Recitation 5 Randomization

Seamus Wagner

September 28, 2021

More data visualization

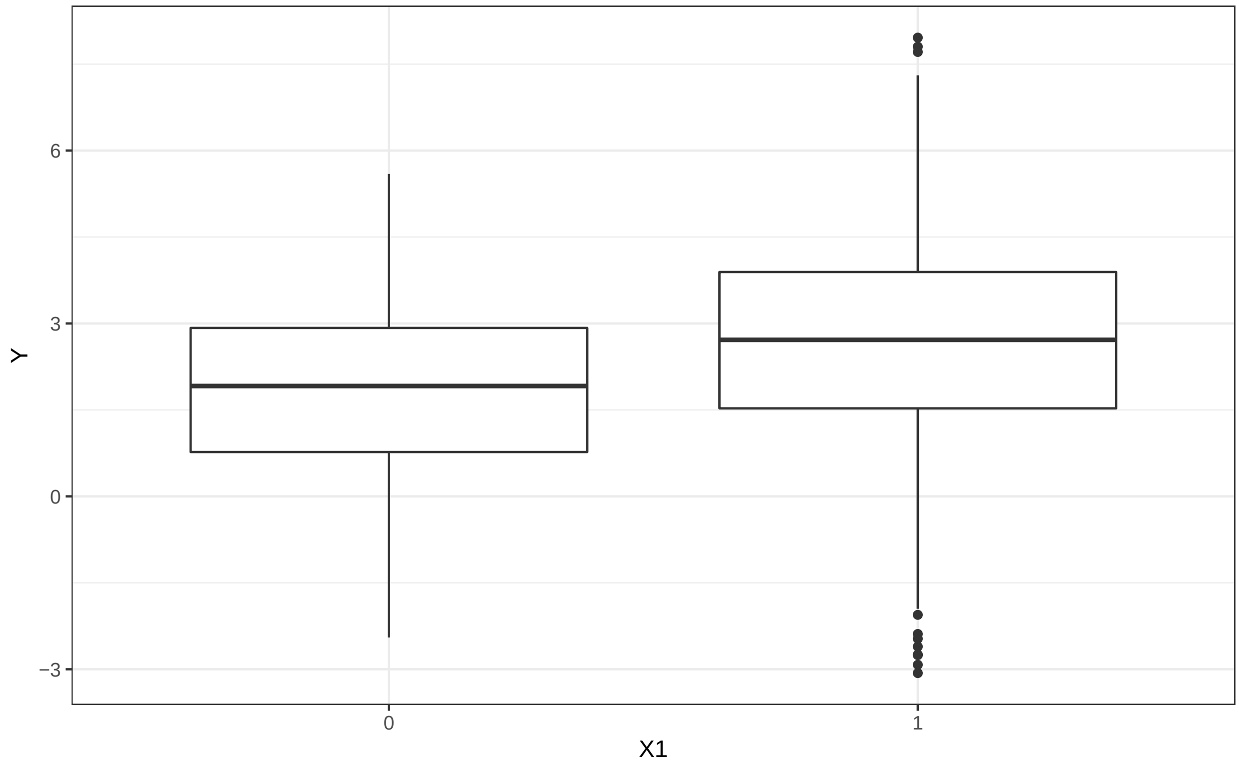
The recitation for today will be more data visualization with simulated data and a practical example at the end, which will also include cleaning and preparing the data.

We begin by creating some data that may be something we would be interested in a basic linear sense. An id column for reference, A binary variable, a continuous normal variable, and five factor levels. I then skip the potential outcomes and only use observable data here, so no assignment. The Y column is modeled to be a linear combination of the variables.

```r
rm(list=ls())
set.seed(10000)
n <- 1000
mu <- 1
sigma <- 1.5
e <- 1
dt <- data.table(id = 1:n,
                 X1 = rbinom(n, 1, .5),
                 X2 = rnorm(n, mu, sigma),
                 X3 = factor(rep(paste0("factor", 1:5))))
dt[, Y := 1 + .5 * X1 + .8 * X2 + .3 * X2 * X1 + -3 * (X3 == 4) + rnorm(n,0,e)]
```

