SR - repetition - Serial position analysis (v5)

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

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

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

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

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

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

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

# get counts of syllable components in different serial positions
syll_comp_dist_N <- PosDat %>% mutate(pos_factor = as.factor(pos), syll_component_factor = as.factor(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	557	35	128	NA	NA	720
2	66	NA	445	97	112	720
3	315	NA	173	216	16	720
4	311	NA	241	69	39	660
5	235	NA	217	72	39	563
6	209	1	139	73	22	444
7	181	NA	102	28	19	330
8	91	NA	56	24	4	175
9	75	NA	2	NA	7	84

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7736111	0.0486111	0.1777778	NA	NA	720
2	0.0916667	NA	0.6180556	0.1347222	0.1555556	720
3	0.4375000	NA	0.2402778	0.3000000	0.0222222	720
4	0.4712121	NA	0.3651515	0.1045455	0.0590909	660
5	0.4174067	NA	0.3854352	0.1278863	0.0692718	563
6	0.4707207	0.0022523	0.3130631	0.1644144	0.0495495	444

pos_factor	O	P	V	1	S	total
7	0.5484848	NA	0.3090909	0.0848485	0.0575758	330
8	0.5200000	NA	0.3200000	0.1371429	0.0228571	175
9	0.8928571	NA	0.0238095	NA	0.0833333	84

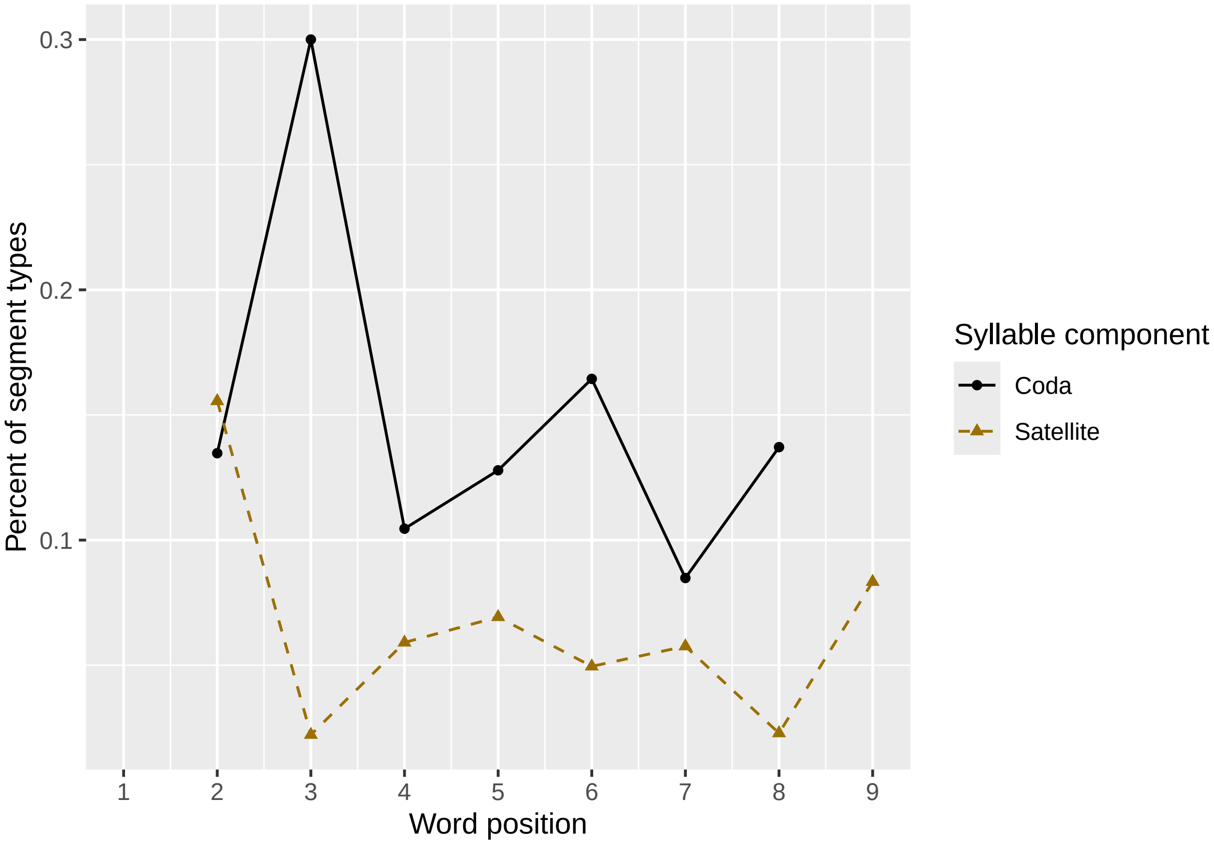
```r
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```r
syll_comp_plot
```

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



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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.95  1     0.983 NA     NA     NA     NA     NA     NA
## 2       5 0.907 0.959 0.959  0.948 NA     NA     NA     NA     NA
## 3       6 0.950 0.983 0.983  0.975  0.975 NA     NA     NA     NA
## 4       7 0.829 0.899 0.943  0.921  0.939  0.925 NA     NA     NA
## 5       8 0.903 0.963 0.939  0.941  0.938  0.961  0.974 NA     NA
## 6       9 0.863 0.923 0.907  0.934  0.879  0.940  0.967  0.967 NA
## 7      10 0.964 0.927 0.861  0.920  0.887  0.917  0.952  0.976  0.952
```

```
# 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.
```

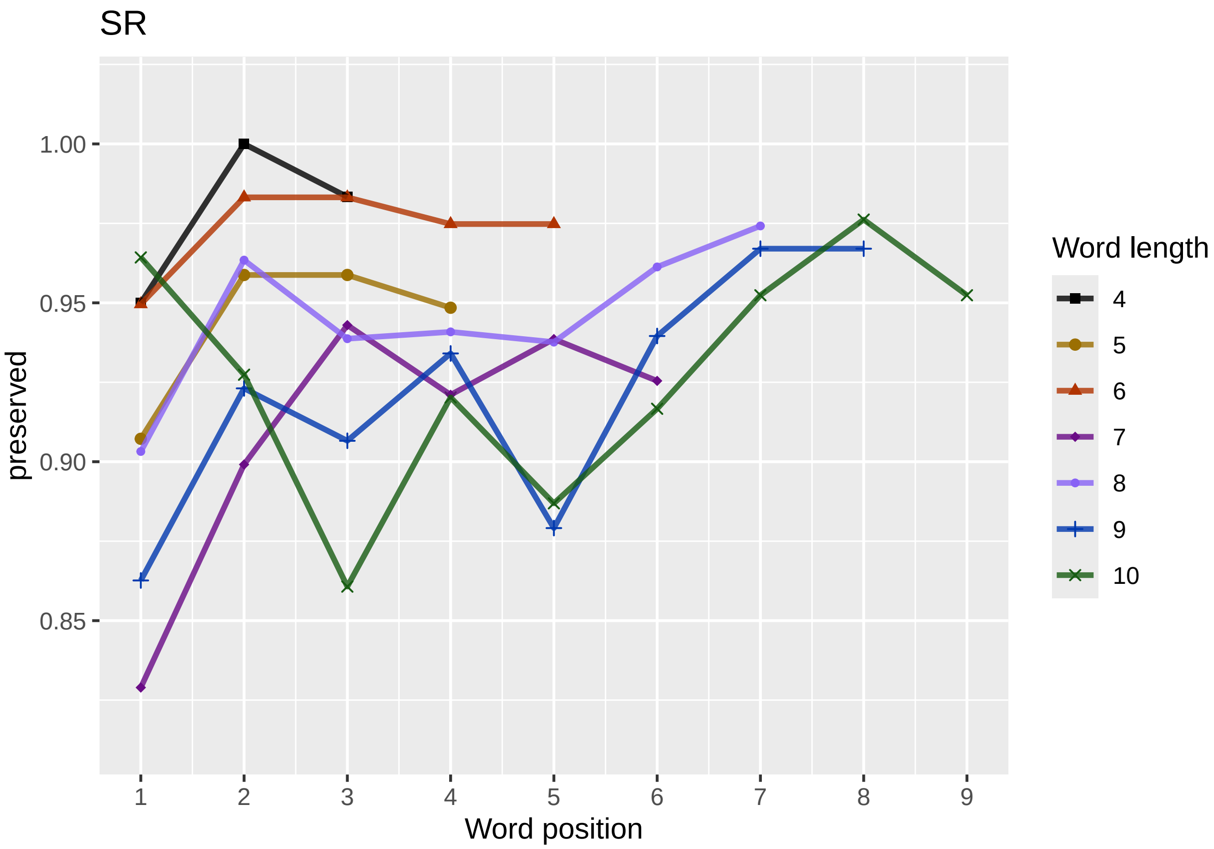
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    60    60    60    NA    NA    NA    NA    NA    NA
## 2       5    97    97    97    97    NA    NA    NA    NA    NA
## 3       6   119   119   119   119   119    NA    NA    NA    NA
## 4       7   114   114   114   114   114   114    NA    NA    NA
## 5       8   155   155   155   155   155   155   155    NA    NA
## 6       9    91    91    91    91    91    91    91    91    NA
## 7      10    84    84    84    84    84    84    84    84    84
```

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

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

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



Length and position

```
# length and position

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

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

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

