Lab 8: Regression Mechanics

Sidak Yntiso sgy210@nyu.edu

March 30, 2020

Regression mechanics

Gauss Markov assumptions

- Linearity in parameters
- Full rank regression matrix (variation in X)
- Zero conditional mean of the errors ($\mathbb{E}[\epsilon_i|X] = 0$)

Regression mechanics

Gauss Markov assumptions

- Linearity in parameters
- Full rank regression matrix (variation in X)
- Zero conditional mean of the errors ($\mathbb{E}[\epsilon_i|X] = 0$)
- Conditional independence of errors ($Cov(\epsilon_i, \epsilon_j|X) = 0$)
- Homoskedasticity of the errors ($Var(\epsilon_i|X) = \sigma^2$)

Regression mechanics

Gauss Markov assumptions

- Linearity in parameters
- Full rank regression matrix (variation in X)
- Zero conditional mean of the errors ($\mathbb{E}[\epsilon_i|X] = 0$)
- Conditional independence of errors ($Cov(\epsilon_i, \epsilon_j|X) = 0$)
- Homoskedasticity of the errors ($Var(\epsilon_i|X) = \sigma^2$)

Effect hetereogeneity in multiple regression

- illustration of effective samples

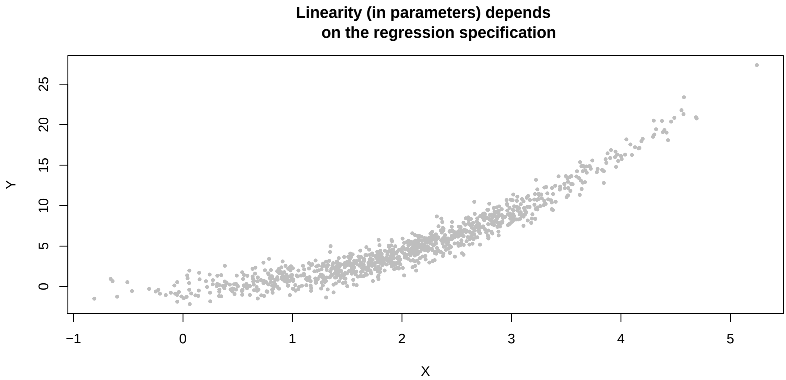
Linearity in parameters

```
rm(list=ls())
N <- 1000; set.seed(123)
X <- rnorm(N,2); Y <- X^2 + rnorm(N)
plot(   X,Y, pch=19, cex=.5,col="gray",
      main="Linearity (in parameters) depends
      on the regression specification")
```

```
p <- recordPlot()
```

Plot Data

```
p
```



Linearity (in parameters) depends
on the regression specification
Y
0
5
10
15
20
25
X
−1
0
1
2
3
4
5

CEF not linear in parameters for this specification

```
fit1 <- lm(Y~X)
```

CEF not linear in parameters for this specification

```r
fit1 <- lm(Y~X)
```

```r
p; points(  X,predict(fit1), type="l", col="red", lwd=2)
text(   5,13, expression(X[1]*beta[1]), col="red")
```

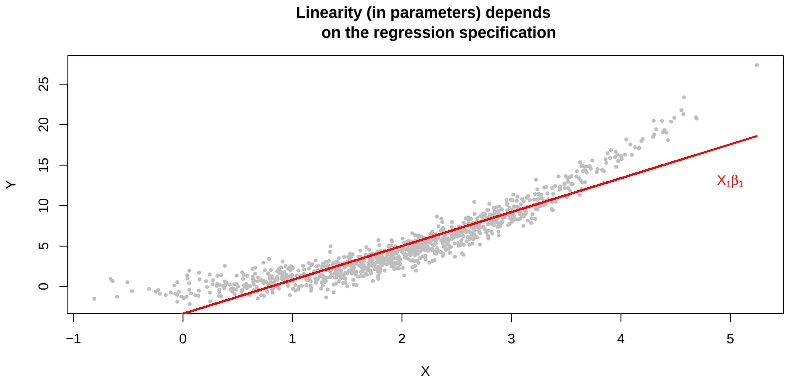
```r
round(mean(residuals(fit1)*X),5) # Orthogonality
```

```
## [1] 0
```

```r
p <- recordPlot()
```