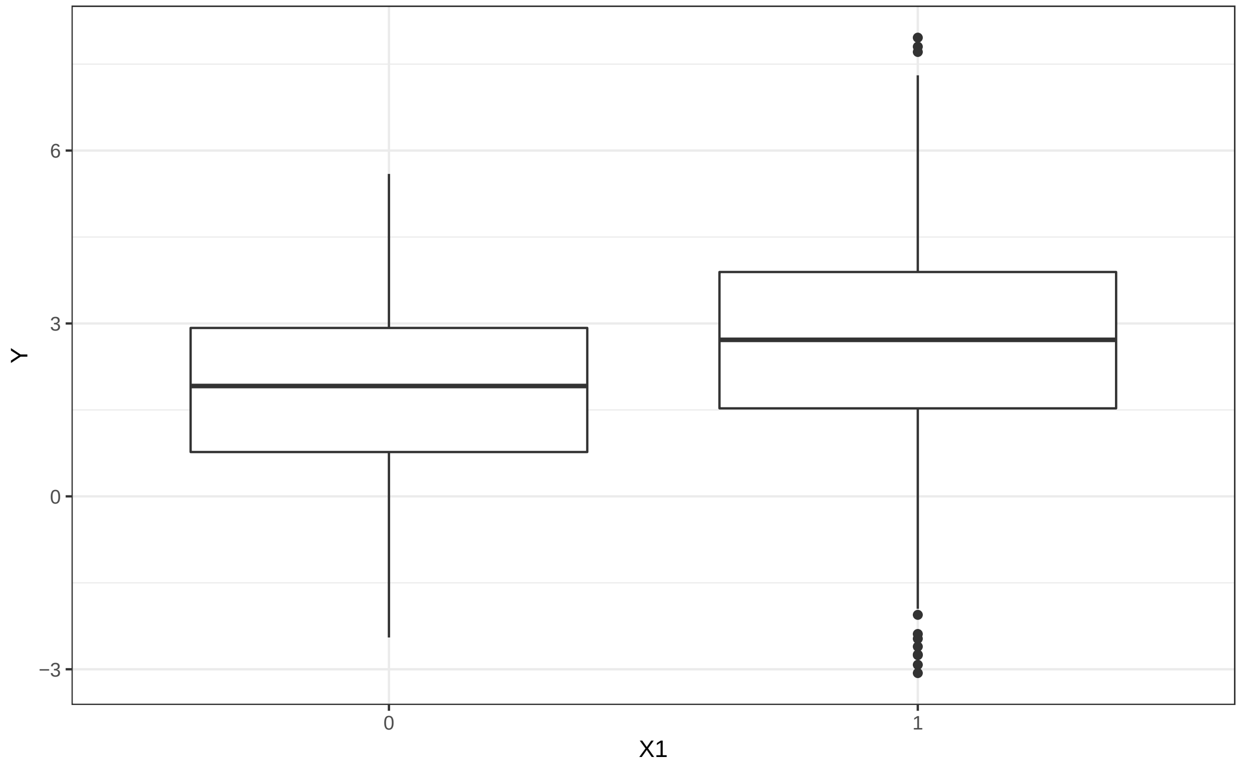
For the first plot, I just show the binary predictor and continuous outcome in a boxplot.

```r
ggplot(dt, aes(factor(X1), Y)) +
  geom_boxplot() +
  theme_bw() +
  xlab("X1")
```



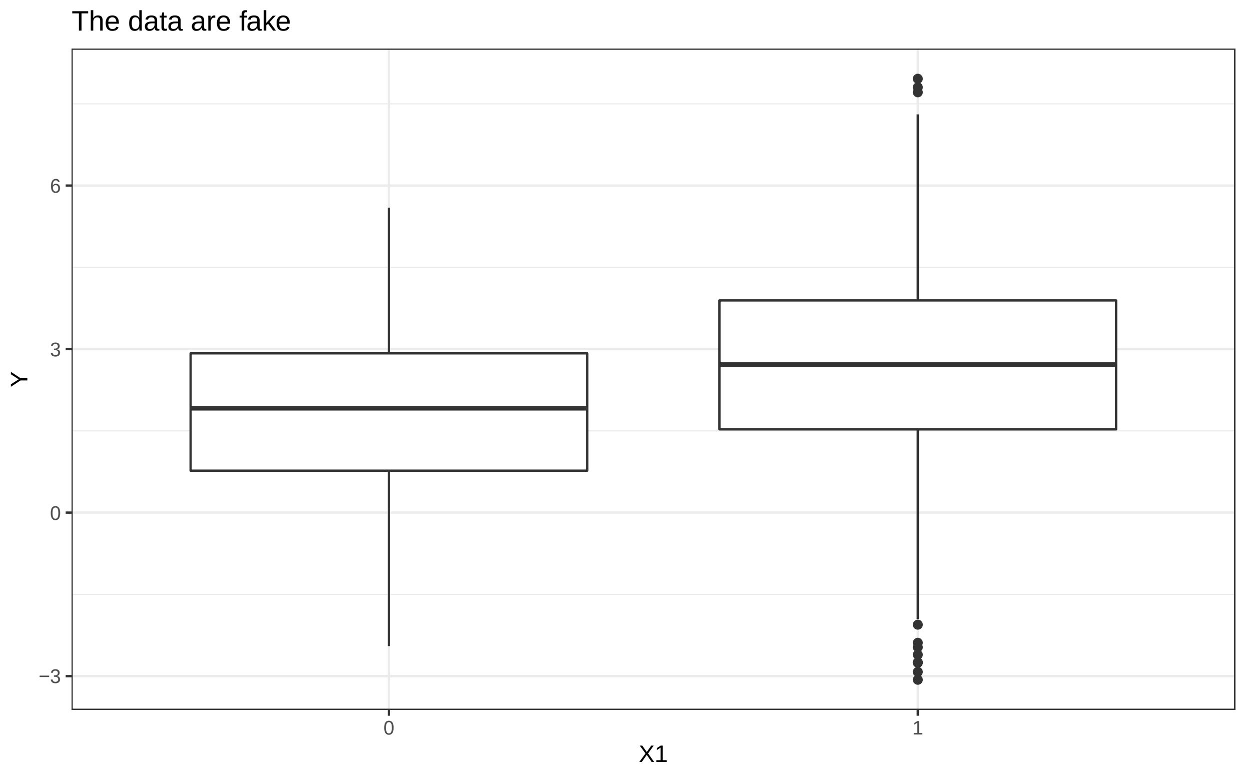
As another refresher, you can pipe ggplot into your tidyverse scripts.

```
dt %>% ggplot(aes(factor(X1), Y)) +
  geom_boxplot() +
  theme_bw() +
  xlab("X1")
```