```
## ***************************
## model index:  8
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          1.05713           0.14274          -0.22159           1.89039           0.02671
##      stimlen:pos
##         -0.21582
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4410 Residual
## Null Deviance:       1995
## Residual Deviance: 1958  AIC: 2117
## log likelihood:  -978.7912
## Nagelkerke R2:  0.02344354
## % pres/err predicted correctly:  -504.5996
## % of predictable range [ (model-null)/(1-null) ]:  0.01000021
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      3.5892      -0.1777       0.1407
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1965  AIC: 2120
## log likelihood:  -982.7258
## Nagelkerke R2:  0.01857952
## % pres/err predicted correctly:  -505.6918
## % of predictable range [ (model-null)/(1-null) ]:  0.007861515
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.69241     -0.18051      0.00680      0.08535
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1965  AIC: 2121
## log likelihood:  -982.6059
## Nagelkerke R2:  0.01872785
## % pres/err predicted correctly:  -505.6269
## % of predictable range [ (model-null)/(1-null) ]:  0.00798872
## ***************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.18079     -0.12913      0.28714     -0.01697
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1965  AIC: 2122
## log likelihood:  -982.4136
## Nagelkerke R2:  0.01896579
## % pres/err predicted correctly:  -505.6081
## % of predictable range [ (model-null)/(1-null) ]:  0.008025528
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.6238      -0.1162
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2143
## log likelihood:  -992.959
## Nagelkerke R2:  0.005888126
## % pres/err predicted correctly:  -508.429
## % of predictable range [ (model-null)/(1-null) ]:  0.002501919
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3741       0.0943
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2145
## log likelihood:  -992.7941
## Nagelkerke R2:  0.006092996
## % pres/err predicted correctly:  -508.4731
## % of predictable range [ (model-null)/(1-null) ]:  0.002415632
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      2.3423      -0.0026       0.1156
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2147
## log likelihood:  -992.7764
## Nagelkerke R2:  0.006115059
## % pres/err predicted correctly:  -508.4671
## % of predictable range [ (model-null)/(1-null) ]:  0.002427309
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.718
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4415 Residual
## Null Deviance:       1995
## Residual Deviance: 1995  AIC: 2154
## log likelihood:  -997.6906
## Nagelkerke R2:  6.107643e-16
## % pres/err predicted correctly:  -509.7067
## % 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)	2117.068	0.000000	1.0000000	0.7088301	0.0234431	5.0571320	0.1427382	1.8903929	-0.2158240	-0.2215892	0.0267131
preserved ~ stimlen + pos	2120.220	3.151828	0.2068184	0.1465993	0.0185793	5.5589215	-0.1777252	0.1406805	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	2121.364	4.298786	0.1165549	0.0826177	0.0187273	3.692410	-0.1805053	0.0853454	NA	0.0067995	NA
preserved ~ stimlen * pos	2121.943	4.874611	0.0873960	0.0619490	0.0189653	3.180791	-0.1291254	0.2871404	-0.0169723	NA	NA
preserved ~ stimlen	2142.842	25.774321	0.0000025	0.0000018	0.0058883	1.623823	-0.1161725	NA	NA	NA	NA
preserved ~ pos	2144.562	27.498188	0.0000010	0.0000008	0.0060932	2.374112	NA	0.0942957	NA	NA	NA
preserved ~ I(pos^2) + pos	2146.729	29.660430	0.0000004	0.0000003	0.0061152	1.342278	NA	0.1156227	NA	-0.0025996	NA
preserved ~ 1	2153.843	36.771972	0.0000000	0.0000000	0.0000000	2.717753	NA	NA	NA	NA	NA

```r
print(BestLPModelFormula)
```

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

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          1.05713           0.14274          -0.22159           1.89039           0.02671
##      stimlen:pos
##         -0.21582
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4410 Residual
## Null Deviance:       1995
## Residual Deviance: 1958  AIC: 2117
```

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

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

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

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

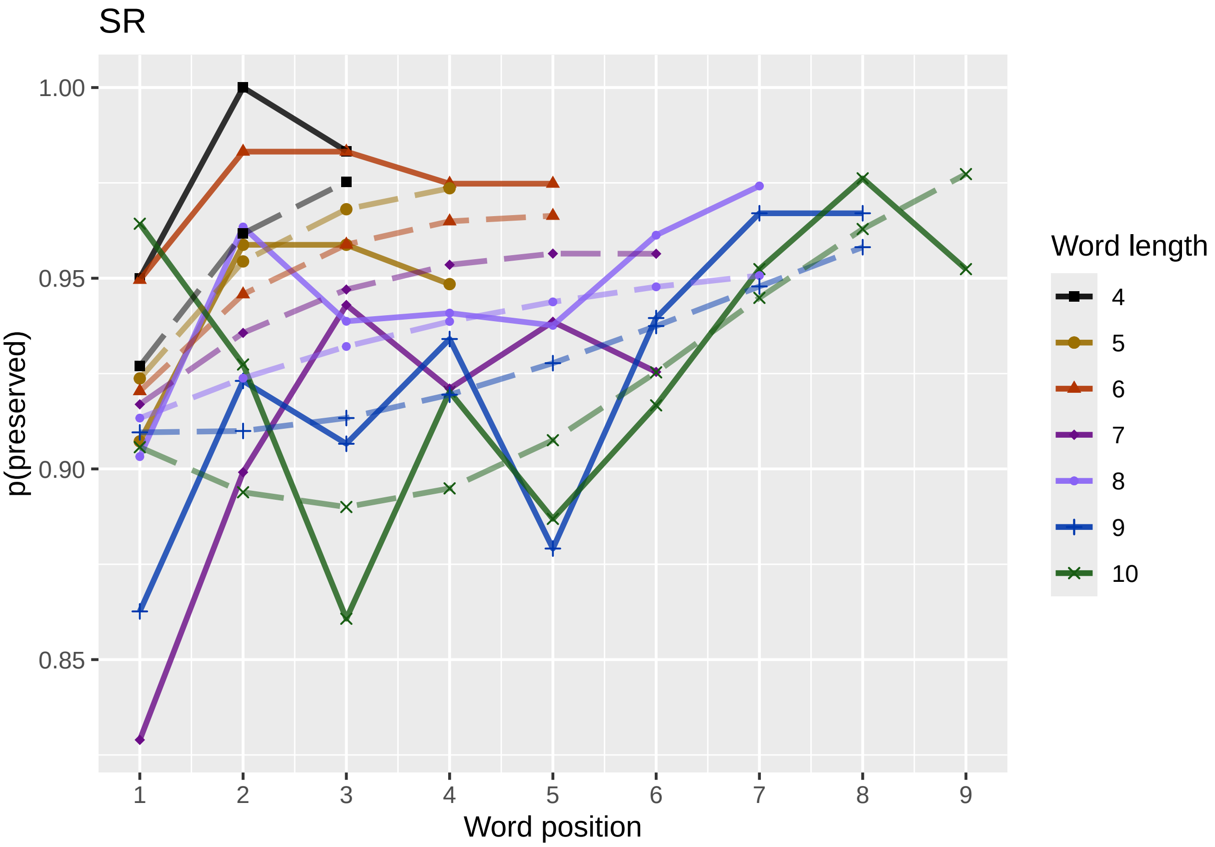
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.927 0.962 0.975 NA     NA     NA     NA     NA     NA
## 2       5 0.924 0.954 0.968  0.974 NA     NA     NA     NA     NA
## 3       6 0.920 0.946 0.959  0.965  0.966 NA     NA     NA     NA
## 4       7 0.917 0.936 0.947  0.954  0.956  0.956 NA     NA     NA
## 5       8 0.913 0.924 0.932  0.939  0.944  0.948  0.951 NA     NA
## 6       9 0.910 0.910 0.913  0.919  0.928  0.937  0.948  0.958 NA
## 7      10 0.906 0.894 0.890  0.895  0.908  0.925  0.945  0.963  0.977
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       12   720
```

```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 12 / 720 = 1.67 percent"
```

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

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

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

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

```
## ****************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          0.88525           0.17270          -0.19956           1.92736           0.02664
##      stimlen:pos
##         -0.22888
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4372 Residual
## Null Deviance:       1787
## Residual Deviance: 1730  AIC: 1884
## log likelihood:  -864.9314
## Nagelkerke R2:  0.03885987
## % pres/err predicted correctly:  -437.9379
## % of predictable range [ (model-null)/(1-null) ]:  0.01461866
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.67801     -0.18107      0.01870      0.08284
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4374 Residual
## Null Deviance:       1787
## Residual Deviance: 1738  AIC: 1889
## log likelihood:  -868.9388
## Nagelkerke R2:  0.03346421
## % pres/err predicted correctly:  -438.9345
## % of predictable range [ (model-null)/(1-null) ]:  0.01238142
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      3.4315      -0.1750       0.2269
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4375 Residual
## Null Deviance:       1787
## Residual Deviance: 1739  AIC: 1890
## log likelihood:  -869.5809
## Nagelkerke R2:  0.0325987
## % pres/err predicted correctly:  -439.0317
## % of predictable range [ (model-null)/(1-null) ]:  0.01216307
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.85207     -0.10560      0.45184     -0.02628
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4374 Residual
## Null Deviance:       1787
## Residual Deviance: 1738  AIC: 1891
## log likelihood:  -868.9834
## Nagelkerke R2:  0.03340411
## % pres/err predicted correctly:  -438.858
## % of predictable range [ (model-null)/(1-null) ]:  0.01255298
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.2241       0.1847
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4376 Residual
## Null Deviance:       1787
## Residual Deviance: 1757  AIC: 1912
## log likelihood:  -878.5075
## Nagelkerke R2:  0.02054066
## % pres/err predicted correctly:  -441.2941
## % of predictable range [ (model-null)/(1-null) ]:  0.007084109
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.320626     0.008924     0.115695
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4375 Residual
## Null Deviance:       1787
## Residual Deviance: 1757  AIC: 1913
## log likelihood:  -878.3602
## Nagelkerke R2:  0.02073997
## % pres/err predicted correctly:  -441.3226
## % of predictable range [ (model-null)/(1-null) ]:  0.007020301
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.5245      -0.0853
```

```
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4376 Residual
## Null Deviance:       1787
## Residual Deviance: 1783  AIC: 1937
## log likelihood:  -891.37
## Nagelkerke R2:  0.003079192
## % pres/err predicted correctly:  -443.8576
## % of predictable range [ (model-null)/(1-null) ]:  0.0013293
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.863
##
## Degrees of Freedom: 4377 Total (i.e. Null);  4377 Residual
## Null Deviance:       1787
## Residual Deviance: 1787  AIC: 1943
## log likelihood:  -893.6304
## Nagelkerke R2:  -6.624666e-16
## % pres/err predicted correctly:  -444.4497
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	1883.650	0.000000	1.000000	0.8661489	0.0388599	2.8852478	0.1726971	1.9273615	-0.2288785	-0.1995650	0.0266415
preserved ~ stimlen + I(pos^2) + pos	1888.785	5.134800	0.0767348	0.0664638	0.0334642	2.6780132	-0.1810694	0.0828442	NA	0.0187031	NA
preserved ~ stimlen + pos	1889.669	6.018157	0.0493370	0.0427333	0.0325987	2.4315362	-0.1750272	0.2268806	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	1890.767	7.118315	0.0284628	0.0246530	0.0334041	1.8520733	-0.1056022	0.4518445	-0.0262772	NA	NA
preserved ~ pos	1911.772	28.12690	0.0000008	0.0000007	0.0205402	2.2241253	NA	0.1847066	NA	NA	NA
preserved ~ I(pos^2) + pos	1912.824	29.17366	0.0000005	0.0000004	0.0207402	2.3206256	NA	0.1156945	NA	0.0089241	NA
preserved ~ stimlen	1937.382	53.73194	0.0000000	0.0000000	0.0030793	2.5245280	-0.0853000	NA	NA	NA	NA
preserved ~ 1	1942.595	58.94041	0.0000000	0.0000000	0.0000000	2.8628651	NA	NA	NA	NA	NA

