihood:  -1351.564
## Nagelkerke R2:  0.04974148
## % pres/err predicted correctly:  -789.2024
## % of predictable range [ (model-null)/(1-null) ]:  0.02483043
## ***************************
## model index:  2
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     4.15052     -0.25539      0.09397
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 2712  AIC: 2894
## log likelihood:  -1355.829
## Nagelkerke R2:  0.04567313
## % pres/err predicted correctly:  -790.7697
## % of predictable range [ (model-null)/(1-null) ]:  0.02289628
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.36526      0.02025     -0.40714
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 2710  AIC: 2898
## log likelihood:  -1355.226
## Nagelkerke R2:  0.04624884
## % pres/err predicted correctly:  -789.881
## % of predictable range [ (model-null)/(1-null) ]:  0.02399298
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.0116      -0.2164
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2714  AIC: 2902
## log likelihood:  -1357.219
## Nagelkerke R2:  0.044345
## % pres/err predicted correctly:  -790.0866
## % of predictable range [ (model-null)/(1-null) ]:  0.02373923
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)      stimlen
##      4.3544      -0.2837
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2724  AIC: 2904
## log likelihood:  -1361.802
## Nagelkerke R2:  0.03996133
## % pres/err predicted correctly:  -792.8356
## % of predictable range [ (model-null)/(1-null) ]:  0.02034676
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##         2.1
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4206 Residual
## Null Deviance:       2806
## Residual Deviance: 2806  AIC: 2990
## log likelihood:  -1403.123
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -809.323
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + I(pos^2) + pos	2842.383	0.0000000	1.0000000	[illegible]	[illegible]	0.5117527	-0.1947626	0.0505754	-0.6276635	NA	NA	0.0296966	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 + (I(pos^2) + pos) * log_freq	2843.5362807610.33633794575292	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1982515	0.3626366	-0.0555664	-0.4017096	NA	0.0789498	0.0287576	NA	-0.0067352	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	2848.51789004553263301609448320944	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1908597	0.0941060	NA	-0.4395906	NA	NA	0.0309461	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	2849.7617708290466738670477339078	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1900185	-0.0391685	NA	-0.4048176	0.0868995	NA	0.0269313	-0.0091837	NA	NA	NA
preserved ~ stimlen * log_freq + pos	2850.8233970713983583669948205344	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1752518	0.5252169	-0.0521391	-0.1537709	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	2851.84513800886045306969189136128	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1792264	0.5301498	-0.0604302	-0.2466942	NA	0.0137805	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	2857.5471637000509630505548025418	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1707194	0.0951326	NA	-0.1539882	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	2857.9045205090426302553855012852	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2195777	NA	NA	-0.4426336	NA	NA	0.0313124	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	2857.96548075504137024766326179007117930	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0117930	NA	NA	0.8061966	NA	NA	-0.1246396	NA	NA	-0.1413132	0.0173297
preserved ~ stimlen + pos * log_freq	2859.57011173860190190104854012323494	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1697240	0.1073626	NA	-0.1555044	-0.0026577	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	2866.23698484829000620036755361263	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2989929	NA	NA	-0.3906788	NA	NA	NA	NA	NA	0.0269816	NA
preserved ~ stimlen + pos	2867.2561781300003400025945331697	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1996427	NA	NA	-0.1534431	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	2877.3427940352400000000583767769881	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0046497	NA	-0.3698290	0.0824036	NA	0.0169361	-0.0096439	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	[illegible]	I(pos^2)	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ pos + log_freq	2880.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.12503[illegible]	NA	-0.2035854	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	2882.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1691[illegible]	NA	-0.2084[illegible]	-0.0[illegible]97131	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	2887.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2598416	0.5237299	-0.0520378	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	2893.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2553855	0.0939[illegible]	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	2898.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.4071437	NA	NA	0.0202[illegible]	NA	NA	NA
preserved ~ pos	2901.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.2164271	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	2903.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2836826	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	2989.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          4.78526          -0.19476           0.51175           0.02970          -0.42766
## stimlen:log_freq
##         -0.05058
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4201 Residual
## Null Deviance:       2806
## Residual Deviance: 2654  AIC: 2842
```

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

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

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

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

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

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

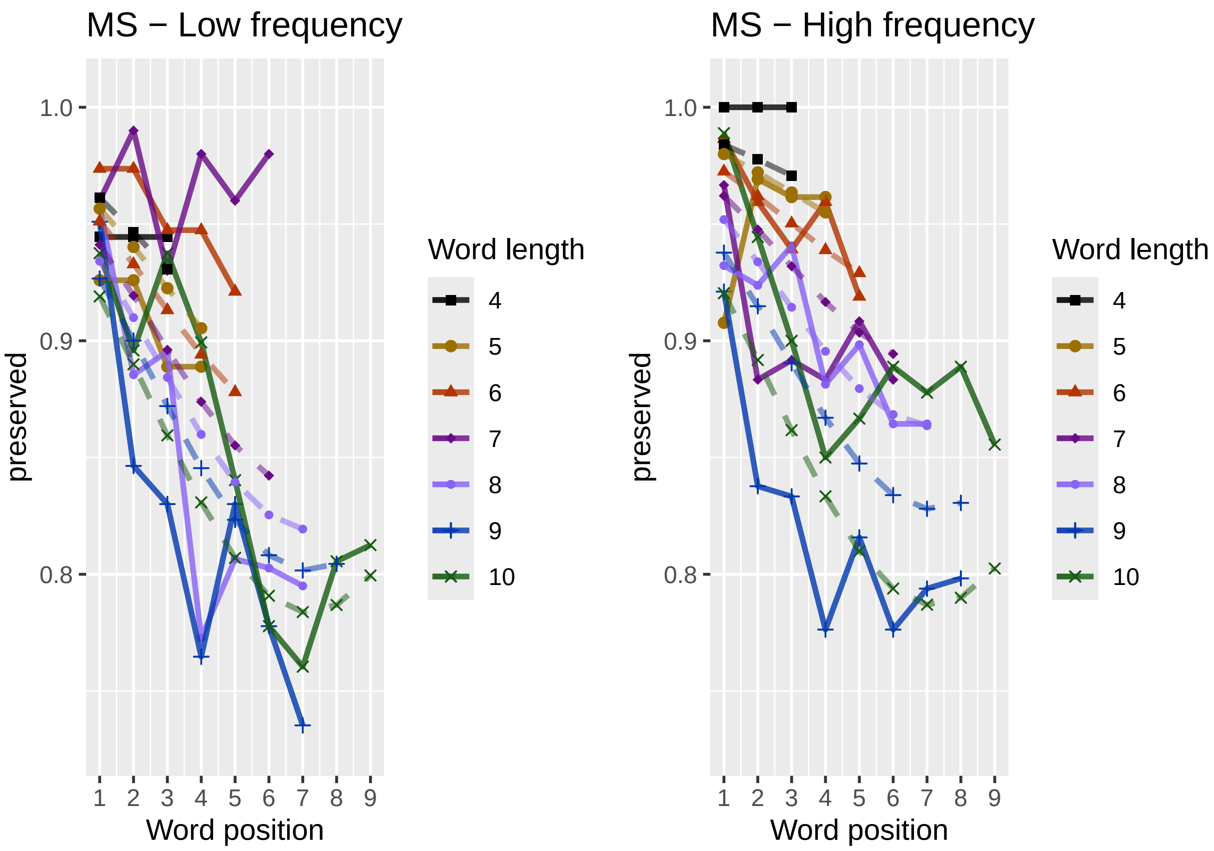
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

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

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

