PV - repetition - Serial position analysis (v5)

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

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

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

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

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

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

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	500	33	118	NA	NA	651
2	60	NA	400	91	100	651
3	284	NA	158	197	12	651
4	281	NA	210	67	36	594
5	216	NA	193	60	36	505
6	183	1	124	64	21	393
7	162	NA	82	24	15	283
8	76	NA	48	22	3	149
9	65	NA	1	NA	6	72

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7680492	0.0506912	0.1812596	NA	NA	651
2	0.0921659	NA	0.6144393	0.1397849	0.1536098	651
3	0.4362519	NA	0.2427035	0.3026114	0.0184332	651
4	0.4730640	NA	0.3535354	0.1127946	0.0606061	594
5	0.4277228	NA	0.3821782	0.1188119	0.0712871	505
6	0.4656489	0.0025445	0.3155216	0.1628499	0.0534351	393

pos_factor	O	P	V	1	S	total
7	0.5724382	NA	0.2897527	0.0848057	0.0530035	283
8	0.5100671	NA	0.3221477	0.1476510	0.0201342	149
9	0.9027778	NA	0.0138889	NA	0.0833333	72

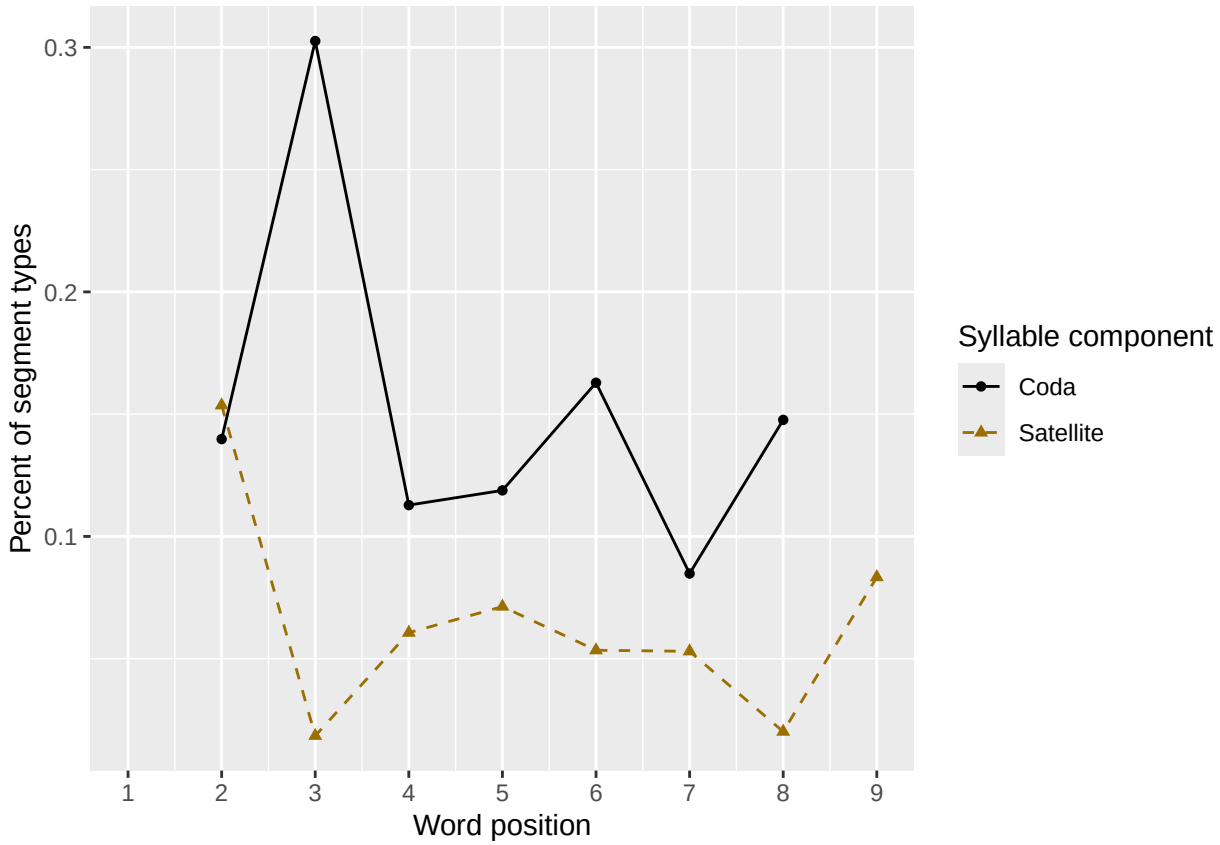
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.947 0.974 0.895 NA     NA     NA     NA     NA     NA
## 2       5 0.876 0.948 0.925  0.925 NA     NA     NA     NA     NA
## 3       6 0.920 0.866 0.835  0.860  0.872 NA     NA     NA     NA
## 4       7 0.859 0.923 0.805  0.852  0.895  0.823 NA     NA     NA
## 5       8 0.862 0.903 0.854  0.801  0.812  0.841  0.831 NA     NA
## 6       9 0.838 0.840 0.779  0.762  0.753  0.827  0.753  0.786 NA
## 7      10 0.872 0.871 0.710  0.644  0.698  0.694  0.774  0.833  0.792
```

```
# 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.
```

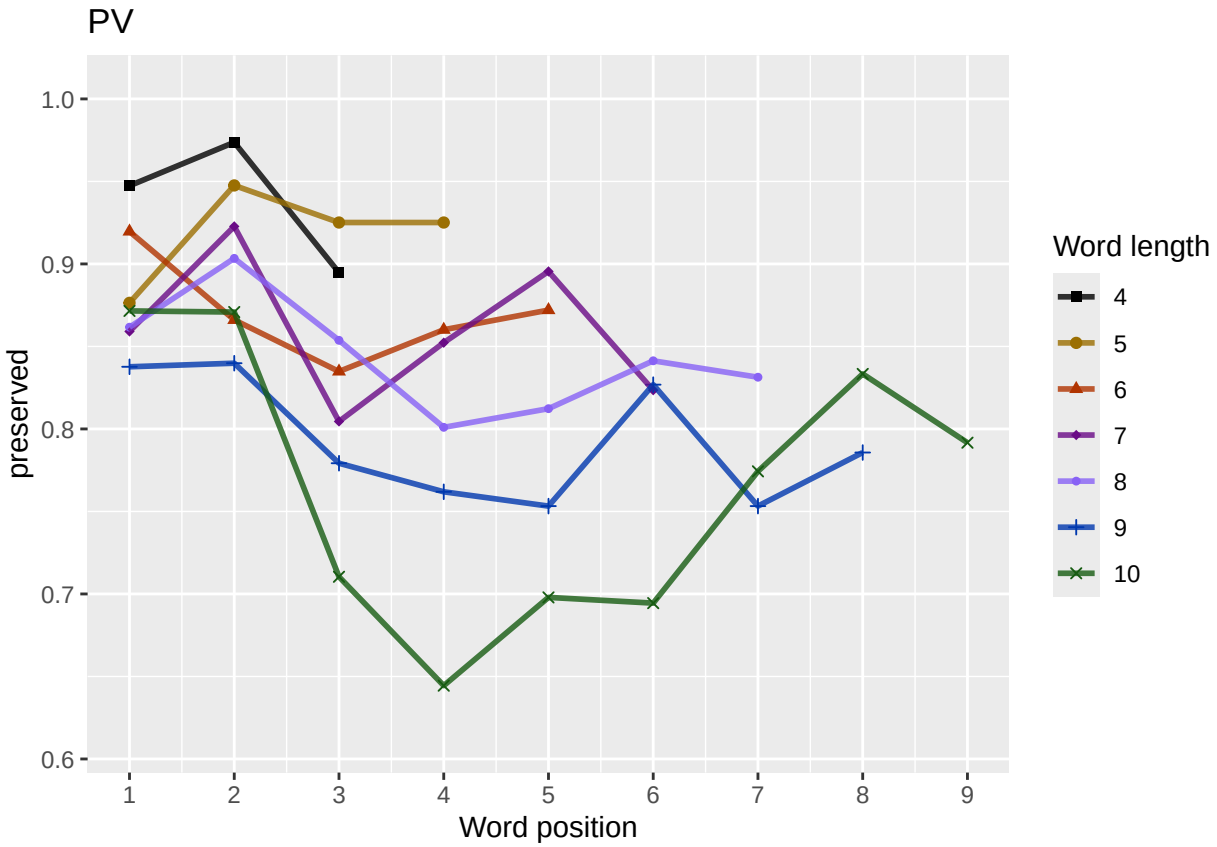
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    57    57    57    NA    NA    NA    NA    NA    NA
## 2       5    89    89    89    89    NA    NA    NA    NA    NA
## 3       6   112   112   112   112   112    NA    NA    NA    NA
## 4       7   110   110   110   110   110   110    NA    NA    NA
## 5       8   134   134   134   134   134   134   134    NA    NA
## 6       9    77    77    77    77    77    77    77    77    NA
## 7      10    72    72    72    72    72    72    72    72    72
```

```
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)
##          2.53445          -0.03149          -0.04225           0.44244           0.00920
##      stimlen:pos
##         -0.09597
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 3213  AIC: 3517
## log likelihood:  -1606.333
## Nagelkerke R2:  0.04027881
## % pres/err predicted correctly:  -1011.797
## % of predictable range [ (model-null)/(1-null) ]:  0.02705706
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.00821     -0.22025      0.03112     -0.31734
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3217  AIC: 3518
## log likelihood:  -1608.308
## Nagelkerke R2:  0.03855352
## % pres/err predicted correctly:  -1013.578
## % of predictable range [ (model-null)/(1-null) ]:  0.0253464
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.41526     -0.20225     -0.04487
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3228  AIC: 3529
## log likelihood:  -1614.069
## Nagelkerke R2:  0.03351106
## % pres/err predicted correctly:  -1017.772
## % of predictable range [ (model-null)/(1-null) ]:  0.02131718
## **************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.71112     -0.23734     -0.13544      0.01045
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3228  AIC: 3531
## log likelihood:  -1613.805
## Nagelkerke R2:  0.03374275
## % pres/err predicted correctly:  -1017.748
## % of predictable range [ (model-null)/(1-null) ]:  0.02134047
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.4166      -0.2252
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3233  AIC: 3532
## log likelihood:  -1616.281
## Nagelkerke R2:  0.03157138
## % pres/err predicted correctly:  -1019.215
## % of predictable range [ (model-null)/(1-null) ]:  0.01993182
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.36877      0.01906     -0.27616
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3271  AIC: 3577
## log likelihood:  -1635.712
## Nagelkerke R2:  0.01443615
## % pres/err predicted correctly:  -1030.084
## % of predictable range [ (model-null)/(1-null) ]:  0.009490528
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.0806      -0.1065
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3276  AIC: 3581
## log likelihood:  -1637.942
## Nagelkerke R2:  0.01245867
## % pres/err predicted correctly:  -1031.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008125014
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.659
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3948 Residual
## Null Deviance:       3304
## Residual Deviance: 3304  AIC: 3611
## log likelihood:  -1651.936
## Nagelkerke R2:  -3.917275e-16
## % pres/err predicted correctly:  -1039.963
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	3516.797	0.000000	1.000000	0.6345028	0.0402728	2.8534446	- 0.0314865	0.4424367	- 0.0959726	- 0.0422529	0.0092005
preserved ~ stimlen + I(pos^2) + pos	3517.913	1.115359	0.572536	0.3632757	0.0385545	2.5008215	- 0.2202536	- 0.3173431	NA	0.0311249	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	3529.03	12.23549	0.0022084	0.0013981	0.0335131	1.1415263	- 0.2022542	- 0.0448673	NA	NA	NA
preserved ~ stimlen * pos	3530.83	14.04156	0.0008983	0.0005686	0.0337427	1.2711118	- 0.2373376	- 0.1354444	0.0104480	NA	NA
preserved ~ stimlen	3532.42	15.62515	0.0004046	0.0002560	0.0315734	1.1416636	- 0.2251860	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	3577.43	60.63785	7e-14	0	0	0.0014436	NA	- 0.2761642	NA	0.0190583	NA
preserved ~ pos	3580.53	63.73942	2e-14	0	0	0.0012458	NA	- 0.1065215	NA	NA	NA
preserved ~ 1	3611.07	94.28218	5e-21	0	0	0	1.0658970	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          2.53445          -0.03149          -0.04225           0.44244           0.00920
##      stimlen:pos
##         -0.09597
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 3213  AIC: 3517
```