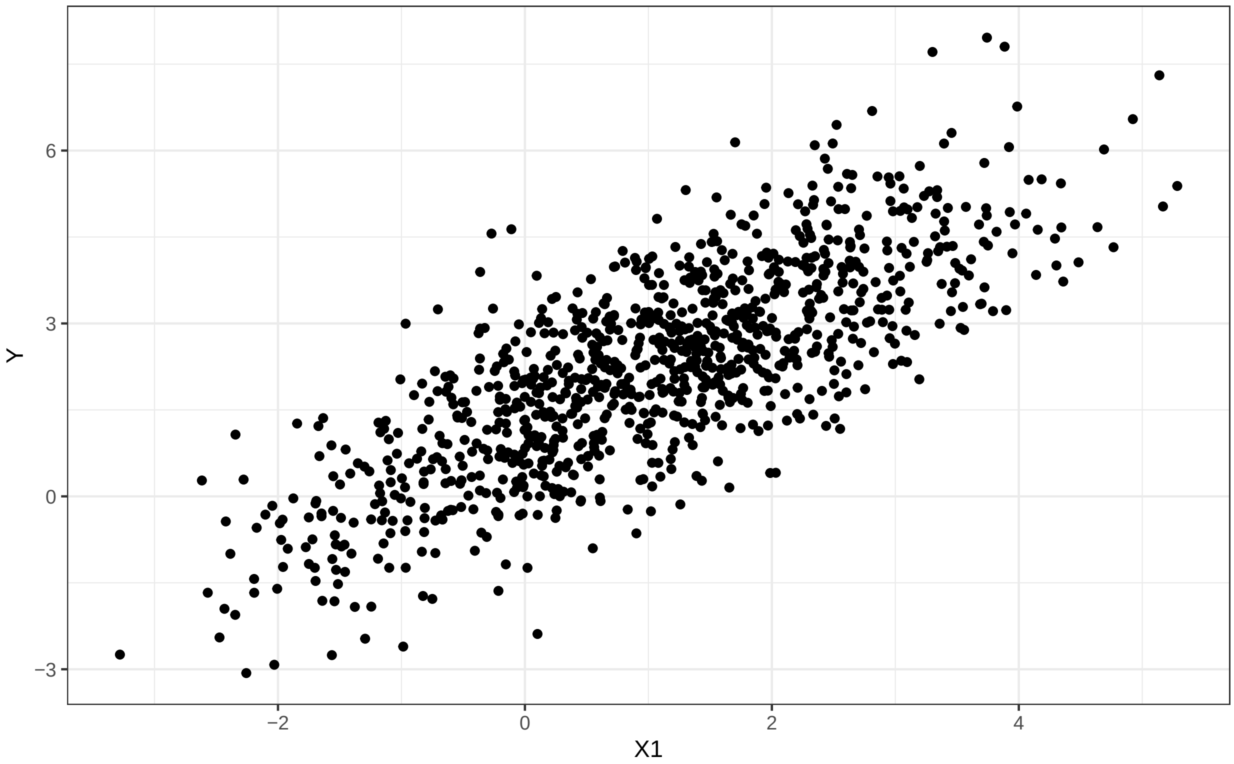
Remember ggplot is layered, so you can create a base object and then add additional commands to it.

```
f1 <- ggplot(dt, aes(factor(X1), Y)) +
  geom_boxplot() +
  theme_bw() +
  xlab("X1")

f1 + ggtitle("The data are fake")
```