Plot Data

```
p
```



Linearity (in parameters) depends
on the regression specification
Y
X
$X_1\beta_1$
0
5
10
15
20
25
−1
0
1
2
3
4
5

Implies non-zero conditional mean of residual over X

```
X.example <- (abs(X-4.05))==min(abs(X-4.05))
resid.example <- mean(residuals(fit1)[X>=4&X<=4.1])
y1 = mean(residuals(fit1)[X>=4&X<=4.1])+
  predict(fit1)[X.example]
```

Implies non-zero conditional mean of residual over X

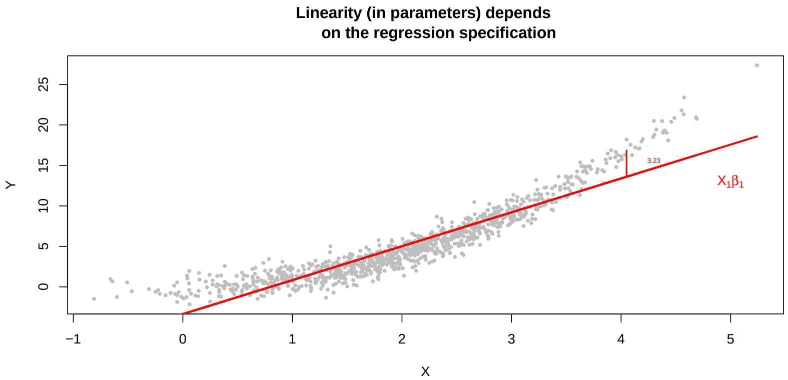
```
X.example <- (abs(X-4.05))==min(abs(X-4.05))
resid.example <- mean(residuals(fit1)[X>=4&X<=4.1])
y1 = mean(residuals(fit1)[X>=4&X<=4.1])+
  predict(fit1)[X.example]
```

```
p; segments(X[X.example], predict(fit1)[X.example],
          X[X.example], y1, col="red",lwd=2)
text(4.3, predict(fit1)[X.example]+2,
      round(resid.example,2), col="red", cex=.5)
```

```
p <- recordPlot()
```

Plot Data

```
p
```



Linearity (in parameters) depends
on the regression specification
Y
0
5
10
15
20
25
−1
0
1
2
3
4
5
X
3.23
X₁β₁

CEF linear in parameters for this specification

```
fit2 <- lm(Y~X+I(X^2))
X.ord <- X[order(X)]
Y.ord <- predict(fit2)[order(X)]
```

CEF linear in parameters for this specification

```r
fit2 <- lm(Y~X+I(X^2))
X.ord <- X[order(X)]
Y.ord <- predict(fit2)[order(X)]
```

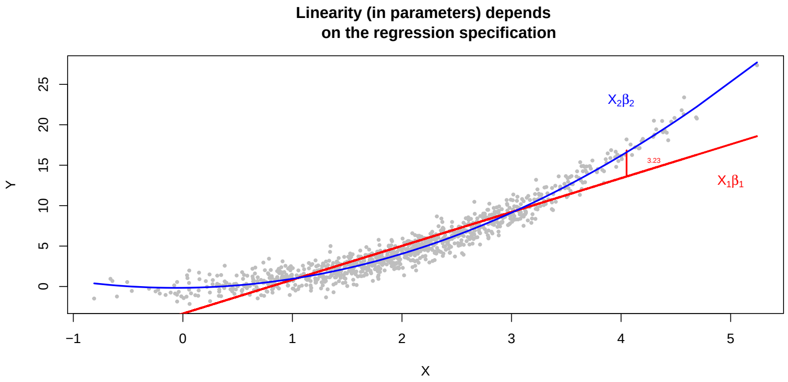
```r
p; points(  X.ord, Y.ord, type="l", col="blue", lwd=2)
text(   4, 23, expression(X[2]*beta[2]), col="blue")
```

```r
p <- recordPlot()
round(mean(residuals(fit2)*X),5) # Orthogonality
```

```
## [1] 0
```

