Stability of Modeling Cognitive Control

Experiment 1

Load in packages

Load in data

Clean the data

```
#Make a dataframe of only the necessary variables because these datafiles are yuuuuge
data <- data.raw %>% dplyr::select(Subject, BlockNum, Congruency,
                                   StimSlideSimon.RT, StimSlideFlanker.RT, StimSlideStroop.RT,
                                   StimSlideSimon.ACC, StimSlideFlanker.ACC, StimSlideStroop.ACC)

#subjects to include based on Whitehead et al., (2018)
includesubs <- c(507,508,513,514,517,518,519,520,521,506,515,523,524,525,526,527,528,529,530,532,
                 533,534,535,537,539,540,541,542,544,545,546,547,548,549,551,553,555,559,560,561,
                 562,567,568,569,571,573,576,577,578,579,583,584,585,586,587,588,591,592,593,594,
                 595,596,597,599,601,602,603,605,606,607,608,609,610,611,613,614,615,616,617,619,
                 620,621,622,623,624,625,626,629,630,631,633,634,635,637,639,640,641,642,643,645,
                 647,648,649,650,651,652,654,655,656,657,658,659,660,661,662,663,664,665,666,667,
                 668,669,670,671,672,673,674,675,676,680,678,679,681,683,685,687,689,690,692,693,
                 694,696,697,698,699,700,701,702,704,705,706,708,710,711,712,713,715,716,718,719,
                 720,721,723,724,725,726,728,729,730,732,733,734,736,737,738,739,740,741)

#Filter and clean data
df.simon <- data %>% mutate(prevcon = lag(Congruency)) %>% #creat previous congruency
  mutate(acc = lag(StimSlideSimon.ACC)) %>% #create previous accuracy
  mutate(RT = (StimSlideSimon.RT)) %>% #create general RT variable
  filter(Subject %in% includesubs & StimSlideSimon.RT != "" &
  (StimSlideSimon.RT > 200 & StimSlideSimon.RT < 3000) & #liberal filter
  StimSlideSimon.ACC == 1 & prevcon != 'NA' & acc == 1 & #accuracy
  BlockNum > 2) #experimental only blocks
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
df.flanker <- data %>% mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideFlanker.ACC)) %>%
  mutate(RT = (StimSlideFlanker.RT)) %>%
  filter(Subject %in% includesubs & StimSlideFlanker.RT != "" &
  (StimSlideFlanker.RT > 200 & StimSlideFlanker.RT < 3000) &
  StimSlideFlanker.ACC == 1 & prevcon != 'NA' & acc == 1 &
  BlockNum > 2)

df.stroop <- data %>% mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideStroop.ACC)) %>%
  mutate(RT = (StimSlideStroop.RT)) %>%
  filter(Subject %in% includesubs &  StimSlideStroop.RT != "" &
  (StimSlideStroop.RT > 200 & StimSlideStroop.RT < 3000) &
```

```
  StimSlideStroop.ACC == 1 & prevcon != 'NA' & acc == 1 &
  BlockNum > 2)
```

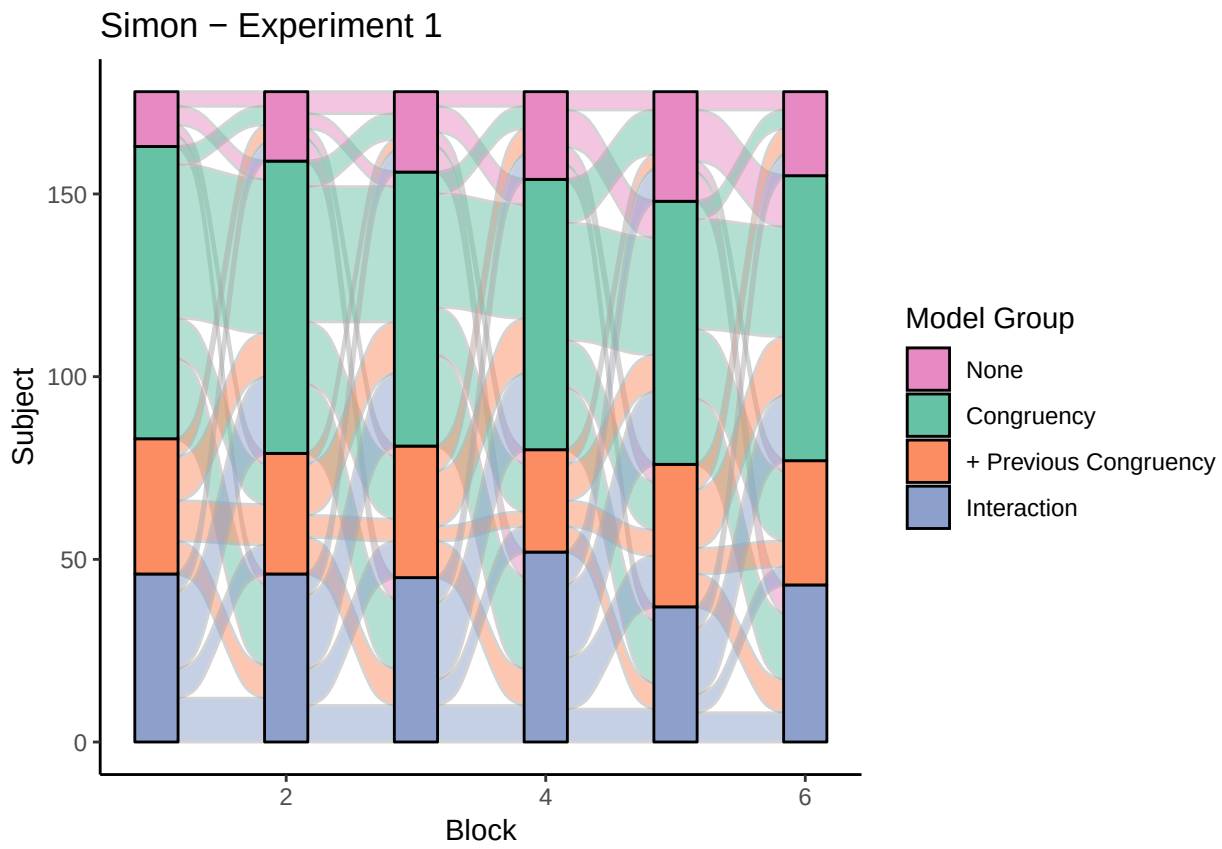
```
computemodels.exp1 <- function(inputdata){
  #create some empty matrixes to put in the adjusted R2 values
  model.0.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.1.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.2.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.3.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))

  count.i <- 0 #counting variable

  #loop to go through and compute each model, per subject per block
  for (i in unique(inputdata$Subject)){
    count.j <- 0
    count.i <- count.i+1
    test <- NULL
    test <- inputdata %>% filter(Subject == i) #only 1 subject
    for (j in unique(test$BlockNum)){
      count.j <- count.j+1
      test.block <- NULL
      test.block <- test %>% filter(test$BlockNum == j)
      model.0 <- lm(RT~1, data = test.block) #null model
      model.1 <- lm(RT~1+Congruency, data = test.block) #congruency model
      model.2 <- lm(RT~1+prevcon+Congruency, data = test.block) #add previous congruency
      model.3 <- lm(RT~1+prevcon*Congruency, data = test.block) #SCE interaction

      #record the adjusted R2 values
      model.0.R[count.j,count.i] <- summary(model.0)$adj.r.squared
      model.1.R[count.j,count.i] <- summary(model.1)$adj.r.squared
      model.2.R[count.j,count.i] <- summary(model.2)$adj.r.squared
      model.3.R[count.j,count.i] <- summary(model.3)$adj.r.squared
    }
  }

  #create dummy data frame to compute the group, in long format
  id.model <- matrix(0, nrow = length(unique(inputdata$Subject)),
                     ncol = length(unique(inputdata$BlockNum)))

  #decide whether the adjusted R2 value for each person, each block
  #is higher or lower than others, in order to determine group membership
  for (i in 1:length(unique(inputdata$Subject))){
    for (j in 1:length(unique(inputdata$BlockNum))){
      if((model.0.R[j,i] > model.1.R[j,i] & model.0.R[j,i] > model.2.R[j,i] &
          model.0.R[j,i] > model.3.R[j,i])){
        id.model[i,j] = 0
      }
      if((model.1.R[j,i] > model.0.R[j,i] & model.1.R[j,i] > model.2.R[j,i] &
          model.1.R[j,i] > model.3.R[j,i])){
        id.model[i,j] = 1
```

```
      }
      if((model.2.R[j,i] > model.1.R[j,i] & model.2.R[j,i] > model.0.R[j,i] &
          model.2.R[j,i] > model.3.R[j,i])){
        id.model[i,j] = 2
      }
      if((model.3.R[j,i] > model.1.R[j,i] & model.3.R[j,i] > model.2.R[j,i] &
          model.3.R[j,i] > model.0.R[j,i])){
        id.model[i,j] = 3
      }
    }
  }

  #put in formate you can plot
  id.model.test <- cbind(as.data.frame(rep(1:178,6)),
                         as.data.frame(rep(1:6,each = 178)),
                         as.data.frame(c(id.model[,1],
                          id.model[,2],
                          id.model[,3],
                          id.model[,4],
                          id.model[,5],
                          id.model[,6]))
                       )
  #rename columns
  colnames(id.model.test) <- c("Subject","Block","Group")

return(id.model.test)
}
```