```r
# plot no fragment data

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

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

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

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

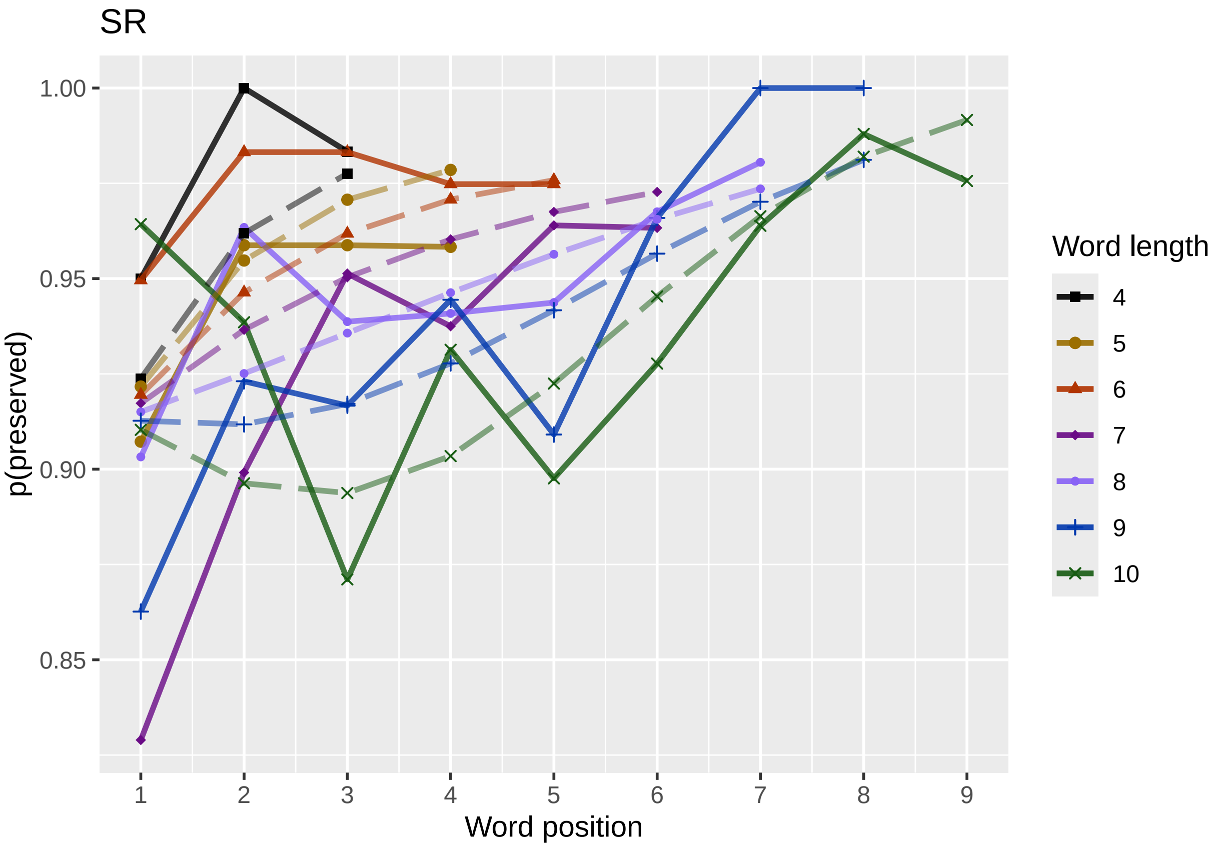
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.924 0.962 0.978 NA     NA     NA     NA     NA     NA
## 2       5 0.922 0.955 0.971  0.979 NA     NA     NA     NA     NA
## 3       6 0.920 0.946 0.962  0.971  0.976 NA     NA     NA     NA
## 4       7 0.917 0.937 0.950  0.960  0.967  0.973 NA     NA     NA
## 5       8 0.915 0.925 0.936  0.946  0.956  0.966  0.974 NA     NA
## 6       9 0.913 0.912 0.917  0.928  0.942  0.957  0.970  0.981 NA
## 7      10 0.910 0.896 0.894  0.903  0.922  0.945  0.966  0.982  0.992
```

```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=n
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: 0.81 - 1.02"
```

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

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

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

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

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.009704823
```

```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] 0.01158559
```

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

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

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

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

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

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

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

  print(" ")
  biggest_return_upward_row <- which(upward_dist == max(upward_dist))
  CurrentLabel <- "row with biggest return upward"
  print(CurrentLabel)
  print(biggest_return_upward_row)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, biggest_return_upward_row)
  print(" ")
  CurrentLabel<-"downward distance for row with the largest upward value"
  print(CurrentLabel)
  print(downward_dist[biggest_return_upward_row])
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, downward_dist[biggest_return_upwa

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

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

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

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

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

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

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

```
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     3.37677     -0.14847      0.14099      0.09272
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1958  AIC: 2113
## log likelihood:  -979.0471
## Nagelkerke R2:  0.02312754
## % pres/err predicted correctly:  -504.6511
## % of predictable range [ (model-null)/(1-null) ]:  0.009899249
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    3.476896    -0.151189     0.006592     0.087362     0.092626
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4411 Residual
## Null Deviance:       1995
## Residual Deviance: 1958  AIC: 2115
## log likelihood:  -978.9349
## Nagelkerke R2:  0.02326606
## % pres/err predicted correctly:  -504.5881
## % of predictable range [ (model-null)/(1-null) ]:  0.01002258
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##           3.3655           -0.1486            0.2251            0.1410           -0.0166
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4411 Residual
## Null Deviance:       1995
## Residual Deviance: 1957  AIC: 2115
## log likelihood:  -978.7209
## Nagelkerke R2:  0.02353041
## % pres/err predicted correctly:  -504.588
## % of predictable range [ (model-null)/(1-null) ]:  0.01002285
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      3.37717      -0.15074       0.14732       0.05561       0.01066
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4411 Residual
## Null Deviance:       1995
## Residual Deviance: 1958  AIC: 2115
## log likelihood:  -978.8238
## Nagelkerke R2:  0.02340328
## % pres/err predicted correctly:  -504.6661
## % of predictable range [ (model-null)/(1-null) ]:  0.009870036
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         3.460142         -0.151225          0.222696          0.006211          0.090501
## stimlen:log_freq
##        -0.016307
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4410 Residual
## Null Deviance:       1995
## Residual Deviance: 1957  AIC: 2116
## log likelihood:  -978.6217
## Nagelkerke R2:  0.02365289
## % pres/err predicted correctly:  -504.5255
## % of predictable range [ (model-null)/(1-null) ]:  0.01014532
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         log_freq              pos  stimlen:log_freq
##         3.36012         -0.15167          0.21828          0.14999          -0.02275
##    log_freq:pos
##         0.01605
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4410 Residual
## Null Deviance:       1995
## Residual Deviance: 1957  AIC: 2116
## log likelihood:  -978.2655
## Nagelkerke R2:  0.02409278
## % pres/err predicted correctly:  -504.602
## % of predictable range [ (model-null)/(1-null) ]:  0.009995524
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos          log_freq
##         3.446953         -0.152698          0.003420          0.115724         -0.057898
## I(pos^2):log_freq      pos:log_freq
##        -0.009021          0.085656
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4409 Residual
## Null Deviance:       1995
## Residual Deviance: 1956  AIC: 2116
## log likelihood:  -977.9031
## Nagelkerke R2:  0.0245403
## % pres/err predicted correctly:  -504.4607
## % of predictable range [ (model-null)/(1-null) ]:  0.01027222
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         I(pos^2)              pos  stimlen:I(pos^2)
##         1.05713          0.14274         -0.22159          1.89039           0.02671
##     stimlen:pos
##        -0.21582
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4410 Residual
## Null Deviance:       1995
## Residual Deviance: 1958  AIC: 2117
## log likelihood:  -978.7912
## Nagelkerke R2:  0.02344354
## % pres/err predicted correctly:  -504.5996
## % of predictable range [ (model-null)/(1-null) ]:  0.01000021
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          3.435631          -0.153724           0.091648           0.003728           0.116133
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.019688          -0.008276           0.084306
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4408 Residual
## Null Deviance:       1995
## Residual Deviance: 1955  AIC: 2118
## log likelihood:  -977.4897
## Nagelkerke R2:  0.02505065
## % pres/err predicted correctly:  -504.4217
## % of predictable range [ (model-null)/(1-null) ]:  0.01034853
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      3.5892      -0.1777       0.1407
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1965  AIC: 2120
## log likelihood:  -982.7258
## Nagelkerke R2:  0.01857952
## % pres/err predicted correctly:  -505.6918
## % of predictable range [ (model-null)/(1-null) ]:  0.007861515
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.69241     -0.18051      0.00680      0.08535
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1965  AIC: 2121
## log likelihood:  -982.6059
## Nagelkerke R2:  0.01872785
## % pres/err predicted correctly:  -505.6269
## % of predictable range [ (model-null)/(1-null) ]:  0.00798872
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.18079     -0.12913      0.28714     -0.01697
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1965  AIC: 2122
## log likelihood:  -982.4136
## Nagelkerke R2:  0.01896579
## % pres/err predicted correctly:  -505.6081
## % of predictable range [ (model-null)/(1-null) ]:  0.008025528
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      2.3525       0.1064       0.1260
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1971  AIC: 2129
## log likelihood:  -985.4835
## Nagelkerke R2:  0.01516508
## % pres/err predicted correctly:  -506.3782
## % of predictable range [ (model-null)/(1-null) ]:  0.006517527
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     2.343361      0.109839      0.105510      0.005968
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1971  AIC: 2131
## log likelihood:  -985.4153
## Nagelkerke R2:  0.01524962
## % pres/err predicted correctly:  -506.3908
## % of predictable range [ (model-null)/(1-null) ]:  0.006492904
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##         2.301586          -0.004787           0.144214          -0.013658          -0.009861
##     pos:log_freq
##         0.086462
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4410 Residual
## Null Deviance:       1995
## Residual Deviance: 1969  AIC: 2132
## log likelihood:  -984.5033
## Nagelkerke R2:  0.01637919
## % pres/err predicted correctly:  -506.1591
## % of predictable range [ (model-null)/(1-null) ]:  0.006946624
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     3.41254     -0.08694      0.09219
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1979  AIC: 2136
## log likelihood:  -989.3026
## Nagelkerke R2:  0.01042957
## % pres/err predicted correctly:  -507.4141
## % of predictable range [ (model-null)/(1-null) ]:  0.004489079
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)           stimlen          log_freq  stimlen:log_freq
##         3.40148          -0.08713           0.22433          -0.01656
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1978  AIC: 2137
## log likelihood:  -988.9771
## Nagelkerke R2:  0.01083345
## % pres/err predicted correctly:  -507.3481
## % of predictable range [ (model-null)/(1-null) ]:  0.004618468
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.6238      -0.1162
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2143
## log likelihood:  -992.959
## Nagelkerke R2:  0.005888126
## % pres/err predicted correctly:  -508.429
## % of predictable range [ (model-null)/(1-null) ]:  0.002501919
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3741       0.0943
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2145
## log likelihood:  -992.7941
## Nagelkerke R2:  0.006092996
## % pres/err predicted correctly:  -508.4731
## % of predictable range [ (model-null)/(1-null) ]:  0.002415632
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      2.3423      -0.0026       0.1156
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2147
## log likelihood:  -992.7764
## Nagelkerke R2:  0.006115059
## % pres/err predicted correctly:  -508.4671
## % of predictable range [ (model-null)/(1-null) ]:  0.002427309
## ***************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.718
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4415 Residual
## Null Deviance:      1995
## Residual Deviance: 1995  AIC: 2154
## log likelihood:  -997.6906
## Nagelkerke R2:  6.107643e-16
## % pres/err predicted correctly:  -509.7067
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC DeltaAICAICexpAICwtNagR2(Intercept)	stimlen log_freq	stimlen:log_freq pos	log_freq:pos	I(pos^2)	log_freq:I(pos^2)	stimlen:I(pos^2)		
preserved ~ stimlen + pos + log_freq	2113.4020100.0000002755012313376766 -0.1484712	0.0927185 NA	0.1409886 NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	2114.5858265573591520432332676896 -0.1511887	0.0926264 NA	0.0873622 NA	NA	0.0065924 NA		NA	NA	NA
preserved ~ stimlen * log_freq + pos	2114.5997049659451631273365535 -0.1486472	0.2251373 -0.0166007	0.1409971 NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	2115.2312869300778503502340377175 -0.1507421	0.0556100 NA	0.1473.1806600 NA	NA	NA		NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	2115.84420372394880811095273520142 -0.1512251	0.2226955 -0.0163066	0.0905014 NA	NA	0.0062111 NA		NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1516677	0.2182812	-0.0227451	0.1499894	NA	0.0160484	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	0.0034201	-0.0090205	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	1.8903929	NA	NA	-0.2215892	NA	NA	-0.2158242	0.0267131
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1537238	0.0916485	-0.0196885	[illegible]	NA	[illegible]	[illegible]	NA	-0.0082758	NA	NA
preserved ~ stimlen + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1777252	NA	NA	[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1805053	NA	NA	[illegible]	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen * pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1291254	NA	NA	[illegible]	NA	NA	NA	NA	NA	-0.0169723	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	NA	[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	NA	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0136582	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0869380	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0871270	0.2243283	-0.0165564	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1161725	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	0.0942957	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	[illegible]	NA	NA	-0.0025996	NA	NA	NA	NA
preserved ~ 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     3.37677     -0.14847      0.14099      0.09272
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1958  AIC: 2113
```