Plot Data

```
p
```



Linearity (in parameters) depends
on the regression specification
Y
X
$X_2\beta_2$
$X_1\beta_1$
3.23
0
5
10
15
20
25
−1
0
1
2
3
4
5

The intercept

- The `lm()`, `lm_robust()` functions by default include an intercept. Why?

The intercept

- The `lm()`, `lm_robust()` functions by default include an intercept. Why?

- In textbooks, zero conditional mean of the errors often coupled with zero expectation of error assumption: $\mathbb{E}[\epsilon_i] = 0$
- The latter can always be assumed to be zero in the linear regression model so long as the intercept is included in the model.
- Let's illustrate by removing the intercept using the -1 syntax

Illustration

```
fit3 <- lm(Y~-1+X+I(X^2))
```

Illustration

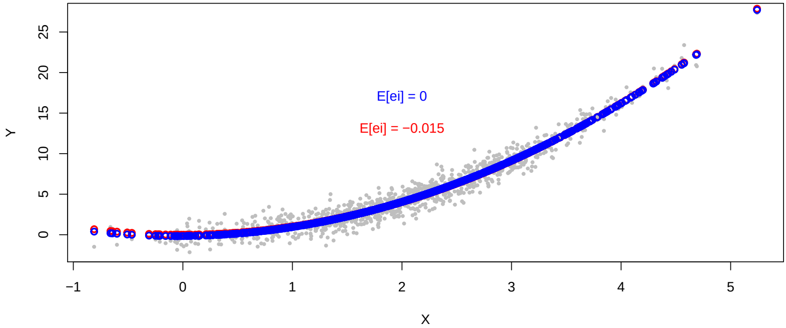
```r
fit3 <- lm(Y~-1+X+I(X^2))
```

```r
r3 <- round(mean(residuals(fit3)),3)
r2 <- round(mean(residuals(fit2)),3)
plot(X,Y, pch=19, cex=.5,col="gray")
points( X,predict(fit3), type="p", col="red", lwd=2)
text(2,13, paste(expression(E[ei]),"=",r3), col="red")
points( X,predict(fit2), type="p", col="blue", lwd=2)
text(2,17, paste(expression(E[ei]),"=",r2), col="blue")
```

```r
p <- recordPlot()
```

Plot Data

```
p
```



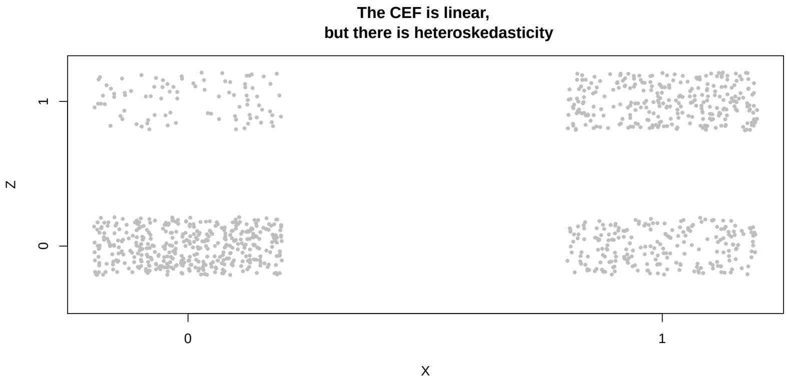
Homoskedasticity example

```
rm(list=ls())
set.seed(12345); N <- 1000
X <- c(rep(1,N/2), rep(0,N/2)); P <- .2 + .3*X
Z <- rbinom(N,1, P) # Let Z = P(X) + u, but Z=0,1.
plot(jitter(X), jitter(Z),pch=19,cex=.5,axes=F,
      xlab="X",ylab="Z", main="The CEF is linear,
      but there is heteroskedasticity",
      col="gray",ylim=c(-.4,1.25))
axis(1, c(0,1)); axis(2, c(0,1)); box()
```

```
p <- recordPlot()
```