```
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
##       2.967       -1.627
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 1870  AIC: 1982
## log likelihood:  -935.193
## Nagelkerke R2:  0.4097315
## % pres/err predicted correctly:  -508.9251
## % of predictable range [ (model-null)/(1-null) ]:  0.3707138
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.36526      0.02025     -0.40714
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 2710  AIC: 2898
## log likelihood:  -1355.226
## Nagelkerke R2:  0.04624884
## % pres/err predicted correctly:  -789.881
## % of predictable range [ (model-null)/(1-null) ]:  0.02399298
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      3.0116      -0.2164
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2714  AIC: 2902
## log likelihood:  -1357.219
## Nagelkerke R2:  0.044345
## % pres/err predicted correctly:  -790.0866
## % of predictable range [ (model-null)/(1-null) ]:  0.02373923
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.3544      -0.2837
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2724  AIC: 2904
## log likelihood:  -1361.802
## Nagelkerke R2:  0.03996133
## % pres/err predicted correctly:  -792.8356
## % of predictable range [ (model-null)/(1-null) ]:  0.02034676
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.7315       0.1603
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2768  AIC: 2948
## log likelihood:  -1384.235
## Nagelkerke R2:  0.0183633
## % pres/err predicted correctly:  -802.3399
## % of predictable range [ (model-null)/(1-null) ]:  0.008617722
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##         2.1
```