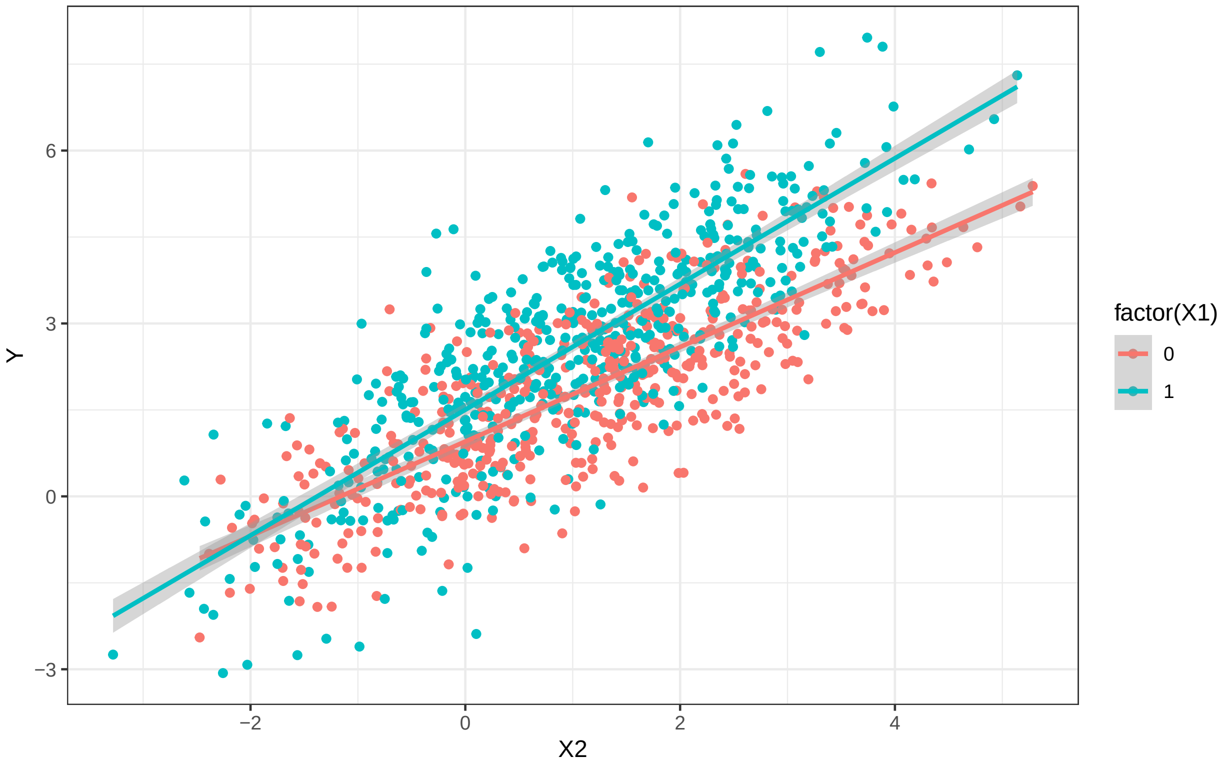
Moving away from boxplots, examine the relationship between the continuous predictor and continuous outcome using a scatterplot “geom_jitter" is another option for this if your data are tightly clustered.

```
ggplot(dt, aes(X2, Y)) +
  geom_point() + theme_bw() +
  xlab("X1")
```



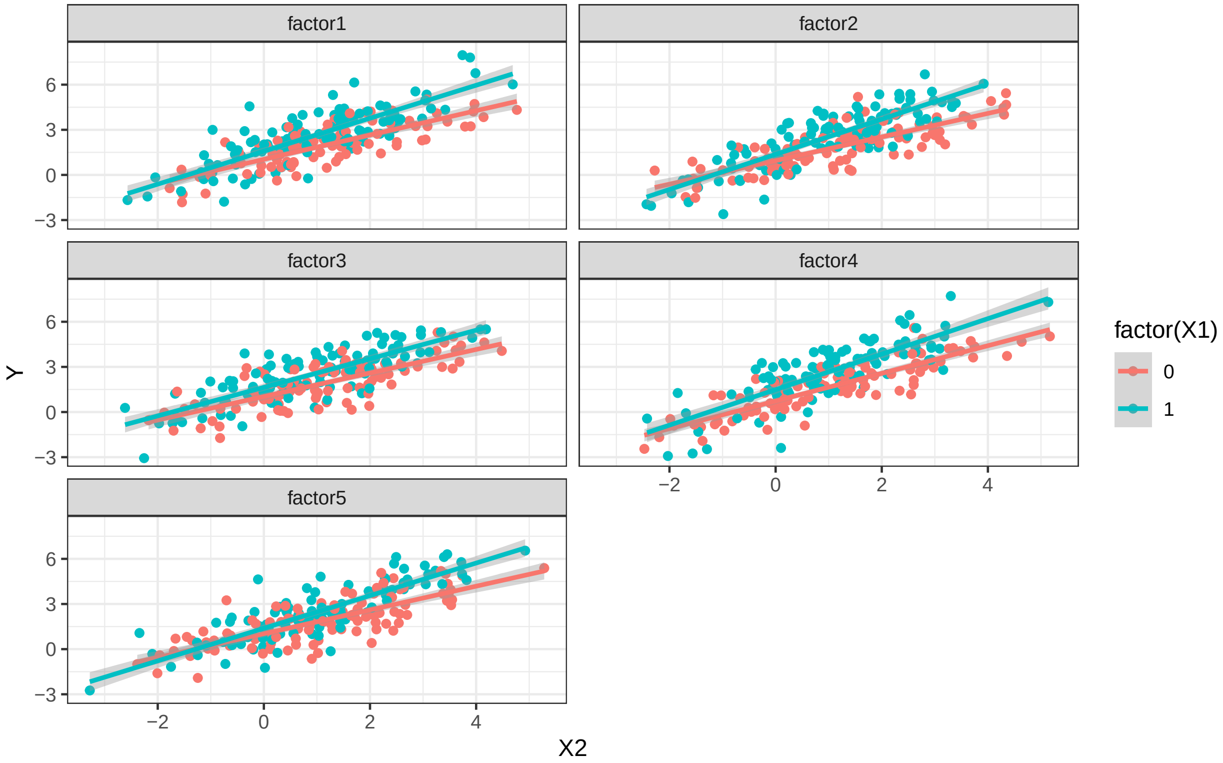
Next, we look at the same relationship as above with the binary predictor included as well. We use the lm methos since we know the data are linear.

```
ggplot(dt, aes(X2, Y, col = factor(X1)))+
  geom_point()+
  stat_smooth(method = "lm") +
  theme_bw()
```



