Advanced Topics in R

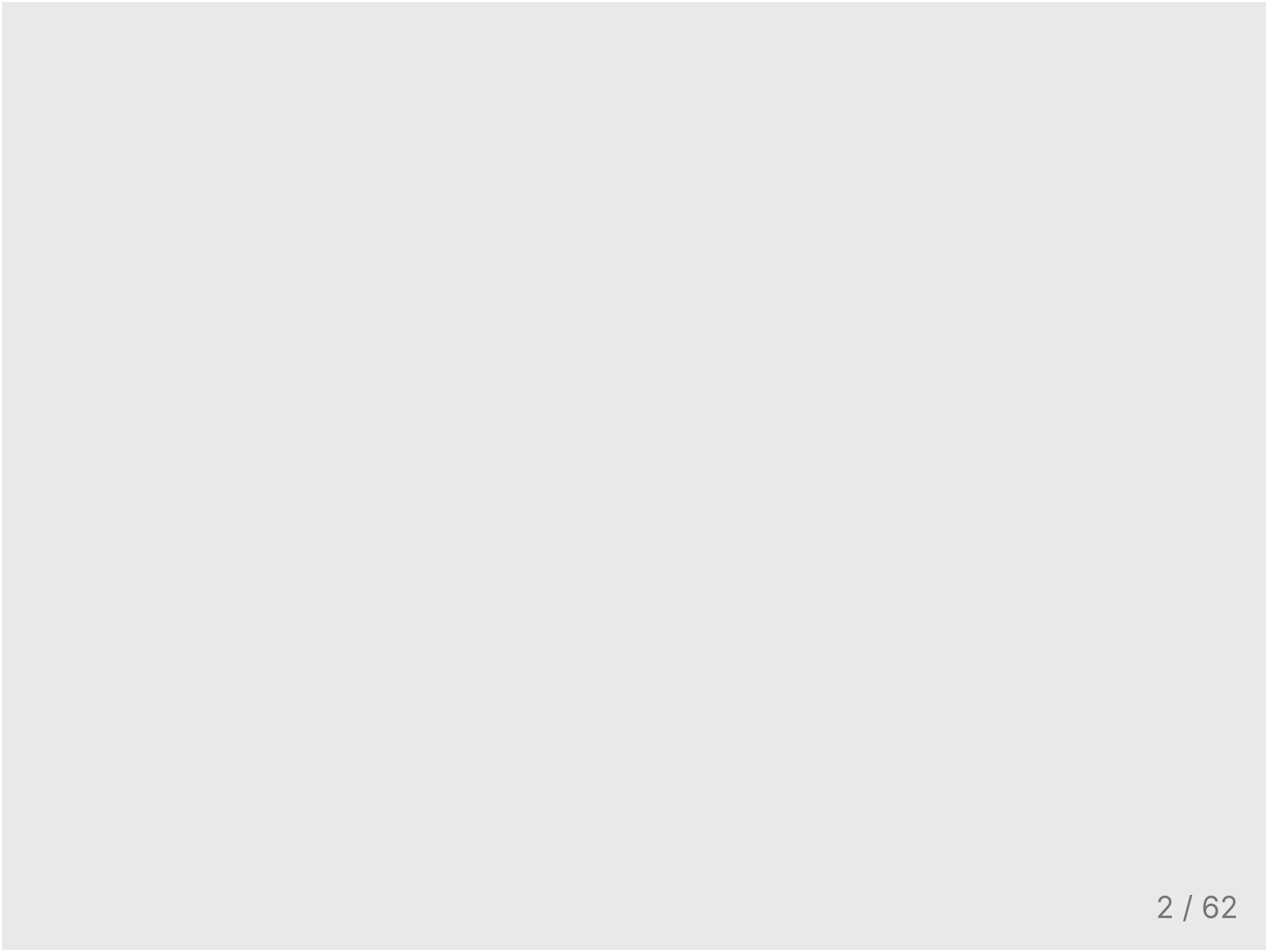
A Preview of What's to Come

Prof. Bisbee

Vanderbilt University

Lecture Date: 2023/04/17

Slides Updated: 2023-04-16

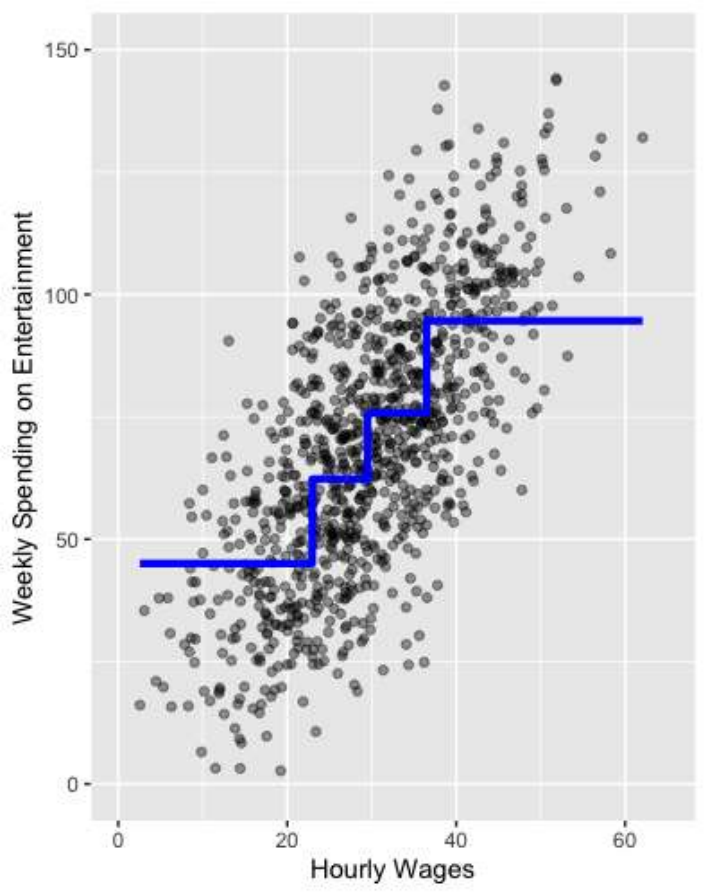

2 / 62

Agenda

- Machine Learning Algorithms
- Mapping in `R` (time permitting)