```
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4206 Residual
## Null Deviance:       2806
## Residual Deviance: 2806  AIC: 2990
## log likelihood:  -1403.123
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -809.323
## % 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	1981.610	0.0000	1	1	0.4097312	3.966741	NA	-1.626647	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2898.465	916.8554	0	0	0.0462488	3.365260	NA	NA	0.0202477	-0.4071437	NA
preserved ~ pos	2901.708	920.0984	0	0	0.0443450	3.011649	NA	NA	NA	-0.2164271	NA
preserved ~ stimlen	2903.792	922.1829	0	0	0.0399614	4.354417	NA	NA	NA	NA	-0.2836826
preserved ~ CumPres	2948.279	966.6691	0	0	0.0183633	1.731486	0.1602528	NA	NA	NA	NA
preserved ~ 1	2989.971	1008.3618	0	0	0.0000000	2.099764	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	0.9135135	555
O	0.8876308	1943
P	0.9333333	30
S	0.8833333	240
V	0.8869122	1439

```
#  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
##       2.946       -1.694
##
## Degrees of Freedom: 3936 Total (i.e. Null);  3935 Residual
## Null Deviance:       2623
## Residual Deviance: 1765  AIC: 1882
## log likelihood:  -882.6009
## Nagelkerke R2:  0.4025918
## % pres/err predicted correctly:  -480.7765
## % of predictable range [ (model-null)/(1-null) ]:  0.363935
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.43298      0.02053     -0.42168
##
## Degrees of Freedom: 3936 Total (i.e. Null);  3934 Residual
## Null Deviance:       2623
## Residual Deviance: 2525  AIC: 2705
## log likelihood:  -1262.422
## Nagelkerke R2:  0.05071851
## % pres/err predicted correctly:  -736.5238
## % of predictable range [ (model-null)/(1-null) ]:  0.02628498
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.0676      -0.2269
##
## Degrees of Freedom: 3936 Total (i.e. Null);  3935 Residual
## Null Deviance:       2623
## Residual Deviance: 2529  AIC: 2708
## log likelihood:  -1264.349
## Nagelkerke R2:  0.04875387
## % pres/err predicted correctly:  -736.6482
## % of predictable range [ (model-null)/(1-null) ]:  0.02612075
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.4318      -0.2928
##
## Degrees of Freedom: 3936 Total (i.e. Null);  3935 Residual
## Null Deviance:       2623
## Residual Deviance: 2541  AIC: 2712
## log likelihood:  -1270.638
## Nagelkerke R2:  0.04233112
## % pres/err predicted correctly:  -740.1801
## % of predictable range [ (model-null)/(1-null) ]:  0.02145781
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.7747       0.1505
##
## Degrees of Freedom: 3936 Total (i.e. Null);  3935 Residual
## Null Deviance:       2623
## Residual Deviance: 2595  AIC: 2766
## log likelihood:  -1297.307
## Nagelkerke R2:  0.01486489
## % pres/err predicted correctly:  -751.1418
## % of predictable range [ (model-null)/(1-null) ]:  0.006985619
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.101
##
## Degrees of Freedom: 3936 Total (i.e. Null);  3936 Residual
## Null Deviance:       2623
## Residual Deviance: 2623  AIC: 2797
## log likelihood:  -1311.591
## Nagelkerke R2:  2.282581e-16
## % pres/err predicted correctly:  -756.4329
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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	1881.619	0.0000	1	1	0.4025911	2.946495	NA	-1.693872	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2705.022	823.4031	0	0	0.0507185	3.432984	NA	NA	0.020533	-0.4216755	NA
preserved ~ pos	2707.807	826.1879	0	0	0.0487539	3.067559	NA	NA	NA	-0.2268898	NA
preserved ~ stimlen	2712.297	830.6779	0	0	0.0423311	1.431803	NA	NA	NA	NA	-0.2927751
preserved ~ CumPres	2766.418	884.7992	0	0	0.0148649	1.774730	0.150492	NA	NA	NA	NA
preserved ~ 1	2797.447	915.8277	0	0	0.0000000	2.101170	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
##       2.840       -1.778
##
## Degrees of Freedom: 3381 Total (i.e. Null);  3380 Residual
## Null Deviance:       2301
## Residual Deviance: 1621  AIC: 1735
## log likelihood:  -810.2935
## Nagelkerke R2:  0.3693177
## % pres/err predicted correctly:  -446.3918
## % of predictable range [ (model-null)/(1-null) ]:  0.3317669
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.41758      0.02501     -0.45300
##
## Degrees of Freedom: 3381 Total (i.e. Null);  3379 Residual
## Null Deviance:       2301
## Residual Deviance: 2214  AIC: 2373
## log likelihood:  -1107.235
## Nagelkerke R2:  0.05131252
## % pres/err predicted correctly:  -650.9246
## % of predictable range [ (model-null)/(1-null) ]:  0.02627266
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.9871      -0.2178
##
## Degrees of Freedom: 3381 Total (i.e. Null);  3380 Residual
## Null Deviance:       2301
## Residual Deviance: 2220  AIC: 2378
## log likelihood:  -1109.956
## Nagelkerke R2:  0.0481324
## % pres/err predicted correctly:  -651.3378
## % of predictable range [ (model-null)/(1-null) ]:  0.02565553
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       4.358       -0.289
##
## Degrees of Freedom: 3381 Total (i.e. Null);  3380 Residual
## Null Deviance:       2301
## Residual Deviance: 2230  AIC: 2379
## log likelihood:  -1114.799
## Nagelkerke R2:  0.04246011
## % pres/err predicted correctly:  -653.7097
## % of predictable range [ (model-null)/(1-null) ]:  0.02211284
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.88210      0.09645
##
## Degrees of Freedom: 3381 Total (i.e. Null);  3380 Residual
## Null Deviance:       2301
## Residual Deviance: 2293  AIC: 2445
## log likelihood:  -1146.593
## Nagelkerke R2:  0.004814679
## % pres/err predicted correctly:  -666.9288
## % of predictable range [ (model-null)/(1-null) ]:  0.002368535
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.064
##
## Degrees of Freedom: 3381 Total (i.e. Null);  3381 Residual
## Null Deviance:       2301
## Residual Deviance: 2301  AIC: 2453
## log likelihood:  -1150.617
## Nagelkerke R2:  2.24922e-16
## % pres/err predicted correctly:  -668.5145
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

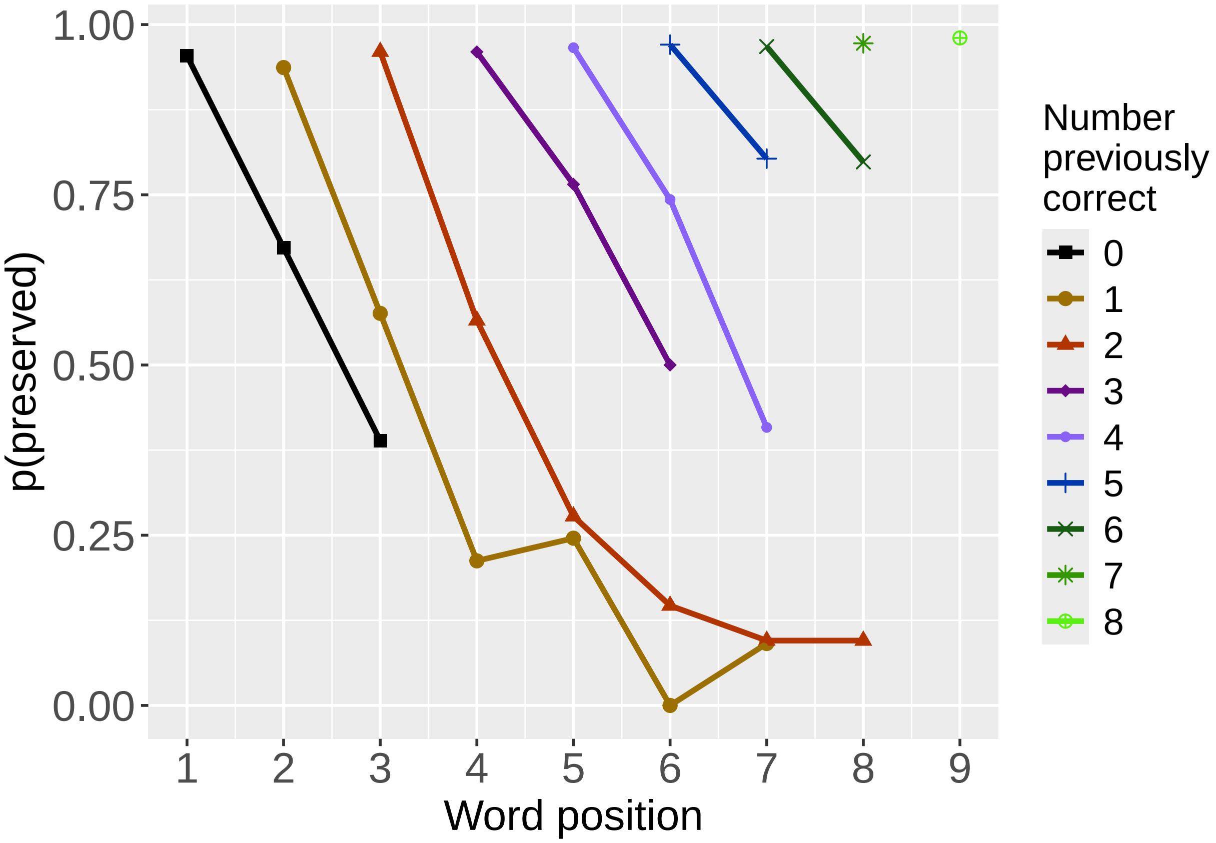