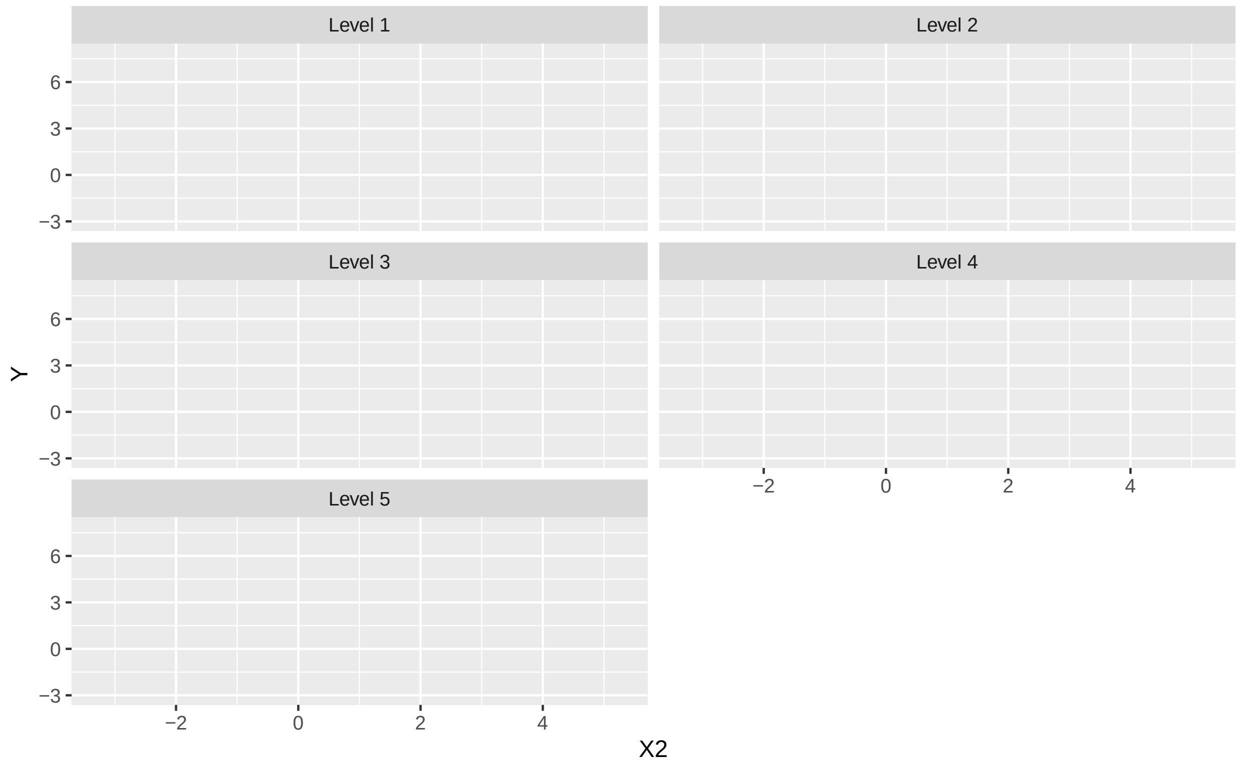
One way we may go about including the other variable is to use a facet wrap. A facet wrap in this case creates five plots on one screen which show the same relationship as above for each factor level.

```
#?facet_wrap
ggplot(dt, aes(X2, Y, col = factor(X1)))+
  facet_wrap(~X3, nrow = 3, ncol = 2)+
  geom_point()+
  stat_smooth(method = "lm") +
  theme_bw()
```