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

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

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

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

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

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

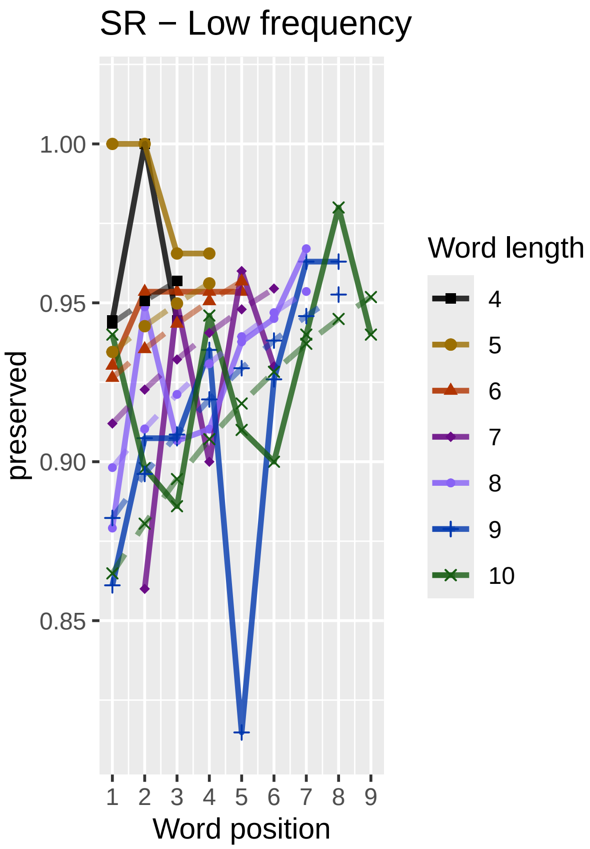
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

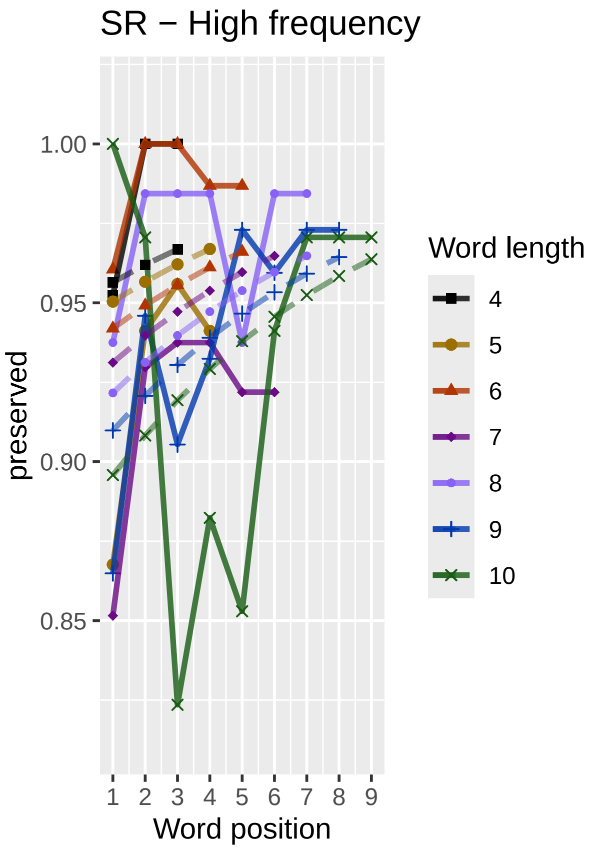
```r
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
```

```
## Warning: Removed 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()`).
```

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


SR − Low frequency


SR − High frequency

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.156       -1.048
```

```
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1775  AIC: 1909
## log likelihood:  -887.7306
## Nagelkerke R2:  0.1336287
## % pres/err predicted correctly:  -454.9682
## % of predictable range [ (model-null)/(1-null) ]:  0.107182
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0260       0.3358
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1904  AIC: 2053
## log likelihood:  -952.1337
## Nagelkerke R2:  0.05617141
## % pres/err predicted correctly:  -498.8994
## % of predictable range [ (model-null)/(1-null) ]:  0.02116158
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.6238      -0.1162
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2143
## log likelihood:  -992.959
## Nagelkerke R2:  0.005888126
## % pres/err predicted correctly:  -508.429
## % of predictable range [ (model-null)/(1-null) ]:  0.002501919
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3741       0.0943
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2145
## log likelihood:  -992.7941
## Nagelkerke R2:  0.006092996
## % pres/err predicted correctly:  -508.4731
## % of predictable range [ (model-null)/(1-null) ]:  0.002415632
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      2.3423      -0.0026       0.1156
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2147
## log likelihood:  -992.7764
## Nagelkerke R2:  0.006115059
## % pres/err predicted correctly:  -508.4671
## % of predictable range [ (model-null)/(1-null) ]:  0.002427309
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.718
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4415 Residual
## Null Deviance:       1995
## Residual Deviance: 1995  AIC: 2154
## log likelihood:  -997.6906
## Nagelkerke R2:  6.107643e-16
## % pres/err predicted correctly:  -509.7067
## % 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))
```