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

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

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

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

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

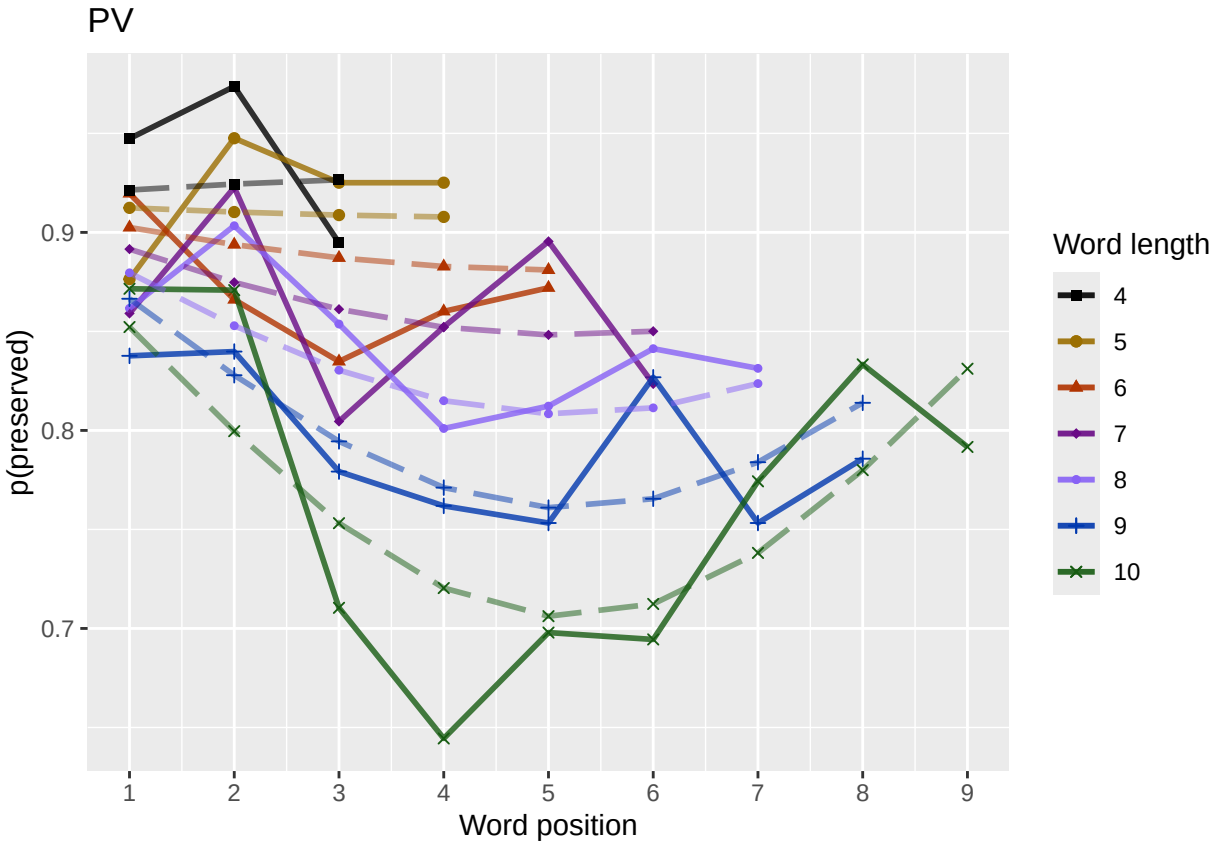
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.921 0.924 0.927 NA     NA     NA     NA     NA     NA
## 2       5 0.912 0.910 0.909  0.908 NA     NA     NA     NA     NA
## 3       6 0.902 0.894 0.887  0.883  0.881 NA     NA     NA     NA
## 4       7 0.892 0.875 0.861  0.852  0.848  0.850 NA     NA     NA
```

```
## 5       8 0.880 0.853 0.830  0.815  0.808  0.811  0.824 NA     NA
## 6       9 0.866 0.828 0.794  0.771  0.761  0.765  0.784  0.814 NA
## 7      10 0.852 0.800 0.753  0.720  0.706  0.712  0.738  0.780  0.831
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       35   651
```

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

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

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

```
## [1] "The number of responses with fragments was 35 / 651 = 5.38 percent"
```

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

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

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

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

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

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.346290          0.005119         -0.030818          0.533866          0.010515
##      stimlen:pos
##        -0.120076
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3862 Residual
## Null Deviance:       3023
## Residual Deviance: 2939  AIC: 3232
## log likelihood:  -1469.619
## Nagelkerke R2:  0.03956076
## % pres/err predicted correctly:  -907.4167
## % of predictable range [ (model-null)/(1-null) ]:  0.02651062
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.22071     -0.24011      0.04834     -0.38572
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3864 Residual
## Null Deviance:       3023
## Residual Deviance: 2946  AIC: 3236
## log likelihood:  -1473.09
## Nagelkerke R2:  0.03631882
## % pres/err predicted correctly:  -910.0448
## % of predictable range [ (model-null)/(1-null) ]:  0.02369424
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.3855      -0.2054
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3866 Residual
## Null Deviance:       3023
## Residual Deviance: 2969  AIC: 3257
## log likelihood:  -1484.735
## Nagelkerke R2:  0.02540186
## % pres/err predicted correctly:  -917.7468
## % of predictable range [ (model-null)/(1-null) ]:  0.01544055
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos
##     3.38255     -0.21634      0.02307
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3865 Residual
## Null Deviance:       3023
## Residual Deviance: 2968  AIC: 3258
## log likelihood:  -1484.219
## Nagelkerke R2:  0.02588723
## % pres/err predicted correctly:  -917.5444
## % of predictable range [ (model-null)/(1-null) ]:  0.01565748
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.67006     -0.25052     -0.06956      0.01072
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3864 Residual
## Null Deviance:       3023
## Residual Deviance: 2968  AIC: 3260
## log likelihood:  -1483.979
## Nagelkerke R2:  0.02611342
## % pres/err predicted correctly:  -917.4414
## % of predictable range [ (model-null)/(1-null) ]:  0.01576783
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.42410      0.03496     -0.33867
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3865 Residual
## Null Deviance:       3023
## Residual Deviance: 3008  AIC: 3302
## log likelihood:  -1503.936
## Nagelkerke R2:  0.007257722
## % pres/err predicted correctly:  -927.5461
## % of predictable range [ (model-null)/(1-null) ]:  0.004939327
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     1.93795     -0.03951
```

```
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3866 Residual
## Null Deviance:       3023
## Residual Deviance: 3020  AIC: 3313
## log likelihood:  -1509.909
## Nagelkerke R2:  0.001576628
## % pres/err predicted correctly:  -931.0892
## % of predictable range [ (model-null)/(1-null) ]:  0.00114241
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.788
##
## Degrees of Freedom: 3867 Total (i.e. Null);  3867 Residual
## Null Deviance:       3023
## Residual Deviance: 3023  AIC: 3316
## log likelihood:  -1511.563
## Nagelkerke R2:  -4.094387e-16
## % pres/err predicted correctly:  -932.1552
## % 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=FALSE
NoFragLPAICSummary <- subset(NoFragLPAICSummary, select = -c(Row.names))

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	3231.547	0.000000	1.000000	0.9902646	0.1039562	2.8346290	0.0051188	5.5338660	- 0.1200767	- 0.0308179	0.0105151
preserved ~ stimlen + I(pos^2) + pos	3236.001	4.454059	0.107848	0.0973488	0.0363188	2.2207120	- 0.2401069	- 3.3857200	NA	0.0483353	NA
preserved ~ stimlen	3256.792	25.246010	0.000000	0.0000000	0.0025403	0.9385500	- 0.2054276	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	3258.202	26.65619	0.000000	0.000000	0.502588	3.382550	-0.2163446	0.0230734	NA	NA	NA
preserved ~ stimlen * pos	3259.962	28.41395	0.000000	0.000000	0.502611	3.467006	-0.2505179	- 0.0695566	0.0107232	NA	NA
preserved ~ I(pos^2) + pos	3302.047	70.50229	0.000000	0.000000	0.007257	2.424099	NA	-0.3386728	NA	0.0349580	NA
preserved ~ pos	3312.905	81.35867	0.000000	0.000000	0.001576	1.937953	NA	-0.0395076	NA	NA	NA
preserved ~ 1	3315.648	84.10192	0.000000	0.000000	0.000000	1.787807	NA	NA	NA	NA	NA