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1734.751	0.0000	1	1	0.3693177	2.839991	NA	-1.778214	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2372.901	638.1502	0	0	0.0513125	3.417579	NA	NA	0.025008	-0.4530009	NA
preserved ~ pos	2377.757	643.0062	0	0	0.0481324	2.987127	NA	NA	NA	-0.2177890	NA
preserved ~ stimlen	2378.919	644.1681	0	0	0.0424601	4.357984	NA	NA	NA	NA	-0.2890267
preserved ~ CumPres	2444.949	710.1982	0	0	0.0048147	1.882100	0.096449	NA	NA	NA	NA
preserved ~ 1	2453.304	718.5530	0	0	0.0000000	2.063704	NA	NA	NA	NA	NA

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

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

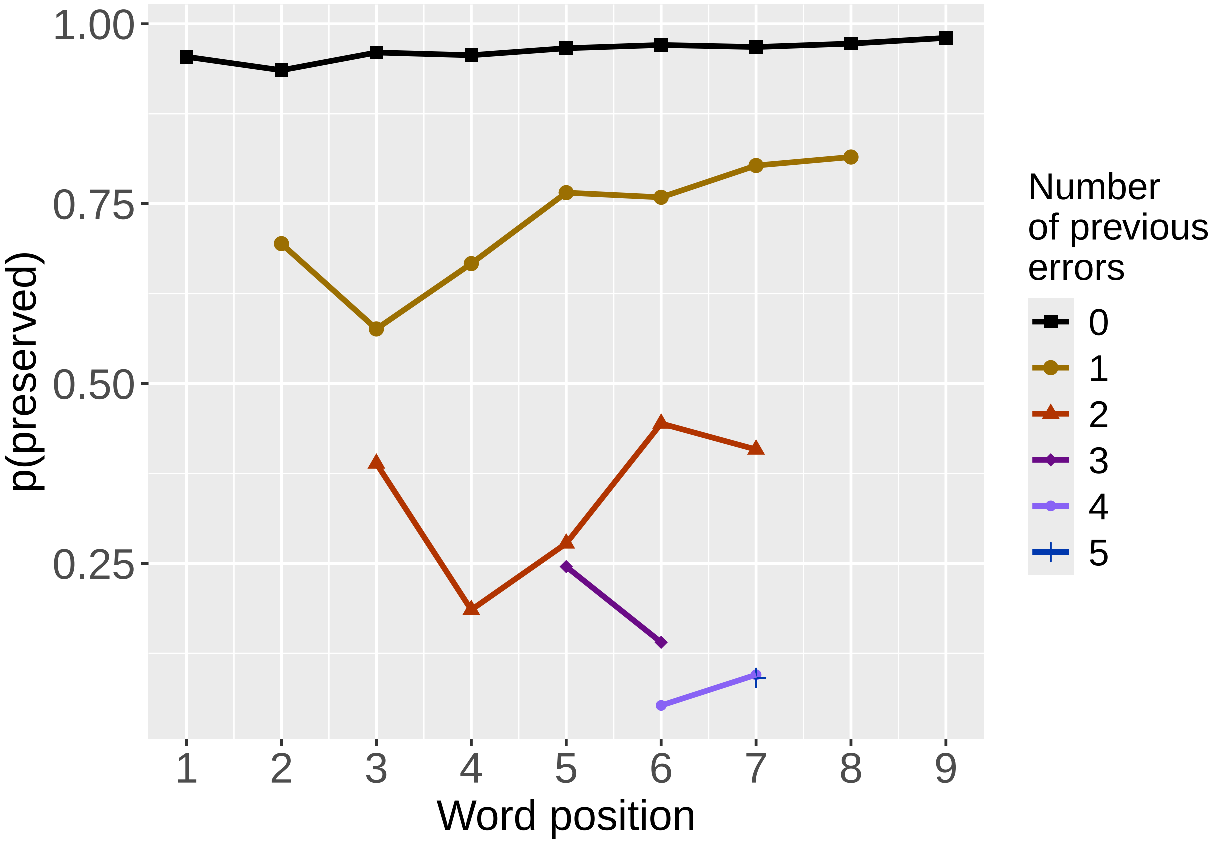
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

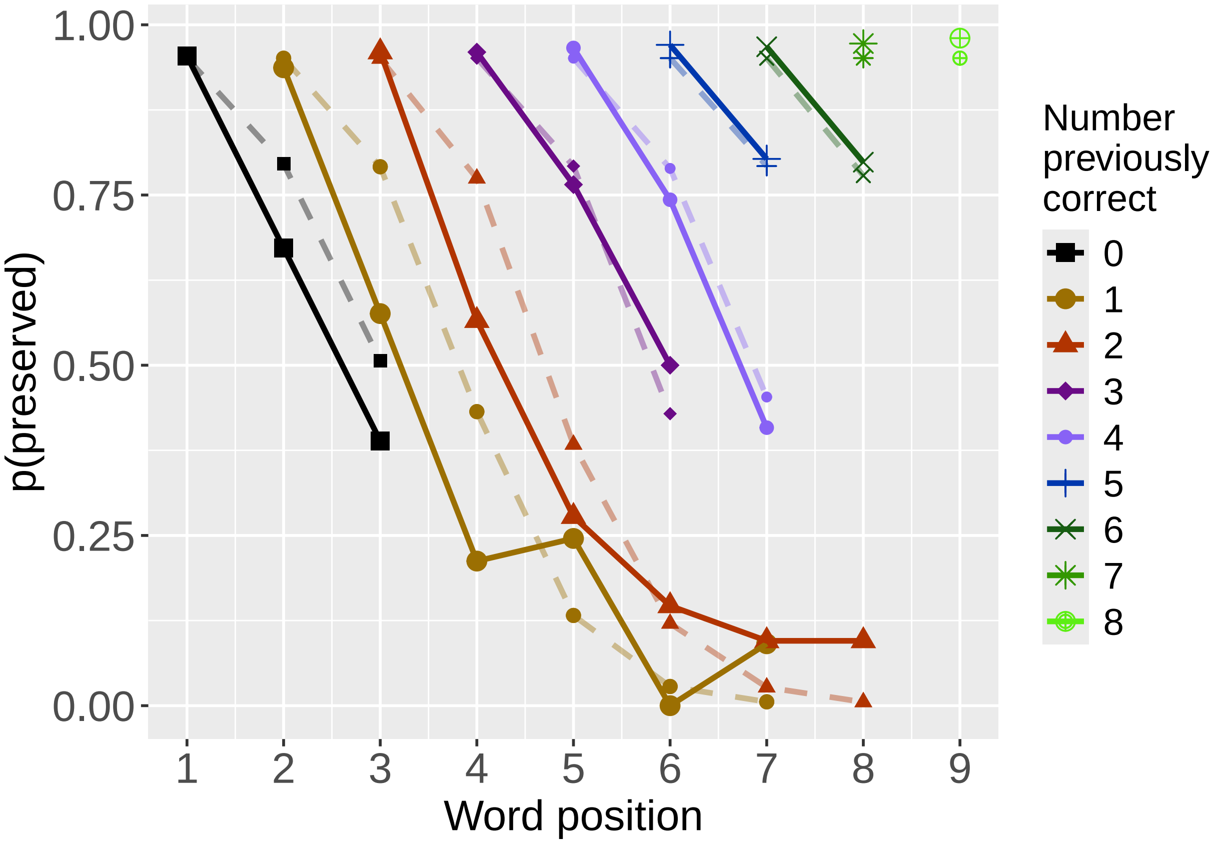
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

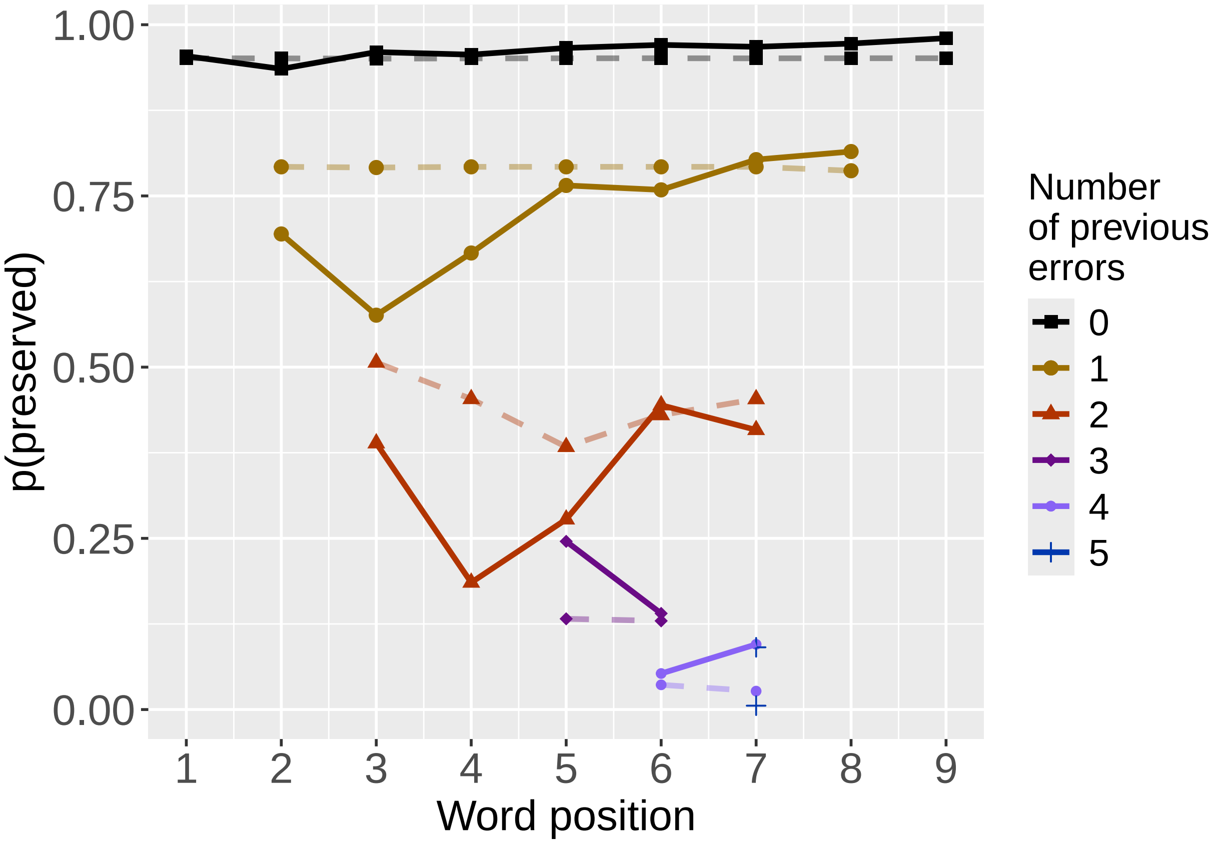
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      3.1250      -1.7776       0.0471      -0.2542
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 1840  AIC: 1947
## log likelihood:  -920.1705
## Nagelkerke R2:  0.421435
## % pres/err predicted correctly:  -500.4278
## % of predictable range [ (model-null)/(1-null) ]:  0.3812002
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.967       -1.627
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 1870  AIC: 1982
## log likelihood:  -935.193
## Nagelkerke R2:  0.4097315
## % pres/err predicted correctly:  -508.9251
## % of predictable range [ (model-null)/(1-null) ]:  0.3707138
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.36526      0.02025     -0.40714
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 2710  AIC: 2898
## log likelihood:  -1355.226
## Nagelkerke R2:  0.04624884
## % pres/err predicted correctly:  -789.881
## % of predictable range [ (model-null)/(1-null) ]:  0.02399298
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1946.609	0.00000	1	1	0.4214350	3.124989	-1.777583	0.0470997	-0.2542320

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1981.610	35.00086	0	0	0.4097315	2.966741	-1.626647	NA	NA
preserved ~ I(posˆ2) + pos	2898.465	951.85623	0	0	0.0462488	3.365260	NA	0.0202477	-0.4071437

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##      3.7695      -1.5884      -0.1044
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 1863  AIC: 1976
## log likelihood:  -931.6187
## Nagelkerke R2:  0.4125238
## % pres/err predicted correctly:  -508.3409
## % of predictable range [ (model-null)/(1-null) ]:  0.3714348
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.967       -1.627
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 1870  AIC: 1982
## log likelihood:  -935.193
## Nagelkerke R2:  0.4097315
## % pres/err predicted correctly:  -508.9251
## % of predictable range [ (model-null)/(1-null) ]:  0.3707138
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.3544      -0.2837
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2724  AIC: 2904
## log likelihood:  -1361.802
## Nagelkerke R2:  0.03996133
## % pres/err predicted correctly:  -792.8356
## % of predictable range [ (model-null)/(1-null) ]:  0.02034676
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	1975.916	0.000000	1.0000000	0.9451531	0.4125238	3.769484	-1.588359	-0.1043556
preserved ~ CumErr	1981.610	5.693603	0.0580296	0.0548469	0.4097315	2.966741	-1.626647	NA
preserved ~ stimlen	2903.792	927.876461	0.0000000	0.0000000	0.0399613	4.354417	NA	-0.2836826

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      2.6292      -1.6338       0.1437
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 1852  AIC: 1959
## log likelihood:  -925.8664
## Nagelkerke R2:  0.4170073
## % pres/err predicted correctly:  -504.0824
## % of predictable range [ (model-null)/(1-null) ]:  0.3766901
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.967       -1.627
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 1870  AIC: 1982
## log likelihood:  -935.193
## Nagelkerke R2:  0.4097315
## % pres/err predicted correctly:  -508.9251
## % of predictable range [ (model-null)/(1-null) ]:  0.3707138
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.7315       0.1603
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2768  AIC: 2948
## log likelihood:  -1384.235
## Nagelkerke R2:  0.0183633
## % pres/err predicted correctly:  -802.3399
## % of predictable range [ (model-null)/(1-null) ]:  0.008617722
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	1958.857	0.00000	1.00e+00	0.9999885	0.4170073	2.629166	-1.633760	0.1436841
preserved ~ CumErr	1981.610	22.75271	1.15e-05	0.0000115	0.4097315	2.966741	-1.626647	NA
preserved ~ CumPres	2948.279	989.42179	0.00e+00	0.0000000	0.0183633	1.731486	NA	0.1602528

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.4855      -1.7774       0.1437
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806
## Residual Deviance: 1852  AIC: 1959
## log likelihood:  -925.8664
## Nagelkerke R2:  0.4170073
## % pres/err predicted correctly:  -504.0824
## % of predictable range [ (model-null)/(1-null) ]:  0.3766901
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.967       -1.627
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 1870  AIC: 1982
## log likelihood:  -935.193
## Nagelkerke R2:  0.4097315
## % pres/err predicted correctly:  -508.9251
## % of predictable range [ (model-null)/(1-null) ]:  0.3707138
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      3.0116      -0.2164
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 2714  AIC: 2902
## log likelihood:  -1357.219
## Nagelkerke R2:  0.044345
## % pres/err predicted correctly:  -790.0866
## % of predictable range [ (model-null)/(1-null) ]:  0.02373923
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	1958.857	0.00000	1.00e+00	0.9999885	0.4170073	2.485482	-1.777445	0.1436841
preserved ~ CumErr	1981.610	22.75271	1.15e-05	0.0000115	0.4097315	2.966741	-1.626647	NA
preserved ~ pos	2901.708	942.85113	0.00e+00	0.0000000	0.0443450	3.011649	NA	-0.2164271