Simon Task

```
id.model.test <- as.data.frame(computemodels.exp1(df.simon))

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
          fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62","#8da0cb"),
                    labels = c("None","Congruency","+ Previous Congruency",
                               "Interaction")) +
  ylab("Subject") +
  ggtitle("Simon - Experiment 1") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



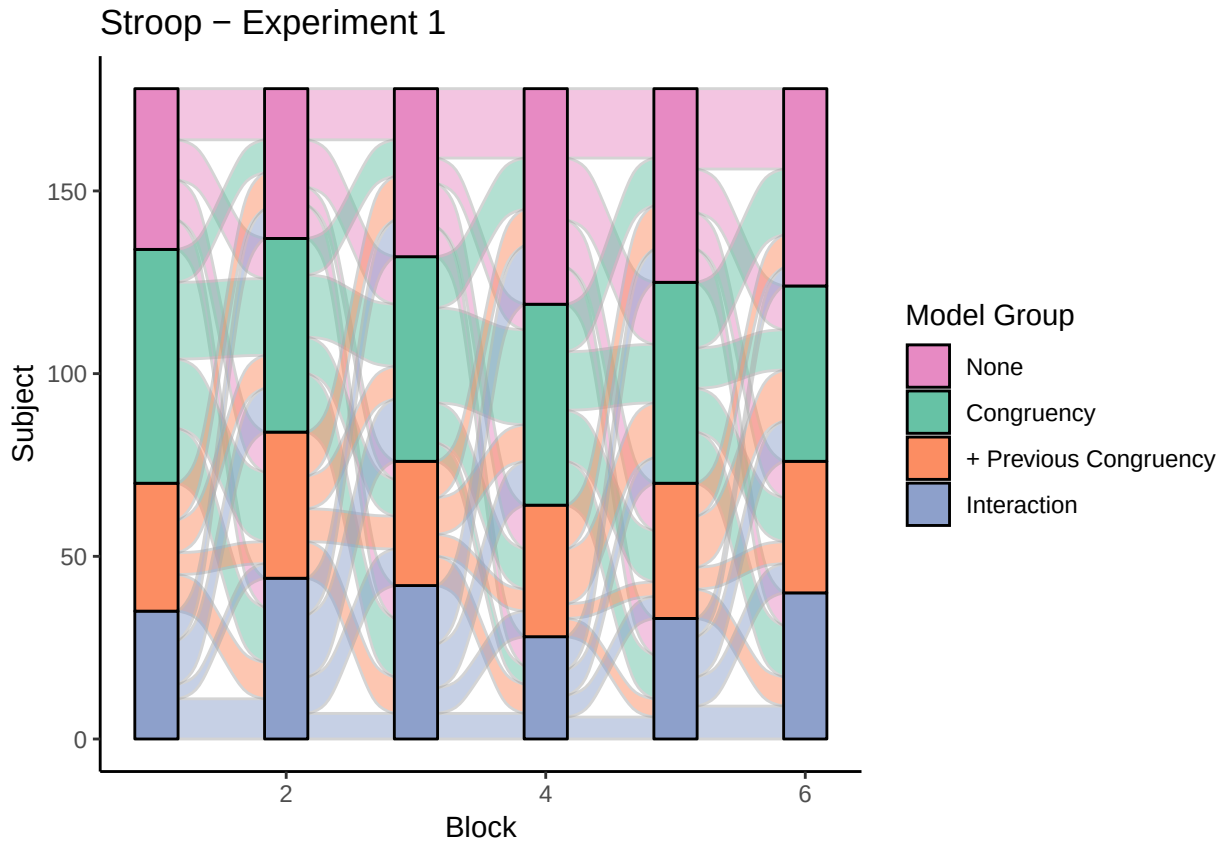

```
simon.1 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Stroop Task

```
id.model.test <- as.data.frame(computemodels.exp1(df.stroop))

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
          fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62","#8da0cb"),
                    labels = c("None","Congruency","+ Previous Congruency",
                               "Interaction")) +
  ylab("Subject") +
  ggtitle("Stroop - Experiment 1") +
  theme_classic() +
```

```
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



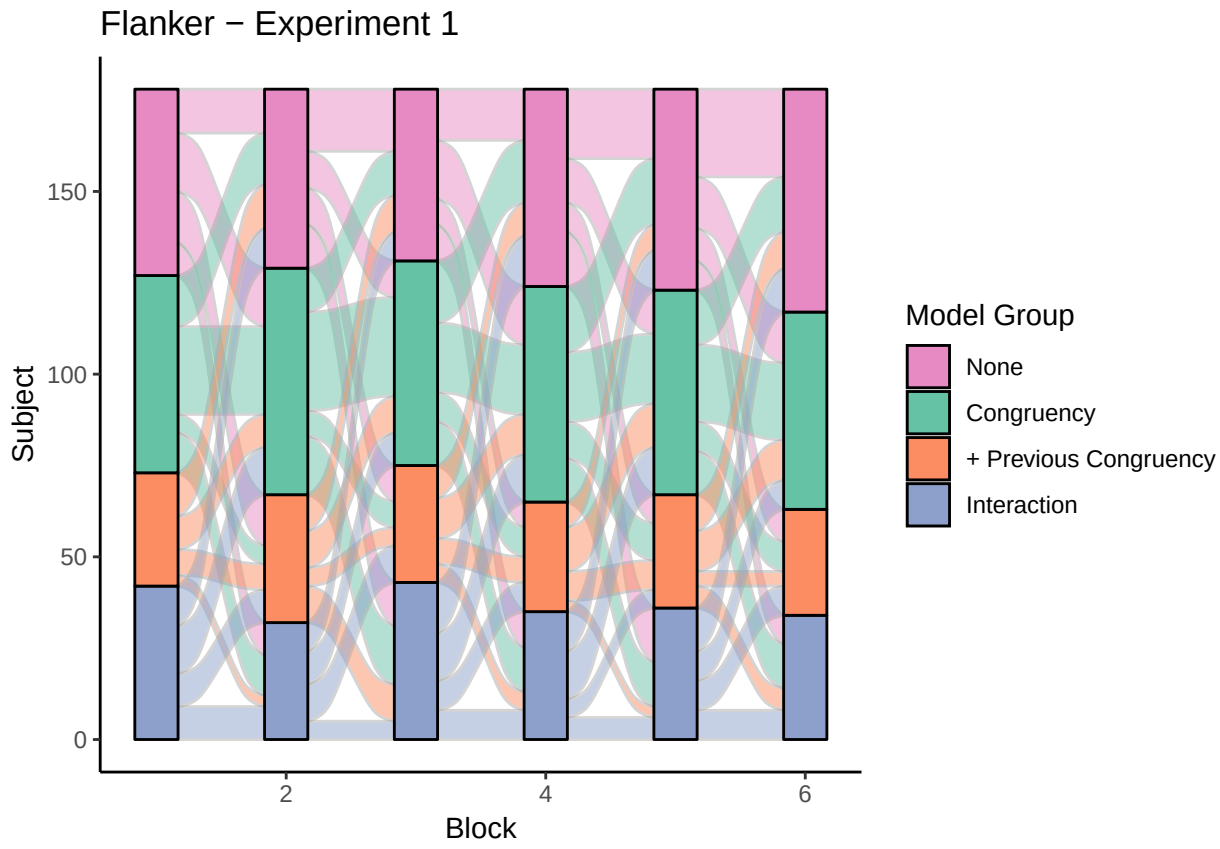

