Advances in Subadult Age Estimation: Evaluating the Performance of the Mixed Cumulative Probit (MCP) on contemporary subadults

AABA - Denver, CO

Elaine Y. Chu

March 2022

Objective

The goal of this project was to demonstrate the utility of the MCP algorithm for subadult age estimation.

```
## Load Libraries
library(tidyverse)
library(caret)
library(yada)
library(doParallel)
library(magrittr)
```

All univariate and multivariate models were generated using the R scripts found in the *scripts* folder. The pipeline of the scripts follow the same order that is outlined at the following: Click for GitHub. This pipeline must be conducted before proceeding.

Materials

Our materials include the US sample from the Subadult Virtual Anthropology Database. The training sample is the complete dataset, whereas the test set is ~20% of the complete dataset, randomly selected using the *caret* package in R.

```
rm(list=ls())  # clear working directory

## Import problem file
train <- read.csv('data/us_train_samp.csv')

## Create testing sample from training sample
set.seed(2021)  # for reproducibility
train_idx <- createDataPartition(train$agey, p=0.8, list=FALSE)
test <- train[-train_idx,]

## Save file as .csv
write.csv(test, 'data/us_test_samp.csv' ,row.names=F)

## Sample size for training and test sets
nrow(train)
```

```
## [1] 1316
```

```r
nrow(test)
```

```
## [1] 260
```

```r
## Training and testing variables
colnames(train)  # 1:3-demographics; 4:65-indicators/variables
```

```
##  [1] "SVAD_identifier" "SEX"             "agey"            "FDL"
##  [5] "FMSB"            "FDB"             "TDL"             "TPB"
##  [9] "TMSB"            "TDB"             "FBDL"            "HDL"
## [13] "HPB"             "HMSB"            "HDB"             "RDL"
## [17] "RPB"             "RMSB"            "RDB"             "UDL"
## [21] "UMSB"            "max_M1"          "max_M2"          "max_M3"
## [25] "max_PM1"         "max_PM2"         "max_C"           "max_I1"
## [29] "max_I2"          "man_M1"          "man_M2"          "man_M3"
## [33] "man_PM1"         "man_PM2"         "man_C"           "man_I1"
## [37] "man_I2"          "FH_EF"           "FGT_EF"          "FLT_EF"
## [41] "FDE_EF"          "TPE_EF"          "TDE_EF"          "FBPE_EF"
## [45] "FBDE_EF"         "HH_Oss"          "HGT_Oss"         "HLT_Oss"
## [49] "HPE_EF"          "HC_Oss"          "HT_Oss"          "HLE_Oss"
## [53] "HDE_EF"          "HME_EF"          "RPE_EF"          "RDE_EF"
## [57] "UPE_EF"          "UDE_EF"          "CT_EF"           "CC_Oss"
## [61] "TC_Oss"          "ISPR_EF"         "ILIS_EF"         "PC_Oss"
## [65] "IC_EF"
```

Methods

In total, 62 univariate models and four (4) multivariate models were tested for accuracy and bias. Using the test sample, a point estimate and 95% confidence interval (CI) was generated. These values (point estimate and range) were used to evaluate:
- bias (raw residuals)
- accuracy (standard deviation of the residuals)
- percent accuracy (number of individuals whose known age falls within the 95% CI)
- standard estimation error, SEE (standard deviation of residuals * 1.96 for 95% prediction range)

Certain functions were specifically coded for this project and can be accessed by "sourcing" the `functions.R` file in the `extras` folder as shown below. The functions are called:
- `idx_all_na` : This function returns the row number of samples with all NAs. This is particularly useful for the six-variable multivariate model, in which some individuals in the test sample may be missing all 6 variables.
- `mcp_univariate_pred` : This function calculates the point estimate and 95% CI for each individual, by each selected univariate model.
- `mcp_multivariate_pred` : This function calculates the point estimate and 95% CI for each individual by the conditionally independent and conditionally dependent multivariate models.
- `calc_perf` : This function calculates the performance metric dictated by the *type*= argument ("%accuracy","rmse","see")

```r
source('extras/functions.R')
```

###Example Scripts: Predicting Age So for example, to predict age for the test sample using all univariate models independently, you would invoke the following script:

```
## Because many functions within the posterior prediction process
## use parallel processing, register the number of cores you want
## to use
registerDoParallel(detectCores()-2)

## Generate a set of possible "ages" to predict
## for the posterior distribution
xcalc <- seq(0,25,by=0.01)

## save_file=T saves the files to a 'results' folder
mcp_univariate_pred(data_dir="models", analysis_name="US_allvar",
                    test_samp=test, response_cols=1:ncol(test),
                    id_col=1, age_col=3,
                    xcalc=xcalc, seed=2021, save_file=T)
```

To predict age using the six variable ("sixvar") multivariate models, you would run the following script:

```
## Generate a set of possible "ages" to predict
## for the posterior distribution
xcalc <- seq(0,25,by=0.01)

mcp_multivariate_pred(data_dir="models", analysis_name="US_sixvar",
                      test_samp=test, id_col=1, age_col=3, xcalc=xcalc,
                      seed=2021, save_file=T)
```

Example Script: Calculating Bias and Accuracy

After you have your predictions, you will want to calculate raw and absolute residuals to evaluate model performance. The example below uses the predictions from the univariate FDL (femur diaphyseal length) model and updates the file.

NOTE: Raw residuals are depicted in the presentation as "Bias" and Absolute residuals are depicted as "Accuracy"

```
fdl_pred <- read.csv("uni_results/FDL_test_predictions.csv")  # import file
fdl_pred %<>% mutate(resid=point_est-known_age,
                      abs_resid=abs(point_est-known_age))  # calculate values
head(fdl_pred)  # inspect work
write.csv(fdl_pred, "uni_results/FDL_test_predictions.csv")  # save file as .csv
```

To do this process across all univariate variables, I used the following lines of code to create a new file *univariate_predictions_combined.csv*:

```
## Data Munge
file_names <- list.files("uni_results","test_predictions.csv")
df0 <- data.frame(SVAD_identifier=NA,
                  known_age=NA,
                  point_est=NA,
                  lower95=NA,
                  upper95=NA,
```

```
                   resid=NA,
                   abs_resid=NA,
                   var=NA,
                   type=NA)

for (i in 1:length(file_names)) {
  temp <- read.csv(paste0("results/",file_names[i]))
  var_name <- gsub("_test_predictions.csv","",file_names[i])
  type <- ifelse(grepl("max_|man_",var_name),"Dental Development",
                 ifelse(grepl("_EF",var_name),"Epiphyseal Fusion",
                        ifelse(grepl("_Oss",var_name),"Ossification",
                               "Diaphyseal Dimension")))

  new <- temp %>% mutate(resid=point_est-known_age,
                          abs_resid=abs(point_est-known_age),
                          var=var_name,
                          type=type)
  write.csv(new, paste0("uni_results/",file_names[i]), row.names=F)
  df0 <- rbind(df0,new)
}
df <- df0 %>% drop_na()
write.csv(df, "uni_results/univariate_predictions_combined.csv", row.names=F)
```