```r
CumAICSummary <- CumAICSummary %>% arrange(AIC)
BestModelFormulaL2 <- CumAICSummary$Model[BestModelIndexL2]
write.csv(CumAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_main_effects_plus_one_model_summary.csv"))
kable(CumAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + I(pos^2) + pos	1946.609	0.000000	1.000000	1.0000000	0.4214353	3.0124989	-1.777583	0.0470997	-0.2542320	NA	NA
preserved ~ CumErr + CumPres	1958.857	0.000000	1.000000	0.9999885	0.4170073	2.629166	-1.633760	NA	NA	NA	0.1436841
preserved ~ CumErr + pos	1958.857	0.000000	1.000000	0.9999885	0.4170073	2.485482	-1.777445	NA	0.1436841	NA	NA
preserved ~ CumErr + stimlen	1975.916	0.000000	1.000000	0.9451531	0.4125233	3.769484	-1.588359	NA	NA	-0.1043556	NA
preserved ~ CumErr	1981.610	35.000858	0.000000	0.0000000	0.4097315	2.966741	-1.626647	NA	NA	NA	NA
preserved ~ CumErr	1981.610	5.693603	0.0580296	0.0548469	0.4097315	2.966741	-1.626647	NA	NA	NA	NA
preserved ~ CumErr	1981.610	22.752710	0.0000115	0.0000115	0.4097315	2.966741	-1.626647	NA	NA	NA	NA
preserved ~ CumErr	1981.610	22.752710	0.0000115	0.0000115	0.4097315	2.966741	-1.626647	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	2898.469	951.856227	0.0000000	0.0000000	0.0462488	3.365260	NA	0.0202477	-0.4071437	NA	NA
preserved ~ pos	2901.708	942.851135	0.0000000	0.0000000	0.0443450	3.011649	NA	NA	-0.2164271	NA	NA
preserved ~ stimlen	2903.792	927.876461	0.0000000	0.0000000	0.0399614	3.354417	NA	NA	NA	-0.2836826	NA
preserved ~ CumPres	2948.279	989.421792	0.0000000	0.0000000	0.0183631	3.731486	NA	NA	NA	NA	0.1602528