```
stroop.1 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Flanker Task

```
id.model.test <- as.data.frame(computemodels.exp1(df.flanker))

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
           fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62","#8da0cb"),
                    labels = c("None","Congruency","+ Previous Congruency",
```

```
                                  "Interaction")) +
  ylab("Subject") +
  ggtitle("Flanker - Experiment 1") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```




```
flanker.1 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Experiment 2

Load in data

Clean the data

```
#Make a dataframe of only the necessary variables because these datafiles are yuuuuge
data.simon <- data.raw.simon %>%
```

```
  dplyr::select(Subject, BlockNum, Congruency,
      StimSlideSimon.ACC, StimSlideSimon.RT, TargetRepeat)
data.flanker <- data.raw.flanker %>%
  dplyr::select(Subject, BlockNum, Congruency,
      StimSlideFlanker.ACC, StimSlideFlanker.RT, TargetRepeat)
data.stroop <- data.raw.stroop %>%
  dplyr::select(Subject, BlockNum, Congruency,
      StimSlideStroop.ACC, StimSlideStroop.RT, TargetRepeat)

#subjects to exclude
excludesubs <- c(115,116,126,140,148,153,160,175,188,189,194,195,203,210,212,220,
                 229,233,237,239,243,250,253,297,901,913,918,145,217,258,280)

#Filter and clean data
#same method as experiment 1
df.simon <- data.simon %>%
  mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideSimon.ACC)) %>%
  mutate(RT = (StimSlideSimon.RT)) %>%
  filter(!Subject %in% excludesubs &
  (StimSlideSimon.RT > 200 & StimSlideSimon.RT < 3000) &
  StimSlideSimon.ACC == 1 & acc == 1 &
  BlockNum > 2)

df.flanker <- data.flanker %>%
  mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideFlanker.ACC)) %>%
  mutate(RT = (StimSlideFlanker.RT)) %>%
  filter(!Subject %in% excludesubs &
  (StimSlideFlanker.RT > 200 & StimSlideFlanker.RT < 3000) &
  StimSlideFlanker.ACC == 1 & acc == 1 &
  BlockNum > 2)

df.stroop <- data.stroop %>%
  mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideStroop.ACC)) %>%
  mutate(RT = (StimSlideStroop.RT)) %>%
  filter(!Subject %in% excludesubs &
  (StimSlideStroop.RT > 200 & StimSlideStroop.RT < 3000) &
  StimSlideStroop.ACC == 1 & acc == 1 &
  BlockNum > 2)
```

```
computemodels.exp2 <- function(inputdata){
  #create some empty matrixes to put in the adjusted R2 values
  model.0.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.1.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.2.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.3.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.4.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
```

```
model.5.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                    ncol = length(unique(inputdata$Subject)))

count.i <- 0 #counting variable

#loop to go through and compute each model, per subject per block
for (i in unique(inputdata$Subject)){
  count.j <- 0
  count.i <- count.i+1
  test <- NULL
  test <- inputdata %>% filter(Subject == i) #only 1 subject
  for (j in unique(test$BlockNum)){
    count.j <- count.j+1
    test.block <- NULL
    test.block <- test %>% filter(test$BlockNum == j)
    model.0 <- lm(RT~1, data = test.block) #null model
    model.1 <- lm(RT~1+Congruency, data = test.block) #congruency model
    model.2 <- lm(RT~1+prevcon*Congruency, data = test.block) #SCE model
    model.3 <- lm(RT~1+TargetRepeat*Congruency, data = test.block) #Target Rep SCE
    model.4 <- lm(RT~1+(prevcon*Congruency)+(TargetRepeat*Congruency), data = test.block) #models 2 &
    model.5 <- lm(RT~1+(prevcon*Congruency):(TargetRepeat*Congruency), data = test.block) #interactio


    model.0.R[count.j,count.i] <- summary(model.0)$adj.r.squared
    model.1.R[count.j,count.i] <- summary(model.1)$adj.r.squared
    model.2.R[count.j,count.i] <- summary(model.2)$adj.r.squared
    model.3.R[count.j,count.i] <- summary(model.3)$adj.r.squared
    model.4.R[count.j,count.i] <- summary(model.4)$adj.r.squared
    model.5.R[count.j,count.i] <- summary(model.5)$adj.r.squared
  }
}

#create dummy data frame to compute the group, in long format
id.model <- matrix(0, nrow = length(unique(inputdata$Subject)),
                   ncol = length(unique(inputdata$BlockNum)))

#decide whether the adjusted R2 value for each person, each block
#is higher or lower than others, in order to determine group membership
for (i in 1:length(unique(inputdata$Subject))){
for (j in 1:length(unique(inputdata$BlockNum))){
  if((model.0.R[j,i] > model.1.R[j,i] & model.0.R[j,i] > model.2.R[j,i] &
      model.0.R[j,i] > model.3.R[j,i] & model.0.R[j,i] > model.4.R[j,i] &
      model.0.R[j,i] > model.5.R[j,i])){
    id.model[i,j] = 0
  }
  if((model.1.R[j,i] > model.0.R[j,i] & model.1.R[j,i] > model.2.R[j,i] &
      model.1.R[j,i] > model.3.R[j,i] & model.1.R[j,i] > model.4.R[j,i] &
      model.1.R[j,i] > model.5.R[j,i])){
    id.model[i,j] = 1
  }
  if((model.2.R[j,i] > model.1.R[j,i] & model.2.R[j,i] > model.0.R[j,i] &
      model.2.R[j,i] > model.3.R[j,i] & model.2.R[j,i] > model.4.R[j,i] &
      model.2.R[j,i] > model.5.R[j,i])){
```