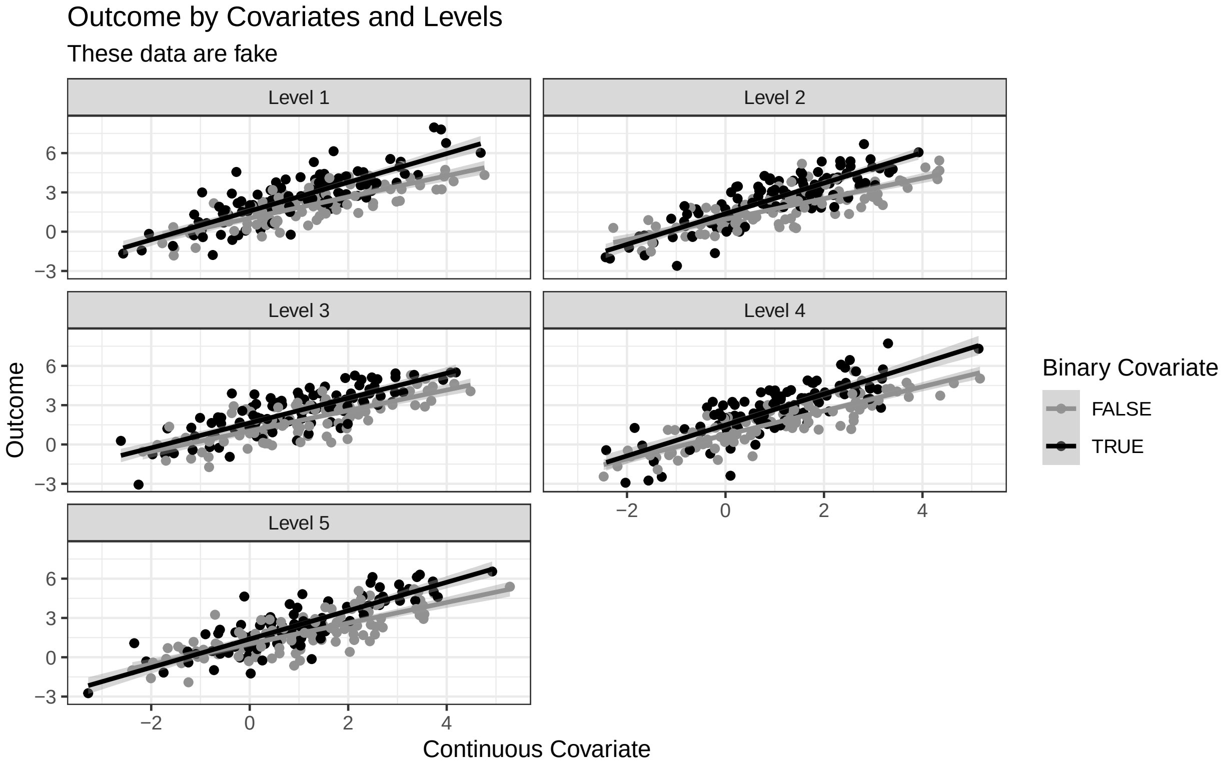
Another way to do this if we wanted to build a better looking plot would be to set the facets and labels for each plot first, then adding out actual plotting commands. The lines below sohuld display blank plots.

```
f2 <- ggplot(dt, aes(x = X2, y = Y, col = factor(X1)))
f2 + facet_wrap(~X3, nrow = 3, ncol = 2,
               labeller = as_labeller(c("factor1" = "Level 1",
                                        "factor2" = "Level 2",
                                        "factor3" = "Level 3",
                                        "factor4" = "Level 4",
                                        "factor5" = "Level 5")))
```



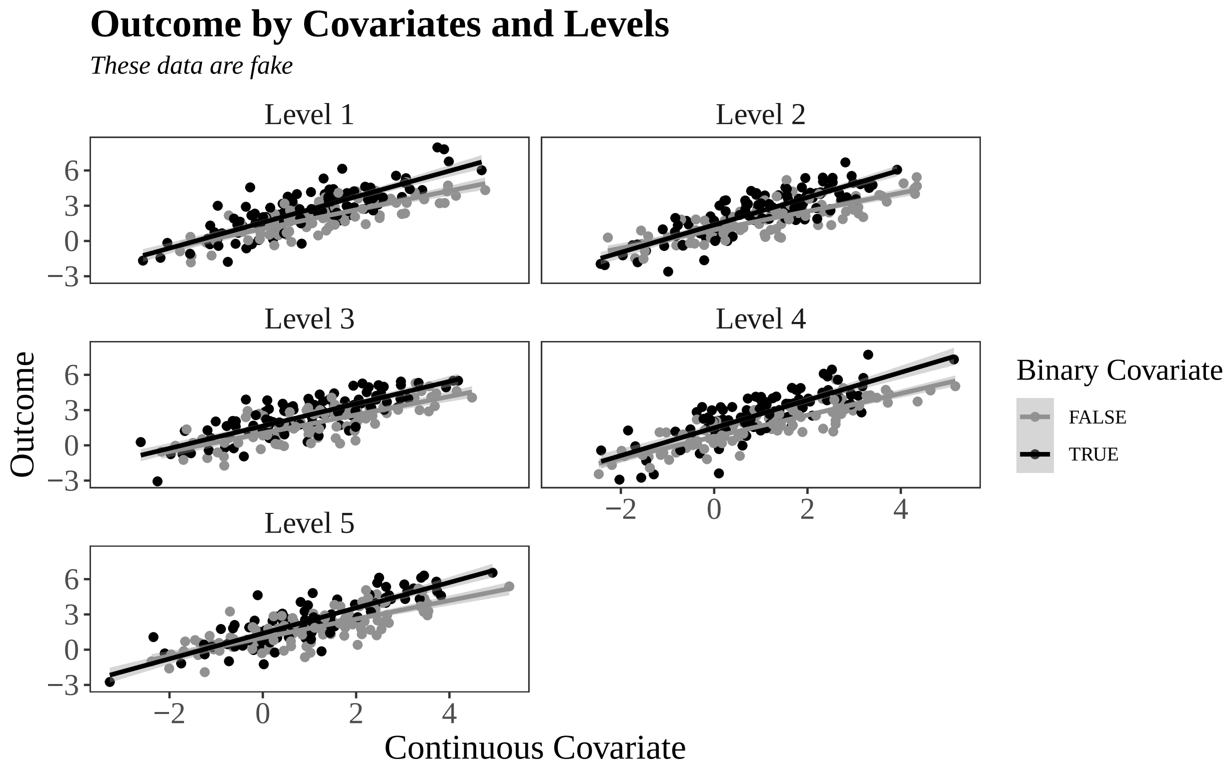
The code below is an example of a basic plot you would present for something.

```
f2 + facet_wrap(~X3, nrow = 3, ncol = 2,
                labeller = as_labeller(c("factor1" = "Level 1",
                                         "factor2" = "Level 2",
                                         "factor3" = "Level 3",
                                         "factor4" = "Level 4",
                                         "factor5" = "Level 5"))) +
  geom_point()+
  stat_smooth(method = "lm")+
  labs(x = "Continuous Covariate", y = "Outcome",
       title = "Outcome by Covariates and Levels",
       subtitle = "These data are fake") +
  scale_color_manual(name = "Binary Covariate",
                     breaks = c(0,1),
                     values = c("gray57","gray0"),
                     labels = c("FALSE","TRUE"))+
  theme_bw()
```