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##      4.4305      -1.7524       0.0571      -0.2837      -0.1737       0.1004
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4201 Residual
## Null Deviance:       2806
## Residual Deviance: 1808  AIC: 1915
## log likelihood:  -903.8147
## Nagelkerke R2:  0.4340823
## % pres/err predicted correctly:  -495.8026
## % of predictable range [ (model-null)/(1-null) ]:  0.3869079
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     4.64912     -1.75837      0.05658     -0.28250     -0.20404
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4202 Residual
## Null Deviance:       2806
## Residual Deviance: 1816  AIC: 1920
## log likelihood:  -908.0139
## Nagelkerke R2:  0.4308446
## % pres/err predicted correctly:  -496.3137
## % of predictable range [ (model-null)/(1-null) ]:  0.3862772
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     3.13052     -1.76628      0.04978     -0.26067      0.13555
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4202 Residual
## Null Deviance:       2806
## Residual Deviance: 1824  AIC: 1932
## log likelihood:  -911.9783
## Nagelkerke R2:  0.427782
## % pres/err predicted correctly:  -498.8503
## % of predictable range [ (model-null)/(1-null) ]:  0.3831468
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      3.1250      -1.7776       0.0471      -0.2542
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806
## Residual Deviance: 1840  AIC: 1947
## log likelihood:  -920.1705
## Nagelkerke R2:  0.421435
## % pres/err predicted correctly:  -500.4278
## % of predictable range [ (model-null)/(1-null) ]:  0.3812002
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##         2.1
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4206 Residual
## Null Deviance:       2806
## Residual Deviance: 2806  AIC: 2990
## log likelihood:  -1403.123
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -809.323
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	1914.656	0.000000	1.0000000	0.9450052	0.4340824	2.430505	-1.752372	0.0571045	-0.2837288	0.1004041	-0.1736678
preserved ~ CumErr + I(pos^2) + pos + stimlen	1920.349	5.692958	0.0580484	0.0548560	0.4308446	1.649117	-1.758366	0.0565806	-0.2825008	NA	-0.2040398
preserved ~ CumErr + I(pos^2) + pos + log_freq	1932.309	17.652904	0.0001468	0.0001387	0.4277832	3.130520	-1.766285	0.0497778	-0.2606694	0.1355503	NA
preserved ~ CumErr + I(pos^2) + pos	1946.609	31.952880	0.0000001	0.0000001	0.4214353	3.124989	-1.777583	0.0470997	-0.2542320	NA	NA
preserved ~ 1	2989.971	1075.315529	0.0000000	0.0000000	0.0000000	2.099764	NA	NA	NA	NA	NA