```r
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	1908.774	0.0000	1	1	0.1336287	3.156011	NA	-1.048339	NA	NA	NA
preserved ~ CumPres	2052.823	144.0487	0	0	0.0561714	2.026011	0.3357981	NA	NA	NA	NA
preserved ~ stimlen	2142.843	234.0683	0	0	0.0058883	3.623823	NA	NA	NA	NA	-0.1161725
preserved ~ pos	2144.567	235.7922	0	0	0.0060932	2.374112	NA	NA	NA	0.0942957	NA
preserved ~ (I(pos^2) + pos)	2146.729	237.9544	0	0	0.0061152	2.342278	NA	NA	-0.0025996	0.1156227	NA
preserved ~ 1	2153.840	245.0660	0	0	0.0000000	2.717753	NA	NA	NA	NA	NA

```r
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)
}
```

```r
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	0.9395509	579
O	0.9302941	2040
P	0.9722222	36
S	0.8850129	258
V	0.9563318	1503

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.229       -1.150
##
## Degrees of Freedom: 4121 Total (i.e. Null);  4120 Residual
## Null Deviance:       1797
## Residual Deviance: 1570  AIC: 1687
## log likelihood:  -785.0982
## Nagelkerke R2:  0.1516605
## % pres/err predicted correctly:  -398.2996
## % of predictable range [ (model-null)/(1-null) ]:  0.1224815
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      CumPres
##      2.0617       0.3764
##
## Degrees of Freedom: 4121 Total (i.e. Null);  4120 Residual
## Null Deviance:       1797
## Residual Deviance: 1707  AIC: 1838
## log likelihood:  -853.6055
## Nagelkerke R2:  0.06114645
## % pres/err predicted correctly:  -443.7873
## % of predictable range [ (model-null)/(1-null) ]:  0.02251562
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.40700      0.09954
##
## Degrees of Freedom: 4121 Total (i.e. Null);  4120 Residual
## Null Deviance:       1797
## Residual Deviance: 1788  AIC: 1929
## log likelihood:  -893.761
## Nagelkerke R2:  0.006676245
## % pres/err predicted correctly:  -452.8592
## % of predictable range [ (model-null)/(1-null) ]:  0.002578837
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       3.557       -0.101
##
## Degrees of Freedom: 4121 Total (i.e. Null);  4120 Residual
## Null Deviance:       1797
## Residual Deviance: 1791  AIC: 1931
## log likelihood:  -895.4159
## Nagelkerke R2:  0.004408484
## % pres/err predicted correctly:  -453.1808
## % of predictable range [ (model-null)/(1-null) ]:  0.001872176
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.23712     -0.01392      0.21402
##
## Degrees of Freedom: 4121 Total (i.e. Null);  4119 Residual
## Null Deviance:       1797
## Residual Deviance: 1787  AIC: 1931
## log likelihood:  -893.2847
## Nagelkerke R2:  0.007328474
## % pres/err predicted correctly:  -452.7391
## % of predictable range [ (model-null)/(1-null) ]:  0.002842825
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.771
##
## Degrees of Freedom: 4121 Total (i.e. Null);  4121 Residual
## Null Deviance:       1797
## Residual Deviance: 1797  AIC: 1938
## log likelihood:  -898.6293
## Nagelkerke R2:  -6.283225e-16
## % pres/err predicted correctly:  -454.0327
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1686.518	0.0000	1	1	0.151660	3.228915	NA	-1.149614	NA	NA	NA
preserved ~ CumPres	1838.496	151.9781	0	0	0.061146	2.061680	0.3763949	NA	NA	NA	NA
preserved ~ pos	1929.163	242.6444	0	0	0.006676	2.406999	NA	NA	NA	0.0995395	NA
preserved ~ stimlen	1930.795	244.2770	0	0	0.004408	3.557455	NA	NA	NA	NA	-0.1009988
preserved ~ (I(pos^2) + pos)	1931.245	244.7266	0	0	0.007328	2.237119	NA	NA	-0.0139245	0.2140236	NA
preserved ~ 1	1937.938	251.4201	0	0	0.000000	2.771000	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
##       3.134       -1.168
##
## Degrees of Freedom: 3542 Total (i.e. Null);  3541 Residual
## Null Deviance:       1544
## Residual Deviance: 1413  AIC: 1507
## log likelihood:  -706.628
## Nagelkerke R2:  0.102607
## % pres/err predicted correctly:  -358.6258
## % of predictable range [ (model-null)/(1-null) ]:  0.07726453
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.1512       0.3872
##
## Degrees of Freedom: 3542 Total (i.e. Null);  3541 Residual
## Null Deviance:       1544
## Residual Deviance: 1480  AIC: 1583
## log likelihood:  -740.0969
## Nagelkerke R2:  0.05057346
## % pres/err predicted correctly:  -381.4588
## % of predictable range [ (model-null)/(1-null) ]:  0.01867936
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.3681       0.1133
##
## Degrees of Freedom: 3542 Total (i.e. Null);  3541 Residual
## Null Deviance:       1544
## Residual Deviance: 1533  AIC: 1642
## log likelihood:  -766.3485
## Nagelkerke R2:  0.009066926
## % pres/err predicted correctly:  -387.4289
## % of predictable range [ (model-null)/(1-null) ]:  0.003361101
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.22916     -0.01207      0.21117
##
## Degrees of Freedom: 3542 Total (i.e. Null);  3540 Residual
## Null Deviance:       1544
## Residual Deviance: 1532  AIC: 1644
## log likelihood:  -766.0347
## Nagelkerke R2:  0.009566694
## % pres/err predicted correctly:  -387.3349
## % of predictable range [ (model-null)/(1-null) ]:  0.003602328
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.3984      -0.0804
##
## Degrees of Freedom: 3542 Total (i.e. Null);  3541 Residual
## Null Deviance:       1544
## Residual Deviance: 1540  AIC: 1648
## log likelihood:  -770.2487
## Nagelkerke R2:  0.002847596
## % pres/err predicted correctly:  -388.2569
## % of predictable range [ (model-null)/(1-null) ]:  0.001236474
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.776
```

```
##
## Degrees of Freedom: 3542 Total (i.e. Null);  3542 Residual
## Null Deviance:       1544
## Residual Deviance: 1544  AIC: 1652
## log likelihood:  -772.0316
## Nagelkerke R2:  -6.285634e-16
## % pres/err predicted correctly:  -388.7388
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

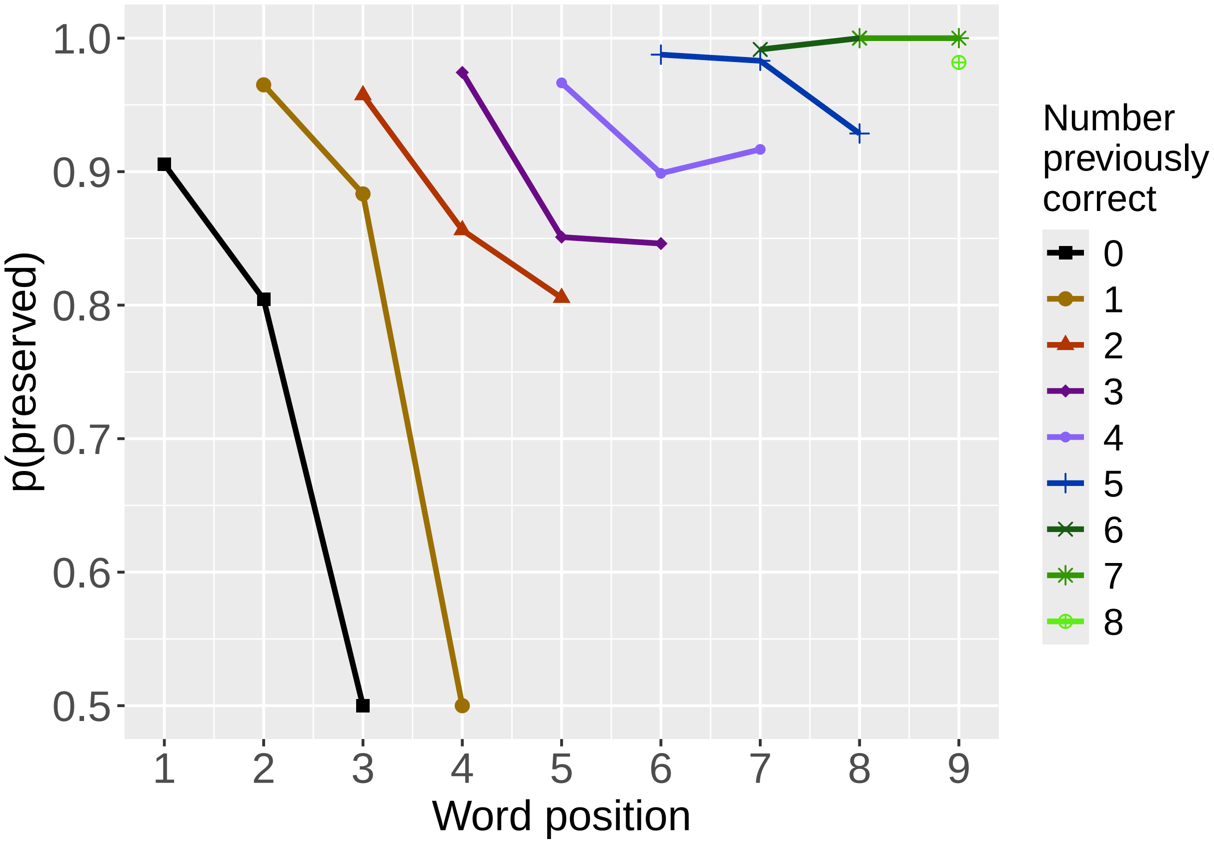
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1507.208	0.00000	1	1	0.102607	3.133828	NA	-1.167817	NA	NA	NA
preserved ~ CumPres	1582.532	75.32366	0	0	0.050573	2.151156	0.3871844	NA	NA	NA	NA
preserved ~ pos	1641.810	134.60227	0	0	0.009066	2.368075	NA	NA	NA	0.1133204	NA
preserved ~ (I(pos^2) + pos)	1643.930	136.72177	0	0	0.009566	2.229159	NA	NA	-0.0120661	0.2111735	NA
preserved ~ stimlen	1648.248	141.04049	0	0	0.002847	3.398432	NA	NA	NA	NA	-0.080402
preserved ~ 1	1651.595	144.38732	0	0	0.000000	2.775542	NA	NA	NA	NA	NA

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

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

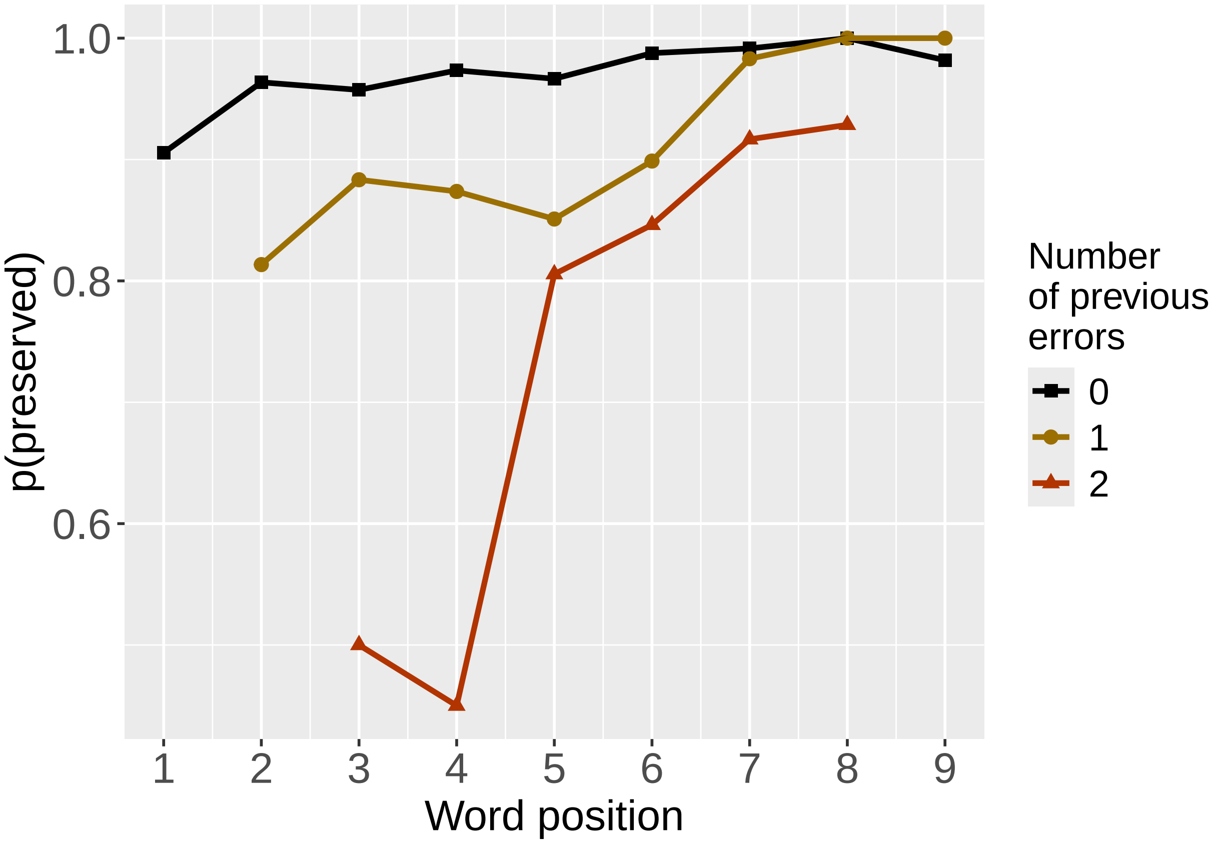
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

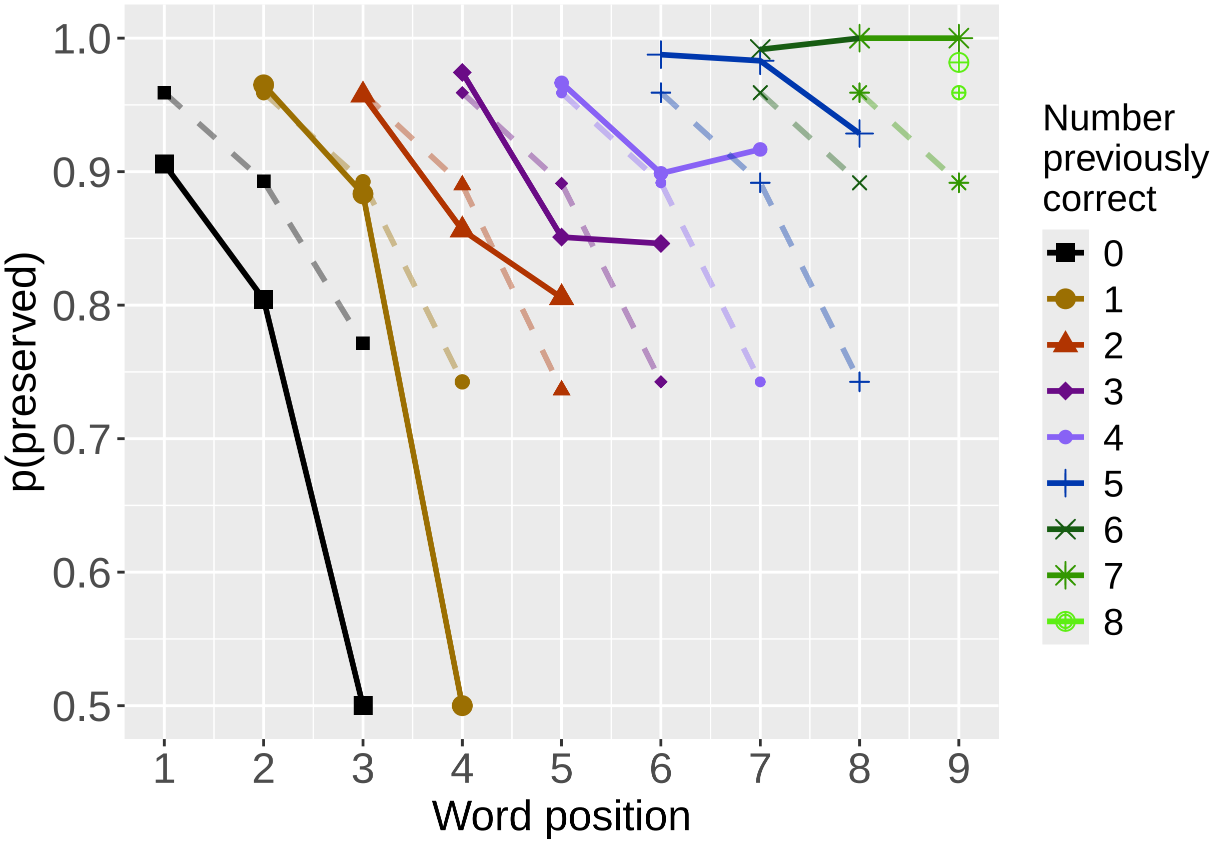
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

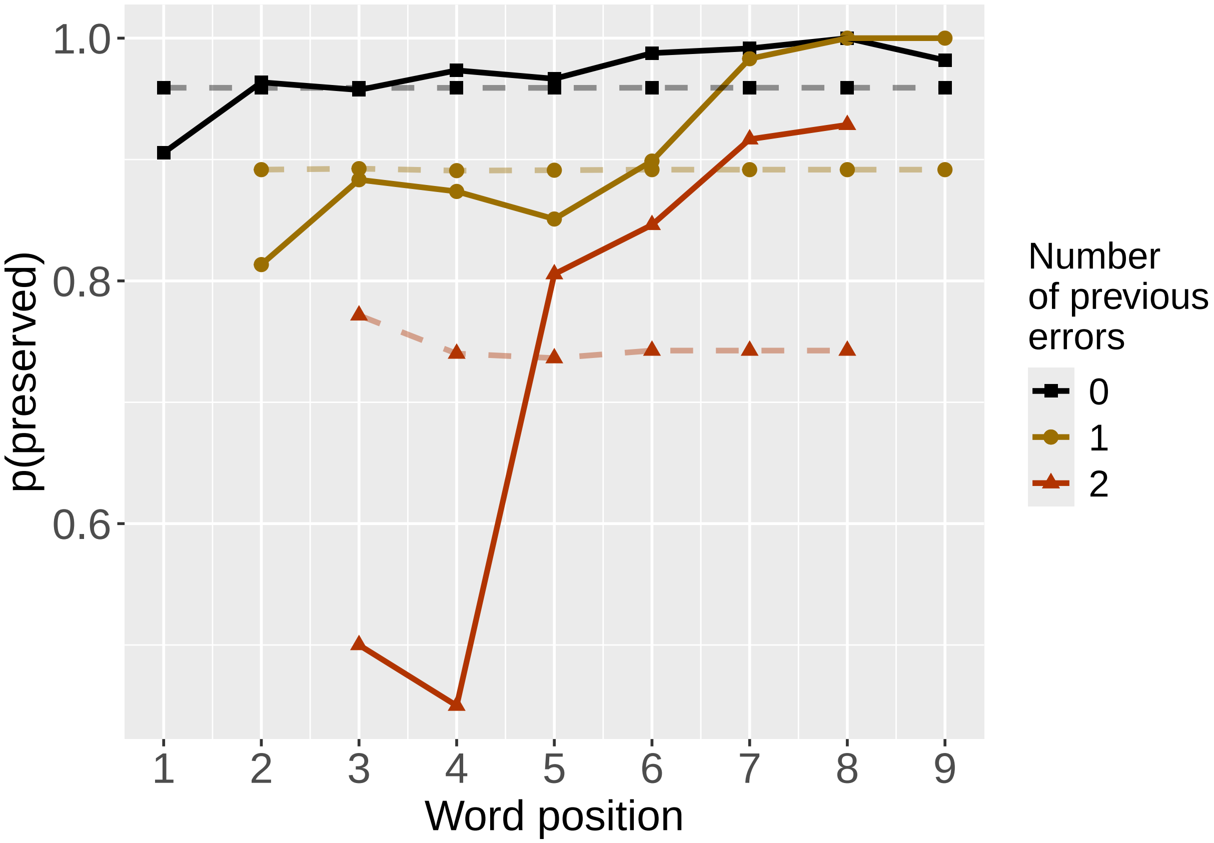
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.16884     -1.47928      0.01313      0.27407
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1678  AIC: 1800
## log likelihood:  -839.2484
## Nagelkerke R2:  0.1904654
## % pres/err predicted correctly:  -437.5489
## % of predictable range [ (model-null)/(1-null) ]:  0.1412902
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.156       -1.048
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1775  AIC: 1909
## log likelihood:  -887.7306
## Nagelkerke R2:  0.1336287
## % pres/err predicted correctly:  -454.9682
## % of predictable range [ (model-null)/(1-null) ]:  0.107182
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      2.3423      -0.0026       0.1156
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2147
## log likelihood:  -992.7764
## Nagelkerke R2:  0.006115059
## % pres/err predicted correctly:  -508.4671
## % of predictable range [ (model-null)/(1-null) ]:  0.002427309
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1799.997	0.0000	1	1	0.1904654	2.168836	-1.479280	0.0131333	0.2740670

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1908.774	108.7777	0	0	0.1336287	3.156011	-1.048339	NA	NA
preserved ~ I(posˆ2) + pos	2146.729	346.7321	0	0	0.0061151	2.342278	NA	-0.0025996	0.1156227

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.156       -1.048
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1775  AIC: 1909
## log likelihood:  -887.7306
## Nagelkerke R2:  0.1336287
## % pres/err predicted correctly:  -454.9682
## % of predictable range [ (model-null)/(1-null) ]:  0.107182
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##      3.3804      -1.0387      -0.0293
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1775  AIC: 1910
## log likelihood:  -887.4732
## Nagelkerke R2:  0.1339337
## % pres/err predicted correctly:  -455.2407
## % of predictable range [ (model-null)/(1-null) ]:  0.1066484
## **************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.6238      -0.1162
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2143
## log likelihood:  -992.959
## Nagelkerke R2:  0.005888126
## % pres/err predicted correctly:  -508.429
## % of predictable range [ (model-null)/(1-null) ]:  0.002501919
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	1908.774	0.0000000	1.0000000	0.6035046	0.1336287	3.156011	-1.048339	NA
preserved ~ CumErr + stimlen	1909.615	0.8401781	0.6569883	0.3964954	0.1339337	3.380411	-1.038663	-0.0293043
preserved ~ stimlen	2142.843	234.0683254	0.0000000	0.0000000	0.0058881	3.623823	NA	-0.1161725