```
# plot no fragment data

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

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

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

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

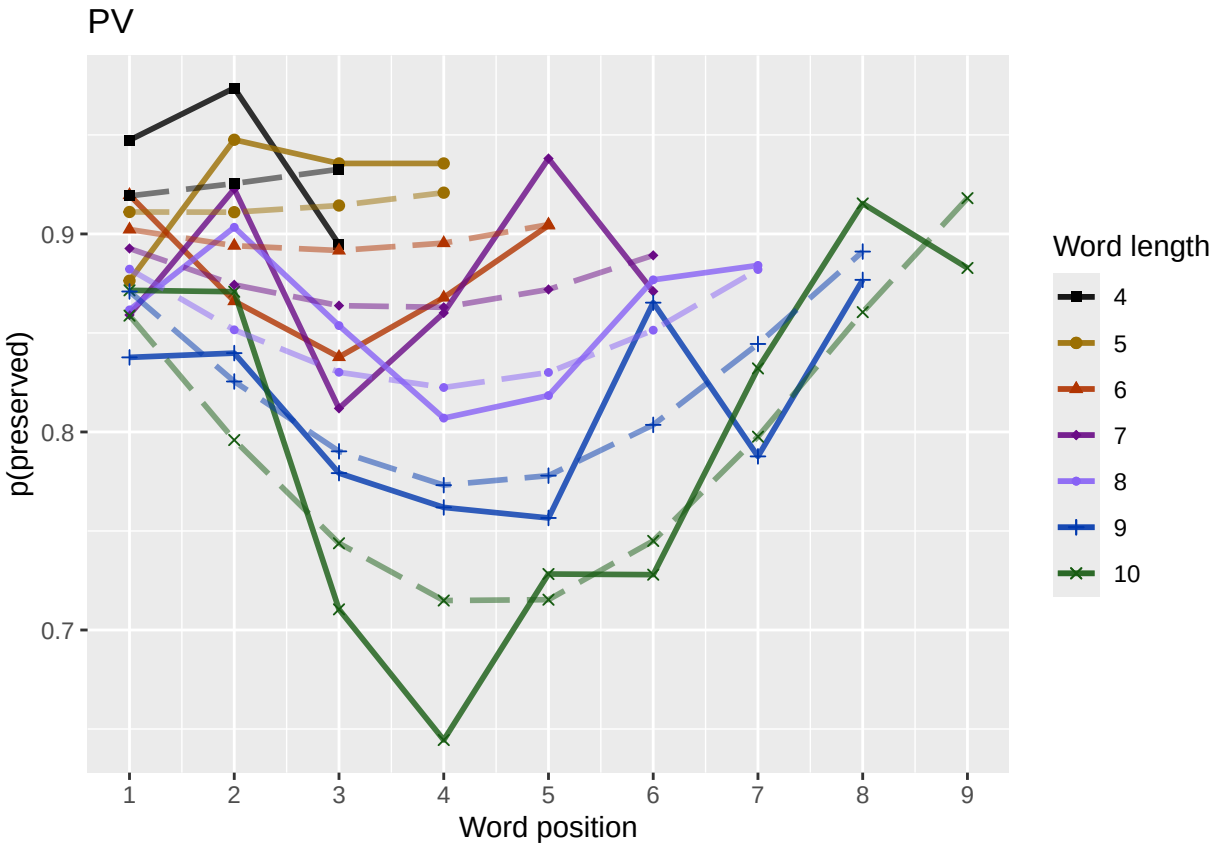
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_fi
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.919 0.925 0.933 NA     NA     NA     NA     NA     NA
## 2       5 0.911 0.911 0.914  0.921 NA     NA     NA     NA     NA
## 3       6 0.902 0.894 0.892  0.895  0.905 NA     NA     NA     NA
## 4       7 0.893 0.874 0.864  0.863  0.872  0.889 NA     NA     NA
## 5       8 0.882 0.852 0.830  0.822  0.830  0.851  0.882 NA     NA
## 6       9 0.871 0.825 0.790  0.773  0.778  0.804  0.844  0.891 NA
## 7      10 0.859 0.796 0.744  0.715  0.715  0.745  0.798  0.861  0.918
```

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

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

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

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



back to full data

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

```
## [1] "Min/max preserved range: 0.61 - 1.01"
```

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

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

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

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

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

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

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

```
print(OA_mean_len_diff)
```

```
## [1] -0.02699349
```

```
results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,OA_mean_len_diff)

CurrentLabel<-"mean change in probability for each additional position (excluding  U-shape): "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional position (excluding  U-shape): "
```

```
print(OA_mean_pos_diff)
```

```
## [1] -0.01663909
```

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

```
if(potential_u_shape){ # downward, so potential U-shape
  # take only positive values from the table
  filtered_pos_upward_u<-table_pos_diffs
  filtered_pos_upward_u[table_pos_diffs <= 0] <- NA
  average_pos_u_diffs <- apply(filtered_pos_upward_u,
                               2,mean,na.rm=TRUE)
  OA_mean_pos_u_diff <- mean(average_pos_u_diffs,na.rm=TRUE)
```

```
  if(is.nan(OA_mean_pos_u_diff) | (OA_mean_pos_u_diff <= 0)){
    print("No U-shape in this participant")
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
    potential_u_shape <- FALSE
  }else{
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 1)

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

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

```
## [1] "Average upward change after U minimum"
## [1] 0.01916151
```

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

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

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

```
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, biggest_return_upward_row)

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

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

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

```
## [1] "differences from left max to min for each row: "
## [1] 0.000000000 0.004583474 0.021442863 0.043314209 0.071197621 0.105475927 0.146025364
## [1] "differences from min to right max for each row: "
## [1] 0.005159456 0.000000000 0.000000000 0.001829299 0.015265381 0.052922150 0.124974322
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.1460254
## [1] 0.1249743
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.8558398
```

```
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",
```

```r
            "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:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##           3.7376           -0.1880            0.4170            0.0299           -0.3066
## stimlen:log_freq
##          -0.0380
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 3189  AIC: 3493
```

```
## log likelihood:  -1594.51
## Nagelkerke R2:  0.05057029
## % pres/err predicted correctly:  -1006.511
## % of predictable range [ (model-null)/(1-null) ]:  0.03213531
## **************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          3.729195          -0.191814           0.341679           0.029213           -0.293312
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.043811          -0.003685           0.048707
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3941 Residual
## Null Deviance:       3304
## Residual Deviance: 3187  AIC: 3493
## log likelihood:  -1593.262
## Nagelkerke R2:  0.05165306
## % pres/err predicted correctly:  -1005.466
## % of predictable range [ (model-null)/(1-null) ]:  0.03313899
## **************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     3.76115     -0.18625      0.03088     -0.31527      0.10974
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3944 Residual
## Null Deviance:       3304
## Residual Deviance: 3195  AIC: 3497
## log likelihood:  -1597.634
## Nagelkerke R2:  0.0478564
## % pres/err predicted correctly:  -1007.239
## % of predictable range [ (model-null)/(1-null) ]:  0.03143571
## **************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          3.731961          -0.187008           0.028569           -0.294541           0.003069
## I(pos^2):log_freq       pos:log_freq
##         -0.005752           0.055819
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3942 Residual
## Null Deviance:       3304
```

```
## Residual Deviance: 3194  AIC: 3498
## log likelihood:  -1596.831
## Nagelkerke R2:  0.04855474
## % pres/err predicted correctly:  -1006.384
## % of predictable range [ (model-null)/(1-null) ]:  0.03225758
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          3.16552          -0.17390           0.43197          -0.03821          -0.04644
##     log_freq:pos
##          0.01293
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 3198  AIC: 3503
## log likelihood:  -1599.122
## Nagelkerke R2:  0.04656275
## % pres/err predicted correctly:  -1009.978
## % of predictable range [ (model-null)/(1-null) ]:  0.02880485
## ****************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          3.16565          -0.17047           0.43064          -0.04505          -0.03964
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3944 Residual
## Null Deviance:       3304
## Residual Deviance: 3200  AIC: 3503
## log likelihood:  -1599.769
## Nagelkerke R2:  0.0460001
## % pres/err predicted correctly:  -1010.443
## % of predictable range [ (model-null)/(1-null) ]:  0.02835782
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          3.16737          -0.19354           0.43058          -0.03964
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3204  AIC: 3506
```

```
## log likelihood:  -1601.989
## Nagelkerke R2:  0.04406677
## % pres/err predicted correctly:  -1012.065
## % of predictable range [ (model-null)/(1-null) ]:  0.02679973
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     3.17209     -0.16817     -0.04515      0.11031
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3207  AIC: 3507
## log likelihood:  -1603.257
## Nagelkerke R2:  0.04296159
## % pres/err predicted correctly:  -1011.399
## % of predictable range [ (model-null)/(1-null) ]:  0.02744019
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##    3.1724693    -0.1684157    -0.0447030     0.1070570     0.0007839
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3944 Residual
## Null Deviance:       3304
## Residual Deviance: 3207  AIC: 3509
## log likelihood:  -1603.255
## Nagelkerke R2:  0.04296402
## % pres/err predicted correctly:  -1011.378
## % of predictable range [ (model-null)/(1-null) ]:  0.02746013
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      3.1738      -0.1913       0.1102
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3211  AIC: 3511
## log likelihood:  -1605.483
## Nagelkerke R2:  0.04102029
## % pres/err predicted correctly:  -1012.993
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.02590833
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          2.53445          -0.03149          -0.04225           0.44244           0.00920
##      stimlen:pos
##         -0.09597
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 3213  AIC: 3517
## log likelihood:  -1606.333
## Nagelkerke R2:  0.04027881
## % pres/err predicted correctly:  -1011.797
## % of predictable range [ (model-null)/(1-null) ]:  0.02705706
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     4.00821     -0.22025      0.03112     -0.31734
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3217  AIC: 3518
## log likelihood:  -1608.308
## Nagelkerke R2:  0.03855352
## % pres/err predicted correctly:  -1013.578
## % of predictable range [ (model-null)/(1-null) ]:  0.0253464
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.41526     -0.20225     -0.04487
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3228  AIC: 3529
## log likelihood:  -1614.069
## Nagelkerke R2:  0.03351106
## % pres/err predicted correctly:  -1017.772
## % of predictable range [ (model-null)/(1-null) ]:  0.02131718
```

```
## **************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.71112     -0.23734     -0.13544      0.01045
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3228  AIC: 3531
## log likelihood:  -1613.805
## Nagelkerke R2:  0.03374275
## % pres/err predicted correctly:  -1017.748
## % of predictable range [ (model-null)/(1-null) ]:  0.02134047
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.4166      -0.2252
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3233  AIC: 3532
## log likelihood:  -1616.281
## Nagelkerke R2:  0.03157138
## % pres/err predicted correctly:  -1019.215
## % of predictable range [ (model-null)/(1-null) ]:  0.01993182
## **************************
## 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
##          2.335344           0.017931          -0.255603           0.060945          -0.006512
##      pos:log_freq
##          0.054297
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 3230  AIC: 3537
## log likelihood:  -1614.799
## Nagelkerke R2:  0.03287121
## % pres/err predicted correctly:  -1016.896
## % of predictable range [ (model-null)/(1-null) ]:  0.02215947
## **************************
```