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

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

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

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

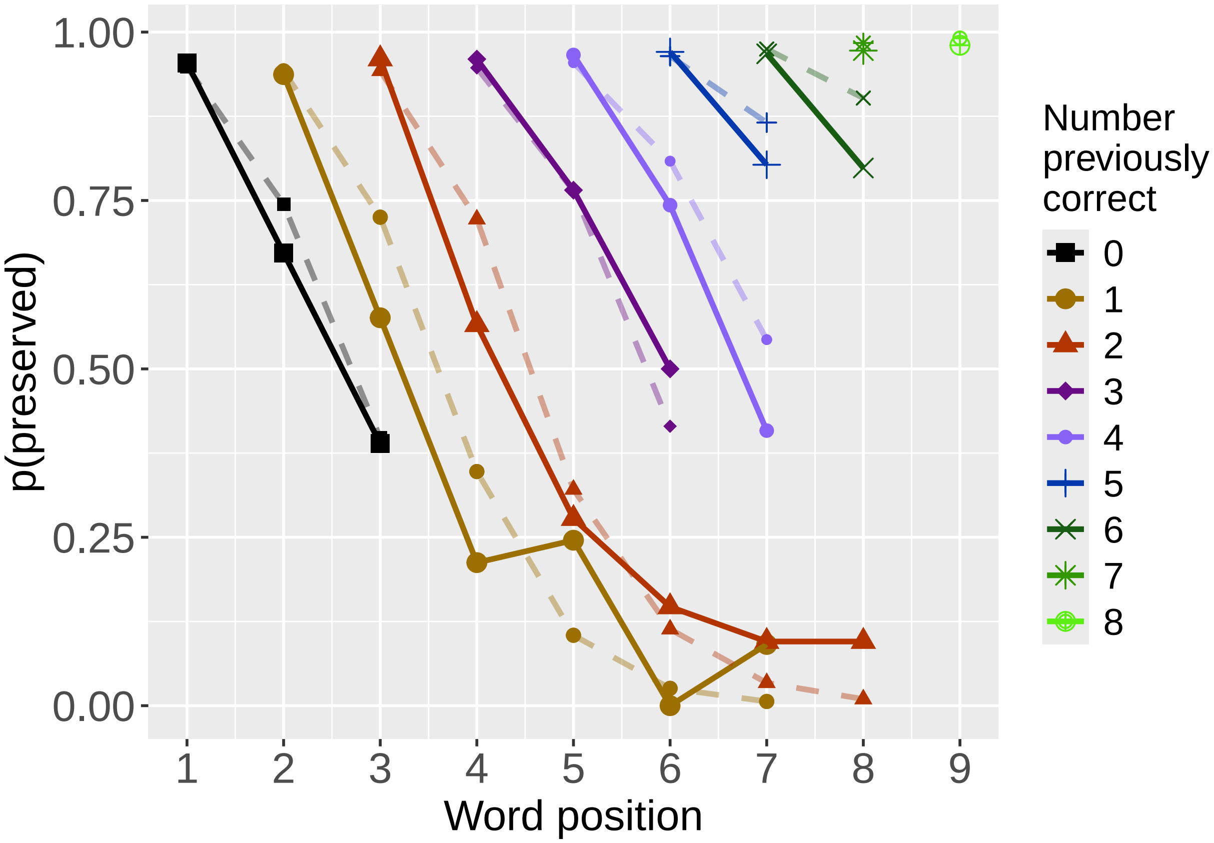
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   2661.6 2766.6
## I(pos^2)  1   1824.0 1929.1
## stimlen   1   1824.0 1929.0
## log_freq  1   1816.0 1921.0
## pos       1   1812.8 1917.8
## <none>        1807.6 1914.7
```

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

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

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

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

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

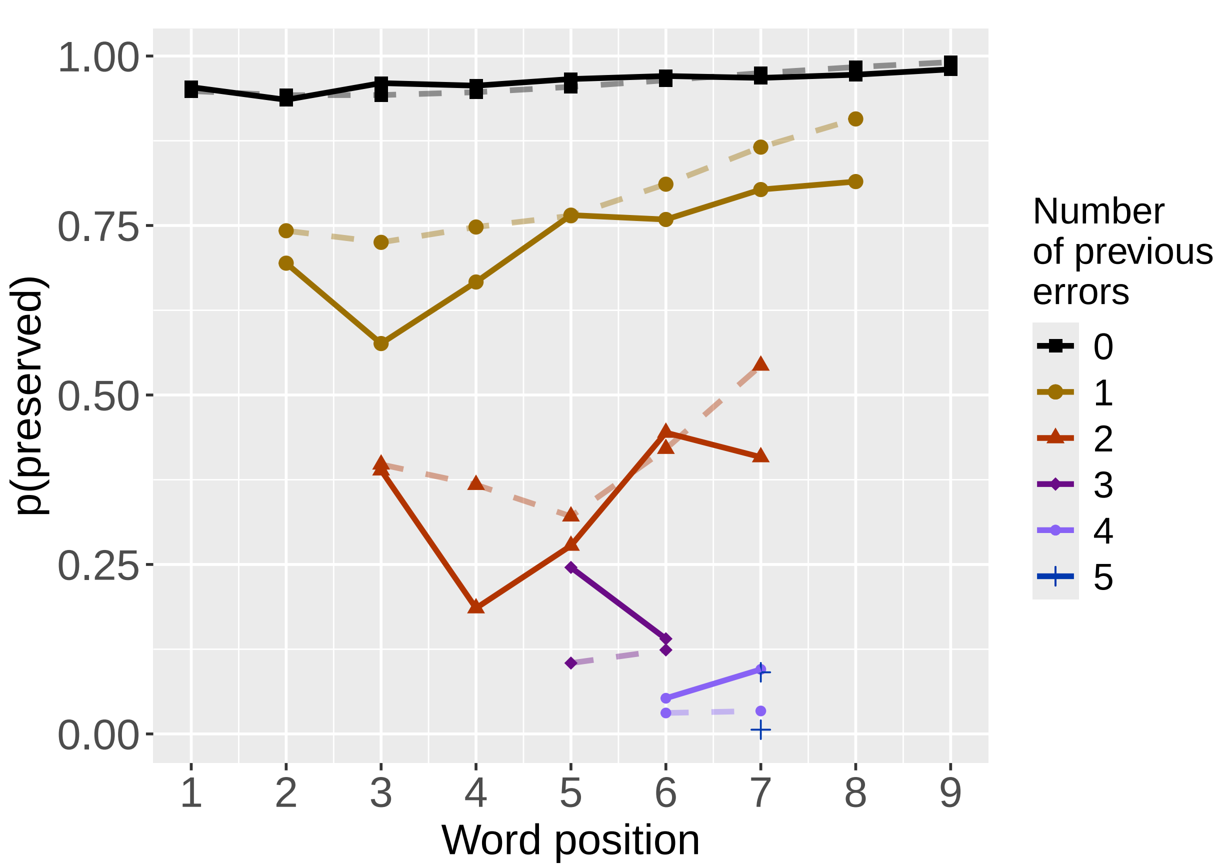
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

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

  RndModelAIC<-numeric(length=RandomSamples)
  for(rindex in seq(1,RandomSamples)){
  # Shuffle cumulative values
    PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
    PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
    BestModelRnd <- glm(as.formula(BestModelFormulaL3Rnd),
                        family="binomial",data=PosDat)
    RndModelAIC[rindex] <- BestModelRnd$aic
  }
  ModelNames<-c(paste0("***",BestModelFormulaL3),
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random average"),
                                     AIC=c(mean(RndModelAIC))))
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random SD"),
                                     AIC=c(sd(RndModelAIC))))
  write.csv(BestModelRndDF,paste0(TablesDir,CurPat,"_",CurTask,
                                  "_best_model_with_random_cum_term.csv"),
            row.names = FALSE)
}
```

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.967       -1.627
##
## Degrees of Freedom: 4206 Total (i.e. Null);  4205 Residual
## Null Deviance:       2806
## Residual Deviance: 1870  AIC: 1982
## log likelihood:  -935.193
## Nagelkerke R2:  0.4097315
## % pres/err predicted correctly:  -508.9251
## % of predictable range [ (model-null)/(1-null) ]:  0.3707138
## ***************************
## model index:  2
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)       CumErr     I(pos^2)  
##     2.68170     -1.79851      0.01939  
## 
## Degrees of Freedom: 4206 Total (i.e. Null);  4204 Residual
## Null Deviance:       2806 
## Residual Deviance: 1845  AIC: 1950
## log likelihood:  -922.2764
## Nagelkerke R2:  0.4197994
## % pres/err predicted correctly:  -501.9392
## % of predictable range [ (model-null)/(1-null) ]:  0.3793349
## ***************************
## model index:  3
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)       CumErr     I(pos^2)      stimlen  
##     4.11600     -1.78198      0.02566     -0.19847  
## 
## Degrees of Freedom: 4206 Total (i.e. Null);  4203 Residual
## Null Deviance:       2806 
## Residual Deviance: 1821  AIC: 1925
## log likelihood:  -910.6033
## Nagelkerke R2:  0.4288449
## % pres/err predicted correctly:  -498.0372
## % of predictable range [ (model-null)/(1-null) ]:  0.3841502
## ***************************
## model index:  4
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)       CumErr     I(pos^2)      stimlen     log_freq  
##     3.89604     -1.77633      0.02601     -0.16814      0.10010  
## 
## Degrees of Freedom: 4206 Total (i.e. Null);  4202 Residual
## Null Deviance:       2806 
## Residual Deviance: 1813  AIC: 1919
## log likelihood:  -906.4136
## Nagelkerke R2:  0.4320792
## % pres/err predicted correctly:  -497.5917
## % of predictable range [ (model-null)/(1-null) ]:  0.3847001
## ***************************
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.