Illustration

```
p
```



Differences in variances

```
fit <- lm(Z~X)
v0 <- var(residuals(fit)[X==0]);
v1 <- var(residuals(fit)[X==1])
print(v0); print(v1)
```

```
## [1] 0.1466092
```

```
## [1] 0.2475792
```

```
fit2 <- estimatr::lm_robust(Z~X)
v20 <- var(Z-predict(fit2)[X==0]);
v21 <- var(Z-predict(fit2)[X==1])
print(v20); print(v21)
```

```
## [1] 0.2322763
```

```
## [1] 0.2322763
```

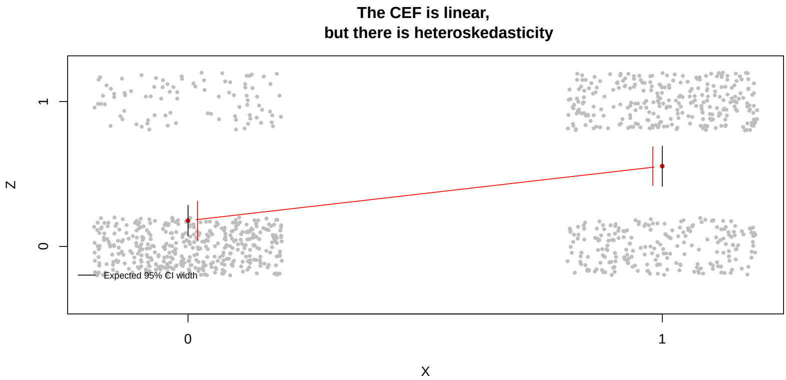
Plot difference

```
p; points(X, predict(fit), pch=19, col="red",
          type="b", cex=.5)
segments(0,mean(predict(fit)[X==0])+1.96*sqrt(v0/50),
          0,mean(predict(fit)[X==0])-1.96*sqrt(v0/50),
          col="black")
segments(1,mean(predict(fit)[X==1])+1.96*sqrt(v1/50),
          1,mean(predict(fit)[X==1])-1.96*sqrt(v1/50),
          col="black")
segments(0.02,mean(predict(fit2)[X==0])+1.96*sqrt(v20/50),
          0.02,mean(predict(fit2)[X==0])-1.96*sqrt(v20/50),
          col="red")
segments(0.98,mean(predict(fit2)[X==1])+1.96*sqrt(v21/50),
          0.98,mean(predict(fit2)[X==1])-1.96*sqrt(v21/50),
          col="red")
legend(-.25,-.15, legend="Expected 95% CI width",
       lty="solid", bty="n", cex=.7)
```

```
p <- recordPlot()
```

Illustration

```
p
```



What if we have heterogenous treatment effects?

- Recall the ATE is a weighted sum of conditional ATEs:
- $ATE = \sum_x \tau_x Pr[X_i = x]$; where $\tau_x = E[Y_i(1) - Y_i(0)|X_i = x]$
- Similar derivation for the $ATT = \sum_x \tau_x Pr[X_i = x|D_i = 1]$;

What if we have heterogenous treatment effects?