For the multivariate models, I'll present an alternative way that doesn't include a for_loop. The resulting file is *multivariate_predictions_combined.csv*:

```
allvar_cindep <- read.csv('multi_results/US_allvar_cindep_model_test_predictions.csv')
allvar_cdep <- read.csv('multi_results/US_allvar_cdep_model_test_predictions.csv')
sixvar_cindep <- read.csv('multi_results/US_sixvar_cindep_model_test_predictions.csv')
sixvar_cdep <- read.csv('multi_results/US_sixvar_cdep_model_test_predictions.csv')

# type is the number of variables
# type2 is the type of dependence

allvar_cindep %<>% mutate(type="allvar",type2="cindep",
                          resid=known_age-point_est,
                          abs_resid=abs(known_age-point_est))
allvar_cdep %<>% mutate(type="allvar",type2="cdep",
                        resid=known_age-point_est,
                        abs_resid=abs(known_age-point_est))
sixvar_cindep %<>% mutate(type="sixvar",type2="cindep",
                          resid=known_age-point_est,
                          abs_resid=abs(known_age-point_est))
sixvar_cdep %<>% mutate(type="sixvar",type2="cdep",
                        resid=known_age-point_est,
                        abs_resid=abs(known_age-point_est))

df <- rbind(sixvar_cindep,sixvar_cdep,allvar_cindep,allvar_cdep)
write.csv(df, 'multi_results/multivariate_predictions_combined.csv', row.names=F)
```

Now, I want to calculate the performance metrics of each univariate model. To do so, I will utilize the `calc_perf` function wrapped in a forloop to calculate:
- % Accuracy

- RMSE
- SEE

```
file_names <- list.files("uni_results","_test_predictions.csv")

df <- data.frame(var=NA,
                 p_accuracy=NA,
                 rmse=NA,
                 see=NA,
                 type=NA)

for (i in 1:length(file_names)) {
  var0 <- gsub("_test_predictions.csv","",file_names[i])
  p_accuracy0 <- calc_perf(data_dir="results",
                           file_name=file_names[i],
                           type="%accuracy")
  rmse0 <- calc_perf(data_dir="results",
                     file_name=file_names[i],
                     type="rmse")
  see0 <- calc_perf(data_dir="results",
                    file_name=file_names[i],
                    type="SEE")
  type0 <- ifelse(grepl("max_|man_",var0),"Dental Development",
                 ifelse(grepl("_EF",var0),"Epiphyseal Fusion",
                        ifelse(grepl("_Oss",var0),"Ossification",
                               "Diaphyseal Dimension")))
  df0 <- data.frame(var=var0,
                    p_accuracy=p_accuracy0,
                    rmse=rmse0,
                    see=see0,
                    type=type0)
  df <- rbind(df, df0)
}

df %<>% drop_na()
write.csv(df, "uni_results/univariate_model_performance.csv", row.names=F)
```

For the multivariate models, a similar format into *multivariate_model_performance.csv*:

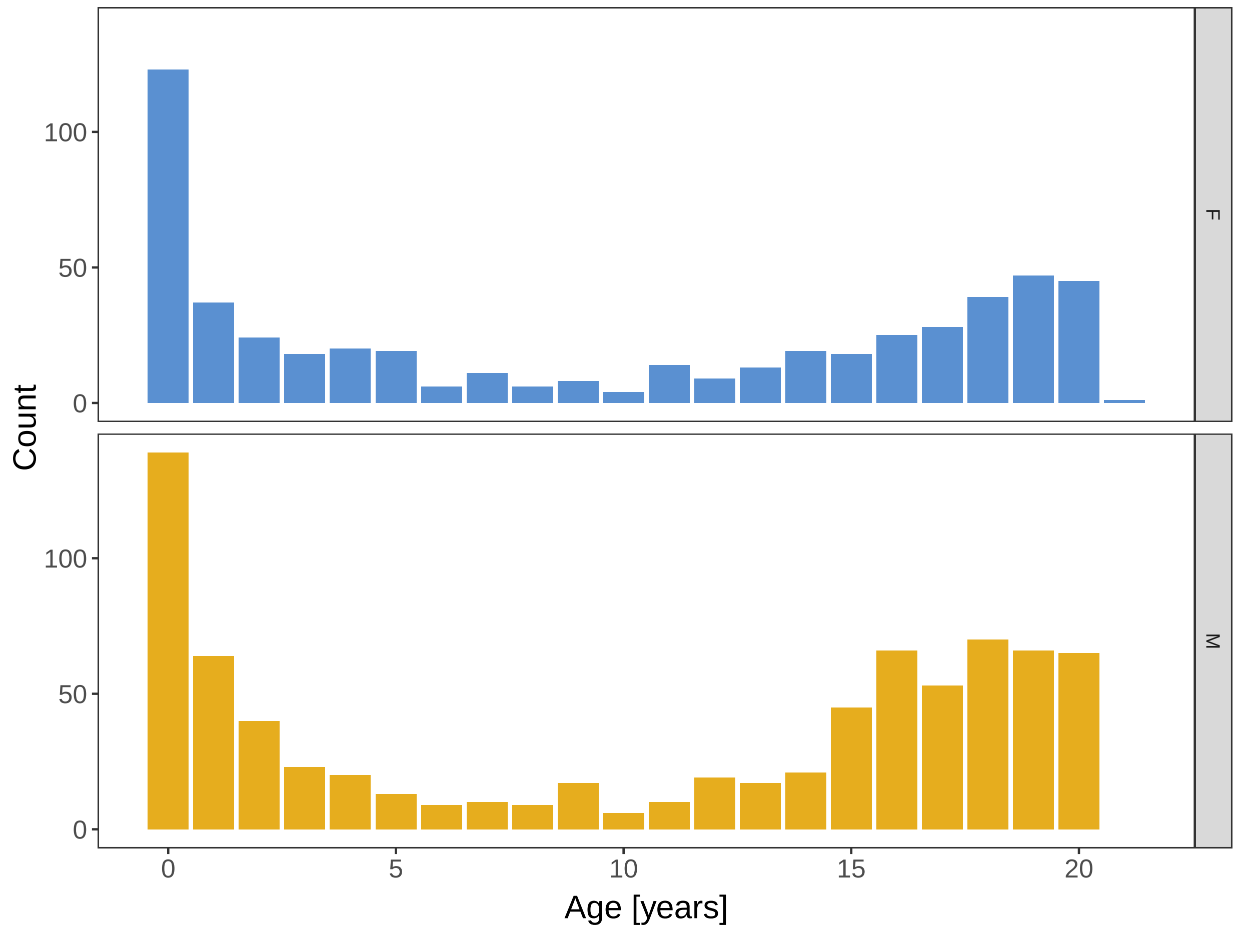
```
# Find all multivariate prediction files
multi_files <- list.files('multi_results','_test_predictions.csv')
type <- c(rep("allvar",2),rep("sixvar",2))
type2 <- rep(c("cdep","cindep"),2)

# Temporary dataframe
temp <- data.frame(matrix(nrow=length(multi_files),ncol=5))
names(temp) <- c('type','type2','p_accuracy','rmse','see')

for(i in 1:length(multi_files)) {
  var0 <- gsub("_test_predictions.csv","",multi_files[i])
  p_accuracy0 <- calc_perf("multi_results",multi_files[i],"%accuracy")
  rmse0 <- calc_perf("multi_results",multi_files[i],"rmse")
  see0 <- calc_perf("multi_results",multi_files[i],"SEE")
  type0 <- type[i]
```

```
  type02 <- type2[i]

  row0 <- c(type0,type02,p_accuracy0,rmse0,see0)
  temp[i,] <- row0
}

temp %<>% mutate(model=paste0(type,'_',type2))
temp$p_accuracy <- as.numeric(temp$p_accuracy)*100
temp$model <- factor(temp$model, levels=c('sixvar_cdep','sixvar_cindep',
                                          'allvar_cdep','allvar_cindep'))
write.csv(temp, 'multi_results/multivariate_model_performance.csv', row.names=F)
```