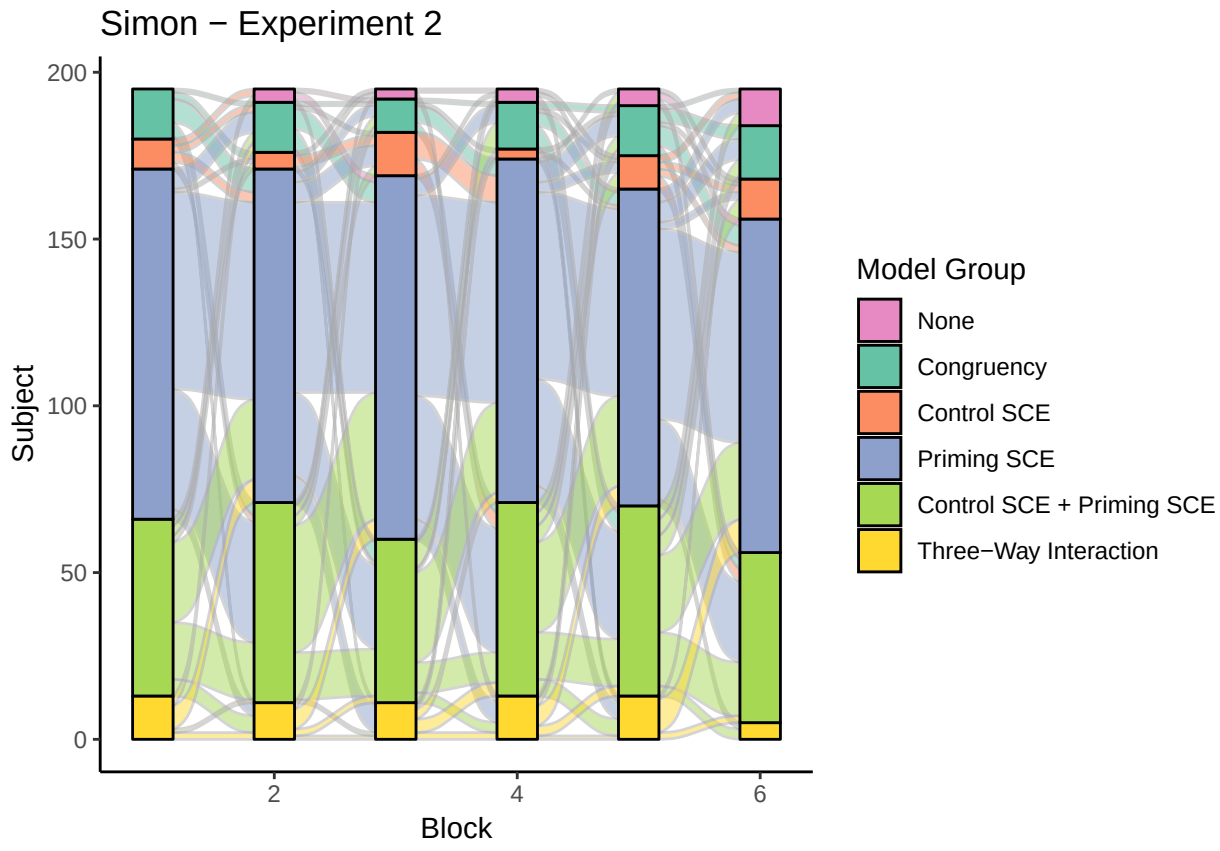
```
      id.model[i,j] = 2
    }
    if((model.3.R[j,i] > model.1.R[j,i] & model.3.R[j,i] > model.2.R[j,i] &
        model.3.R[j,i] > model.0.R[j,i] & model.3.R[j,i] > model.4.R[j,i] &
        model.3.R[j,i] > model.5.R[j,i])){
      id.model[i,j] = 3
    }
    if((model.4.R[j,i] > model.1.R[j,i] & model.4.R[j,i] > model.2.R[j,i] &
        model.4.R[j,i] > model.3.R[j,i] & model.4.R[j,i] > model.0.R[j,i] &
        model.4.R[j,i] > model.5.R[j,i])){
      id.model[i,j] = 4
    }
    if((model.5.R[j,i] > model.1.R[j,i] & model.5.R[j,i] > model.2.R[j,i] &
        model.5.R[j,i] > model.3.R[j,i] & model.5.R[j,i] > model.4.R[j,i] &
        model.5.R[j,i] > model.0.R[j,i])){
      id.model[i,j] = 5
    }
  }
}

  #put in formate you can plot
  if (length(unique(inputdata$BlockNum))>5){
  id.model.test <- cbind(as.data.frame(rep(1:length(unique(inputdata$Subject)),6)),
                         as.data.frame(rep(1:6,each = length(unique(inputdata$Subject)))),
                         as.data.frame(c(id.model[,1],
                          id.model[,2],
                          id.model[,3],
                          id.model[,4],
                          id.model[,5],
                          id.model[,6]))
                    )
  }
  else{
  id.model.test <- cbind(as.data.frame(rep(1:length(unique(inputdata$Subject)),5)),
                         as.data.frame(rep(1:5,each = length(unique(inputdata$Subject)))),
                         as.data.frame(c(id.model[,1],
                          id.model[,2],
                          id.model[,3],
                          id.model[,4],
                          id.model[,5]))
                    )
  }
  #rename columns
  colnames(id.model.test) <- c("Subject","Block","Group")

return(id.model.test)
}
```

Simon Task

```
id.model.test <- computemodels.exp2(df.simon)
```

```
ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
           fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Simon - Experiment 2") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```

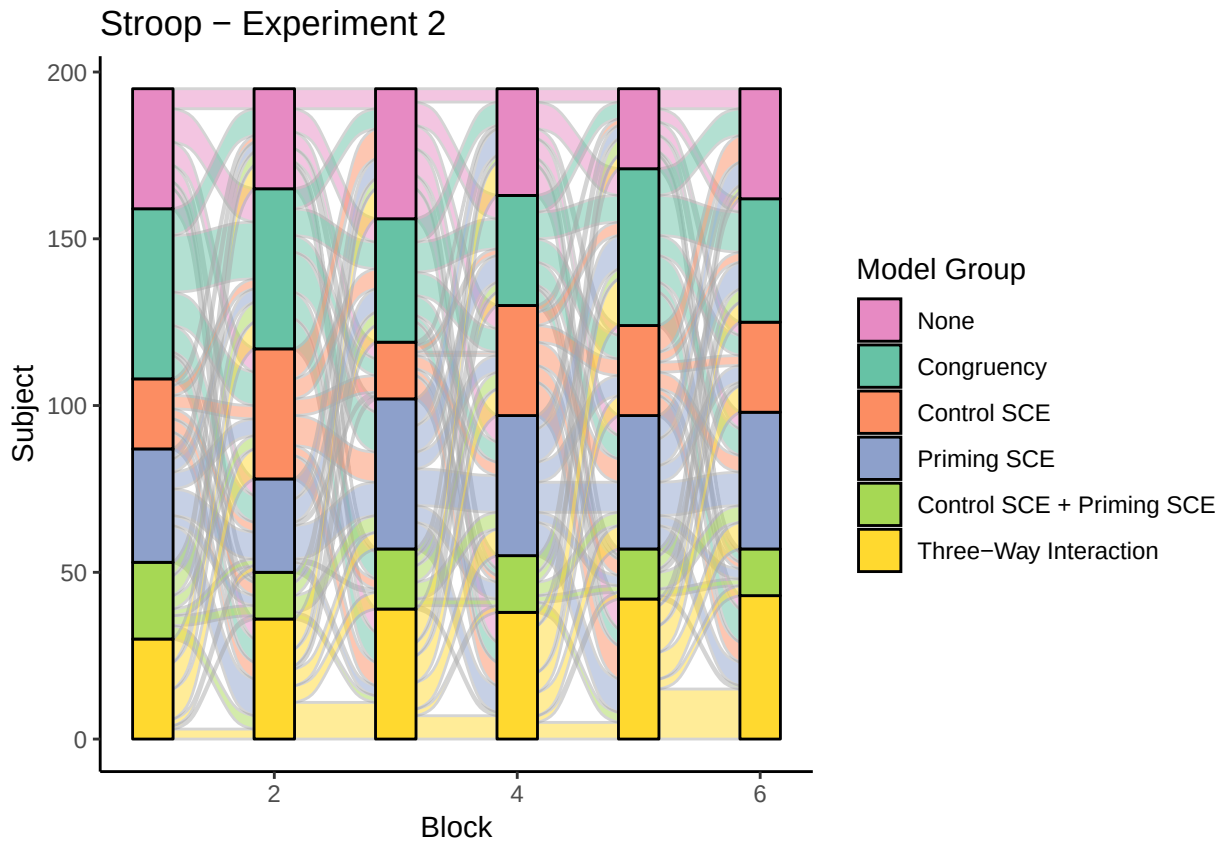



```
simon.2 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Stroop Task

```
id.model.test <- computemodels.exp2(df.stroop)

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
           fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Stroop - Experiment 2") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



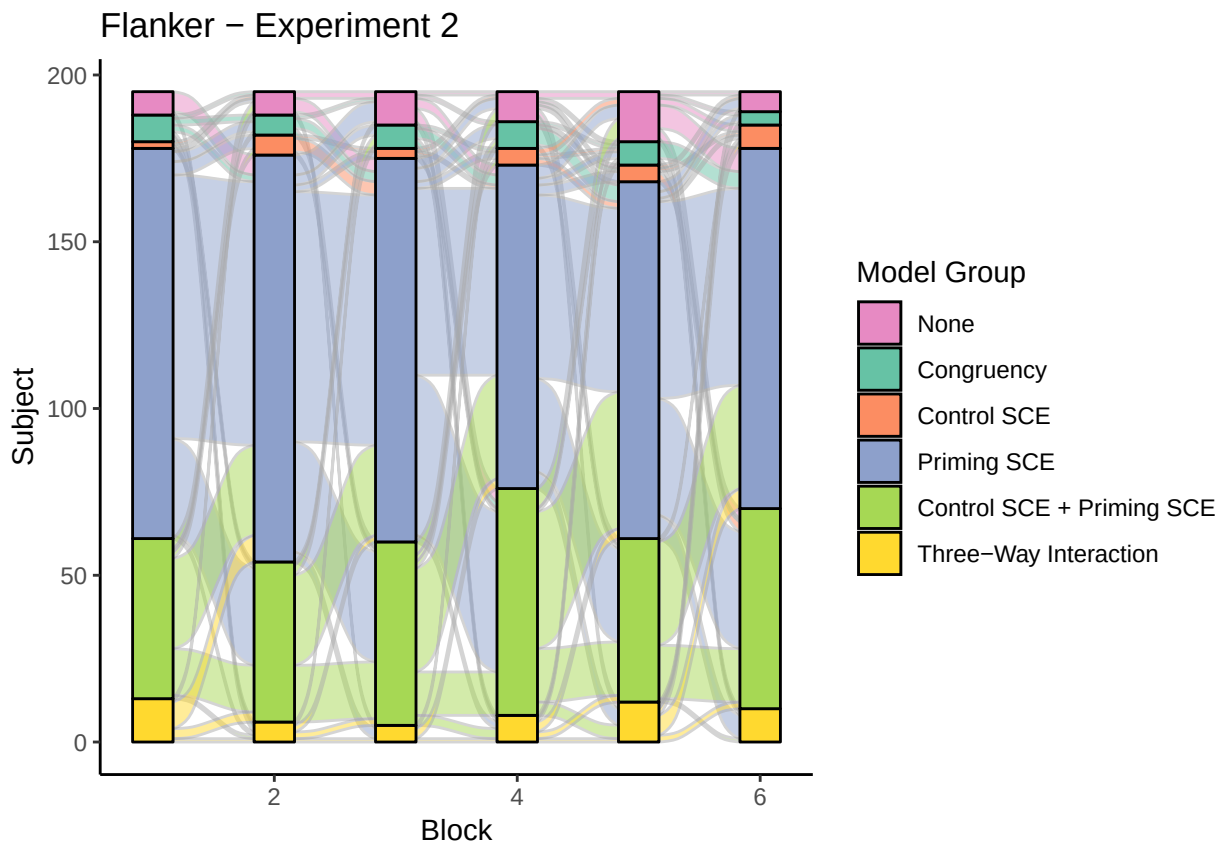