If we wanted a standalone image for publication for instance, we could add some more aesthetics such as below.

```
f2 + facet_wrap(~X3, nrow = 3, ncol = 2,
              labeller = as_labeller(c("factor1" = "Level 1",
                                       "factor2" = "Level 2",
                                       "factor3" = "Level 3",
                                       "factor4" = "Level 4",
                                       "factor5" = "Level 5"))) +
geom_point()+
stat_smooth(method = "lm")+
labs(x = "Continuous Covariate", y = "Outcome",
     title = "Outcome by Covariates and Levels",
     subtitle = "These data are fake") +
scale_color_manual(name = "Binary Covariate",
                   breaks = c(0,1),
                   values = c("gray57","gray0"),
                   labels = c("FALSE","TRUE"))+
theme_bw() +
theme(text = element_text(family = "serif"),
      plot.title = element_text(size = 18, face = "bold"),
      plot.subtitle = element_text(size = 12, face = "italic"),
      axis.text = element_text(size = 14),
      axis.title = element_text(size = 16),
      strip.text = element_text(size = 14),
      strip.background = element_blank(),
      legend.title = element_text(size = 14),
      panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```



Applied Example

This applied example is from a recent piece that I worked on with Vlad and Dan and is a practical example of a publishable figure. The replication data are open source and I will include them in the recitation folder if you wish to use them.

```
library(lfe)
library(dplyr)
library(dotwhisker)
rm(list=ls())
setwd("C:/Users/spw51/OneDrive/Desktop")
afro.master <- read.csv("autocratic_rule_data_replication.csv")
afro.master <- afro.master%>%
  rename(round = ï..round)
```

The plot we will eventually end up with is a dot and whisker coefficient plot by a few subgroups in our sample. We begin with setting our variable lists.

```
iv_list <- c("centralization", "TSI", "centralization*col_britain")

control_list_grid_pre <- "wateraccess+soil_quality+abslat+mnt2000+avgprec+prec2+avgtemp+meanrh+malaria_
control_list_grid_post <- "+capdist+bdist1+conflict+lpop+loglightsavg+polity+land_area+slave_exports+in
control_list_indiv_pre <- "+female+age"
```

```r
control_list_indiv_post <- "+female+age+education+employed+urban"

dv_list <- c("big_man_ord", "centralization")
```

Then define out formulas for fixed effect regressions.

```r
# model formulas
fmla1 <- as.formula(paste(paste(dv_list[1],"~"),
                          paste(iv_list[1]),
                          paste("| round + country | 0 | NAME")))



fmla2 <- as.formula(paste(paste(dv_list[1],"~"),
                          paste(iv_list[1], "+"),
                          paste(control_list_grid_pre),
                          paste(control_list_grid_post),
                          paste(control_list_indiv_pre),
                          paste(control_list_indiv_post),
                          paste("| round + country | 0 | NAME")))
```

We then create a set of subsets for previous colonial experience and estimate the coefficients for both subsets across both models.

```r
British <- subset(afro.master, afro.master$col_britain==1)

French <- subset(afro.master, afro.master$col_france==1)

# estimate coefficients for French colonies
m1 <- felm(fmla1, data=French)
m2 <- felm(fmla2, data=French)

# estimate coefficients for British colonies
m3 <- felm(fmla1, data=British)
m4 <- felm(fmla2, data=British)
```

We then repeat a few times for the other subgroups.

```r
# different degrees of contact with traditional leaders
no.contact <- subset(afro.master, afro.master$contact_trad==0)

contact <- subset(afro.master, afro.master$contact_trad==1)