Figures

Slide 9

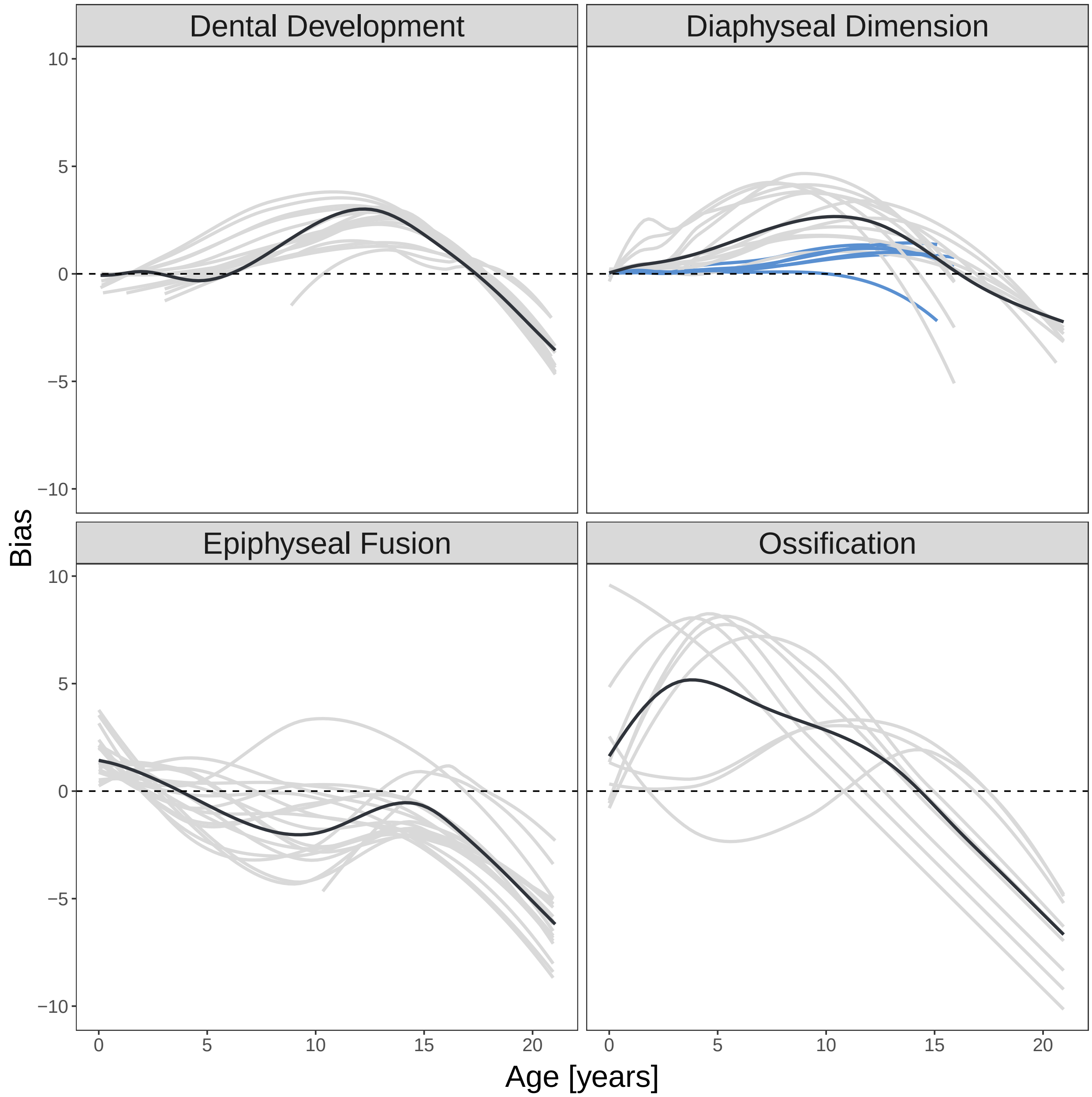
```
rm(list=ls())  # clear working directory

## Import themes and colors
source('extras/aaba_2022_theme.R')

## Import data
train <- read.csv('data/us_train_samp.csv')

ggplot(train) + geom_bar(aes(x=as.integer(agey), fill=SEX)) +
  elaine_theme + labs(x="Age [years]", y="Count") +
  facet_grid(rows=vars(SEX)) + theme(legend.position="none") +
  scale_fill_manual(values=aaba_colors_2022[4:5])
```