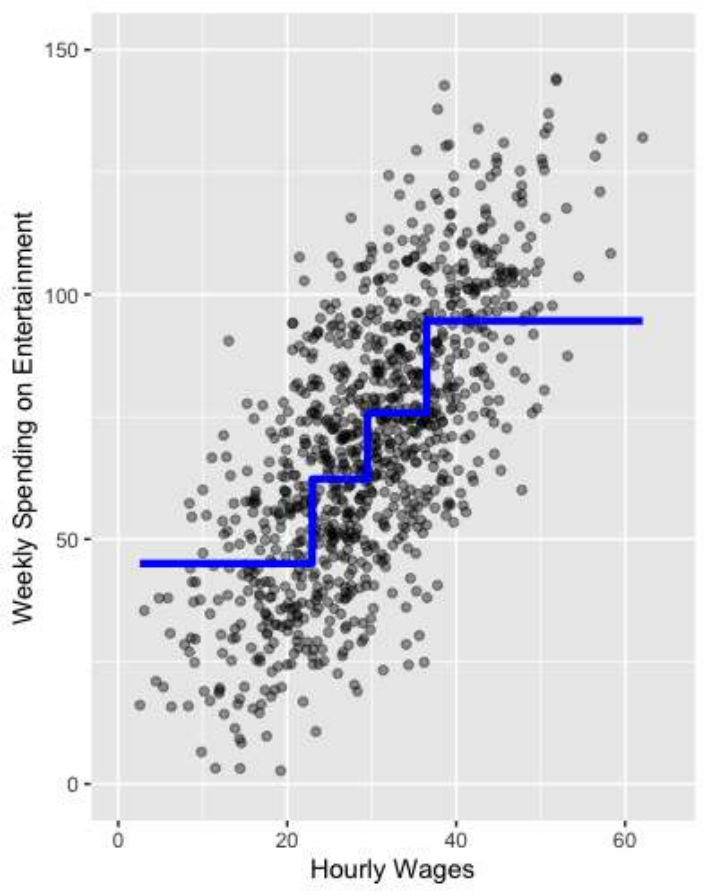
What is regression?

- Conditional means for continuous data



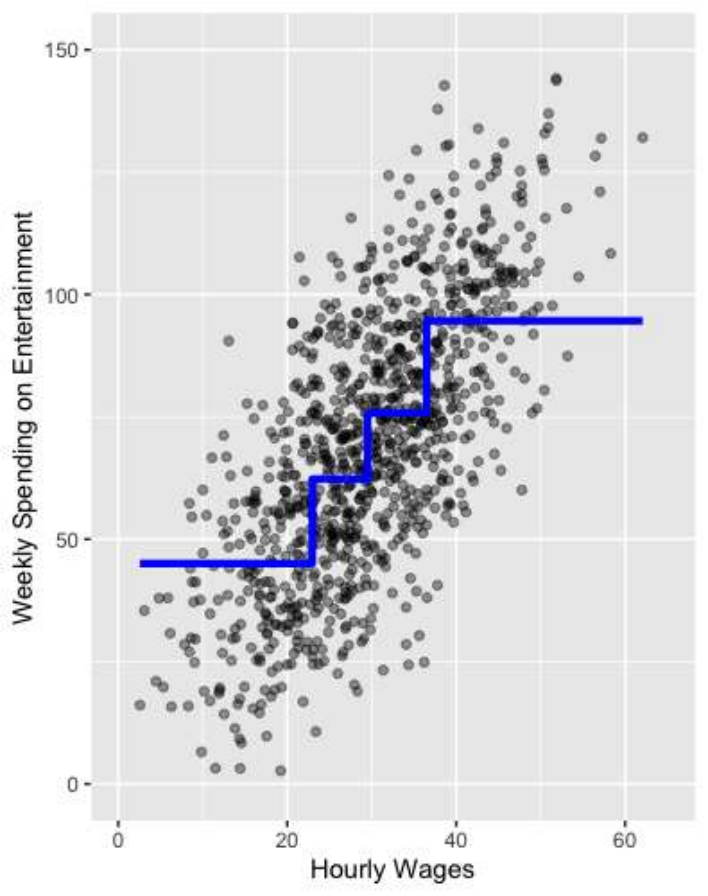
What is regression?