# estimate coefficients for people with no contact
m5 <- felm(fmla1, data=no.contact)
m6 <- felm(fmla2, data=no.contact)

# estimate coefficients for people with contact
m7 <- felm(fmla1, data=contact)
m8 <- felm(fmla2, data=contact)
```

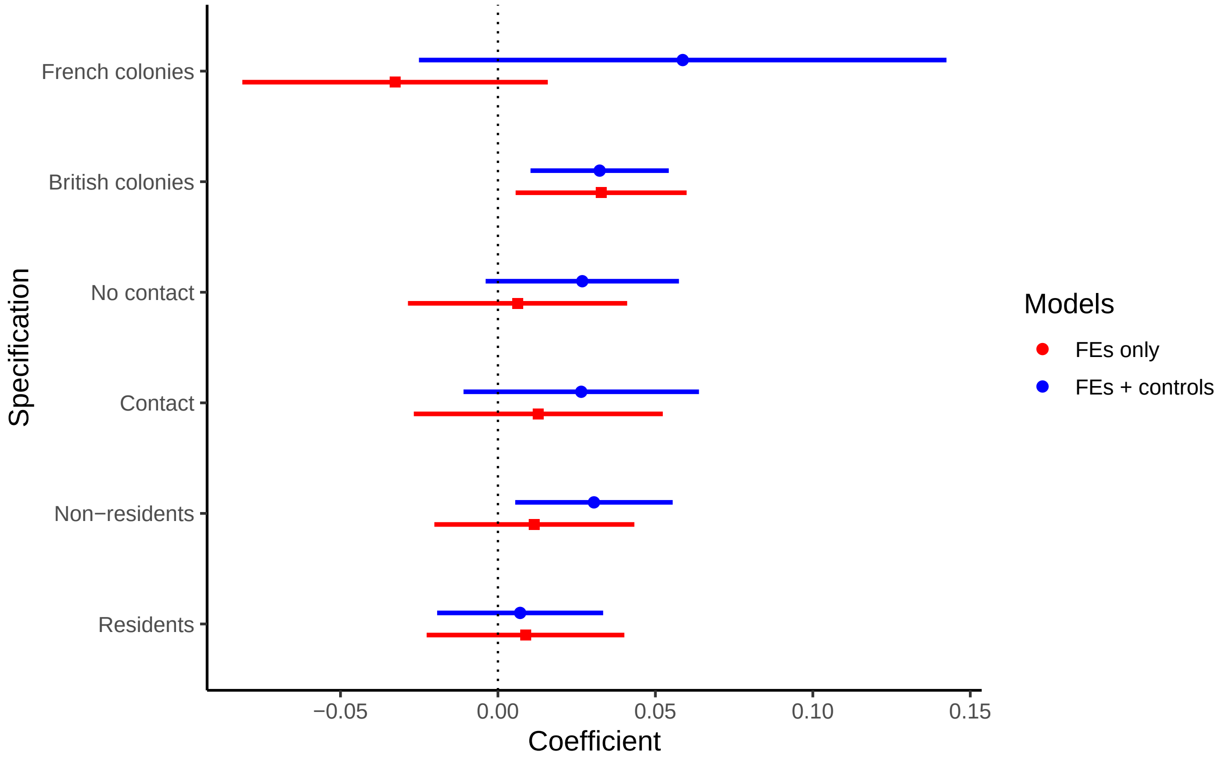
```
# comparing residents and non-residents of "ancestral" areas
nonresidents <- subset(afro.master, afro.master$resident==0)

residents <- subset(afro.master, afro.master$resident==1)


# estimate coefficients for nonresidents
m9 <- felm(fmla1, data=nonresidents)
m10 <- felm(fmla2, data=nonresidents)

# estimate coefficients for residents
m11 <- felm(fmla1, data=residents)
m12 <- felm(fmla2, data=residents)
```

Once we get the model estimates, we create objects for the term names as they should appear in the plot. Next, we gather the coefficients from the models. Then we grab the standard errors. Then we create a list of the model order for the plot. We then bind all of those together into a data frame to call into the plot command. Finally, we define the variable class of the objects. One of the reasons this was necessary for us is the object type class that felm gives and the package for plotting do not work well together and extracting manually ensures accuracy in this case.

```
# visualization via a coefficient plot (Figure 3)
term <- c("French colonies", "French colonies", "British colonies", "British colonies",
          "No contact", "No contact", "Contact", "Contact", "Non-residents", "Non-residents",
          "Residents", "Residents")

estimate <- c(m1$coefficients[1], m2$coefficients[1], m3$coefficients[1], m4$coefficients[1],
              m5$coefficients[1], m6$coefficients[1], m7$coefficients[1], m8$coefficients[1],
              m9$coefficients[1], m10$coefficients[1], m11$coefficients[1], m12$coefficients[1])

std.error <- c(m1$cse[1], m2$cse[1], m3$cse[1], m4$cse[1], m5$cse[1], m6$cse[1], m7$cse[1],
               m8$cse[1], m9$cse[1], m10$cse[1], m11$cse[1], m12$cse[1])

model <- c("Model 1", "Model 2", "Model 1", "Model 2", "Model 1", "Model 2", "Model 1", "Model 2",
           "Model 1", "Model 2", "Model 1", "Model 2")

data <- as.data.frame(cbind(term, estimate, std.error, model))

data$term <- as.character(data$term)
data$estimate <- as.numeric(data$estimate)
data$std.error <- as.numeric(std.error)
```

Below is the plotting code to produce the figure. The package we used is built from gpglot but is made for dot and whisker plots to look more like Stata or marginal effects plots since they are just easier and better looking in our opinion than the chunkier base ggplot ones. One of the reasons ours is more bare than the above example is for publications they sometimes want no titles and cleaner images if their editors compile the article separately from how you format it for submission (which is often).

```
dwplot(data, color = model, dot_args = (list(size = 2, pch = c(19, 15,
                                                               19, 15,
                                                               19, 15,
                                                               19, 15,
                                                               19, 15,
                                                             19, 15))),
     whisker_args =  list(lwd = 1)) +
labs(x = "Coefficient",
     y = "Specification") +
scale_color_manual(name = "Models", values = c("Model 1" = "red", "Model 2" = "blue"),
                   labels = c("Model 1" = "FEs only", "Model 2" = "FEs + controls")) +
theme_classic(base_size = 13 ) +
geom_vline(xintercept = 0, colour = "black", linetype = 3)
```


French colonies
British colonies
No contact
Contact
Non−residents
Residents
Specification
Coefficient
Models
● FEs only
● FEs + controls