Slides 14-16

```
uni_tests <- read.csv("uni_results/univariate_predictions_combined.csv")
uni_tests %<>% mutate(type2=ifelse(grepl("PB|MSB|DB",uni_tests$var),"Breadth",
                                   ifelse(grepl("DL",uni_tests$var),"Length",
                                          "NA")))

## Bias1
ggplot(uni_tests) +
  geom_smooth(aes(x=known_age, y=resid, group=var, color=type2), se=F) +
  scale_color_manual(values=c("grey85",aaba_colors_2022[4],"grey85")) +
  elaine_theme + labs(x="Age [years]", y="Bias") +
  geom_smooth(aes(x=known_age, y=resid), se=F,
              color=aaba_colors_2022[1]) +
  geom_hline(yintercept=0, linetype="dashed") +
  facet_wrap(~type) + theme(legend.position="none") +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



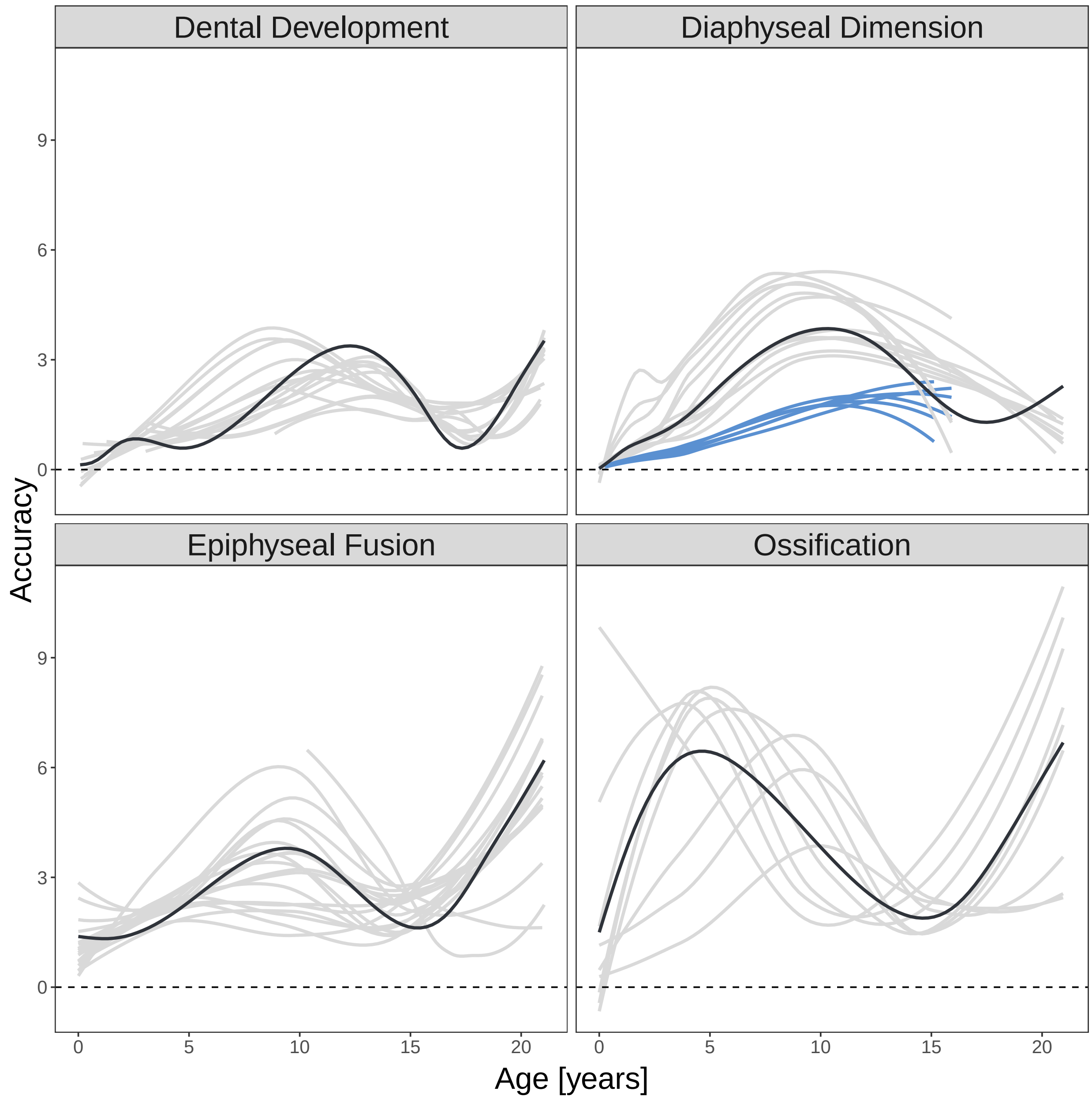