```
stroop.2 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
```

```
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Flanker Task

```
id.model.test <- computemodels.exp2(df.flanker)

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
          fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Flanker - Experiment 2") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```




```
flanker.2 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Experiment 3

Load in data

Clean the data

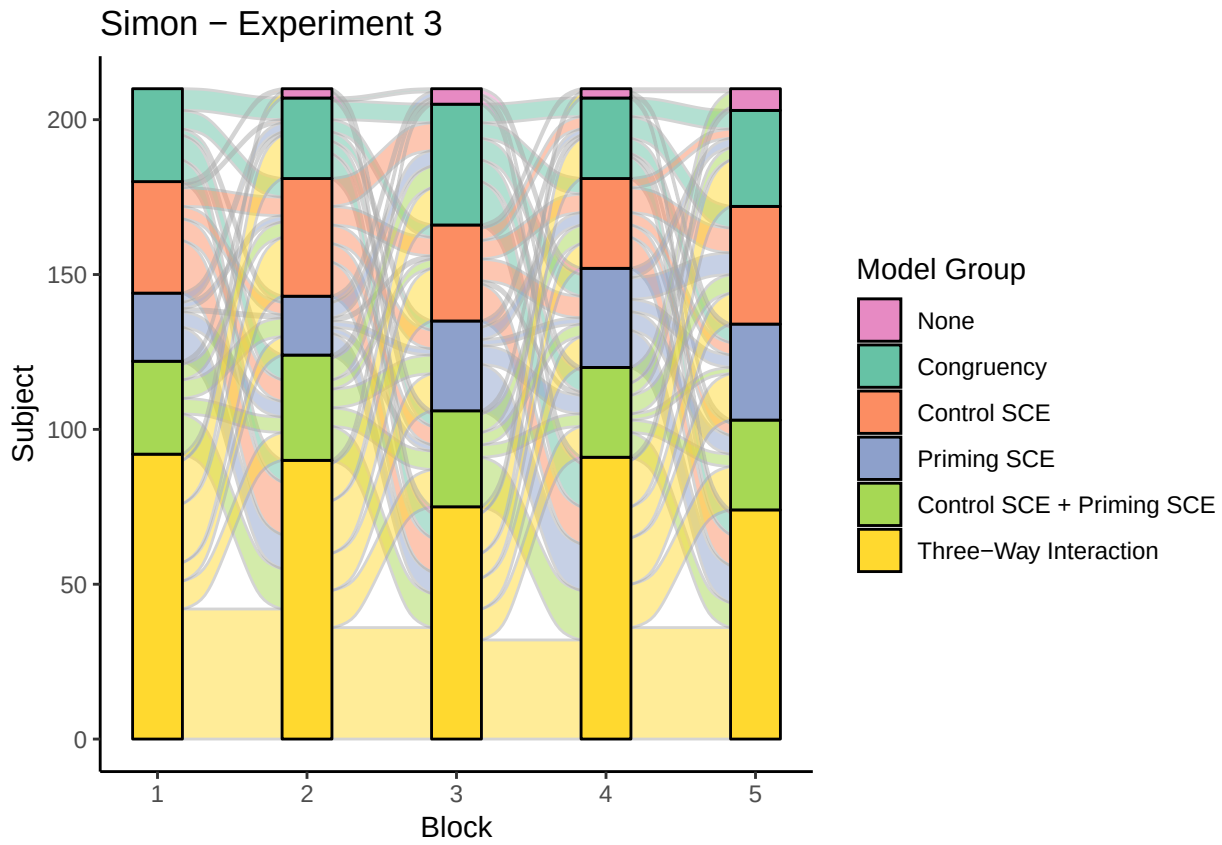
```
#Make a dataframe of only the necessary variables because these datafiles are yuuuuge
data.simon <- data.raw.simon %>%
  dplyr::select(Subject, PracExp, BlockNum, Congruency,
  StimSlideSimon.ACC, StimSlideSimon.RT, Position)
data.flanker <- data.raw.flanker %>%
  dplyr::select(Subject, PracExp, BlockNum, Congruency,
  StimSlideFlanker.ACC, StimSlideFlanker.RT, Color)
data.stroop <- data.raw.stroop %>%
  dplyr::select(Subject, PracExp, BlockNum, Congruency,
  StimSlideStroop.ACC, StimSlideStroop.RT, Color)
```

```r
#subjects to exclude
includesubs <- c(105,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128
                 131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152
                 155,156,157,158,159,160,161,162,163,165,166,167,168,169,170,171,172,173,174,175,176,177
                 180,181,182,183,184,185,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202
                 205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226
                 231,232,233,234,235,236,237,238,240,241,242,243,244,246,247,248,249,250,251,253,254,255
                 259,260,261,262,263,264,265,266,267,268,269,270,271,273,274,275,277,278,279,280,281,282
                 285,286,287,288,289,290,291,293,294,295,296,297,299,301,303,305,306,307,308,309,311,312
                 316,317,318,319,320,321,322,323,324,325,326,327,328,330,331,332,333,334)

#Filter and clean data
#Same as previous
df.simon <- data.simon %>%
  mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideSimon.ACC)) %>%
  mutate(RT = (StimSlideSimon.RT)) %>%
  mutate(targetlag = lag(Position)) %>%
  mutate(TargetRepeat = ifelse(targetlag == Position, 1, 0)) %>%
  filter(Subject %in% includesubs &
  (StimSlideSimon.RT > 200 & StimSlideSimon.RT < 3000) &
  StimSlideSimon.ACC == 1 & acc == 1 &
  PracExp == "Exp")

df.flanker <- data.flanker %>%
  mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideFlanker.ACC)) %>%
  mutate(RT = (StimSlideFlanker.RT)) %>%
  mutate(targetlag = lag(Color)) %>%
  mutate(TargetRepeat = ifelse(targetlag == Color, 1, 0)) %>%
  filter(Subject %in% includesubs &
  (StimSlideFlanker.RT > 200 & StimSlideFlanker.RT < 3000) &
  StimSlideFlanker.ACC == 1 & acc == 1 &
  PracExp == "Exp")

df.stroop <- data.stroop %>%
  mutate(prevcon = lag(Congruency)) %>%
  mutate(acc = lag(StimSlideStroop.ACC)) %>%
  mutate(RT = (StimSlideStroop.RT)) %>%
  mutate(targetlag = lag(Color)) %>%
  mutate(TargetRepeat = ifelse(targetlag == Color, 1, 0)) %>%
  filter(Subject %in% includesubs &
  (StimSlideStroop.RT > 200 & StimSlideStroop.RT < 3000) &
  StimSlideStroop.ACC == 1 & acc == 1 &
  PracExp == "Exp")

#duplicate function
computemodels.exp3 <- computemodels.exp2
```