```
########
# 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
##      2.4014      -1.1003       0.3759
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1679  AIC: 1800
## log likelihood:  -839.5785
## Nagelkerke R2:  0.1900826
## % pres/err predicted correctly:  -438.0329
## % of predictable range [ (model-null)/(1-null) ]:  0.1403425
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.156       -1.048
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1775  AIC: 1909
## log likelihood:  -887.7306
## Nagelkerke R2:  0.1336287
## % pres/err predicted correctly:  -454.9682
## % of predictable range [ (model-null)/(1-null) ]:  0.107182
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.0260       0.3358
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1904  AIC: 2053
## log likelihood:  -952.1337
## Nagelkerke R2:  0.05617141
## % pres/err predicted correctly:  -498.8994
## % of predictable range [ (model-null)/(1-null) ]:  0.02116158
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	1799.725	0.0000	1	1	0.1900826	2.401445	-1.100343	0.3758718
preserved ~ CumErr	1908.774	109.0494	0	0	0.1336287	3.156011	-1.048339	NA
preserved ~ CumPres	2052.823	253.0981	0	0	0.0561714	2.026011	NA	0.3357981

```
########
# 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  
##      2.0256      -1.4762       0.3759  
## 
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995 
## Residual Deviance: 1679  AIC: 1800
## log likelihood:  -839.5785
## Nagelkerke R2:  0.1900826
## % pres/err predicted correctly:  -438.0329
## % of predictable range [ (model-null)/(1-null) ]:  0.1403425
## ***************************
## model index:  1
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)       CumErr  
##       3.156       -1.048  
## 
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995 
## Residual Deviance: 1775  AIC: 1909
## log likelihood:  -887.7306
## Nagelkerke R2:  0.1336287
## % pres/err predicted correctly:  -454.9682
## % of predictable range [ (model-null)/(1-null) ]:  0.107182
## ***************************
## model index:  3
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
```

```
## Coefficients:
## (Intercept)          pos
##      2.3741       0.0943
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1986  AIC: 2145
## log likelihood:  -992.7941
## Nagelkerke R2:  0.006092996
## % pres/err predicted correctly:  -508.4731
## % of predictable range [ (model-null)/(1-null) ]:  0.002415632
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	1799.725	0.0000	1	1	0.1900826	2.025573	-1.476215	0.3758718
preserved ~ CumErr	1908.774	109.0494	0	0	0.1336287	3.156011	-1.048339	NA
preserved ~ pos	2144.567	344.8416	0	0	0.0060930	2.374112	NA	0.0942957