- Theory: the more you earn, the more you spend



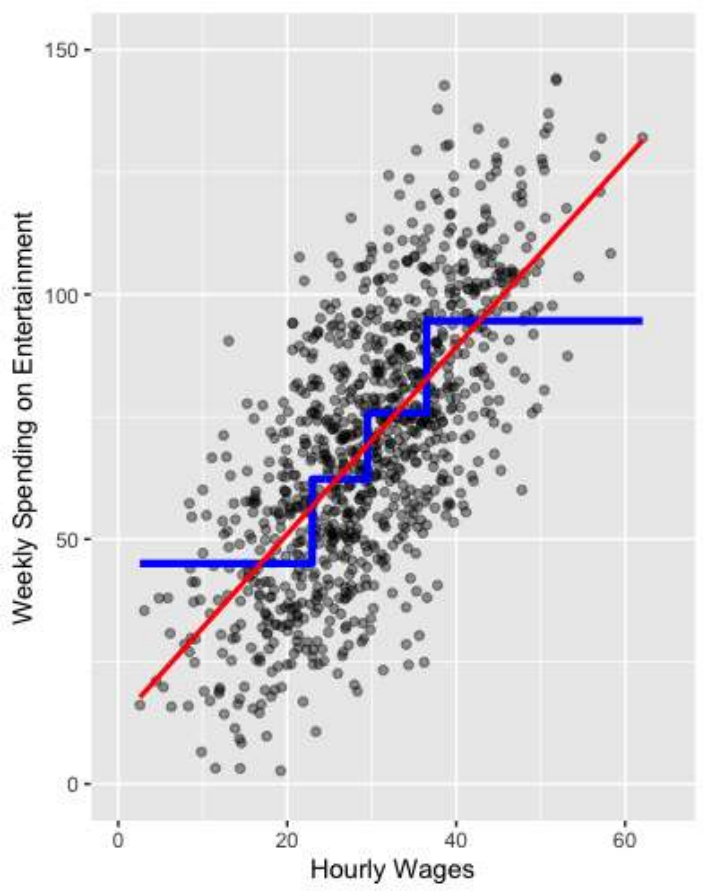
What is regression?

- But **conditional means** make a lot of mistakes. Can we do better?



What is regression?

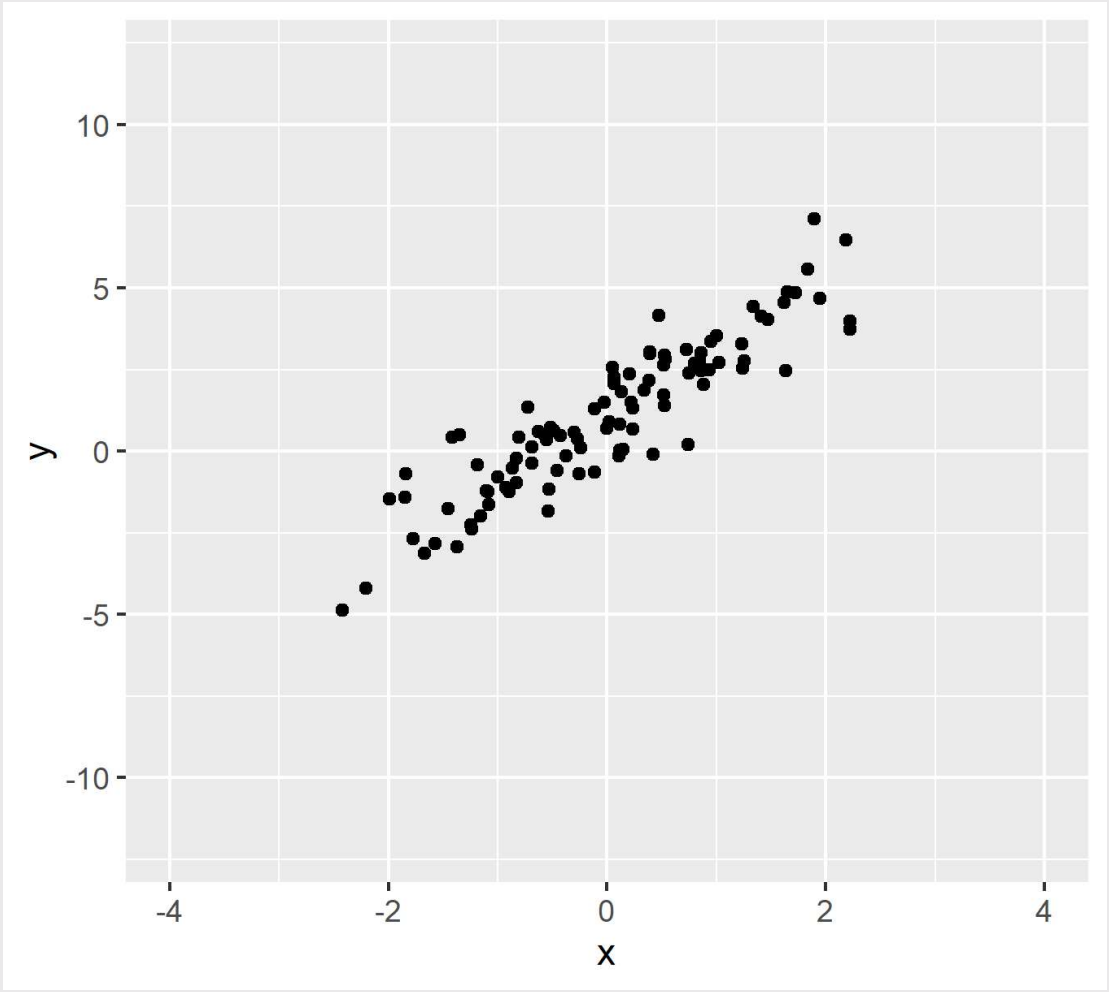
- But **conditional means** make a lot of mistakes. Can we do better?