```
## Bias2
uni_tests2 <- uni_tests[-grep('PB|MSB|DB',uni_tests$var),]
ggplot(uni_tests2) +
  geom_smooth(aes(x=known_age, y=resid, group=var), color="grey85", se=F) +
  elaine_theme + labs(x="Age [years]", y="Bias") +
  geom_smooth(aes(x=known_age, y=resid), se=F,
              color=aaba_colors_2022[1]) +
  geom_hline(yintercept=0, linetype="dashed") +
  facet_wrap(~type) + theme(legend.position="none") +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```


```
## Accuracy
ggplot(uni_tests) +
  geom_smooth(aes(x=known_age, y=abs_resid, group=var, color=type2), se=F) +
  scale_color_manual(values=c("grey85",aaba_colors_2022[4],"grey85")) +
  elaine_theme + labs(x="Age [years]", y="Accuracy") +
  geom_smooth(aes(x=known_age, y=abs_resid), se=F,
              color=aaba_colors_2022[1]) +
  geom_hline(yintercept=0, linetype="dashed") +
  facet_wrap(~type) + theme(legend.position="none") +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

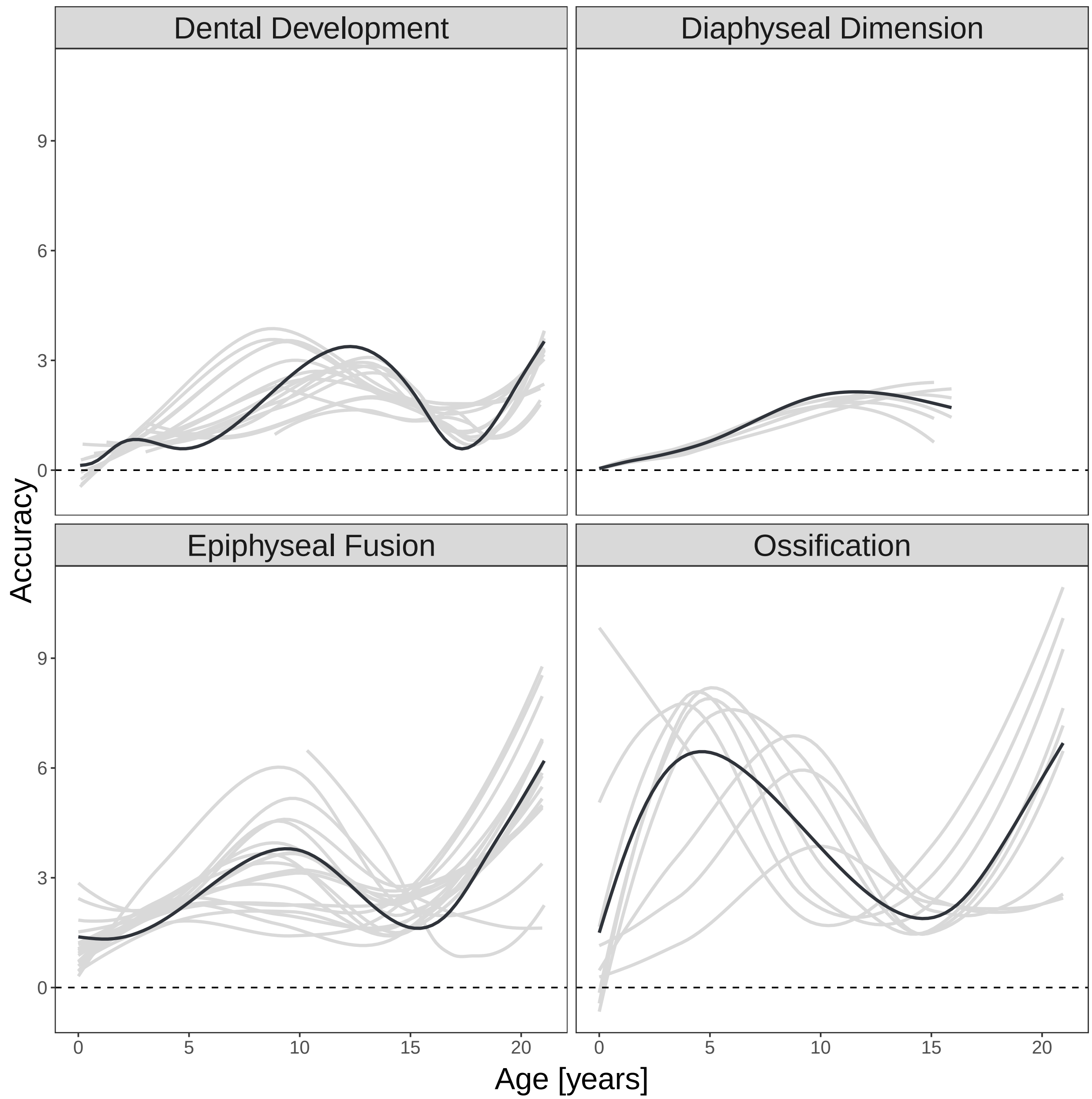
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## Accuracy2
ggplot(uni_tests2) +
  geom_smooth(aes(x=known_age, y=abs_resid, group=var), color="grey85", se=F) +
  elaine_theme + labs(x="Age [years]", y="Accuracy") +
  geom_smooth(aes(x=known_age, y=abs_resid), se=F,
              color=aaba_colors_2022[1]) +
  geom_hline(yintercept=0, linetype="dashed") +
  facet_wrap(~type) + theme(legend.position="none") +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

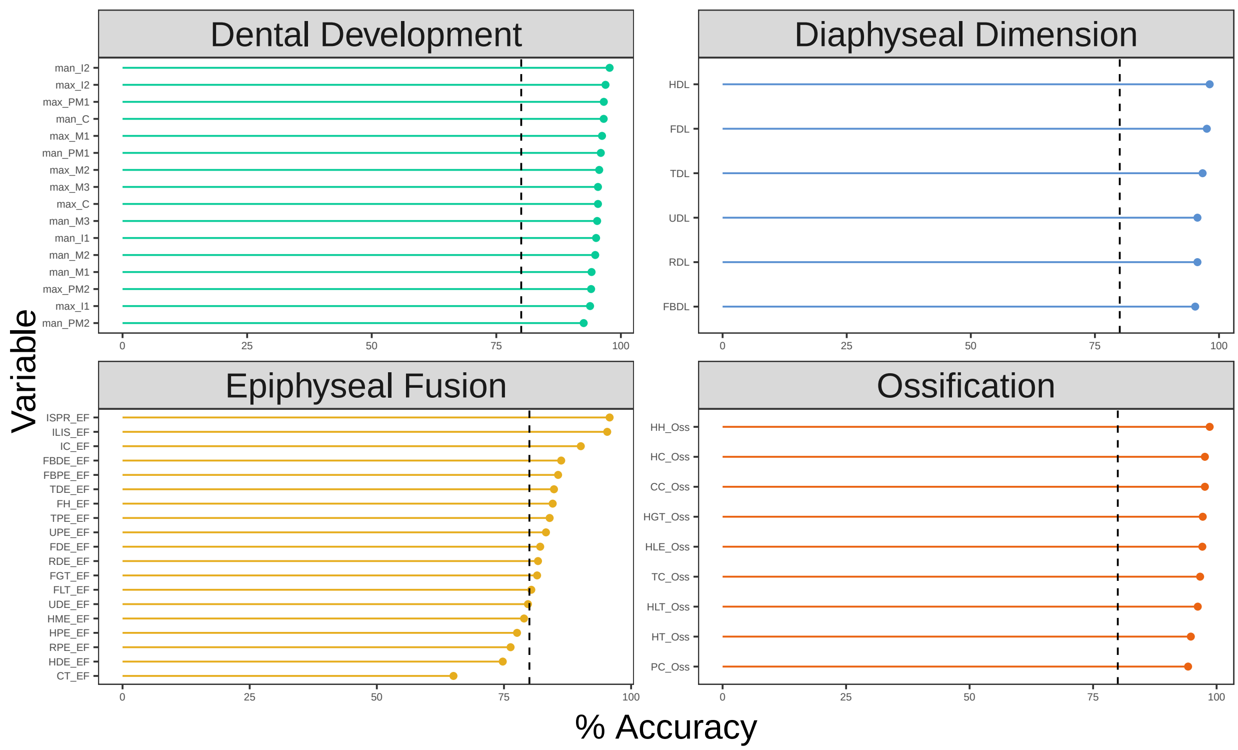
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Slide 17

```
uni_perf <- read.csv("uni_results/univariate_model_performance.csv")
uni_perf$p_accuracy <- uni_perf$p_accuracy*100  # convert to percentage
uni_perf2 <- uni_perf[-grep('PB|MSB|DB',uni_perf$var),]  # remove Breadths

ggplot(uni_perf2 %>% arrange(p_accuracy) %>%
         mutate(var=factor(var, levels=var))) +
  geom_segment(aes(x=var, xend=var, y=0, yend=p_accuracy, color=type)) +
  geom_point(aes(x=var, y=p_accuracy, color=type)) +
```