- Recall the ATE is a weighted sum of conditional ATEs:
- $ATE = \sum_x \tau_x Pr[X_i = x]$; where $\tau_x = E[Y_i(1) - Y_i(0)|X_i = x]$
- Similar derivation for the $ATT = \sum_x \tau_x Pr[X_i = x|D_i = 1]$;
- Using Bayes rule $Pr[X_i = x|D_i = 1] = \frac{Pr[D_i=1|X_i=x]Pr[X_i=x]}{\sum_x Pr[D_i=1|X_i=x]Pr[X_i=x]}$
- Which means that the ATT is a propensity-score weighted function of the CATEs: $ATT = \frac{\sum_x \tau_x Pr[D_i=1|X_i=x]Pr[X_i=x]}{\sum_x Pr[D_i=1|X_i=x]Pr[X_i=x]}$

OLS also weighted average of CATEs but use different weights

$$Y_i = \sum_x D_{xi}\alpha_x + \tau_R D_i + e_i$$

$$\tau_R = \frac{Cov(Y_i, \tilde{D}_i)}{V(\tilde{D}_i)}$$

where $\tilde{D}_i$ is residual from regression: $D_i = \sum_x D_{xi}\beta_x + \tilde{D}_i$

OLS also weighted average of CATEs but use different weights

$$Y_i = \sum_x D_{xi}\alpha_x + \tau_R D_i + e_i$$

$$\tau_R = \frac{Cov(Y_i, \tilde{D}_i)}{V(\tilde{D}_i)}$$

where $\tilde{D}_i$ is residual from regression: $D_i = \sum_x D_{xi}\beta_x + \tilde{D}_i$

$$\begin{aligned}\tau_R &= \frac{Cov(E[Y_i|X_i, D_i], D_i - E[D_i|X_i])}{V(D_i - E[D_i|X_i])} \\ &= \frac{E[E[Y_i|X_i, D_i](D_i - E[D_i|X_i])]}{E[(D_i - E[D_i|X_i])^2]}\end{aligned}$$

Simplify the CEF

$$
\begin{aligned}
E[Y_i|X_i, D_i] &= E[D_i Y_i(1) + (1 - D_i) Y_i(0)|X_i, D_i] \\
&= E[Y_i(0)|X_i, D_i = 0] + D_i E[Y_i(1) - Y_i(0)|X_i, D_i] \\
&= E[Y_i|X_i, D_i = 0] + \tau(X_i) D_i
\end{aligned}
$$

Simplify the CEF

$$
\begin{aligned}
E[Y_i|X_i, D_i] &= E[D_i Y_i(1) + (1 - D_i)Y_i(0)|X_i, D_i] \\
&= E[Y_i(0)|X_i, D_i = 0] + D_i E[Y_i(1) - Y_i(0)|X_i, D_i] \\
&= E[Y_i|X_i, D_i = 0] + \tau(X_i)D_i
\end{aligned}
$$

Substitute $E[Y_i|X_i, D_i]$:

$$
\begin{aligned}
\tau_R &= \frac{E[E[Y_i|X_i, D_i](D_i - E[D_i|X_i])]}{E[(D_i - E[D_i|X_i])^2]} \\
&= \frac{E[\tau(X_i)D_i(D_i - E[D_i|X_i])]}{E[(D_i - E[D_i|X_i])^2]} \\
&= \frac{E[\tau(X_i)(D_i^2 - D_i E[D_i|X_i])]}{E[(D_i - E[D_i|X_i])^2]}
\end{aligned}
$$

Putting it altogether

$$
\begin{aligned}
E[D_i^2 - D_i E[D_i|X_i]] &= E[(D_i|X_i = x)^2] - (E[D_i|X_i = x])^2 \\
&= Var(D_i|X_i = x) \\
&= \Pr[D_i = 1|X_i](1 - \Pr[D_i = 1|X_i])
\end{aligned}
$$