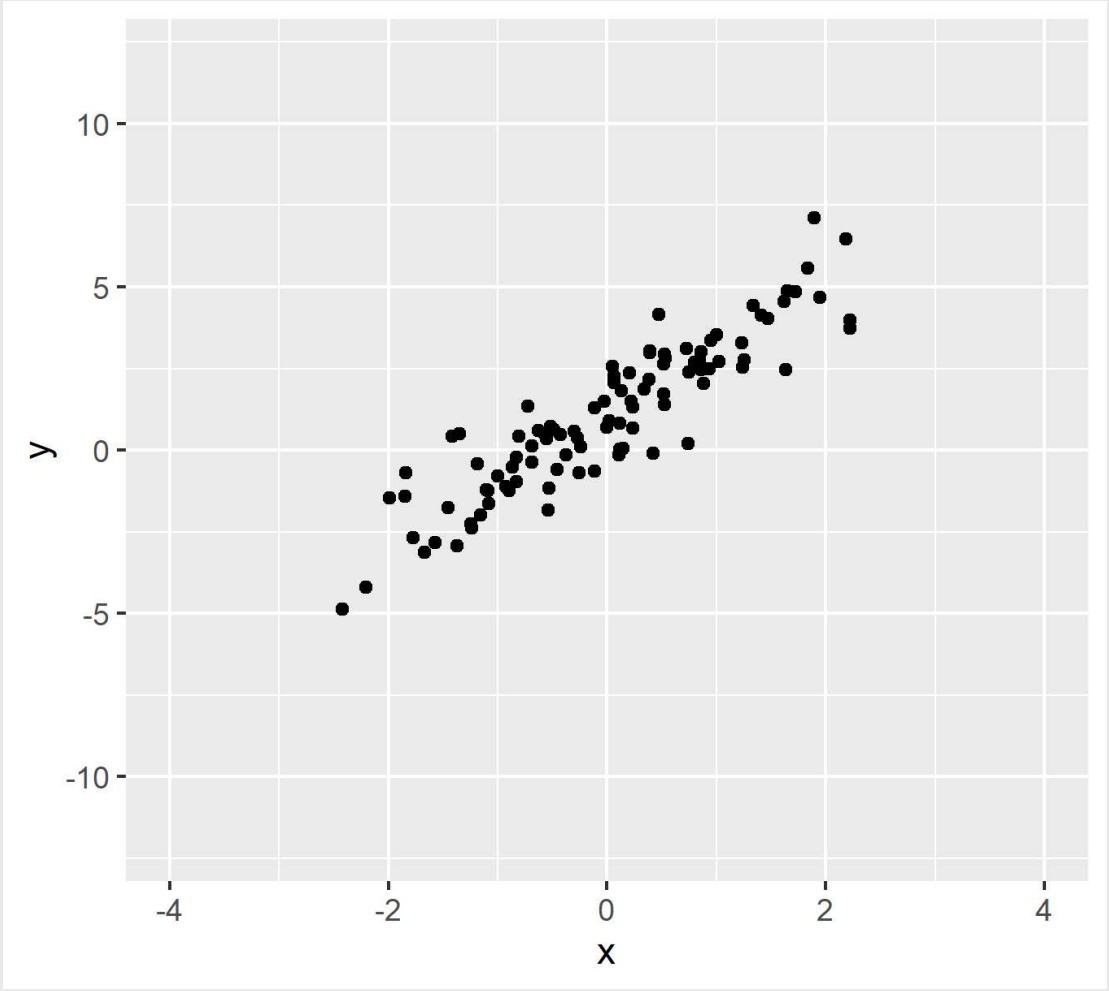
Regression

- Calculating a **line** that minimizes mistakes *for every observation*
  - NB: could be a curvey line! For now, just assume straight
- Recall from geometry how to graph a straight line
- $Y = a + bX$
  - a: the "intercept" (where the line intercepts the y-axis)
  - b: the "slope" (how much Y changes for each increase in X)
- (Data scientists use α and β instead of a and b b/c nerds)
- Regression analysis simply chooses the best line
  - "Best"?
  - The line that minimizes the mistakes (the **line of best fit**)

Visual Intuition



Visual Intuition



Regression

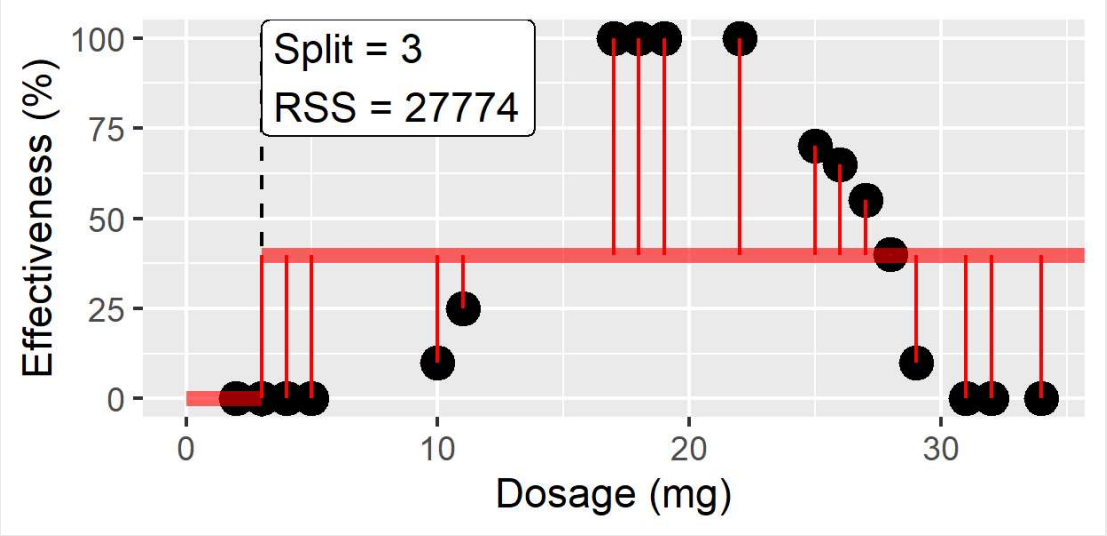
- The line is substantively meaningful
- Red line on scatter plot of spending and wages: $Y = 12 + 2 * X$
- α tells us the value of Y when X is zero
    - People who don't make any money spend $12 per week on entertainment
- β tells us how much Y increases for each additional X
    - People spend an additional $2 per week for each additional $1 in hourly wages