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

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

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

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

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

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).
```

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

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

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

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

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

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

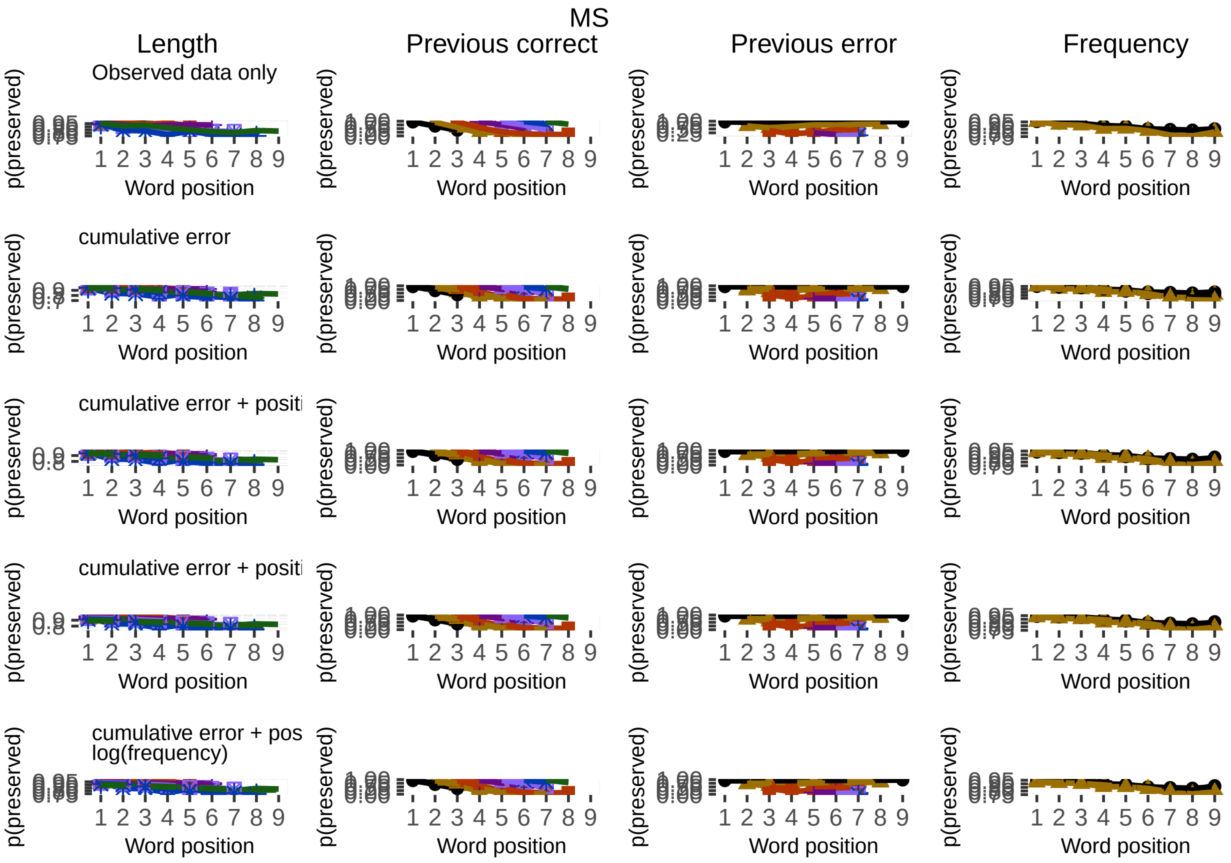
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
```

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

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

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

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



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

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

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

	CumErr	I(pos^2)	pos	stimlen	log_freq
McFadden	0.3205489	0.0119448	0.0121646	0.0129050	0.0056649
SquaredCorrelation	0.1992640	0.0076897	0.0081402	0.0085679	0.0037264
Nagelkerke	0.1992640	0.0076897	0.0081402	0.0085679	0.0037264
Estrella	0.2426778	0.0089527	0.0089223	0.0095089	0.0041970

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

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

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

```
##                                                                          model deviance
## CumErr + I(pos^2) + stimlen + log_freq CumErr + I(pos^2) + stimlen + log_freq 1812.827
## CumErr + I(pos^2) + stimlen                       CumErr + I(pos^2) + stimlen 1821.207
## CumErr + I(pos^2)                                           CumErr + I(pos^2) 1844.553
## CumErr                                                                 CumErr 1870.386
## null                                                                     null 2806.245
##                                        deviance_explained percent_explained
## CumErr + I(pos^2) + stimlen + log_freq           993.4181          35.40026
## CumErr + I(pos^2) + stimlen                      985.0388          35.10166
## CumErr + I(pos^2)                                961.6926          34.26973
## CumErr                                           935.8593          33.34916
## null                                               0.0000           0.00000
##                                        percent_of_explained_deviance increment_in_explained
## CumErr + I(pos^2) + stimlen + log_freq                     100.00000              0.8434836
## CumErr + I(pos^2) + stimlen                                 99.15652              2.3500855
## CumErr + I(pos^2)                                           96.80643              2.6004482
## CumErr                                                      94.20598             94.2059828
## null                                                              NA              0.0000000
```

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

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

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

	deviance	deviance_explained
CumErr + I(pos^2) + stimlen + log_freq	1812.827	993.4181
CumErr + I(pos^2) + stimlen	1821.207	985.0388
CumErr + I(pos^2)	1844.553	961.6926
CumErr	1870.386	935.8593
null	2806.245	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + I(pos^2) + stimlen + log_freq	35.40026	100.00000	0.8434836
CumErr + I(pos^2) + stimlen	35.10166	99.15652	2.3500855
CumErr + I(pos^2)	34.26973	96.80643	2.6004482
CumErr	33.34916	94.20598	94.2059828
null	0.00000	NA	0.0000000

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

```
##          Nagelkerke
## CumErr   0.87631638
## I(pos^2) 0.03381750
## pos      0.03579861
## stimlen  0.03767975
## log_freq 0.01638777
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.9744199	1870.386
preserved ~ CumErr+I(pos^2)	0.9839958	1844.553
preserved ~ CumErr+I(pos^2)+stimlen+log_freq	0.9842432	1812.827
preserved ~ CumErr+I(pos^2)+stimlen	0.9843352	1821.207

```
## `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
```

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

```
##                                          model p_accounted_for model_deviance diff_CumErr
## 1                           preserved ~ CumErr       0.9744199       1870.386 0.000000000
## 2                  preserved ~ CumErr+I(pos^2)       0.9839958       1844.553 0.009575900
## 3 preserved ~ CumErr+I(pos^2)+stimlen+log_freq       0.9842432       1812.827 0.009823312
## 4          preserved ~ CumErr+I(pos^2)+stimlen       0.9843352       1821.207 0.009915316
##   diff_CumErr+I(pos^2) diff_CumErr+I(pos^2)+stimlen+log_freq diff_CumErr+I(pos^2)+stimlen
## 1        -0.0095759003                         -9.823312e-03                -9.915316e-03
## 2         0.0000000000                         -2.474113e-04                -3.394154e-04
## 3         0.0002474113                          0.000000e+00                -9.200414e-05
## 4         0.0003394154                          9.200414e-05                 0.000000e+00
```

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

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

model	diff_CumErr	diff_CumErr+I(pos^2)	diff_CumErr+I(pos^2)+stimlen+log_freq
preserved ~ CumErr	0.0000000	-0.0095759	-0.0098233
preserved ~ CumErr+I(pos^2)	0.0095759	0.0000000	-0.0002474
preserved ~ CumErr+I(pos^2)+stimlen+log_freq	0.0098233	0.0002474	0.0000000
preserved ~ CumErr+I(pos^2)+stimlen	0.0099153	0.0003394	0.0000920

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