```
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.05046     -0.09197      0.14305
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3237  AIC: 3540
## log likelihood:  -1618.501
## Nagelkerke R2:  0.02962186
## % pres/err predicted correctly:  -1019.538
## % of predictable range [ (model-null)/(1-null) ]:  0.01962126
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##     2.059913     -0.094845      0.166017     -0.005618
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3237  AIC: 3542
## log likelihood:  -1618.36
## Nagelkerke R2:  0.02974614
## % pres/err predicted correctly:  -1019.601
## % of predictable range [ (model-null)/(1-null) ]:  0.01956104
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.36877      0.01906     -0.27616
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3271  AIC: 3577
## log likelihood:  -1635.712
## Nagelkerke R2:  0.01443615
## % pres/err predicted correctly:  -1030.084
## % of predictable range [ (model-null)/(1-null) ]:  0.009490528
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.0806      -0.1065
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3276  AIC: 3581
## log likelihood:  -1637.942
## Nagelkerke R2:  0.01245867
## % pres/err predicted correctly:  -1031.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008125014
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.659
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3948 Residual
## Null Deviance:       3304
## Residual Deviance: 3304  AIC: 3611
## log likelihood:  -1651.936
## Nagelkerke R2:  -3.917275e-16
## % pres/err predicted correctly:  -1039.963
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC DeltaAIC AICexp AICw NagR2 (Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos I(pos^2) I(pos^2):log_freq stimlen:pos stimlen:I(pos^2)						
preserved ~ stimlen * log_freq + I(pos^2) + pos	3492.0.89001.00000.47745590 3.7337614	-0.1880480	0.4170018	-0.0380023	-0.3065846	NA	NA	0.0299020	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	3493.0.9404731586740997951635219195	-0.1918141	0.3416788	-0.04381[illegible]	-0.2933124	NA	0.0487[illegible]	0.0632927	NA	-0.0036854	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	3496.3.0711864563072462987356 1147	-0.1862458	0.1097144	NA	-0.3152670	NA	NA	0.0308759	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	3498.2.9502498638560488483574 1961	-0.1870082	0.0030688	NA	-0.2945413	0.0558191	NA	0.0285691	-0.0057521	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	3502.9.8920870676073229863628 5519	-0.1738995	0.4319704	-0.0464430	-0.0382071	NA	0.0129295	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	3502.9.4858984654013 23630165649	-0.1704689	0.4306381	-0.0396370	-0.0450456	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	3506.13.6471032187956724366 8375	-0.1935400	0.4305816	-0.0396430	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	3507.14.7527490070050330442961 72093	-0.1681742	0.1103092	NA	-0.0451514	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	3509.16.8469270426530026729647 2469	-0.1684157	0.1070570	NA	-0.0447030	0.0007839	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	3510.17.4935874100263006631020 73778	-0.1912944	0.1101628	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	3516.24.700827700061000290 2.7881446	-0.0314865	NA	NA	0.4424367	NA	NA	-0.0422529	NA	NA	-0.0959726	0.0092005
preserved ~ stimlen + I(pos^2) + pos	3517.25.1328630700350015784550 8215	-0.2202536	NA	NA	-0.3173431	NA	NA	0.0311249	NA	NA	NA	NA
preserved ~ stimlen + pos	3529.36.3240708000000003351 115263	-0.2022542	NA	NA	-0.0448673	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAICAICexpAICwNagR2(Intercept)stimlenlog_freqstimlenpos:log_freqlog_freqI(pos^2)I(pos^2)log_freq:I(pos^2)stimlen:I(pos^2)										
preserved ~ stimlen * pos	3530.383949840000000000337427111	-0.2373376	NA	NA	-0.1354444	NA	NA	NA	NA	NA	0.0104480 NA
preserved ~ stimlen	3532.3922330429000000000315741	-0.2251860	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	3537.4402129000000000003228732534NA		0.0609451 NA		-0.2556029	0.0542970 NA		0.0179307	-0.0065121	NA	NA	NA
preserved ~ pos + log_freq	3539.4681892370000000000299215046NA		0.1430533 NA		-0.0919722	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	3541.47894408110000000000297465991NA		0.1660166 NA		-0.0948940	-0.0056176	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	3577.8435461350000000000104236876NA	NA	NA		-0.2761642	NA	NA	0.0190583 NA		NA	NA	NA
preserved ~ pos	3580.8537477030000000000124587063NA	NA	NA		-0.1065215	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	3611.0178.2904120000000000001065897NA	NA	NA		NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##           3.7376           -0.1880            0.4170            0.0299           -0.3066
## stimlen:log_freq
##          -0.0380
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 3189  AIC: 3493
```

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

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

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

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

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

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

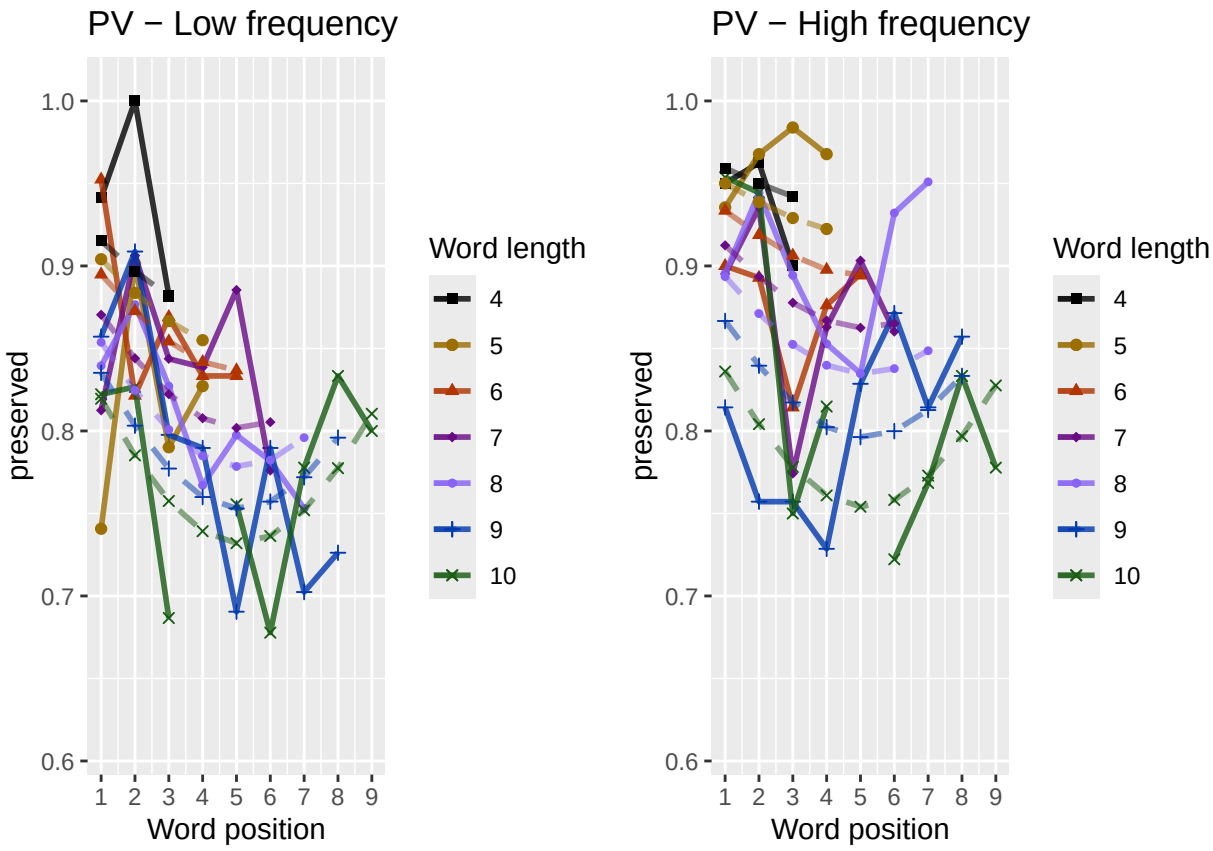
```
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

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

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

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

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




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

```
  "preserved ~ 1"
)
MERes<-TestModels(MEModelEquations,PosDat)
```

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.044       -0.621
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3077  AIC: 3338
## log likelihood:  -1538.371
## Nagelkerke R2:  0.09860547
## % pres/err predicted correctly:  -956.9053
## % of predictable range [ (model-null)/(1-null) ]:  0.0797892
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.4166      -0.2252
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3233  AIC: 3532
## log likelihood:  -1616.281
## Nagelkerke R2:  0.03157138
## % pres/err predicted correctly:  -1019.215
## % of predictable range [ (model-null)/(1-null) ]:  0.01993182
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
```

```
##     2.36877      0.01906     -0.27616
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3271  AIC: 3577
## log likelihood:  -1635.712
## Nagelkerke R2:  0.01443615
## % pres/err predicted correctly:  -1030.084
## % of predictable range [ (model-null)/(1-null) ]:  0.009490528
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.0806      -0.1065
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3276  AIC: 3581
## log likelihood:  -1637.942
## Nagelkerke R2:  0.01245867
## % pres/err predicted correctly:  -1031.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008125014
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.54687      0.04928
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3300  AIC: 3607
## log likelihood:  -1649.78
## Nagelkerke R2:  0.001924835
## % pres/err predicted correctly:  -1038.643
## % of predictable range [ (model-null)/(1-null) ]:  0.001268182
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.659
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3948 Residual
```

```
## Null Deviance:       3304
## Residual Deviance: 3304  AIC: 3611
## log likelihood:  -1651.936
## Nagelkerke R2:  -3.917275e-16
## % pres/err predicted correctly:  -1039.963
## % of predictable range [ (model-null)/(1-null) ]:  0
## ****************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	3337.989	0.0000	1	1	0.098605	2.044396	NA	-0.621003	NA	NA	NA
preserved ~ stimlen	3532.422	194.4334	0	0	0.031571	3.416636	NA	NA	NA	NA	-0.225186
preserved ~ (I(pos^2) + pos)	3577.435	239.4462	0	0	0.014436	2.368769	NA	NA	0.0190583	-0.2761642	NA
preserved ~ pos	3580.537	242.5477	0	0	0.012458	2.080638	NA	NA	NA	-0.1065215	NA
preserved ~ CumPres	3607.329	269.3401	0	0	0.001924	3.546875	0.04928	NA	NA	NA	NA
preserved ~ 1	3611.079	273.0904	0	0	0.000000	1.658970	NA	NA	NA	NA	NA

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

    RndModelAIC<-numeric(length=RandomSamples)
    for(rindex in seq(1,RandomSamples)){
    # Shuffle cumulative values
      PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
      PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
      BestModelRnd <- glm(as.formula(BestMEModelFormulaRnd),
                          family="binomial",data=PosDat)
```

```
      RndModelAIC[rindex] <- BestModelRnd$aic
    }
    ModelNames<-c(paste0("***",BestMEModelFormula),
                  rep(BestMEModelFormulaRnd,RandomSamples))
    AICValues <- c(BestMEModel$aic,RndModelAIC)
    BestMEModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
    BestMEModelRndDF <- BestMEModelRndDF %>% arrange(AIC)
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random average"),
                                         AIC=c(mean(RndModelAIC))))
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random SD"),
                                         AIC=c(sd(RndModelAIC))))

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

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

syll_component	MeanPres	N
1	0.8599316	525
O	0.8188153	1827
P	0.8529412	34
S	0.8270015	229
V	0.8633664	1334