Two Camps Revisited

- Regression is great for **theory testing**
  - Results tell us something **meaningful** about our theory
- But if all we care about is **prediction**...?
  - Want to test every possible predictor (and combinations)
  - Don't care about **relationships**
  - Just care about **accuracy**
- Algorithms can save us time!
  - Random Forests
  - LASSO

Random Forests

- Identify the best "partition" (split) that divides the data



- In `R`: `ranger`
  - `formula = Y ~ .`

Random Forests

```
require(tidyverse)
covidData <- read_rds('../data/covid_prepped.Rds')
glimpse(covidData)
```

```
## Rows: 3,067
## Columns: 18
## $ trump.votes          <dbl> 19838, 83544, 5622, 7525, 2471…
## $ perc.trump.2020      <dbl> 0.7255770, 0.7726827, 0.538608…
## $ covid.deaths         <int> 120, 431, 67, 76, 149, 42, 79,…
## $ population           <int> 58805, 231767, 25223, 22293, 5…
## $ perc.non.hisp.white <dbl> 72.789, 82.685, 46.965, 74.460…
## $ perc.non.hisp.black <dbl> 18.678, 7.379, 45.783, 20.242,…
## $ perc.non.hisp.asian <dbl> 1.422, 0.853, 0.422, 0.103, 0.…
## $ perc.hispanic        <dbl> 3.007, 4.492, 4.639, 2.641, 7.…
## $ perc.male            <dbl> 0.4876331, 0.4856838, 0.525897…
## $ perc.65up            <dbl> 0.1510923, 0.2000085, 0.189450…
## $ unemp.rate           <dbl> 0.03904205, 0.04069932, 0.0584…
## $ lfpr                 <dbl> 0.4688933, 0.4333219, 0.330603…
## $ weekly.wages         <dbl> 711, 663, 676, 791, 651, 623, …
## $ perc.rural           <dbl> 42.00, 42.28, 67.79, 68.35, 89…
## $ perc.manuf           <dbl> 0.06168181, 0.04557498, 0.2842…
## $ perc.trump.2016      <dbl> 0.7277, 0.7655, 0.5210, 0.7640…
```

Research Question

- What predicts the Covid-19 death rate by county?

```
form.demogs <- 'covid.death.rate ~ perc.65up + perc.male +
perc.non.hisp.white + perc.non.hisp.black + perc.non.hisp.asian +
perc.hispanic + log.pop'

form.econ <- 'covid.death.rate ~ perc.65up + perc.male +
perc.non.hisp.white + perc.non.hisp.black + perc.non.hisp.asian +
perc.hispanic + log.pop + lfpr + weekly.wages + unemp.rate +
perc.manuf + perc.rural'

form.pol <- 'covid.death.rate ~ perc.65up + perc.male +
perc.non.hisp.white + perc.non.hisp.black + perc.non.hisp.asian +
perc.hispanic + log.pop + lfpr + weekly.wages + unemp.rate +
perc.manuf + perc.rural + perc.trump.2016'
```

Comparing models

```
m.demogs <- lm(as.formula(form.demogs),covidData)
summary(m.demogs)
```

```
##
## Call:
## lm(formula = as.formula(form.demogs), data = covidData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2013 -0.7134 -0.0769  0.6127  6.9393
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           7.792220   0.659644  11.813  < 2e-16
## perc.65up             2.102616   0.540668   3.889 0.000103
## perc.male            -4.658581   0.996893  -4.673 3.10e-06
## perc.non.hisp.white -0.020637   0.003192  -6.466 1.17e-10
## perc.non.hisp.black  0.011876   0.003314   3.583 0.000345
## perc.non.hisp.asian -0.071837   0.009242  -7.773 1.04e-14
## perc.hispanic         0.003056   0.003425   0.892 0.372286
## log.pop              -0.193326   0.017954 -10.768  < 2e-16
##
```

Comparing models

```
m.econ <- lm(as.formula(form.econ),covidData)
summary(m.econ)
```

```
##
## Call:
## lm(formula = as.formula(form.econ), data = covidData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6810 -0.6706 -0.0941  0.5779  8.5308
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         11.1115407  0.7684102  14.460  < 2e-16
## perc.65up            1.4349387  0.5556532   2.582  0.00986
## perc.male           -7.1002727  1.0172358  -6.980 3.61e-12
## perc.non.hisp.white -0.0147605  0.0032693  -4.515 6.57e-06
## perc.non.hisp.black  0.0104428  0.0032715   3.192  0.00143
## perc.non.hisp.asian -0.0504391  0.0098165  -5.138 2.95e-07
## perc.hispanic        0.0074509  0.0034500   2.160  0.03088
## log.pop             -0.2382359  0.0236365 -10.079  < 2e-16
## lfpr                -4.2922126  0.4097871 -10.474  < 2e-16
```

Comparing models

```
m.pol <- lm(as.formula(form.pol),covidData)
summary(m.pol)
```