Putting it altogether

$$\begin{aligned} E[D_i^2 - D_i E[D_i|X_i]] &= E[(D_i|X_i = x)^2] - (E[D_i|X_i = x])^2 \\ &= Var(D_i|X_i = x) \\ &= \Pr[D_i = 1|X_i](1 - \Pr[D_i = 1|X_i]) \end{aligned}$$

$$\tau_R = \frac{\sum_x \tau_x [\Pr[D_i = 1|X_i = x](1 - \Pr[D_i = 1|X_i = x])] \Pr[X_i = x]}{\sum_x [\Pr[D_i = 1|X_i = x](1 - \Pr[D_i = 1|X_i = x])] \Pr[X_i = x]}$$

Implications

Both weighted averages of CATEs:

- ATT aggregrates via population weighting
- OLS aggregrates via conditional variance weighting wrt D_i

Implications

Both weighted averages of CATEs:

- ATT aggregrates via population weighting
- OLS aggregrates via conditional variance weighting wrt D_i

OLS produces ATT if

- constant treatment effects $\tau_x = \tau$ for all X or
- unconditional independence

Implications

Both weighted averages of CATEs:

- ATT aggregrates via population weighting
- OLS aggregrates via conditional variance weighting wrt D_i

OLS produces ATT if

- constant treatment effects $\tau_x = \tau$ for all X or
- unconditional independence

Variance weighting is biased - it privileges X_i for which τ_x estimates are precise

- $Pr[D_i = 1|X_i = x](1 - \Pr[D_i = 1|X_i = x])$ is maximized when $Pr[D_i = 1|X_i = x] = 1/2$
- Regression weights produce an effective sample different from the observed sample

Effective Samples

- Let's check the properties of your effective sample in regression.
- The key result is: $\hat{\rho}_{reg} \xrightarrow{p} \frac{E[w_i \rho_i]}{E[w_i]}$
  - where $w_i = (D_i - E[D_i|X_i])^2$

Effective Samples

- Let's check the properties of your effective sample in regression.
- The key result is: $\hat{\rho}_{reg} \xrightarrow{p} \frac{E[w_i \rho_i]}{E[w_i]}$
  - where $w_i = (D_i - E[D_i|X_i])^2$
- We estimate these weights with: $\hat{w}_i = \hat{D}_i^2$
  - where D_i^2 is the ith squared residual.
  - Because these estimates are "bad" for each unit, using them to reweight the sample is a bad idea.

Example paper

How do people translate personal experiences into political attitudes?

- non-random assignment of social and economic phenomena
- Egan and Mullin 2013 focus on local weather shocks

The variables of interest are:

- `ddt_week_direction` - Treatment variable (1 if the normal local temperature (in Fahrenheit) in week prior to survey $>$ local average; 0 otherwise)
- `getwarmord` - Opinion on whether there is "solid evidence" for global warming i.e., the earth getting warmer (no $= 1$, mixed/some/don't know $= 2$, yes $= 3$).

Load in data

```
d <- haven::read_dta("gwdataset.dta")
zips <- haven::read_dta("zipcodetostate.dta")
zips<-unique(zips[,c("statenum","statefromzipfile")])
pops <- read.csv("population_ests_2013.csv")
pops$state <- tolower(pops$NAME)
d$getwarmord <- as.double(d$getwarmord)
d$treatment <- as.double(d$ddt_week >0 )
model = "educ_hsless+educ_coll+educ_postgrad+educ_dk+
  party_rep+party_leanrep+party_leandem+party_dem+
  male+raceeth_black+raceeth_hisp+raceeth_notwbh+
  raceeth_dkref+age_1824+age_2534+age_3544+age_5564+
  age_65plus+age_dk+as.factor(statenum)"
```

Base Model

- We won't worry about standard errors yet.

```
# And estimate primary model of interest:

out<-lm(paste("getwarmord ~ treatment+",model,sep=""),d)
summary(out)$coefficients[1:8,]
```

```
##                   Estimate Std. Error   t value      Pr(>
## (Intercept)     2.48053277 0.08563826 28.965241 7.596011e
## treatment       0.05010349 0.02230735  2.246053  2.473324
## educ_hsless     0.05473799 0.02370818  2.308823  2.098389
## educ_coll       0.02961485 0.02643154  1.120436  2.625684
## educ_postgrad   0.06678542 0.02996072  2.229100  2.584052
## educ_dk         0.18495725 0.24172677  0.765150  4.442093
## party_rep      -0.29899990 0.03383892 -8.835976  1.252564
## party_leanrep  -0.11410765 0.03985345 -2.863181  4.207206
```