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

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

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

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

```
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.0386      -0.6461
##
## Degrees of Freedom: 3685 Total (i.e. Null);  3684 Residual
## Null Deviance:       3073
## Residual Deviance: 2862  AIC: 3119
## log likelihood:  -1430.787
## Nagelkerke R2:  0.09847114
## % pres/err predicted correctly:  -890.8903
## % of predictable range [ (model-null)/(1-null) ]:  0.07890667
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.4599      -0.2298
##
## Degrees of Freedom: 3685 Total (i.e. Null);  3684 Residual
## Null Deviance:       3073
## Residual Deviance: 3004  AIC: 3298
## log likelihood:  -1501.979
## Nagelkerke R2:  0.03270848
## % pres/err predicted correctly:  -947.154
## % of predictable range [ (model-null)/(1-null) ]:  0.02080074
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.35379      0.01816     -0.26590
##
## Degrees of Freedom: 3685 Total (i.e. Null);  3683 Residual
## Null Deviance:       3073
## Residual Deviance: 3044  AIC: 3344
## log likelihood:  -1522.03
## Nagelkerke R2:  0.01372366
## % pres/err predicted correctly:  -958.5911
## % of predictable range [ (model-null)/(1-null) ]:  0.008989233
## **************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.0787      -0.1041
##
## Degrees of Freedom: 3685 Total (i.e. Null);  3684 Residual
## Null Deviance:       3073
## Residual Deviance: 3048  AIC: 3347
## log likelihood:  -1523.928
## Nagelkerke R2:  0.01191539
## % pres/err predicted correctly:  -959.7393
## % of predictable range [ (model-null)/(1-null) ]:  0.007803363
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.54920      0.05414
##
## Degrees of Freedom: 3685 Total (i.e. Null);  3684 Residual
## Null Deviance:       3073
## Residual Deviance: 3068  AIC: 3370
## log likelihood:  -1534.17
## Nagelkerke R2:  0.002127901
## % pres/err predicted correctly:  -965.967
## % of predictable range [ (model-null)/(1-null) ]:  0.001371758
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.664
##
## Degrees of Freedom: 3685 Total (i.e. Null);  3685 Residual
## Null Deviance:       3073
## Residual Deviance: 3073  AIC: 3374
## log likelihood:  -1536.39
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -967.2953
## % 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	3119.060	0.0000	1	1	0.0984712	1.038622	NA	-0.6460605	NA	NA	NA
preserved ~ stimlen	3298.142	179.0823	0	0	0.0327085	3.459932	NA	NA	NA	NA	-0.2297784
preserved ~ (I(pos^2) + pos)	3344.130	225.0708	0	0	0.0137232	2.353790	NA	NA	0.0181581	-0.2658965	NA
preserved ~ pos	3346.744	227.6847	0	0	0.0119152	4.078693	NA	NA	NA	-0.1041498	NA
preserved ~ CumPres	3370.278	251.2184	0	0	0.0021279	1.549203	0.0541364	NA	NA	NA	NA
preserved ~ 1	3373.867	254.8070	0	0	0.0000000	0.664155	NA	NA	NA	NA	NA

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.9862      -0.6962
##
## Degrees of Freedom: 3160 Total (i.e. Null);  3159 Residual
## Null Deviance:       2662
## Residual Deviance: 2494  AIC: 2718
## log likelihood:  -1246.868
## Nagelkerke R2:  0.09101634
## % pres/err predicted correctly:  -779.7134
## % of predictable range [ (model-null)/(1-null) ]:  0.07365211
## ***************************
```

```
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.4115      -0.2272
##
## Degrees of Freedom: 3160 Total (i.e. Null);  3159 Residual
## Null Deviance:       2662
## Residual Deviance: 2602  AIC: 2861
## log likelihood:  -1301.202
## Nagelkerke R2:  0.03275317
## % pres/err predicted correctly:  -823.9144
## % of predictable range [ (model-null)/(1-null) ]:  0.0212058
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.35667      0.01854     -0.27581
##
## Degrees of Freedom: 3160 Total (i.e. Null);  3158 Residual
## Null Deviance:       2662
## Residual Deviance: 2632  AIC: 2895
## log likelihood:  -1315.907
## Nagelkerke R2:  0.01663729
## % pres/err predicted correctly:  -832.4137
## % of predictable range [ (model-null)/(1-null) ]:  0.01112104
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.0833      -0.1112
##
## Degrees of Freedom: 3160 Total (i.e. Null);  3159 Residual
## Null Deviance:       2662
## Residual Deviance: 2636  AIC: 2897
## log likelihood:  -1317.776
## Nagelkerke R2:  0.01457787
## % pres/err predicted correctly:  -833.5589
## % of predictable range [ (model-null)/(1-null) ]:  0.009762223
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.641
##
## Degrees of Freedom: 3160 Total (i.e. Null);  3160 Residual
## Null Deviance:       2662
## Residual Deviance: 2662  AIC: 2927
## log likelihood:  -1330.945
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -841.7864
## % of predictable range [ (model-null)/(1-null) ]:  0
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##    1.643208    -0.001437
##
## Degrees of Freedom: 3160 Total (i.e. Null);  3159 Residual
## Null Deviance:       2662
## Residual Deviance: 2662  AIC: 2929
## log likelihood:  -1330.944
## Nagelkerke R2:  1.146731e-06
## % pres/err predicted correctly:  -841.7874
## % of predictable range [ (model-null)/(1-null) ]:  -1.267892e-06
## ****************************
```

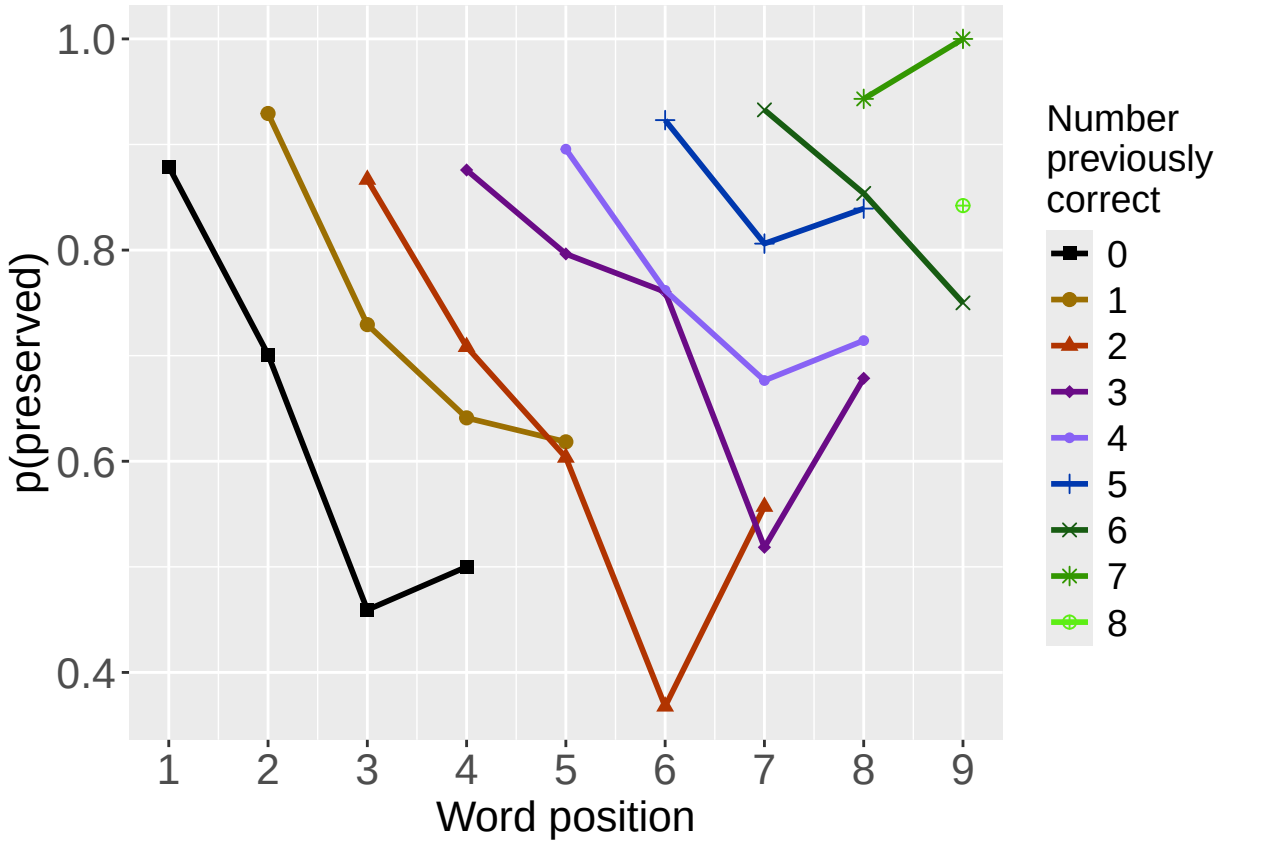
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2718.336	0.0000	1	1	0.0910163	1.986200	NA	-0.6961758	NA	NA	NA
preserved ~ stimlen	2860.606	142.2698	0	0	0.0327532	3.411452	NA	NA	NA	NA	-0.2272118
preserved ~ (I(pos^2) + pos)	2894.794	176.4581	0	0	0.0166373	3.356671	NA	NA	0.018545	-0.2758145	NA
preserved ~ pos	2897.314	178.9786	0	0	0.0145772	2.083340	NA	NA	NA	-0.1112374	NA
preserved ~ 1	2926.886	208.5500	0	0	0.0000000	1.640601	NA	NA	NA	NA	NA
preserved ~ CumPres	2928.891	210.5552	0	0	0.0000011	1.643208	-0.0014369	NA	NA	NA	NA

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

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

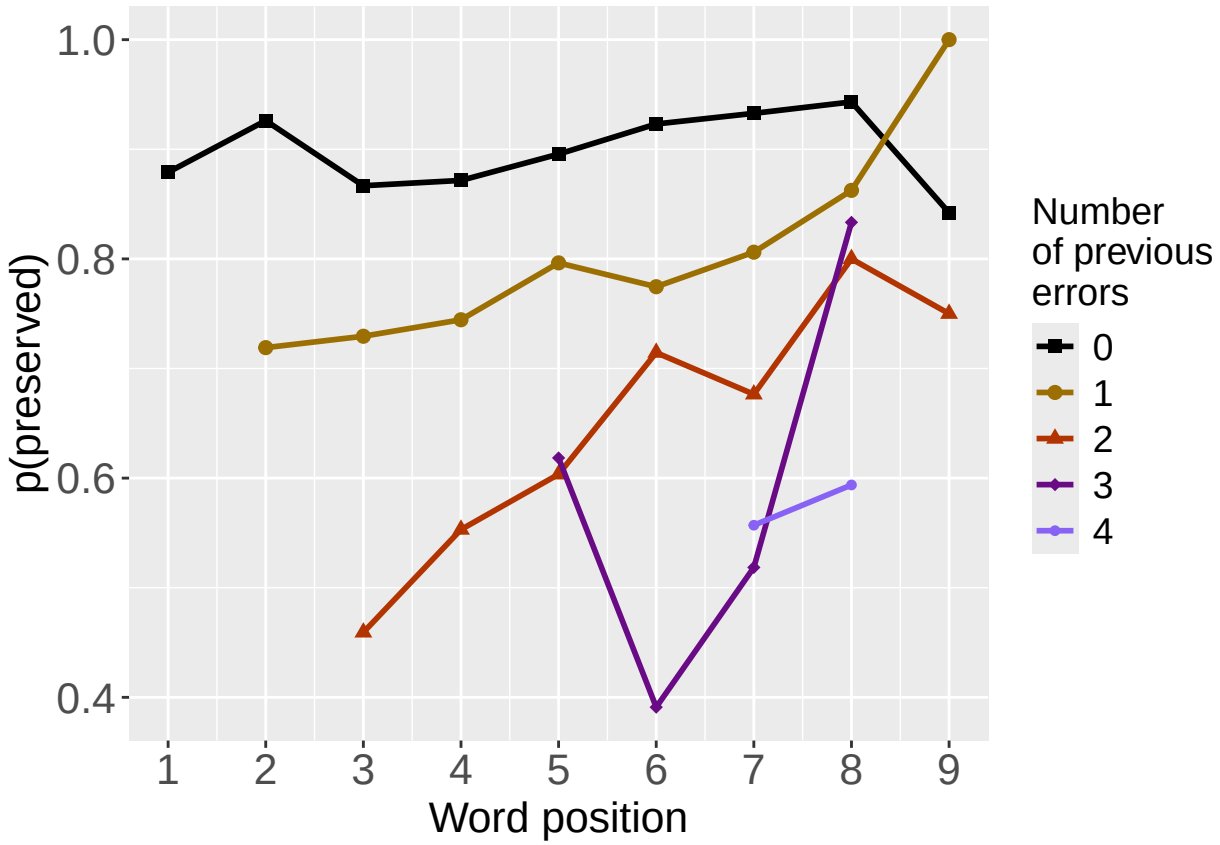
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