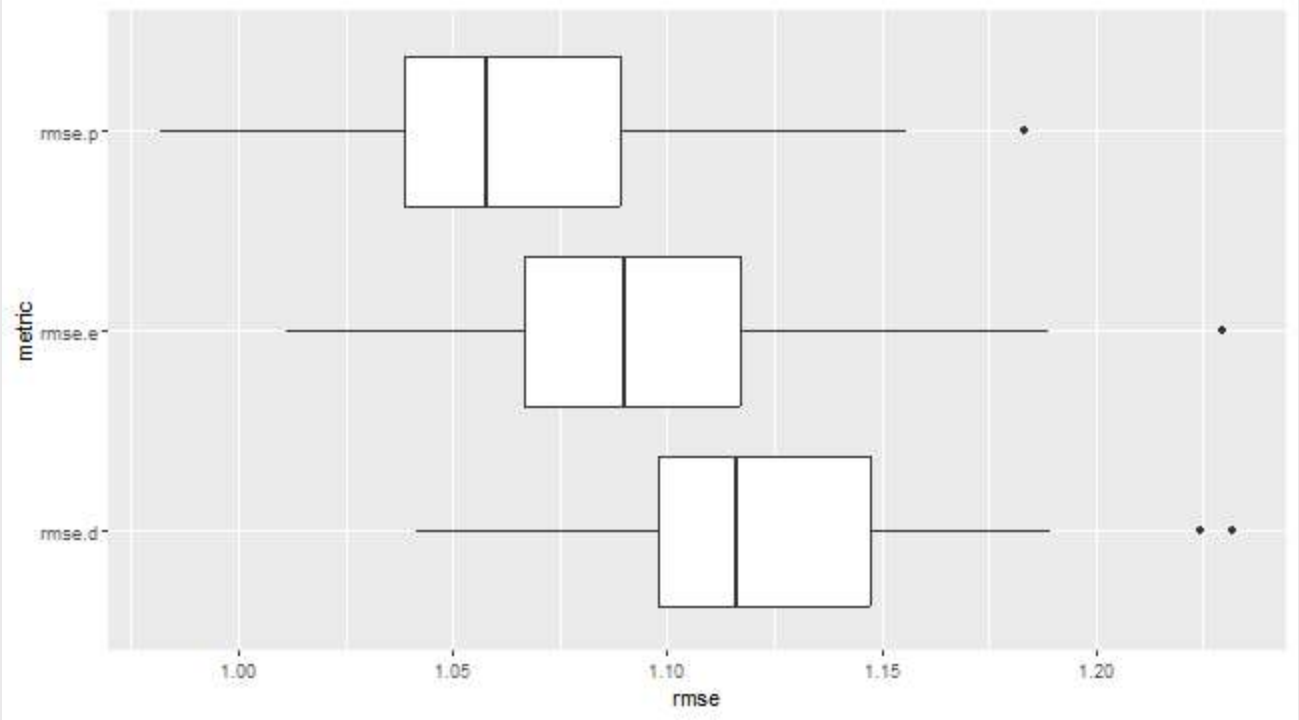
```
##
## Call:
## lm(formula = as.formula(form.pol), data = covidData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5375 -0.6331 -0.0998  0.5212  7.4524
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          7.5048349  0.7865457   9.542  < 2e-16
## perc.65up            2.9946782  0.5493709   5.451 5.40e-08
## perc.male           -5.6817433  0.9904641  -5.736 1.06e-08
## perc.non.hisp.white -0.0303016  0.0033509  -9.043  < 2e-16
## perc.non.hisp.black  0.0102318  0.0031692   3.229  0.00126
## perc.non.hisp.asian -0.0378293  0.0095508  -3.961 7.64e-05
## perc.hispanic       -0.0017425  0.0034042  -0.512  0.60879
## log.pop             -0.1472463  0.0237775  -6.193 6.71e-10
## lfpr                -1.9962819  0.4286574  -4.657 3.34e-06
```

Evaluate Model Fit

```
cvRes <- NULL
for(i in 1:100) {
  inds <- sample(1:nrow(covidData),size =
round(nrow(covidData)*.8),replace = F)
  train <- covidData %>% slice(inds)
  test <- covidData %>% slice(-inds)

  # Train
  mTmp.demogs <- lm(as.formula(form.demogs),train)
  mTmp.econ <- lm(as.formula(form.econ),train)
  mTmp.pol <- lm(as.formula(form.pol),train)

  # Test
  tmp <- test %>%
    mutate(pred.d = predict(mTmp.demogs,newdata = test),
           pred.e = predict(mTmp.econ,newdata = test),
           pred.p = predict(mTmp.pol,newdata = test)) %>%
    summarise(rmse.d = sqrt(mean((covid.death.rate - pred.d)^2)),
              rmse.e = sqrt(mean((covid.death.rate - pred.e)^2)),
              rmse.p = sqrt(mean((covid.death.rate - pred.p)^2))) %>%
    mutate(cvIndex = i)

  cvRes <- cvRes %>%
```

Evaluate Model Fit

```
cvRes %>%
  gather(metric,rmse,-cvIndex) %>%
  ggplot(aes(x = rmse,y = metric)) +
  geom_boxplot()
```



Random Forests

```
require(ranger) # Fast random forests package
m.all <- ranger(formula = as.formula(form.demogs),data = covidData)
m.all
```