Simon Task

```
id.model.test <- computemodels.exp3(df.simon)

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
           fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Simon - Experiment 3") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



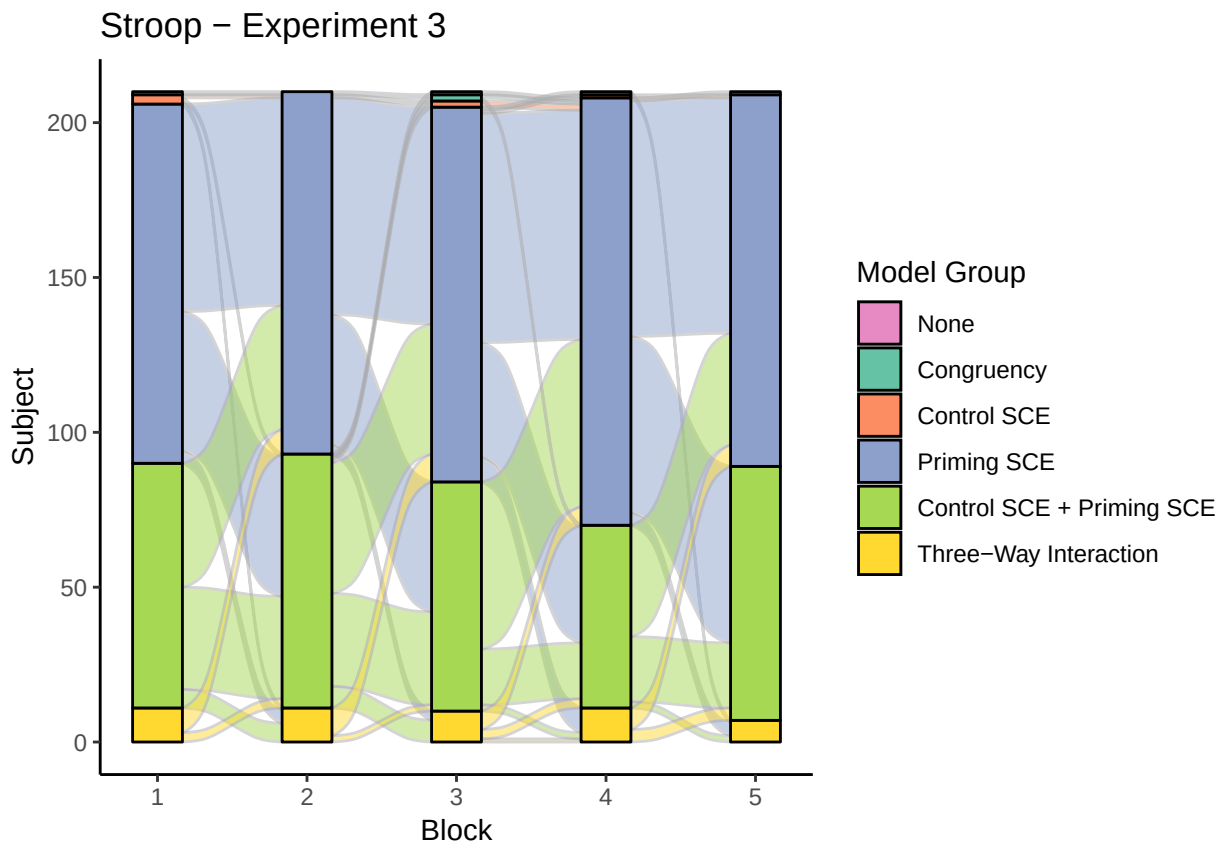

```
simon.3 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
```

```
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Stroop Task

```
id.model.test <- computemodels.exp3(df.stroop)

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
          fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Stroop - Experiment 3") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



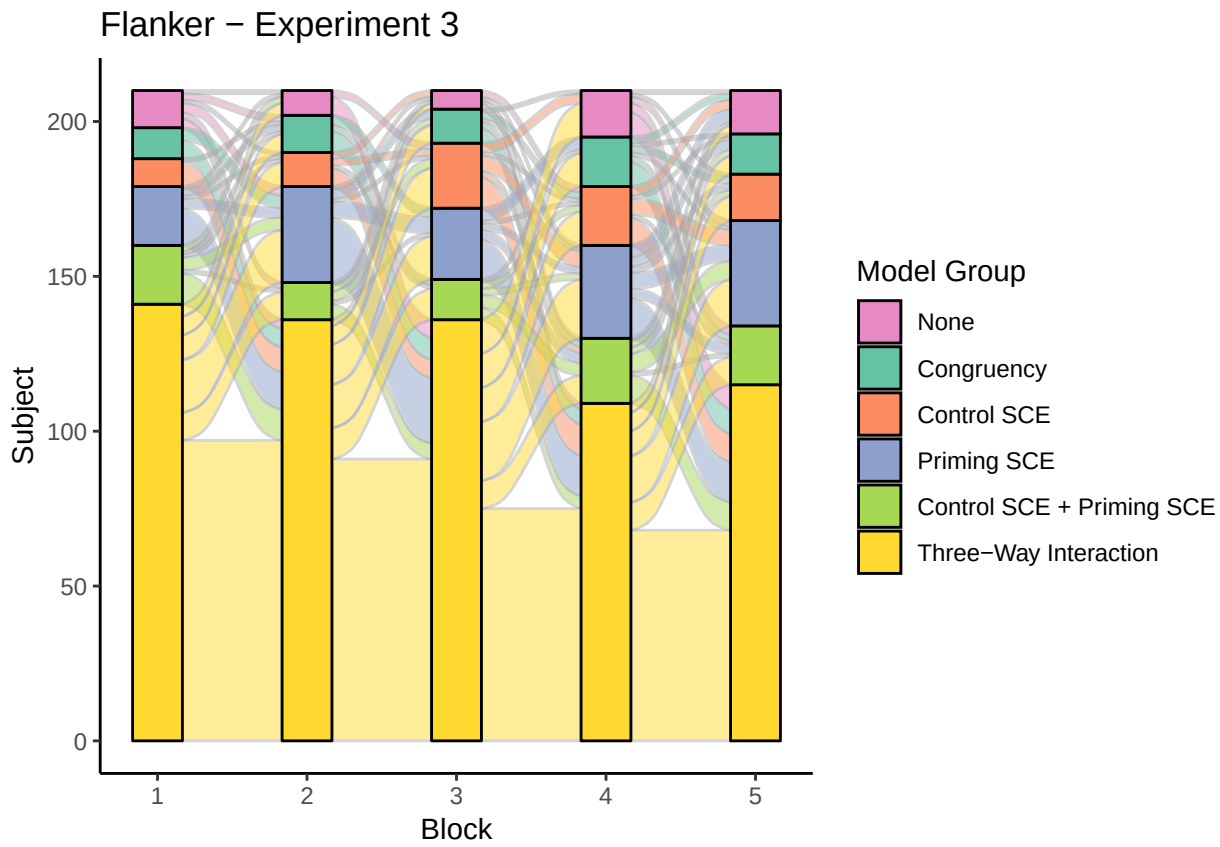

```
stroop.3 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Flanker Task

```
id.model.test <- computemodels.exp3(df.flanker)

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
          fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Flanker - Experiment 3") +
```

```
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```