Estimate D^2

- We can simply square the residuals of a partial regression to get D^2:

```
outD<-lm(paste("treatment ~",model,sep=""),d)
D2 <- residuals(outD)^2
```

Effective Sample Statistics

- We can use these estimated weights for examining the sample.

```
compare_samples<- d[,c("wave","treatment","raceeth_black",
                       "raceeth_hisp","party_rep",
                       "age_1824","educ_hsless")]
compare_samples <- apply(compare_samples,2,function(x)
  round(c(mean(x),sd(x),weighted.mean(x,D2),
    sqrt(weighted.mean((x-weighted.mean(x,D2))^2,D2))),3) )
compare_samples <- t(compare_samples)
colnames(compare_samples) <- c("Nominal Mean"," SD",
                               "Effective Mean", "SD")
```

Comparisons

```
compare_samples
```

```
##                   Nominal Mean    SD Effective Mean    SD
## wave                     3.097 1.425          2.437 1.441
## treatment                0.743 0.437          0.294 0.456
## raceeth_black            0.089 0.284          0.080 0.272
## raceeth_hisp             0.056 0.230          0.062 0.242
## party_rep                0.295 0.456          0.304 0.460
## age_1824                 0.072 0.258          0.071 0.257
## educ_hsless             0.342 0.474          0.344 0.475
```

Effective Sample Maps

- Where in the US does the effective sample emphasize?

```
# Effective sample by state
wt.by.state <- tapply(D2,d$statenum,sum)
wt.by.state <- wt.by.state/sum(wt.by.state)*100
wt.by.state <- cbind(D2=wt.by.state,
                     statenum=names(wt.by.state))
data_for_map <- merge(wt.by.state,zips,by="statenum")
# Nominal Sample by state
wt.by.state <- tapply(rep(1,6726),d$statenum,sum)
wt.by.state <- wt.by.state/sum(wt.by.state)*100
wt.by.state <- cbind(Nom=wt.by.state,
                     statenum=names(wt.by.state))
data_for_map <- merge(data_for_map,wt.by.state,by="statenum
```

Set up data

```
data(state.fips, package = "maps") #load maps
#merge maps with data
data_for_map <- merge(state.fips,data_for_map,by.x="abb",
                      by.y="statefromzipfile")
#convert factor columns into numeric
data_for_map$D2<-as.double(as.character(data_for_map$D2))
data_for_map$Nom<-as.double(as.character(data_for_map$Nom))
#recode state names
data_for_map$state <- sapply(
  as.character(data_for_map$polyname),function(x)
    strsplit(x,":")[[1]][1])
#merge populations with data
data_for_map <- merge(data_for_map,pops,by="state")
```

Set up data cntd

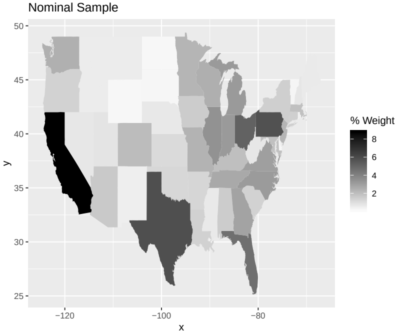
```
#Difference in weights
data_for_map$Diff <-
  data_for_map$D2 - data_for_map$Nom
data_for_map$PopPct <- data_for_map$POPESTIMATE2013/
  sum(data_for_map$POPESTIMATE2013)*100
data_for_map$PopDiffEff <-
  data_for_map$D2 - data_for_map$PopPct
data_for_map$PopDiffNom <-
  data_for_map$Nom - data_for_map$PopPct
data_for_map$PopDiff <-
  data_for_map$PopDiffEff - data_for_map$PopDiffNom
require(ggplot2,quietly=TRUE) #plotting package
state_map <- map_data("state")
```