```
  scale_color_manual(values=aaba_colors_2022[3:6]) +
  geom_hline(aes(yintercept=80), linetype="dashed", color="black") +
  elaine_theme + facet_wrap(~type, scales="free") +
  coord_flip() + labs(x="Variable", y="% Accuracy") +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20),
        legend.position="none",
        axis.text=element_text(size=6))
```



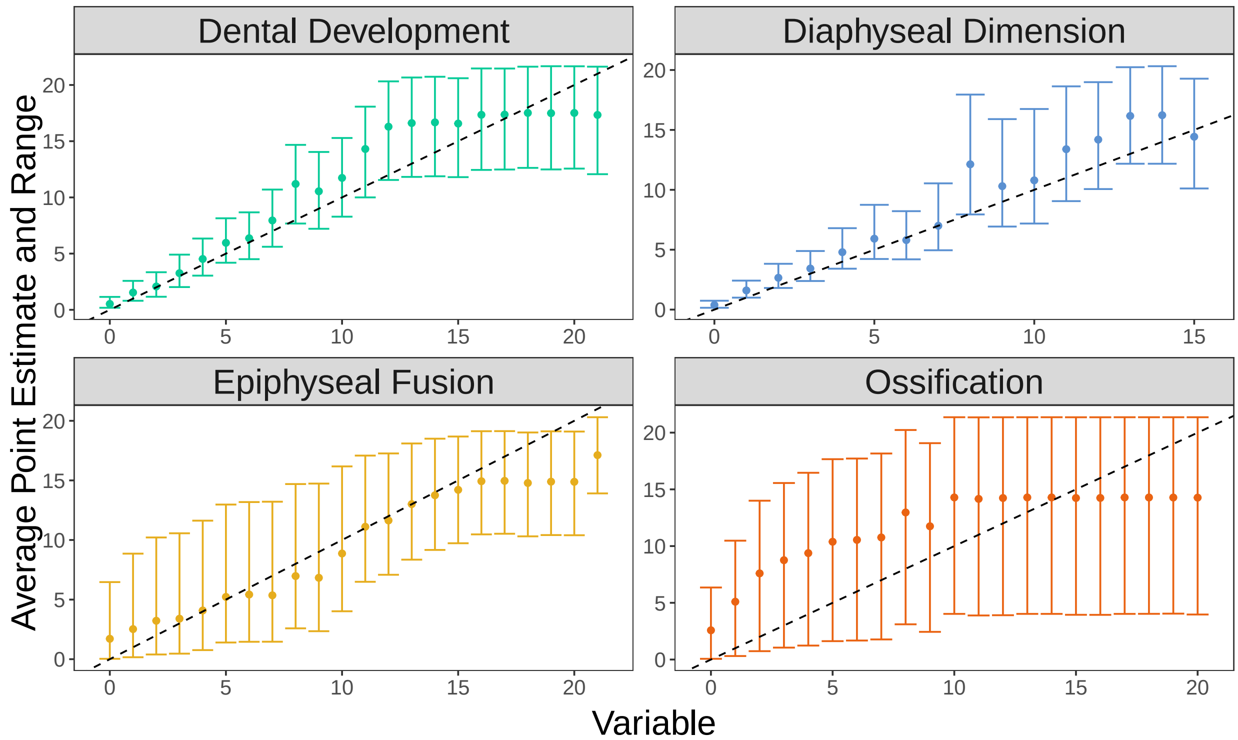
Slide 18

```
uni_pred <- read.csv("uni_results/univariate_predictions_combined.csv")
uni_pred2 <- uni_pred[-grep('PB|MSB|DB',uni_pred$var),]

uni_summary <- uni_pred2 %>% group_by(type, agey=as.integer(known_age)) %>%
  summarize(point=mean(point_est),lower95=mean(lower95),upper95=mean(upper95))
```

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