```
flanker.3 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Experiment 4

```
computemodels.exp4 <- function(inputdata){
  #create some empty matrixes to put in the adjusted R2 values
  model.0.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.1.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.2.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                      ncol = length(unique(inputdata$Subject)))
  model.3.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
```

```
                    ncol = length(unique(inputdata$Subject)))
model.4.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                    ncol = length(unique(inputdata$Subject)))
model.5.R <- matrix(0, nrow = length(unique(inputdata$BlockNum)),
                    ncol = length(unique(inputdata$Subject)))

count.i <- 0 #counting variable

#loop to go through and compute each model, per subject per block
for (i in unique(inputdata$Subject)){
  count.j <- 0
  count.i <- count.i+1
  test <- NULL
  test <- inputdata %>% filter(Subject == i) #only 1 subject
  for (j in unique(test$BlockNum)){
    count.j <- count.j+1
    test.block <- NULL
    test.block <- test %>% filter(test$BlockNum == j)
    model.0 <- lm(RT~1, data = test.block) #null model
    model.1 <- lm(RT~1+Congruency, data = test.block) #congruency model
    model.2 <- lm(RT~1+prevcon*Congruency, data = test.block) #SCE model
    model.3 <- lm(RT~1+TargetRepeat*Congruency, data = test.block) #Target Rep SCE
    model.4 <- lm(RT~1+(prevcon*Congruency)+(TargetRepeat*Congruency), data = test.block) #models 2 &
    model.5 <- lm(RT~1+(prevcon*Congruency):(TargetRepeat*Congruency), data = test.block) #interactio


    model.0.R[count.j,count.i] <- summary(model.0)$adj.r.squared
    model.1.R[count.j,count.i] <- summary(model.1)$adj.r.squared
    model.2.R[count.j,count.i] <- summary(model.2)$adj.r.squared
    model.3.R[count.j,count.i] <- summary(model.3)$adj.r.squared
    model.4.R[count.j,count.i] <- summary(model.4)$adj.r.squared
    model.5.R[count.j,count.i] <- summary(model.5)$adj.r.squared
  }
}

#create dummy data frame to compute the group, in long format
id.model <- matrix(0, nrow = length(unique(inputdata$Subject)),
                   ncol = length(unique(inputdata$BlockNum)))

#decide whether the adjusted R2 value for each person, each block
#is higher or lower than others, in order to determine group membership
for (i in 1:length(unique(inputdata$Subject))){
for (j in 1:length(unique(inputdata$BlockNum))){
  if((model.0.R[j,i] > model.1.R[j,i] & model.0.R[j,i] > model.2.R[j,i] &
      model.0.R[j,i] > model.3.R[j,i] & model.0.R[j,i] > model.4.R[j,i] &
      model.0.R[j,i] > model.5.R[j,i])){
    id.model[i,j] = 0
  }
  if((model.1.R[j,i] > model.0.R[j,i] & model.1.R[j,i] > model.2.R[j,i] &
      model.1.R[j,i] > model.3.R[j,i] & model.1.R[j,i] > model.4.R[j,i] &
      model.1.R[j,i] > model.5.R[j,i])){
    id.model[i,j] = 1
  }
```

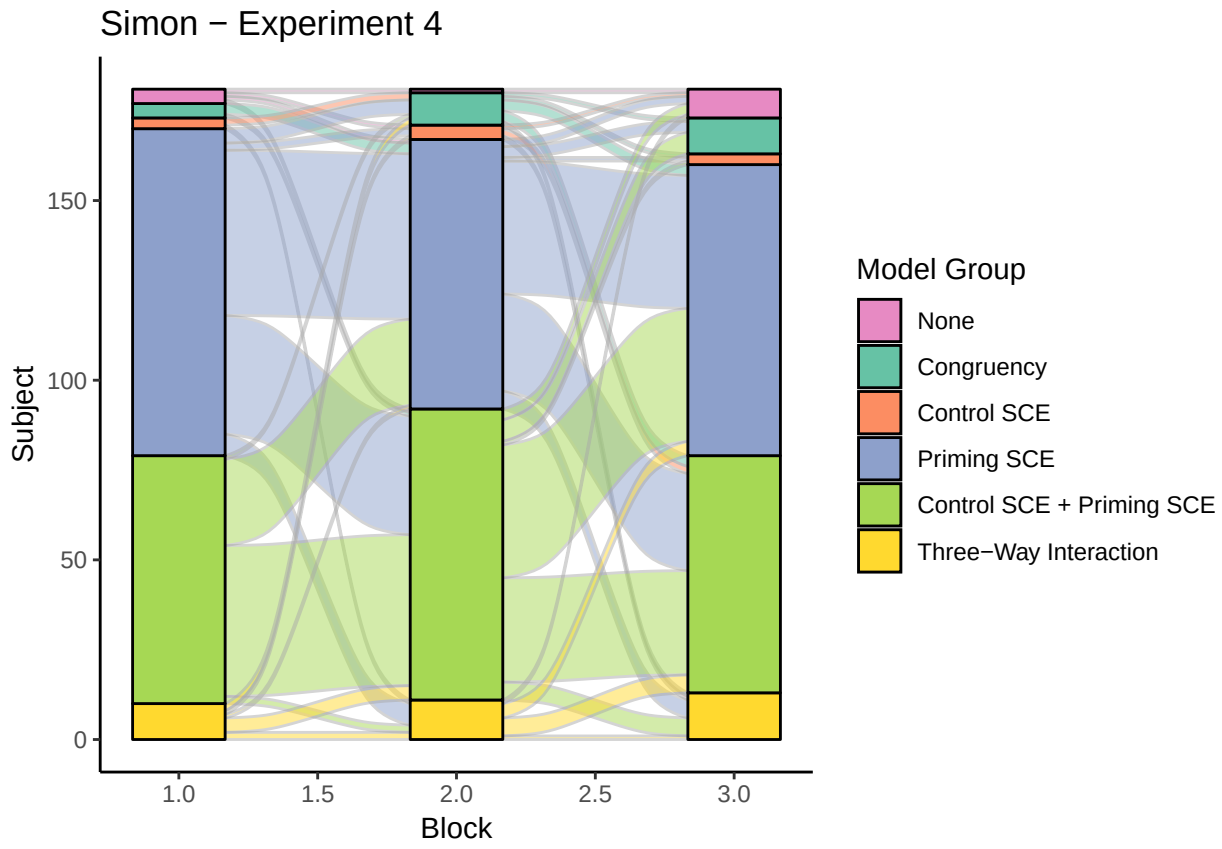
```
    if((model.2.R[j,i] > model.1.R[j,i] & model.2.R[j,i] > model.0.R[j,i] &
        model.2.R[j,i] > model.3.R[j,i] & model.2.R[j,i] > model.4.R[j,i] &
        model.2.R[j,i] > model.5.R[j,i])){
      id.model[i,j] = 2
    }
    if((model.3.R[j,i] > model.1.R[j,i] & model.3.R[j,i] > model.2.R[j,i] &
        model.3.R[j,i] > model.0.R[j,i] & model.3.R[j,i] > model.4.R[j,i] &
        model.3.R[j,i] > model.5.R[j,i])){
      id.model[i,j] = 3
    }
    if((model.4.R[j,i] > model.1.R[j,i] & model.4.R[j,i] > model.2.R[j,i] &
        model.4.R[j,i] > model.3.R[j,i] & model.4.R[j,i] > model.0.R[j,i] &
        model.4.R[j,i] > model.5.R[j,i])){
      id.model[i,j] = 4
    }
    if((model.5.R[j,i] > model.1.R[j,i] & model.5.R[j,i] > model.2.R[j,i] &
        model.5.R[j,i] > model.3.R[j,i] & model.5.R[j,i] > model.4.R[j,i] &
        model.5.R[j,i] > model.0.R[j,i])){
      id.model[i,j] = 5
    }
  }
}

  #put in formate you can plot

  id.model.test <- cbind(as.data.frame(rep(1:length(unique(inputdata$Subject)),3)),
                         as.data.frame(rep(1:3,each = length(unique(inputdata$Subject)))),
                         as.data.frame(c(id.model[,1],
                          id.model[,2],
                          id.model[,3]))
                    )

  #rename columns
  colnames(id.model.test) <- c("Subject","Block","Group")

return(id.model.test)
}
```