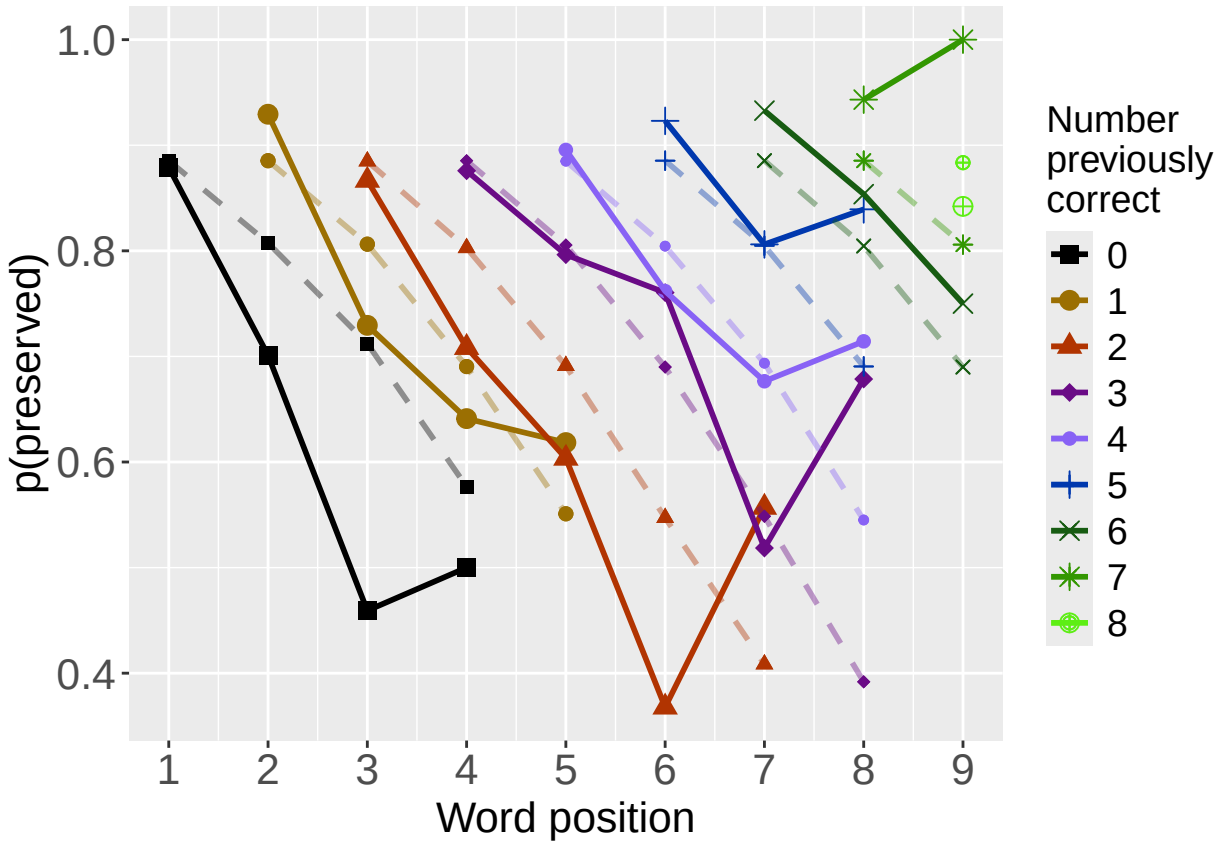
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

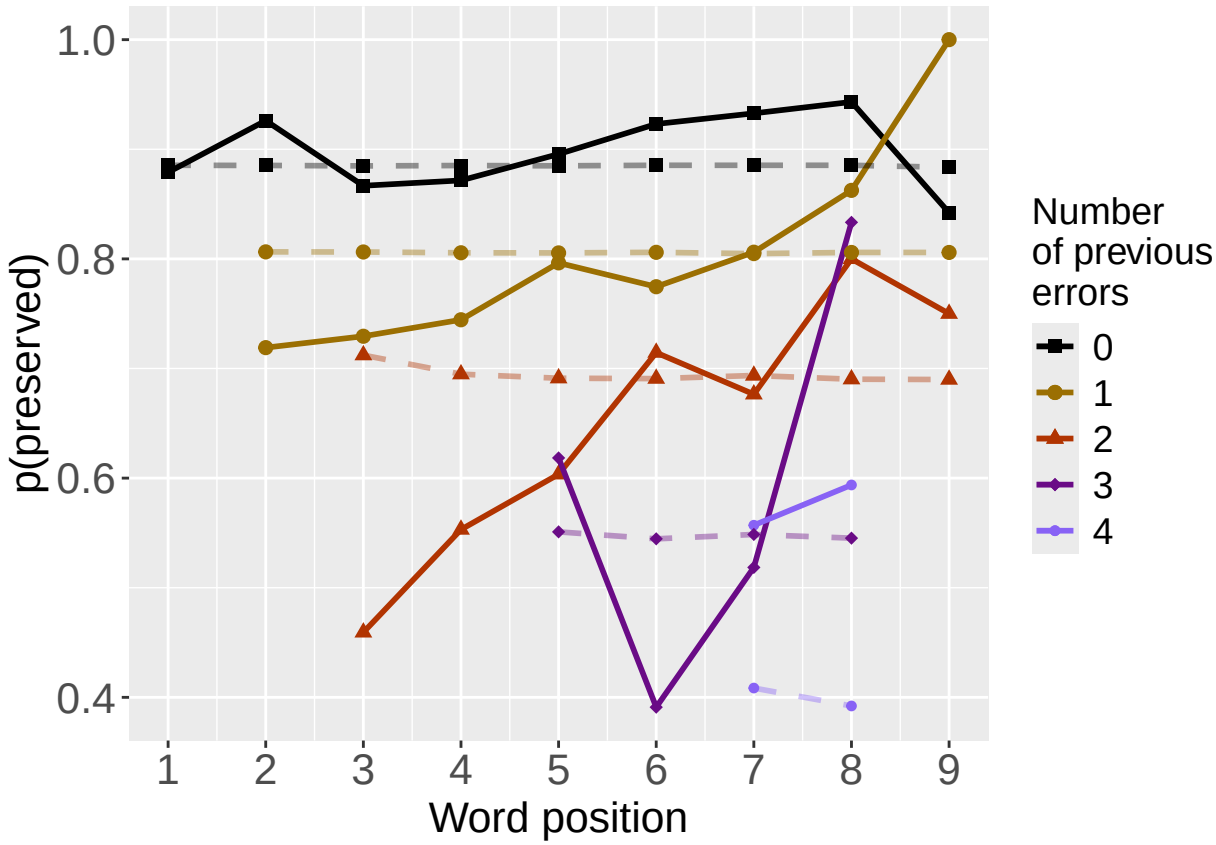
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.34594     -0.74642      0.04032     -0.25980
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3049  AIC: 3308
## log likelihood:  -1524.435
## Nagelkerke R2:  0.1103197
## % pres/err predicted correctly:  -945.7202
## % of predictable range [ (model-null)/(1-null) ]:  0.09053415
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.044       -0.621
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3077  AIC: 3338
## log likelihood:  -1538.371
## Nagelkerke R2:  0.09860547
## % pres/err predicted correctly:  -956.9053
## % of predictable range [ (model-null)/(1-null) ]:  0.0797892
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.36877      0.01906     -0.27616
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3271  AIC: 3577
## log likelihood:  -1635.712
## Nagelkerke R2:  0.01443615
## % pres/err predicted correctly:  -1030.084
## % of predictable range [ (model-null)/(1-null) ]:  0.009490528
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	3307.836	0.00000	1e+00	0.9999997	0.1103197	2.345937	-0.7464206	0.0403161	-0.2598038

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	3337.989	30.15261	3e-07	0.0000003	0.0986055	2.044396	-0.6210030	NA	NA
preserved ~ I(posˆ2) + pos	3577.435	269.59876	0e+00	0.0000000	0.0144362	2.368769	NA	0.0190583	-0.2761642

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##      2.9386      -0.5639      -0.1194
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3060  AIC: 3322
## log likelihood:  -1529.836
## Nagelkerke R2:  0.1057895
## % pres/err predicted correctly:  -953.6206
## % of predictable range [ (model-null)/(1-null) ]:  0.0829446
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.044       -0.621
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3077  AIC: 3338
## log likelihood:  -1538.371
## Nagelkerke R2:  0.09860547
## % pres/err predicted correctly:  -956.9053
## % of predictable range [ (model-null)/(1-null) ]:  0.0797892
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.4166      -0.2252
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3233  AIC: 3532
## log likelihood:  -1616.281
## Nagelkerke R2:  0.03157138
## % pres/err predicted correctly:  -1019.215
## % of predictable range [ (model-null)/(1-null) ]:  0.01993182
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	3322.184	0.00000	1.0000000	0.9996303	0.1057895	2.938605	-0.5639005	-0.1193955
preserved ~ CumErr	3337.989	15.80476	0.0003699	0.0003697	0.0986055	2.044396	-0.6210030	NA
preserved ~ stimlen	3532.422	210.23821	0.0000000	0.0000000	0.0315714	3.416636	NA	-0.2251860

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     1.86357     -0.63342      0.08239
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
```

```
## Null Deviance:       3304
## Residual Deviance: 3066  AIC: 3326
## log likelihood:  -1533.121
## Nagelkerke R2:  0.103028
## % pres/err predicted correctly:  -952.833
## % of predictable range [ (model-null)/(1-null) ]:  0.08370124
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.044       -0.621
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3077  AIC: 3338
## log likelihood:  -1538.371
## Nagelkerke R2:  0.09860547
## % pres/err predicted correctly:  -956.9053
## % of predictable range [ (model-null)/(1-null) ]:  0.0797892
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.54687      0.04928
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3300  AIC: 3607
## log likelihood:  -1649.78
## Nagelkerke R2:  0.001924835
## % pres/err predicted correctly:  -1038.643
## % of predictable range [ (model-null)/(1-null) ]:  0.001268182
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	3326.030	0.00000	1.0000000	0.9974762	0.1030280	1.863573	-0.6334216	0.0823855
preserved ~ CumErr	3337.989	11.95893	0.0025302	0.0025238	0.0986055	2.044396	-0.6210030	NA
preserved ~ CumPres	3607.329	281.29905	0.0000000	0.0000000	0.0019248	1.546875	NA	0.0492800