```
ggplot(uni_summary) +
  geom_point(aes(x=agey, y=point, color=type)) +
  geom_errorbar(aes(x=agey, ymin=lower95, ymax=upper95, color=type)) +
  scale_color_manual(values=aaba_colors_2022[3:6]) +
  geom_abline(aes(slope=1, intercept=0), linetype="dashed", color="black") +
  elaine_theme + facet_wrap(~type, scales="free") +
```

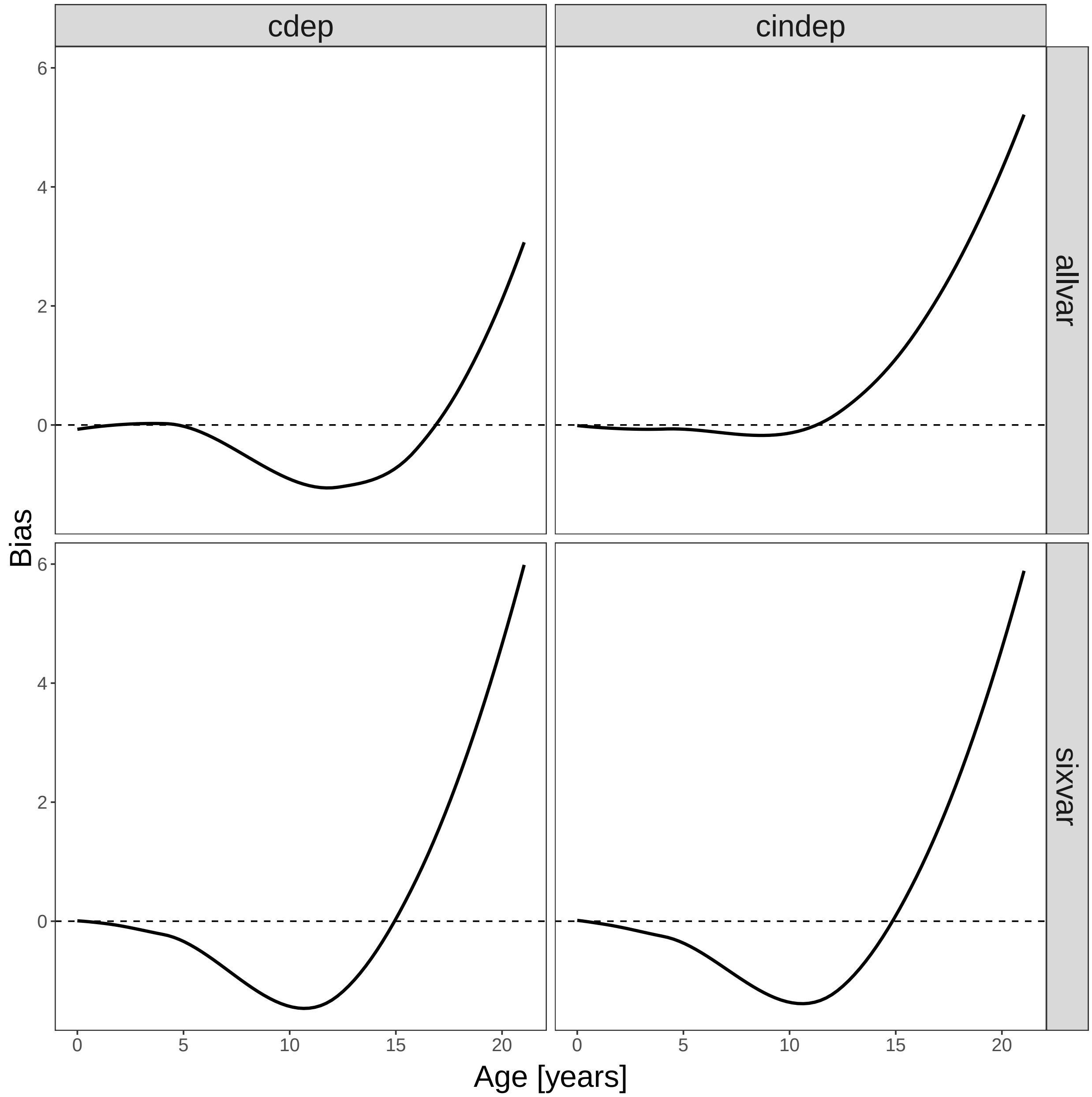
```
  labs(x="Variable", y="Average Point Estimate and Range") +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20),
        legend.position="none")
```



Slide 20

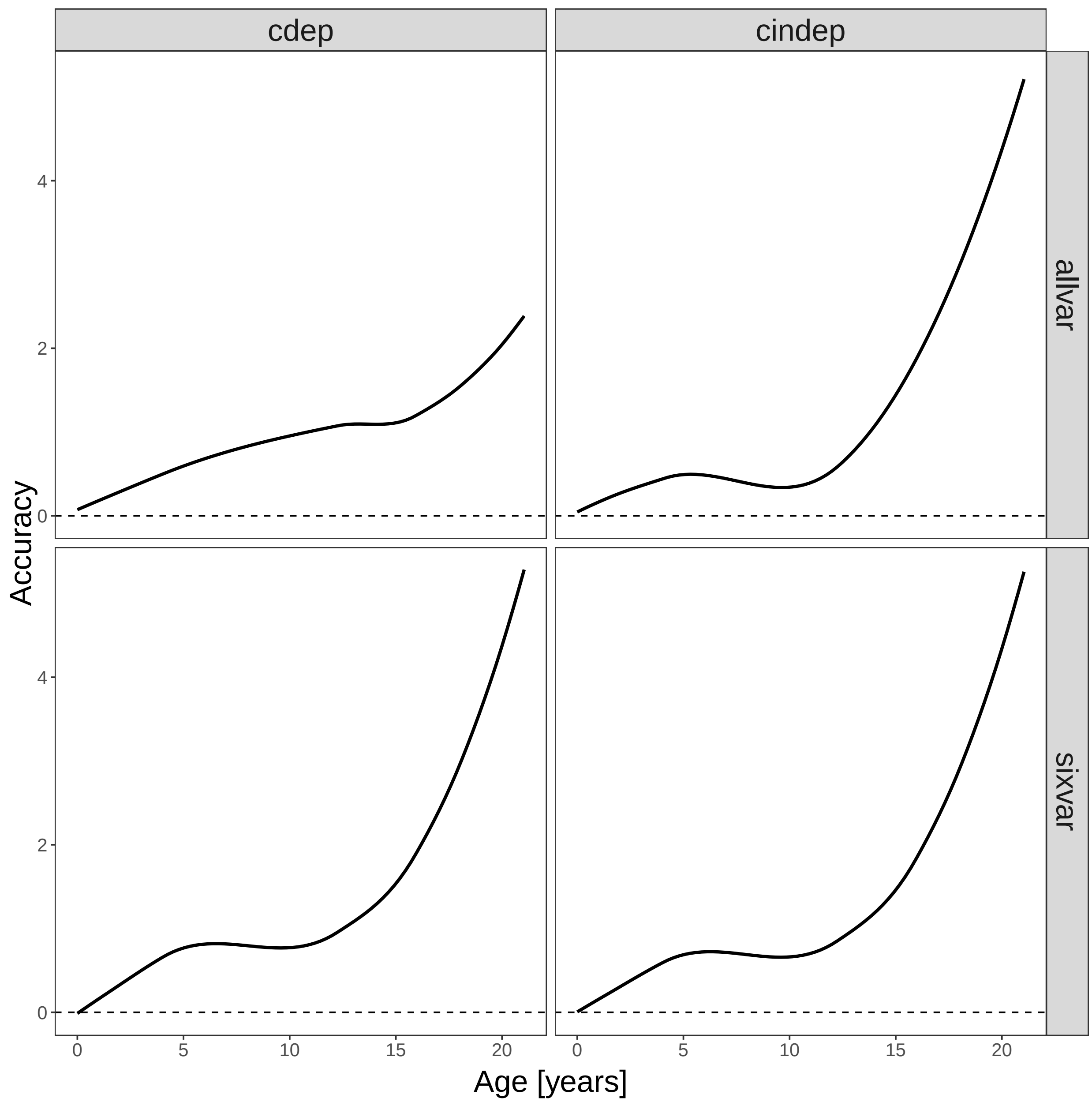
```
multi_pred <- read.csv('multi_results/multivariate_predictions_combined.csv')

## Bias
ggplot(multi_pred) +
  geom_smooth(aes(x=known_age, y=resid), se=F, color="black") +
  elaine_theme + labs(x="Age [years]", y="Bias") +
  geom_hline(yintercept=0, linetype="dashed") +
  facet_grid(rows=vars(type), cols=vars(type2)) +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20))
```