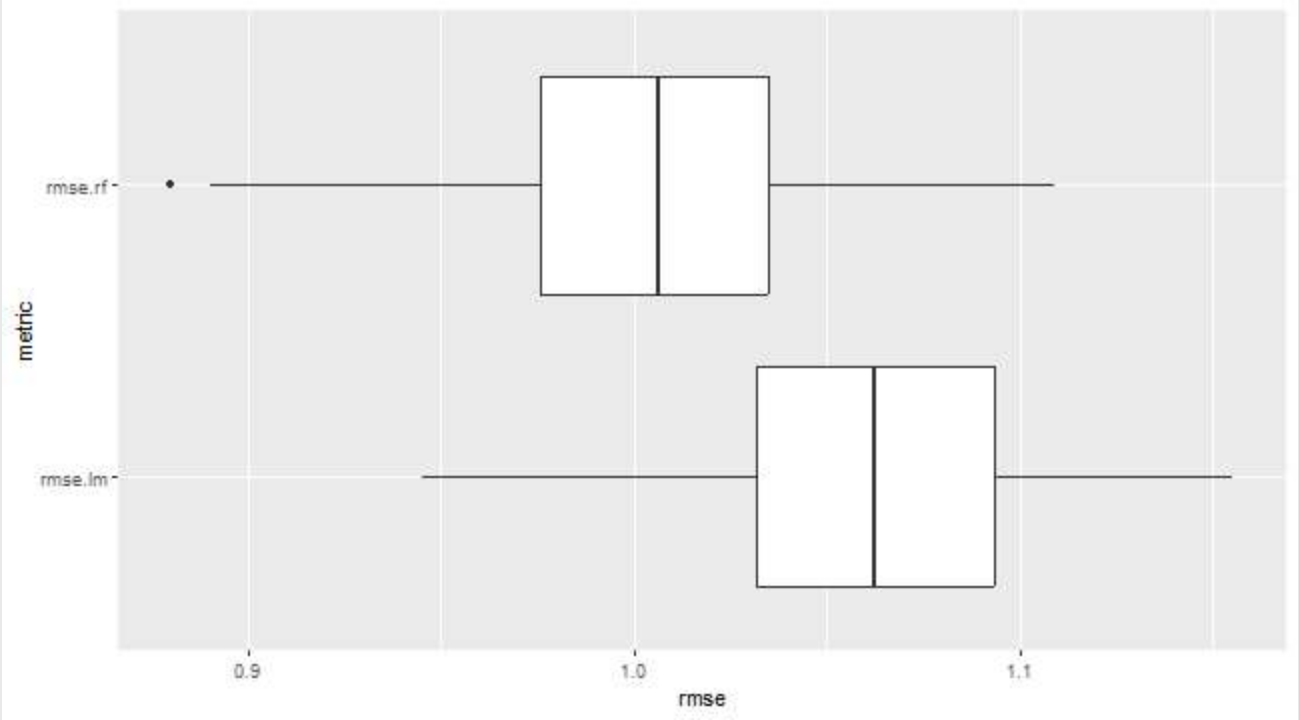
```
## Ranger result
##
## Call:
##  ranger(formula = as.formula(form.demogs), data = covidData)
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      3067
## Number of independent variables:  7
## Mtry:                             2
## Target node size:                 5
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       1.126429
## R squared (OOB):                  0.2848192
```

Random Forest Comparison

```
cvRes <- NULL
for(i in 1:100) {
  inds <- sample(1:nrow(covidData),size =
round(nrow(covidData)*.8),replace = F)
  train <- covidData %>% slice(inds)
  test <- covidData %>% slice(-inds)

  # Train
  mLM.p <- lm(as.formula(form.pol),train)
  mRF.p <- ranger(as.formula(form.pol),train)

  # Test
  # NEED TO RUN PREDICTION ON RF FIRST
  tmpPred <- predict(mRF.p,test)

  tmp <- test %>%
    mutate(pred.lm = predict(mLM.p,newdata = test),
           pred.rf = tmpPred$predictions) %>%
    summarise(rmse.lm = sqrt(mean((covid.death.rate - pred.lm)^2)),
              rmse.rf = sqrt(mean((covid.death.rate - pred.rf)^2)))
%>%
    mutate(cvIndex = i)
```

Random Forest Comparison

```
cvRes %>%
  gather(metric,rmse,-cvIndex) %>%
  ggplot(aes(x = rmse,y = metric)) +
  geom_boxplot()
```



What matters most?

- Random Forests are particularly suitable for investigating **variable importance**
  - I.e., which X predictors are most helpful?
- A few options, but we rely on **permutation tests**
  - Idea: run the best model you have, then re-run it after "permuting" one of the variables
  - "Permute" means randomly reshuffle...breaks relationship
  - How much **worse** is the model when you break a variable?

Variable Importance

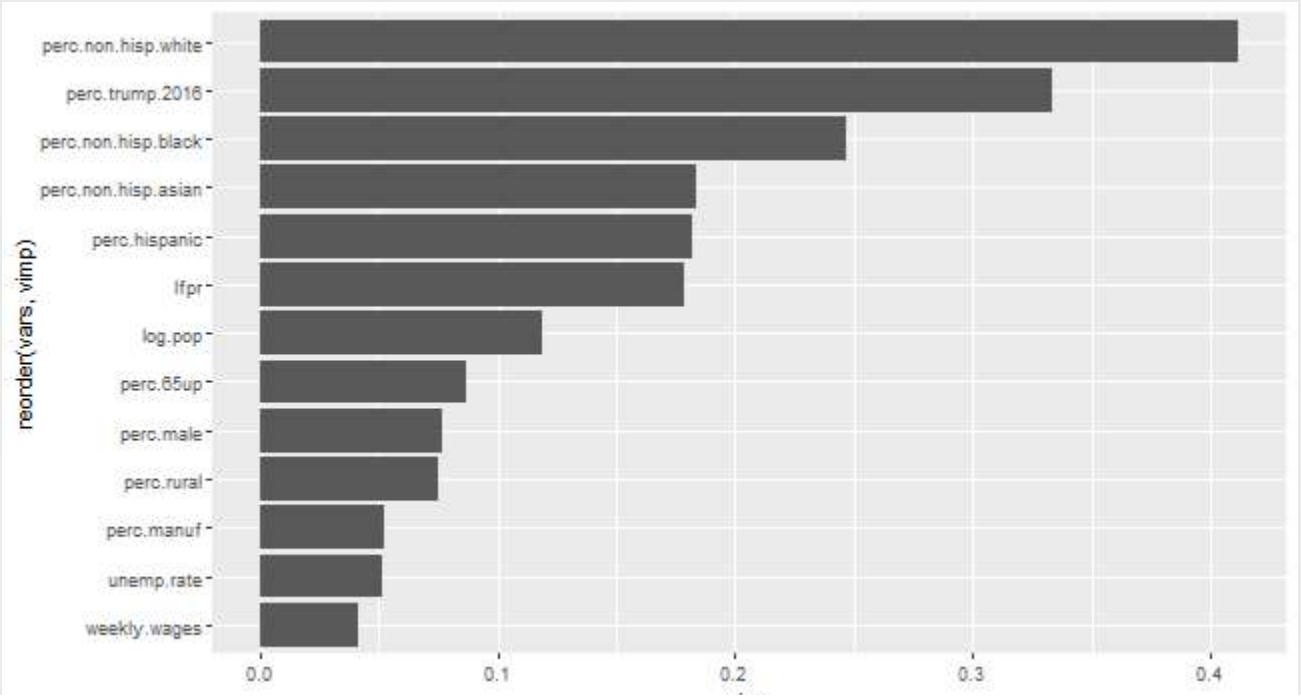
- In `ranger()`, use `importance = "permutation"`

```
rf.pol <- ranger(formula = as.formula(form.pol),data =
covidData,importance = 'permutation')

rf.pol$variable.importance
```

```
##           perc.65up           perc.male perc.non.hisp.white
##          0.08698038          0.07701430          0.41155854
## perc.non.hisp.black perc.non.hisp.asian       perc.hispanic
##          0.24682569          0.18361829          0.18203055
##             log.pop                lfpr        weekly.wages
##          0.11904814          0.17866041          0.04163480
##          unemp.rate          perc.manuf          perc.rural
##          0.05170493          0.05189645          0.07484234
##     perc.trump.2016
##          0.33363024
```