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     1.78119     -0.71581      0.08239
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3066  AIC: 3326
## log likelihood:  -1533.121
## Nagelkerke R2:  0.103028
## % pres/err predicted correctly:  -952.833
## % of predictable range [ (model-null)/(1-null) ]:  0.08370124
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.044       -0.621
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3077  AIC: 3338
## log likelihood:  -1538.371
## Nagelkerke R2:  0.09860547
## % pres/err predicted correctly:  -956.9053
## % of predictable range [ (model-null)/(1-null) ]:  0.0797892
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.0806      -0.1065
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3276  AIC: 3581
## log likelihood:  -1637.942
## Nagelkerke R2:  0.01245867
## % pres/err predicted correctly:  -1031.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008125014
## **************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	3326.030	0.00000	1.0000000	0.9974762	0.1030280	1.781188	-0.7158071	0.0823855
preserved ~ CumErr	3337.989	11.95893	0.0025302	0.0025238	0.0986055	2.044396	-0.6210030	NA
preserved ~ pos	3580.537	254.50665	0.0000000	0.0000000	0.0124587	2.080638	NA	-0.1065215

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + I(pos^2) + pos	3307.830	0.00000	1.0000000	0.9999997	0.1103192	2.345937	-0.7464206	0.0403161	-0.2598038	NA	NA
preserved ~ CumErr + stimlen	3322.184	14.00000	1.0000000	0.9996308	0.1057892	2.938605	-0.5639005	NA	NA	-0.1193955	NA
preserved ~ CumErr + CumPres	3326.030	0.00000	1.0000000	0.9974762	0.1030280	0.863573	-0.6334216	NA	NA	NA	0.0823855
preserved ~ CumErr + pos	3326.030	0.00000	1.0000000	0.9974762	0.1030280	1.781188	-0.7158071	NA	0.0823855	NA	NA
preserved ~ CumErr	3337.989	10.15261	0.0000000	0.0000000	0.0986055	2.044396	-0.6210030	NA	NA	NA	NA
preserved ~ CumErr	3337.989	15.80476	0.0003699	0.0003697	0.0986055	2.044396	-0.6210030	NA	NA	NA	NA
preserved ~ CumErr	3337.989	11.95893	0.0025302	0.0025238	0.0986055	2.044396	-0.6210030	NA	NA	NA	NA
preserved ~ CumErr	3337.989	11.95893	0.0025302	0.0025238	0.0986055	2.044396	-0.6210030	NA	NA	NA	NA
preserved ~ stimlen	3532.422	210.23821	0.0000000	0.0000000	0.0031571	3.416636	NA	NA	NA	-0.2251860	NA
preserved ~ I(pos^2) + pos	3577.432	269.59876	0.0000000	0.0000000	0.0014436	2.368769	NA	0.0190583	-0.2761642	NA	NA
preserved ~ pos	3580.537	254.50665	0.0000000	0.0000000	0.0124587	2.080638	NA	NA	-0.1065215	NA	NA
preserved ~ CumPres	3607.329	281.29905	0.0000000	0.0000000	0.0001924	3.546875	NA	NA	NA	NA	0.0492800

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     3.51806     -0.70886      0.04868     -0.28711     -0.15877      0.09183
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3943 Residual
## Null Deviance:       3304
## Residual Deviance: 2998  AIC: 3256
## log likelihood:  -1498.846
## Nagelkerke R2:  0.1316153
## % pres/err predicted correctly:  -933.3736
## % of predictable range [ (model-null)/(1-null) ]:  0.1023949
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     3.71652     -0.72003      0.04928     -0.28989     -0.18574
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3944 Residual
## Null Deviance:       3304
## Residual Deviance: 3012  AIC: 3269
## log likelihood:  -1505.846
## Nagelkerke R2:  0.1258172
## % pres/err predicted correctly:  -936.088
## % of predictable range [ (model-null)/(1-null) ]:  0.09978727
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     2.34152     -0.72599      0.04124     -0.26090      0.12113
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3944 Residual
## Null Deviance:       3304
## Residual Deviance: 3023  AIC: 3282
## log likelihood:  -1511.522
## Nagelkerke R2:  0.1211007
## % pres/err predicted correctly:  -939.7446
## % of predictable range [ (model-null)/(1-null) ]:  0.09627456
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.34594     -0.74642      0.04032     -0.25980
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3049  AIC: 3308
## log likelihood:  -1524.435
## Nagelkerke R2:  0.1103197
## % pres/err predicted correctly:  -945.7202
## % of predictable range [ (model-null)/(1-null) ]:  0.09053415
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       1.659
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3948 Residual
## Null Deviance:       3304
## Residual Deviance: 3304  AIC: 3611
## log likelihood:  -1651.936
## Nagelkerke R2:  -3.917275e-16
## % pres/err predicted correctly:  -1039.963
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	3255.645	0.000000	1.000000	0.9985161	0.1131613	3.518056	-0.7088639	0.0486789	-0.2871140	0.0918337	-0.1587703
preserved ~ CumErr + I(pos^2) + pos + stimlen	3268.670	13.025840	0.001484	0.0014819	0.1258137	3.716515	-0.7200326	0.0492758	-0.2898872	NA	-0.1857420
preserved ~ CumErr + I(pos^2) + pos + log_freq	3281.912	26.269150	0.000002	0.0000021	0.1211027	3.341520	-0.7259940	0.0412418	-0.2609040	0.1211348	NA
preserved ~ CumErr + I(pos^2) + pos	3307.836	52.191800	0.000000	0.0000001	0.1103197	3.345937	-0.7464206	0.0403161	-0.2598038	NA	NA
preserved ~ 1	3611.073	355.434870	0.000000	0.0000000	0.0000000	1.658970	NA	NA	NA	NA	NA