```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 + pos	1799.725	0.0000000	1.0000000	1.0000000	0.1900826	2.025573	-1.476215	NA	0.3758718	NA	NA
preserved ~ CumErr + CumPres	1799.725	0.0000000	1.0000000	1.0000000	0.1900826	2.401445	-1.100343	NA	NA	NA	0.3758718
preserved ~ CumErr + I(pos^2) + pos	1799.997	0.0000000	1.0000000	1.0000000	0.1904654	2.168836	-1.479280	0.0131333	0.2740670	NA	NA
preserved ~ CumErr	1908.774	108.7777001	0.0000000	0.0000000	0.1336287	3.156011	-1.048339	NA	NA	NA	NA
preserved ~ CumErr	1908.774	0.0000000	1.0000000	0.6035046	0.1336287	3.156011	-1.048339	NA	NA	NA	NA
preserved ~ CumErr	1908.774	109.0494138	0.0000000	0.0000000	0.1336287	3.156011	-1.048339	NA	NA	NA	NA
preserved ~ CumErr	1908.774	109.0494138	0.0000000	0.0000000	0.1336287	3.156011	-1.048339	NA	NA	NA	NA
preserved ~ CumErr + stimlen	1909.616	0.8401781	0.6569888	0.3396495	0.1339337	3.380411	-1.038663	NA	NA	-0.0293043	NA
preserved ~ CumPres	2052.823	353.0980992	0.0000000	0.0000000	0.0056171	2.026011	NA	NA	NA	NA	0.3357981
preserved ~ stimlen	2142.843	334.0683254	0.0000000	0.0000000	0.0058881	3.623823	NA	NA	NA	-0.1161725	NA
preserved ~ pos	2144.567	344.8416059	0.0000000	0.0000000	0.0060930	2.374112	NA	NA	0.0942957	NA	NA
preserved ~ I(pos^2) + pos	2146.729	346.7321343	0.0000000	0.0000000	0.0061151	2.342278	NA	-0.0025996	0.1156227	NA	NA

```
# 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          pos      stimlen     log_freq  
##     3.08946     -1.47102      0.42503     -0.15631      0.08199  
## 
## Degrees of Freedom: 4415 Total (i.e. Null);  4411 Residual
## Null Deviance:       1995 
## Residual Deviance: 1655  AIC: 1773
## log likelihood:  -827.7348
## Nagelkerke R2:  0.2037807
## % pres/err predicted correctly:  -436.4206
## % of predictable range [ (model-null)/(1-null) ]:  0.1434996
## ***************************
## model index:  4
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen
##      3.2868      -1.4807       0.4234      -0.1829
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1660  AIC: 1777
## log likelihood:  -830.2406
## Nagelkerke R2:  0.2008887
## % pres/err predicted correctly:  -436.6601
## % of predictable range [ (model-null)/(1-null) ]:  0.1430306
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq
##      2.0006      -1.4654       0.3897       0.1178
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1668  AIC: 1788
## log likelihood:  -833.9696
## Nagelkerke R2:  0.1965789
## % pres/err predicted correctly:  -437.4132
## % of predictable range [ (model-null)/(1-null) ]:  0.1415558
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.0256      -1.4762       0.3759
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1679  AIC: 1800
## log likelihood:  -839.5785
## Nagelkerke R2:  0.1900826
## % pres/err predicted correctly:  -438.0329
## % of predictable range [ (model-null)/(1-null) ]:  0.1403425
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       2.718
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4415 Residual
## Null Deviance:       1995
## Residual Deviance: 1995  AIC: 2154
## log likelihood:  -997.6906
## Nagelkerke R2:  6.107643e-16
## % pres/err predicted correctly:  -509.7067
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos	log_freq	stimlen
preserved ~ CumErr + pos + stimlen + log_freq	1772.985	0.000000	1.0000000	0.8738431	0.2037807	7.089458	-1.471016	0.4250347	0.0819946	-0.1563084
preserved ~ CumErr + pos + stimlen	1776.864	3.878738	0.1437946	0.1256540	0.2008887	7.286819	-1.480673	0.4234051	NA	-0.1828513
preserved ~ CumErr + pos + log_freq	1787.911	14.925790	0.0005740	0.0005016	0.1965782	9.000642	-1.465411	0.3897456	0.1178383	NA
preserved ~ CumErr + pos	1799.725	26.739612	0.0000016	0.0000014	0.1900826	6.025573	-1.476215	0.3758718	NA	NA
preserved ~ 1	2153.840	380.855003	0.0000000	0.0000000	0.0000000	2.717753	NA	NA	NA	NA

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

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

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

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

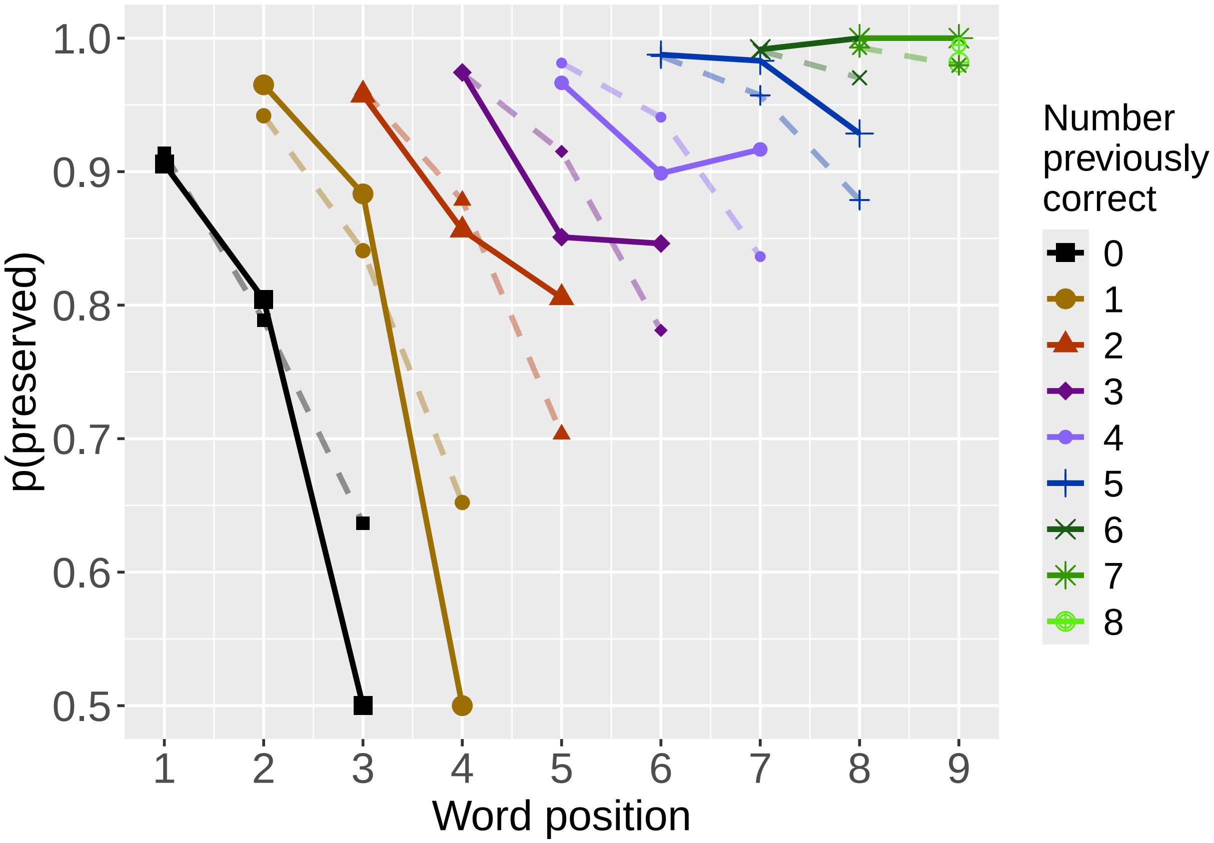
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   1958.1 2073.6
## pos       1   1770.3 1885.9
## stimlen   1   1667.9 1783.5
## log_freq  1   1660.5 1776.0
## <none>        1655.5 1773.0
```