Simon Task

```
id.model.test <- computemodels.exp4(df.simon)

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
          fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
```

```
                                "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Simon - Experiment 4") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



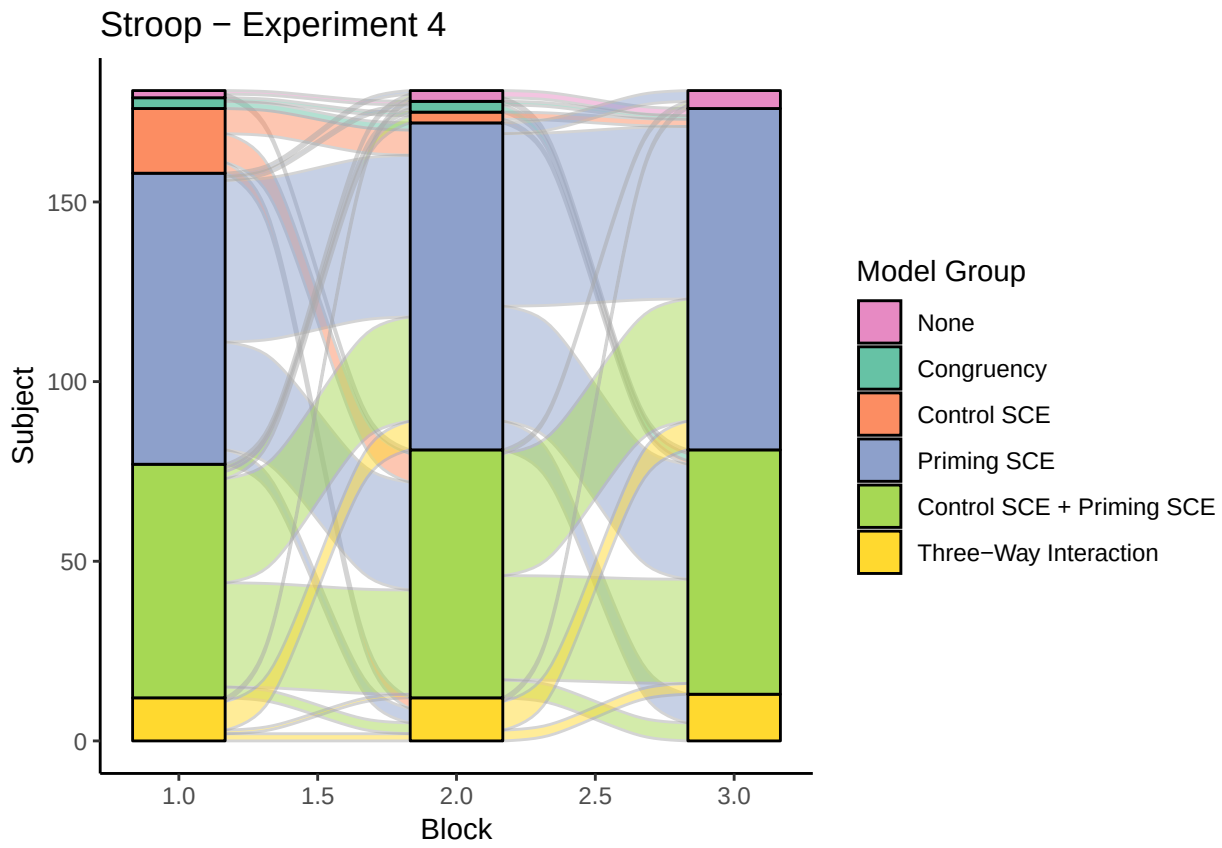

```
simon.4 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Stroop Task

```
id.model.test <- computemodels.exp4(df.stroop)

ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
           fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
```

```
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Stroop - Experiment 4") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



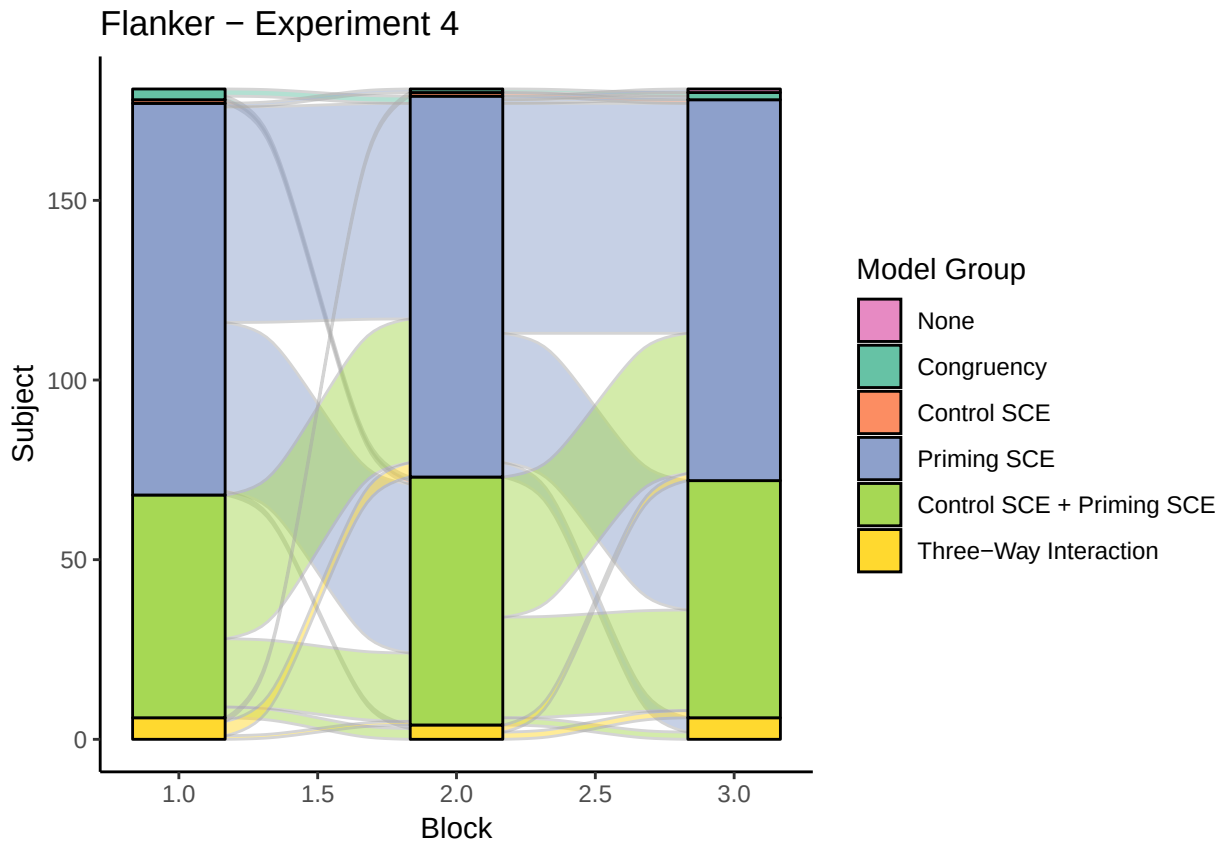

```
stroop.4 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Flanker Task

```
id.model.test <- computemodels.exp4(df.flanker)
```

```
ggplot(id.model.test,
       aes(x = Block, stratum = factor(Group), alluvium = Subject,
           fill =factor(Group), label = factor(Group))) +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  geom_flow(stat = "flow",
            color = "darkgray") +
  scale_fill_manual(name  = "Model Group",
                    values = c("#e78ac3","#66c2a5","#fc8d62",
                               "#8da0cb","#a6d854","#ffd92f"),
                    labels = c("None", "Congruency","Control SCE",
                               "Priming SCE", "Control SCE + Priming SCE", "Three-Way Interaction")) +
  ylab("Subject") +
  ggtitle("Flanker - Experiment 4") +
  theme_classic() +
  geom_stratum()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```




```
flanker.4 <- id.model.test %>% group_by(Subject) %>%
  mutate(switch = lag(Group), switchstay = ifelse(switch == Group, 0, 1)) %>%
  filter(Block != 1) %>%
  group_by(Block) %>%
  summarize(switchprop = sum(switchstay)/length(unique(id.model.test$Subject))) %>%
  summarize(totalswitch = sum(switchprop)/length(unique(id.model.test$Block)))
```

Overall Switching Rate for each Experiment

```
plot.switch <- as.data.frame(cbind(rep(1:3,each=4),rep(1:4,3),
                     rbind(simon.1,simon.2,simon.3,simon.4,
                           stroop.1,stroop.2,stroop.3,stroop.4,
                           flanker.1,flanker.2,flanker.3,flanker.4)))

colnames(plot.switch) <- c("Task","Experiment","Percent")


ggplot(plot.switch, aes(x = factor(Experiment), y = Percent, group = factor(Task), fill = factor(Task)))
    geom_bar(position = "dodge", stat = "identity", width = (.75))  +
    coord_cartesian(ylim = c(.30, 1.00)) +
    scale_x_discrete(name = "", labels=c("Exp1 - No Feature Rep", "Exp2", "Exp3","Exp4 - Fixed Order"))
    scale_fill_manual(name="Task", breaks = c("1","2", "3"),
                      labels=c("Simon", "Stroop", "Flanker"),
                      values = c("1" = "#fdae61", "2" = "#abdda4", "3" = "#2b83ba")) +
    ggtitle("Model Switching Between Tasks/Experiments") +
    labs(y="Percent of Model Switching Between Blocks (%)")  +
    theme_classic(base_size = 12) +
    #theme(legend.position = c(.75, .9)) +
    NULL
```