More setup

```
plotbase <- ggplot(data_for_map,aes(map_id=state))+
  expand_limits(x = state_map$long,y = state_map$lat)+
  scale_fill_gradient2("% Weight",low = "red",
                        mid = "white", high = "black")

plotEff <- plotbase+geom_map(aes(fill=D2),map=state_map)+
  labs(title = "Effective Sample")

plotNom <- plotbase+geom_map(aes(fill=Nom),map=state_map)+
  labs(title = "Nominal Sample")

plotDiff <- plotbase+geom_map(aes(fill=Diff),map=state_map
  labs(title = "Effective Weight Minus Nominal Weight")
```

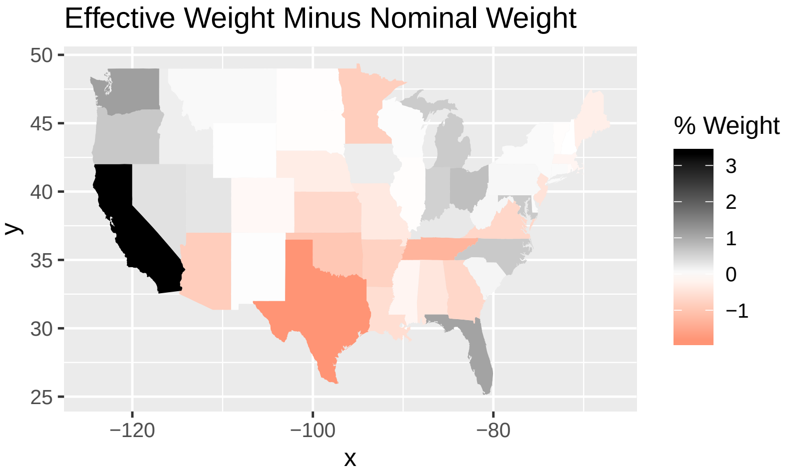
And the maps

```
require(gridExtra,quietly=TRUE)
grid.arrange(plotNom,plotEff,ncol=2)
```



Difference in Weights

```
plotDiff
```

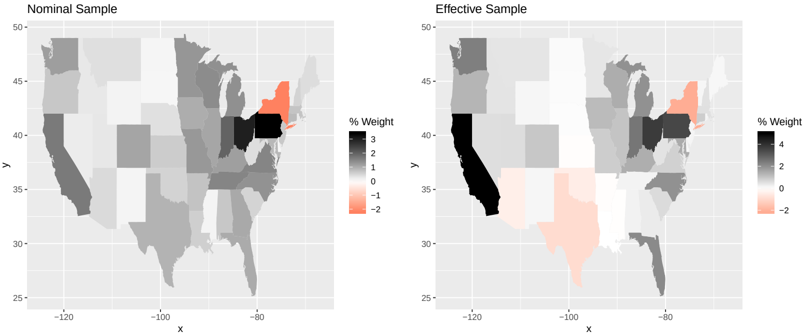


Population Comparison

```r
plotEff <- plotbase+
  geom_map(aes(fill=PopDiffEff),map=state_map)+
  labs(title = "Effective Sample")

plotNom <- plotbase +
  geom_map(aes(fill=PopDiffNom),map = state_map)+
  labs(title = "Nominal Sample")

plotDiff <- plotbase +
  geom_map(aes(fill=PopDiff),map = state_map)+
  labs(title = "Effective Weight Minus Nominal Weight")
```

Population Comparison Plots

```
grid.arrange(plotNom,plotEff,ncol=2)
```



Plot Difference

```
plotDiff
```