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

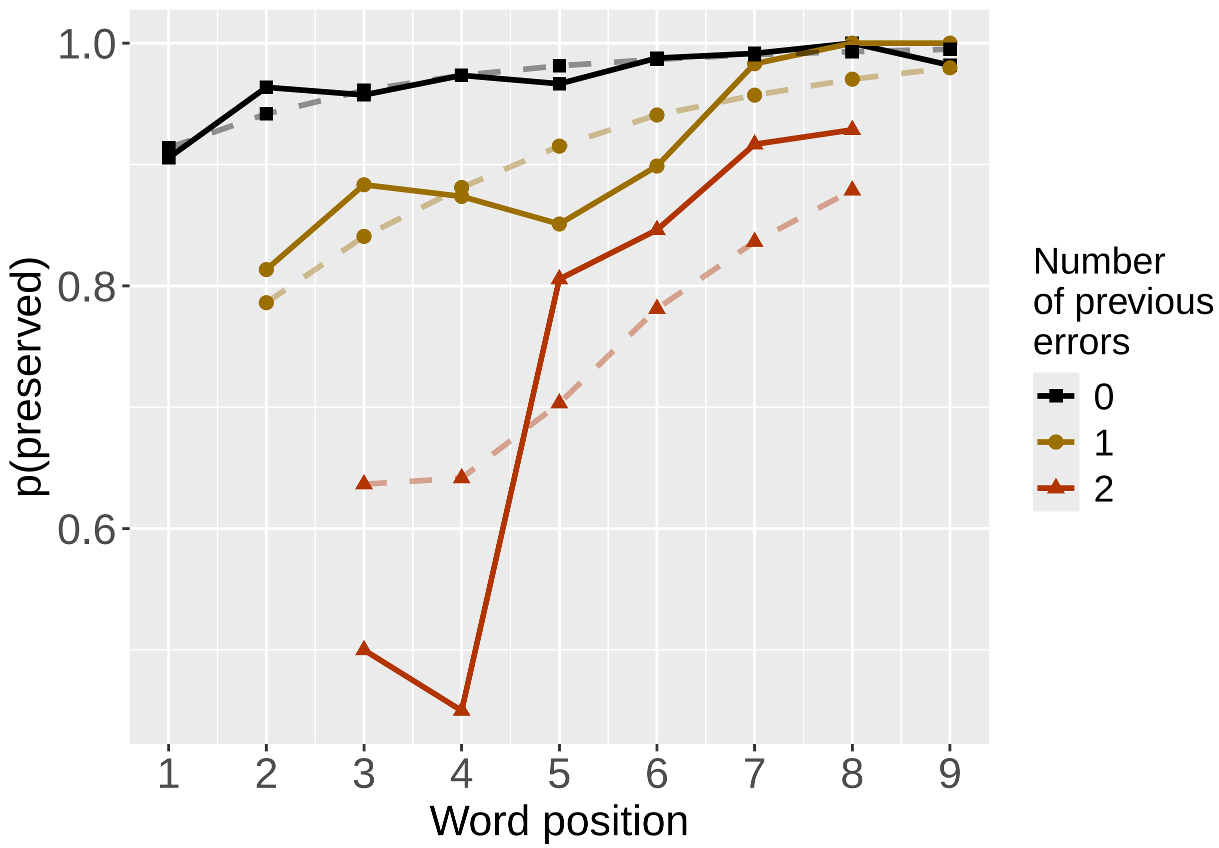
```
print(PrevCorPlot)
```



```r
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.
```

```r
print(PrevErrPlot)
```



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

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

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

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

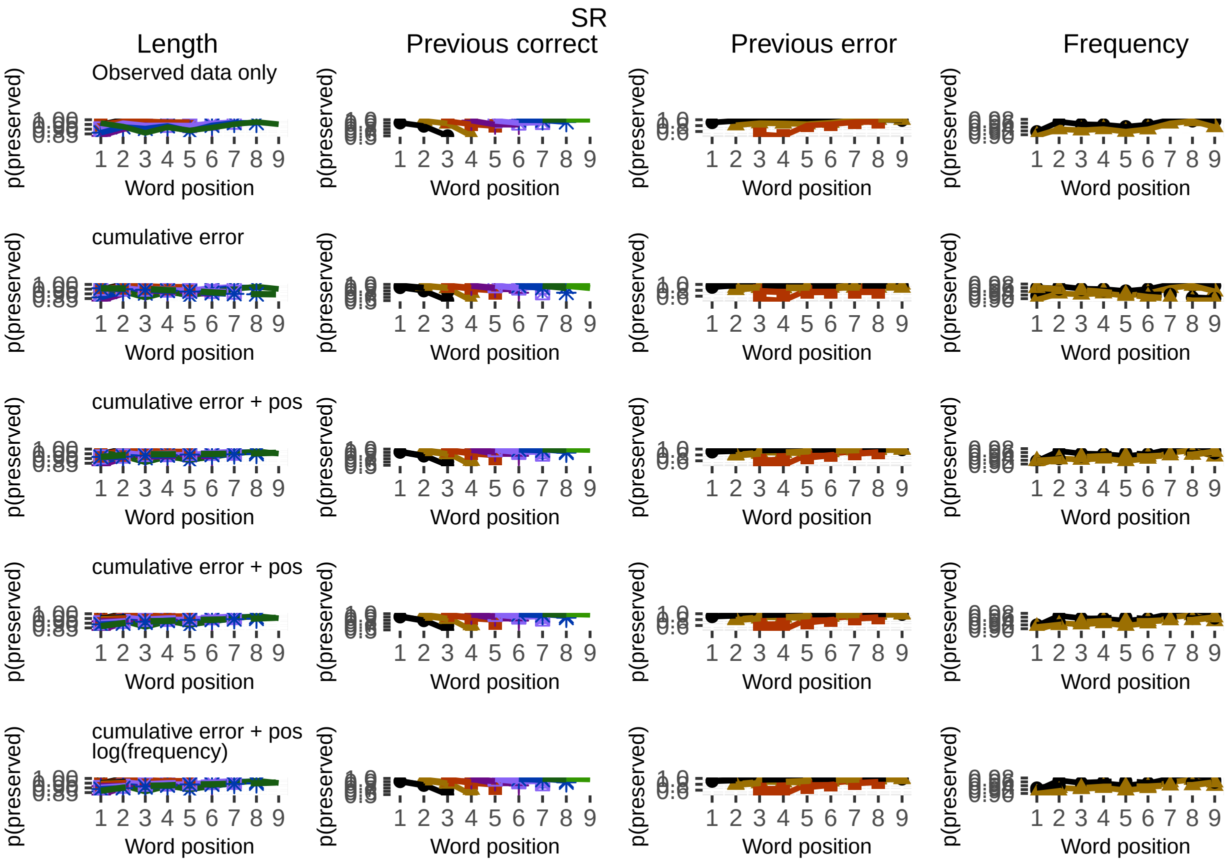
```
if(DoSimulations){
  BestModelFormulaL3Rnd <- BestModelFormulaL3
  if(grepl("CumPres",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumPres","RndCumPres",BestModelFormulaL3Rnd)
  }
  if(grepl("CumErr",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumErr","RndCumErr",BestModelFormulaL3Rnd)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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.156       -1.048
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4414 Residual
## Null Deviance:       1995
## Residual Deviance: 1775  AIC: 1909
## log likelihood:  -887.7306
## Nagelkerke R2:  0.1336287
## % pres/err predicted correctly:  -454.9682
## % of predictable range [ (model-null)/(1-null) ]:  0.107182
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.0256      -1.4762       0.3759
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4413 Residual
## Null Deviance:       1995
## Residual Deviance: 1679  AIC: 1800
## log likelihood:  -839.5785
## Nagelkerke R2:  0.1900826
## % pres/err predicted correctly:  -438.0329
## % of predictable range [ (model-null)/(1-null) ]:  0.1403425
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen
##      3.2868      -1.4807       0.4234      -0.1829
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4412 Residual
## Null Deviance:       1995
## Residual Deviance: 1660  AIC: 1777
## log likelihood:  -830.2406
## Nagelkerke R2:  0.2008887
## % pres/err predicted correctly:  -436.6601
## % of predictable range [ (model-null)/(1-null) ]:  0.1430306
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos      stimlen     log_freq
##     3.08946     -1.47102      0.42503     -0.15631      0.08199
##
## Degrees of Freedom: 4415 Total (i.e. Null);  4411 Residual
## Null Deviance:       1995
## Residual Deviance: 1655  AIC: 1773
## log likelihood:  -827.7348
## Nagelkerke R2:  0.2037807
## % pres/err predicted correctly:  -436.4206
## % of predictable range [ (model-null)/(1-null) ]:  0.1434996
## ***************************
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

	CumErr	pos	stimlen	log_freq
McFadden	0.1359000	0.0335822	0.0064404	0.0047855
SquaredCorrelation	0.0636593	0.0153523	0.0030130	0.0022656
Nagelkerke	0.0636593	0.0153523	0.0030130	0.0022656
Estrella	0.0692369	0.0176089	0.0032957	0.0024138

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

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

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

```
##                                                                 model deviance
## CumErr + pos + stimlen + log_freq CumErr + pos + stimlen + log_freq 1655.470
## CumErr + pos + stimlen                        CumErr + pos + stimlen 1660.481
## CumErr + pos                                            CumErr + pos 1679.157
## CumErr                                                        CumErr 1775.461
## null                                                            null 1995.381
##                                   deviance_explained percent_explained
## CumErr + pos + stimlen + log_freq           339.9115          17.03492
## CumErr + pos + stimlen                      334.8999          16.78376
## CumErr + pos                                316.2241          15.84780
## CumErr                                      219.9199          11.02145
## null                                          0.0000           0.00000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + pos + stimlen + log_freq                     100.00000               1.474395
## CumErr + pos + stimlen                                 98.52561               5.494322
## CumErr + pos                                           93.03128              28.332131
## CumErr                                                 64.69915              64.699153
## null                                                         NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + stimlen + log_freq	1655.470	339.9115
CumErr + pos + stimlen	1660.481	334.8999
CumErr + pos	1679.157	316.2241
CumErr	1775.461	219.9199
null	1995.381	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + stimlen + log_freq	17.03492	100.00000	1.474395
CumErr + pos + stimlen	16.78376	98.52561	5.494322
CumErr + pos	15.84780	93.03128	28.332131
CumErr	11.02145	64.69915	64.699153
null	0.00000	NA	0.000000

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

```
##          Nagelkerke
## CumErr   0.75523852
## pos      0.18213640
## stimlen  0.03574601
## log_freq 0.02687908
```

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

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

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

## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.6408576	1775.461
preserved ~ CumErr+pos+stimlen+log_freq	0.8809226	1655.470
preserved ~ CumErr+pos+stimlen	0.8836785	1660.481
preserved ~ CumErr+pos	0.8849513	1679.157

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

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

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

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

```
##                                     model p_accounted_for model_deviance diff_CumErr
## 1                      preserved ~ CumErr       0.6408576       1775.461   0.0000000
## 2 preserved ~ CumErr+pos+stimlen+log_freq       0.8809226       1655.470   0.2400650
## 3          preserved ~ CumErr+pos+stimlen       0.8836785       1660.481   0.2428210
## 4                  preserved ~ CumErr+pos       0.8849513       1679.157   0.2440938
##   diff_CumErr+pos+stimlen+log_freq diff_CumErr+pos+stimlen diff_CumErr+pos
## 1                     -0.240065003            -0.242820981    -0.244093766
## 2                      0.000000000            -0.002755978    -0.004028763
## 3                      0.002755978             0.000000000    -0.001272785
## 4                      0.004028763             0.001272785     0.000000000
```

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

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

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

model	diff_CumErr	diff_CumErr+pos+stimlen+log_freq	diff_CumErr+pos+stimlen
preserved ~ CumErr	0.0000000	-0.2400650	-0.2428210
preserved ~ CumErr+pos+stimlen+log_freq	0.2400650	0.0000000	-0.0027560
preserved ~ CumErr+pos+stimlen	0.2428210	0.0027560	0.0000000
preserved ~ CumErr+pos	0.2440938	0.0040288	0.0012728