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

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

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

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

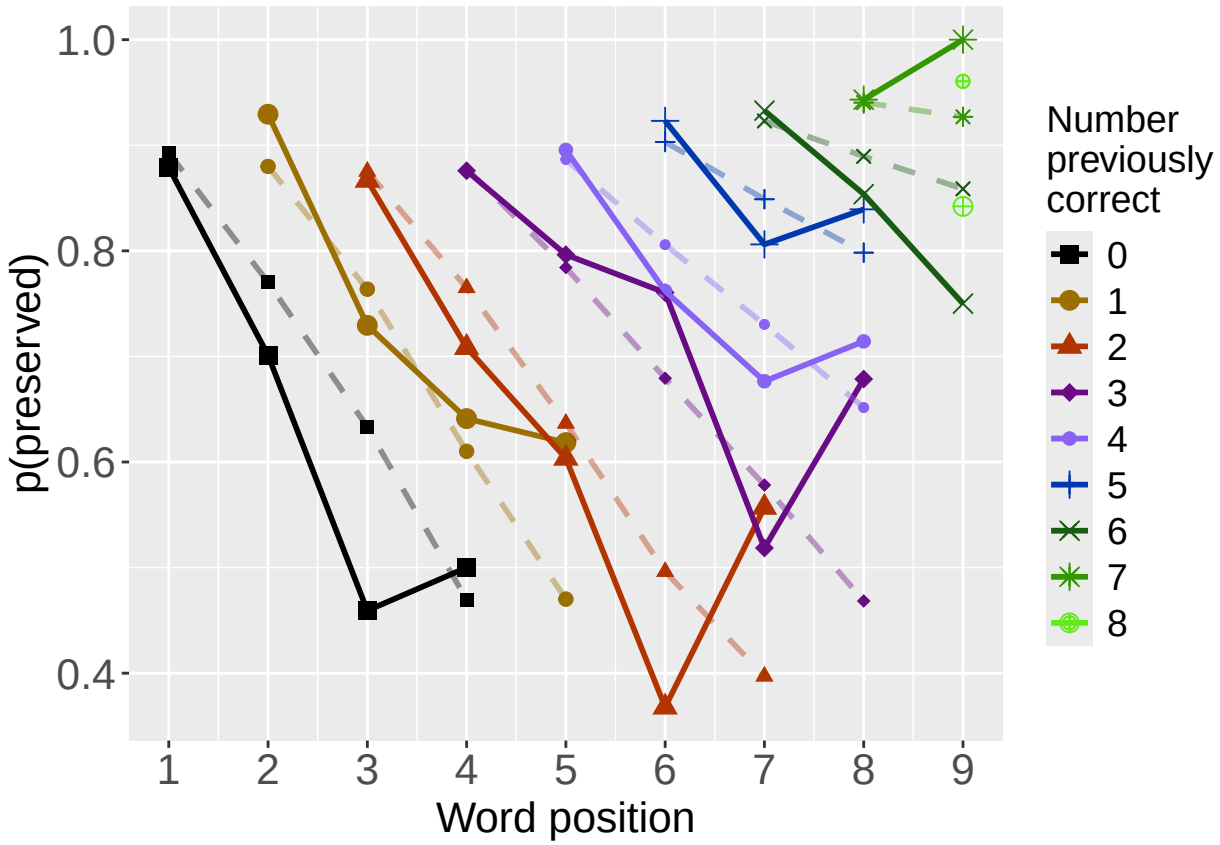
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   3195.3 3451.2
## stimlen   1   3023.0 3279.0
## I(pos^2)  1   3022.4 3278.4
## log_freq  1   3011.7 3267.6
## pos       1   3008.6 3264.5
## <none>        2997.7 3255.6
```

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

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

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

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

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

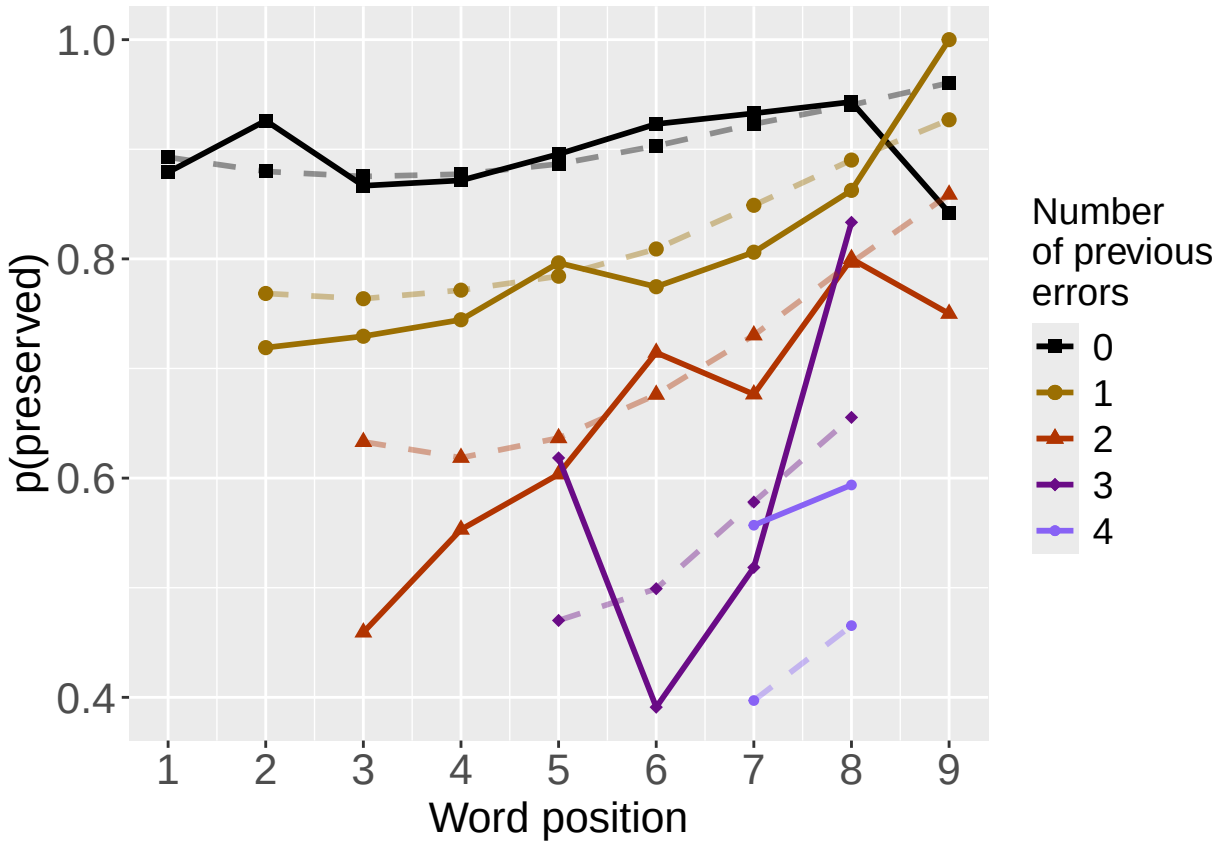
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

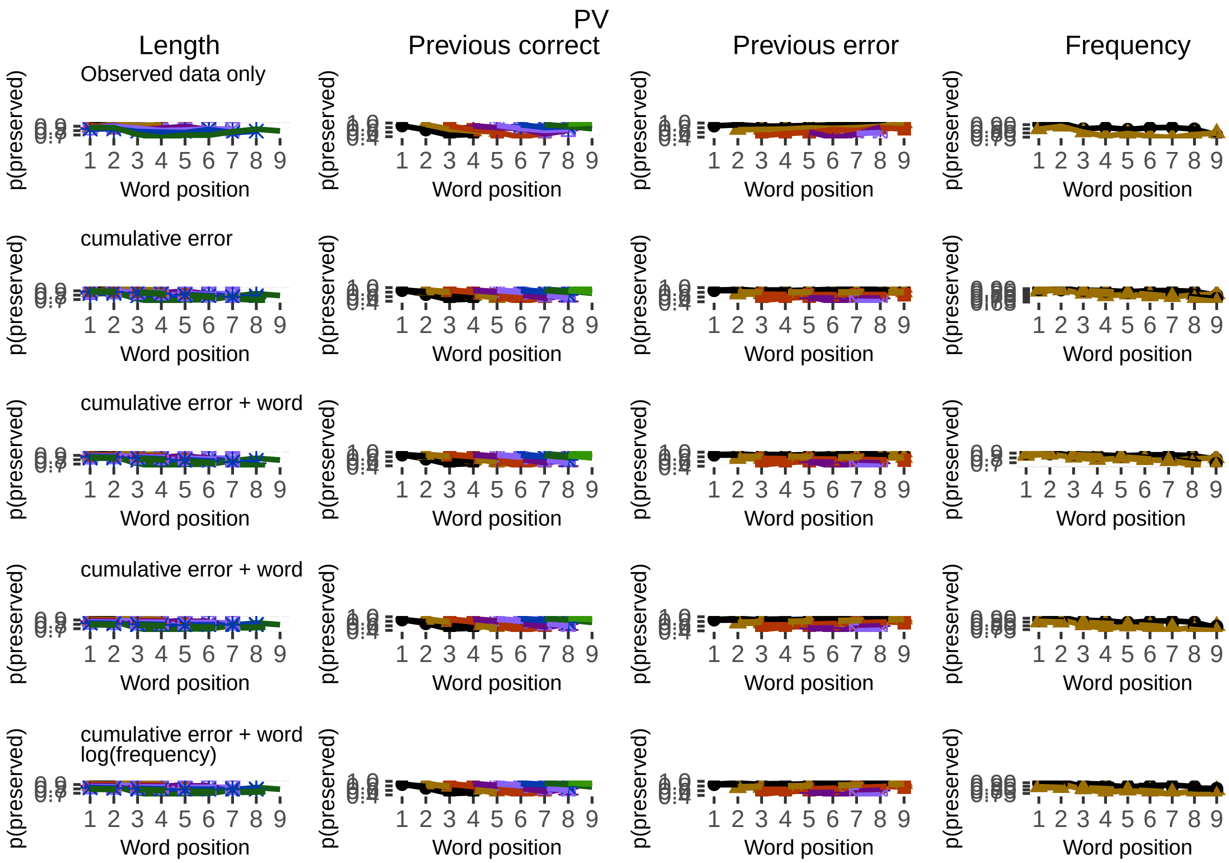
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.044       -0.621
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3947 Residual
## Null Deviance:       3304
## Residual Deviance: 3077  AIC: 3338
## log likelihood:  -1538.371
```

```
## Nagelkerke R2:  0.09860547
## % pres/err predicted correctly:  -956.9053
## % of predictable range [ (model-null)/(1-null) ]:  0.0797892
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##      2.9386      -0.5639      -0.1194
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3946 Residual
## Null Deviance:       3304
## Residual Deviance: 3060  AIC: 3322
## log likelihood:  -1529.836
## Nagelkerke R2:  0.1057895
## % pres/err predicted correctly:  -953.6206
## % of predictable range [ (model-null)/(1-null) ]:  0.0829446
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen     I(pos^2)
##     3.14933     -0.72780     -0.17904      0.01808
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3945 Residual
## Null Deviance:       3304
## Residual Deviance: 3023  AIC: 3279
## log likelihood:  -1511.411
## Nagelkerke R2:  0.1211931
## % pres/err predicted correctly:  -940.7373
## % of predictable range [ (model-null)/(1-null) ]:  0.0953209
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen     I(pos^2)     log_freq
##     2.95424     -0.71639     -0.15182      0.01777      0.09255
##
## Degrees of Freedom: 3948 Total (i.e. Null);  3944 Residual
## Null Deviance:       3304
## Residual Deviance: 3009  AIC: 3266
## log likelihood:  -1504.281
## Nagelkerke R2:  0.1271148
## % pres/err predicted correctly:  -937.9382
## % of predictable range [ (model-null)/(1-null) ]:  0.09800987
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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




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

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

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

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

	CumErr	I(pos^2)	pos	stimlen	log_freq
McFadden	0.0695898	0.0061456	0.0050008	0.0123207	0.0081974
SquaredCorrelation	0.0605841	0.0053002	0.0043933	0.0108710	0.0072373
Nagelkerke	0.0605841	0.0053002	0.0043933	0.0108710	0.0072373
Estrella	0.0639025	0.0056492	0.0045884	0.0112996	0.0075176

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

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

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

```
##                                                                    model deviance
## CumErr + stimlen + I(pos^2) + log_freq CumErr + stimlen + I(pos^2) + log_freq 3008.563
## CumErr + stimlen + I(pos^2)                         CumErr + stimlen + I(pos^2) 3022.821
## CumErr + stimlen                                               CumErr + stimlen 3059.672
## CumErr                                                               CumErr 3076.742
## null                                                                   null 3303.872
##                                       deviance_explained percent_explained
## CumErr + stimlen + I(pos^2) + log_freq           295.3090          8.938270
## CumErr + stimlen + I(pos^2)                      281.0502          8.506694
## CumErr + stimlen                                 244.1994          7.391310
## CumErr                                           227.1296          6.874651
## null                                               0.0000          0.000000
##                                       percent_of_explained_deviance increment_in_explained
## CumErr + stimlen + I(pos^2) + log_freq                     100.00000              4.828407
## CumErr + stimlen + I(pos^2)                                 95.17159             12.478751
## CumErr + stimlen                                            82.69284              5.780302
## CumErr                                                      76.91254             76.912540
## null                                                              NA              0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + stimlen + I(pos^2) + log_freq	3008.563	295.3090
CumErr + stimlen + I(pos^2)	3022.821	281.0502
CumErr + stimlen	3059.672	244.1994
CumErr	3076.742	227.1296
null	3303.872	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + stimlen + I(pos^2) + log_freq	8.938270	100.00000	4.828407
CumErr + stimlen + I(pos^2)	8.506694	95.17159	12.478751
CumErr + stimlen	7.391310	82.69284	5.780302
CumErr	6.874651	76.91254	76.912540
null	0.000000	NA	0.000000

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

```
##          Nagelkerke
## CumErr   0.68544900
## I(pos^2) 0.05996667
## pos      0.04970620
## stimlen  0.12299503
## log_freq 0.08188310
```

```
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.
```

```
## 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
## 1                   preserved ~ CumErr+stimlen       0.7429300       3059.672
## 2                           preserved ~ CumErr       0.7721327       3076.742
## 3          preserved ~ CumErr+stimlen+I(pos^2)       0.8699145       3022.821
## 4 preserved ~ CumErr+stimlen+I(pos^2)+log_freq       0.8725964       3008.563
##   diff_CumErr+stimlen diff_CumErr diff_CumErr+stimlen+I(pos^2)
## 1          0.00000000 -0.02920263                 -0.126984454
## 2          0.02920263  0.00000000                 -0.097781822
## 3          0.12698445  0.09778182                  0.000000000
## 4          0.12966636  0.10046373                  0.002681904
##   diff_CumErr+stimlen+I(pos^2)+log_freq
## 1                          -0.129666358
## 2                          -0.100463726
## 3                          -0.002681904
## 4                           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	p_accounted_for	model_deviance
preserved ~ CumErr+stimlen	0.7429300	3059.672
preserved ~ CumErr	0.7721327	3076.742
preserved ~ CumErr+stimlen+I(pos^2)	0.8699145	3022.821
preserved ~ CumErr+stimlen+I(pos^2)+log_freq	0.8725964	3008.563

model	diff_CumErr+stimlen	diff_CumErr	diff_CumErr+stimlen+I(pos^2)
preserved ~ CumErr+stimlen	0.0000000	-0.0292026	-0.1269845
preserved ~ CumErr	0.0292026	0.0000000	-0.0977818
preserved ~ CumErr+stimlen+I(pos^2)	0.1269845	0.0977818	0.0000000
preserved ~ CumErr+stimlen+I(pos^2)+log_freq	0.1296664	0.1004637	0.0026819