```
## 'geom_smooth()' using method = 'loess' and formula 'y ~ x'
```




```r
## Accuracy
ggplot(multi_pred) +
  geom_smooth(aes(x=known_age, y=abs_resid), se=F, color="black") +
  elaine_theme + labs(x="Age [years]", y="Accuracy") +
  geom_hline(yintercept=0, linetype="dashed") +
  facet_grid(rows=vars(type), cols=vars(type2)) +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20))
```

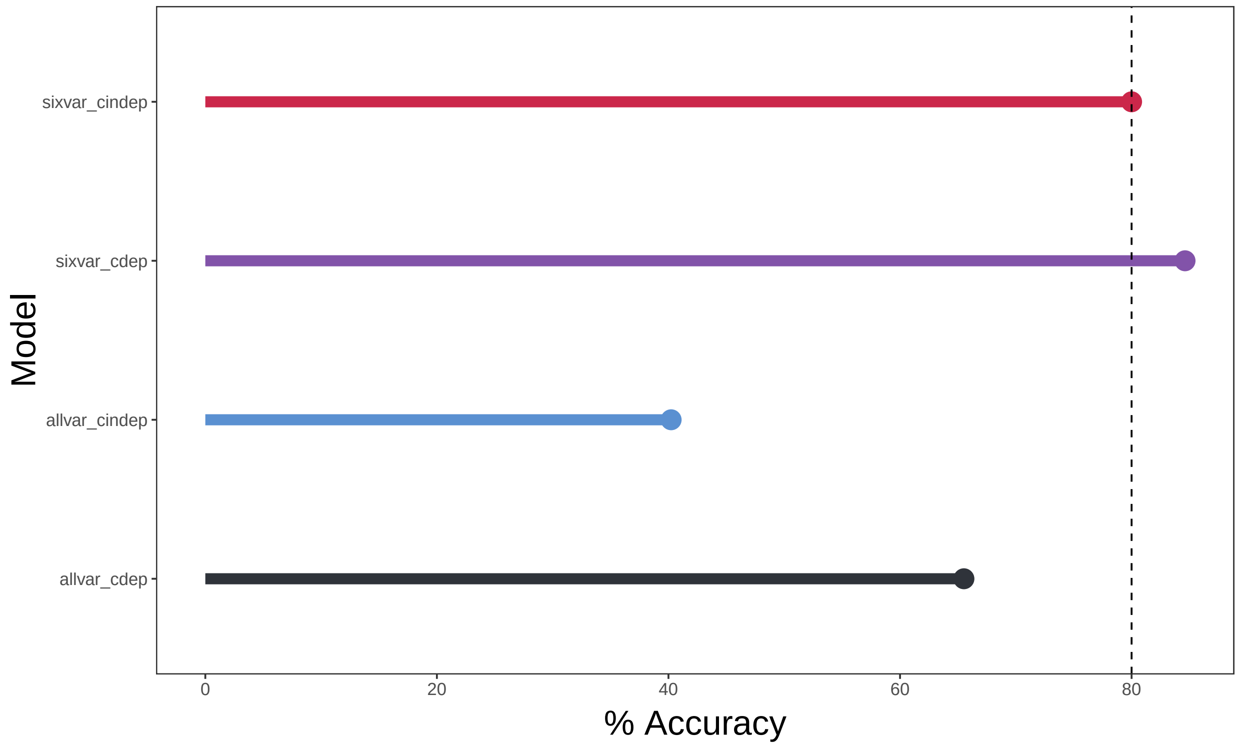
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Slide 21

```
multi_perf <- read.csv('multi_results/multivariate_model_performance.csv')

## Percent Accuracy
ggplot(multi_perf) +
  geom_segment(aes(x=model, xend=model, y=0, yend=p_accuracy, color=model),
               lwd=3) +
  geom_point(aes(x=model, y=p_accuracy, color=model), size=5) +
  scale_color_manual(values=aaba_colors_2022[c(1,4,7:8)]) +
  elaine_theme + coord_flip() + labs(x="Model", y="% Accuracy") +
  theme(strip.text=element_text(size=20),
```

```
        axis.title=element_text(size=20),
        legend.position="none",
        axis.text=element_text(size=10)) +
  geom_hline(aes(yintercept=80), linetype="dashed", color="black")
```

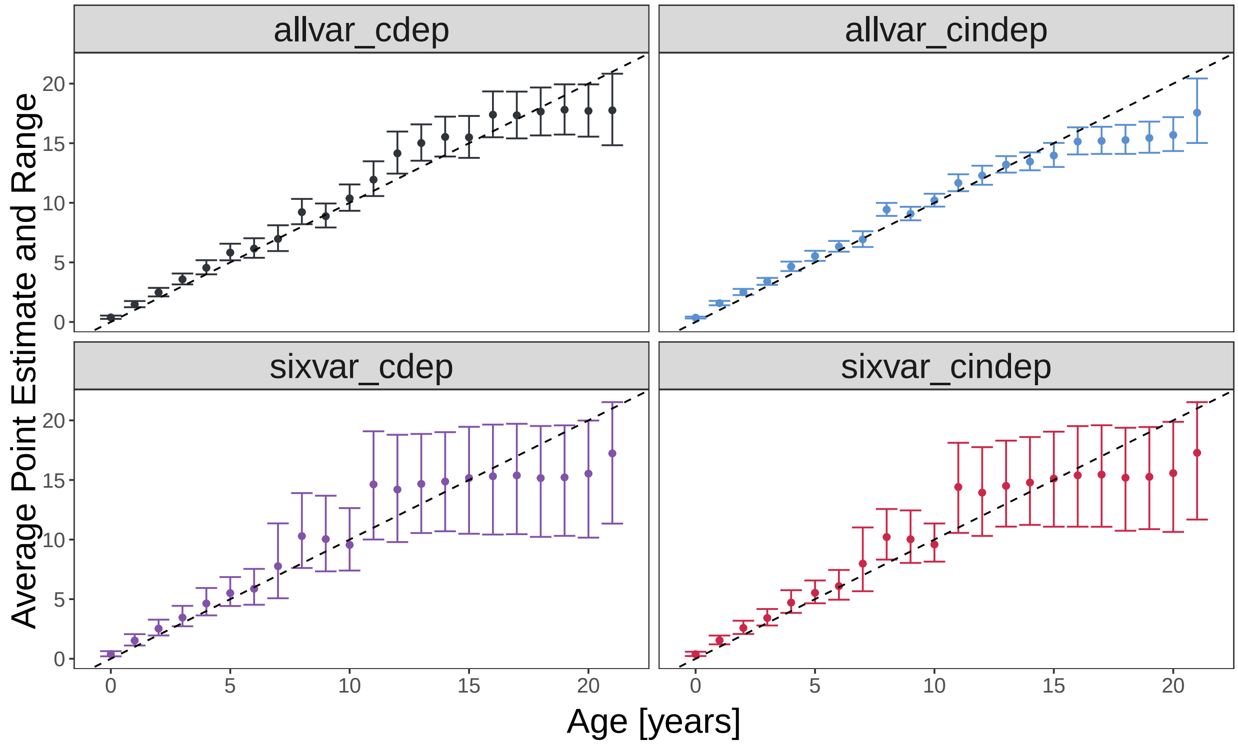


Slide 22

```
multi_pred %<>% mutate(model=paste0(type,'_',type2))
multi_summary <- multi_pred %>% group_by(model, agey=as.integer(known_age)) %>%
  summarize(point=mean(point_est),lower95=mean(lower95),upper95=mean(upper95))
```

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

```
## Performance by Age
ggplot(multi_summary) +
  geom_point(aes(x=agey, y=point, color=model)) +
  geom_errorbar(aes(x=agey, ymin=lower95, ymax=upper95, color=model)) +
  scale_color_manual(values=aaba_colors_2022[c(1,4,7:8)]) +
  geom_abline(aes(slope=1, intercept=0), linetype="dashed", color="black") +
  elaine_theme + facet_wrap(~model) +
  labs(x="Age [years]", y="Average Point Estimate and Range") +
  theme(strip.text=element_text(size=20),
        axis.title=element_text(size=20),
        legend.position="none")
```


allvar_cdep
allvar_cindep
sixvar_cdep
sixvar_cindep
Average Point Estimate and Range
Age [years]
0
5
10
15
20