Variable Importance

```
toplot <- data.frame(vimp = rf.pol$variable.importance,
                     vars = names(rf.pol$variable.importance))

toplot %>%
  ggplot(aes(x = vimp,y = reorder(vars,vimp))) +
  geom_bar(stat = 'identity')
```



LASSO

- "Least Absolute Shrinkage and Selection Operator"
- Concept: Make it hard for predictors to matter
  - Practice: λ penalizes how many variables you can include
  - $\sum_{i=1}^{n}(y_i - \sum_j x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$
  - Minimize the errors, but penalize for each additional predictor
  - You *could* kitchen-sink a regression and get super low errors
  - LASSO penalizes you from throwing everything into the kitchen sink
- In `R`, need to install a new package! `install.packages('glmnet')`

```
require(glmnet)
```

LASSO

- Function doesn't use formulas
- Give it the raw data instead, divided into `Y` (outcome) and `X` (predictors)

```
Y <- covidData$covid.death.rate
X <- covidData %>%
select(perc.65up,perc.male,perc.non.hisp.white,perc.non.hisp.black,perc
```

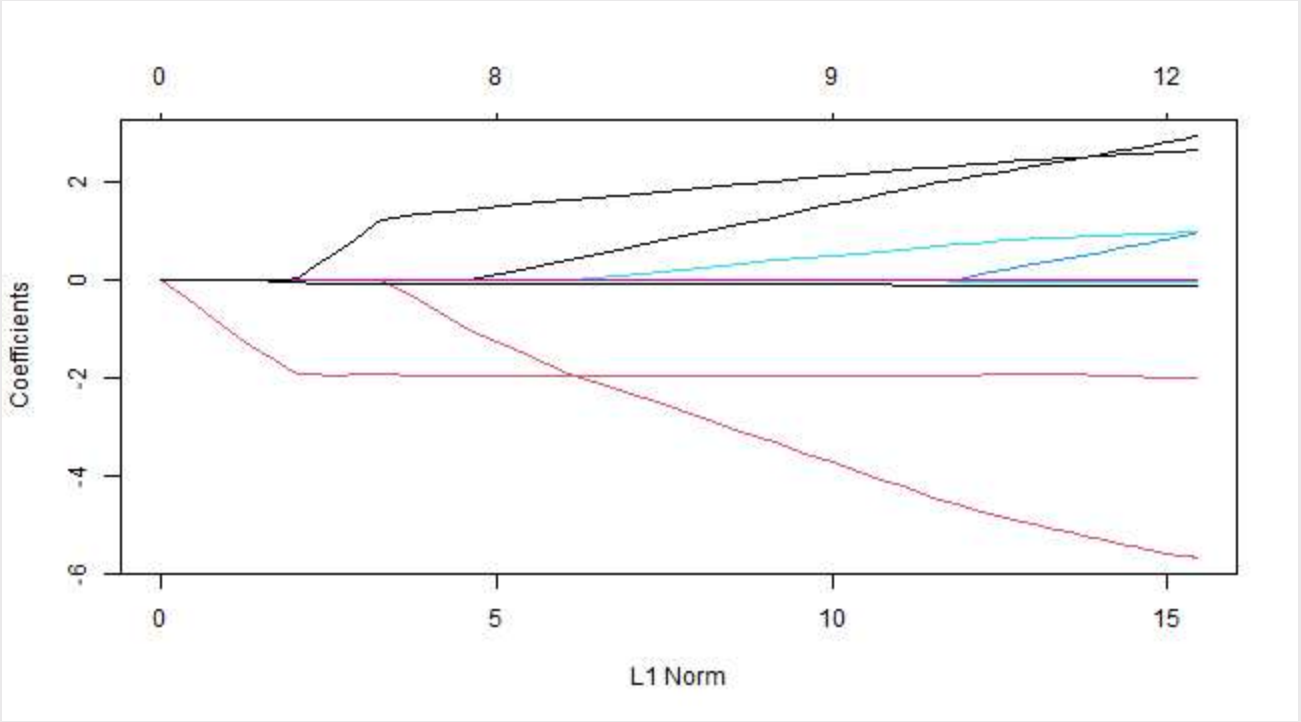
LASSO

- Now estimate!

```
lassFit <- glmnet(x = as.matrix(X),
                  y = as.matrix(Y))
```

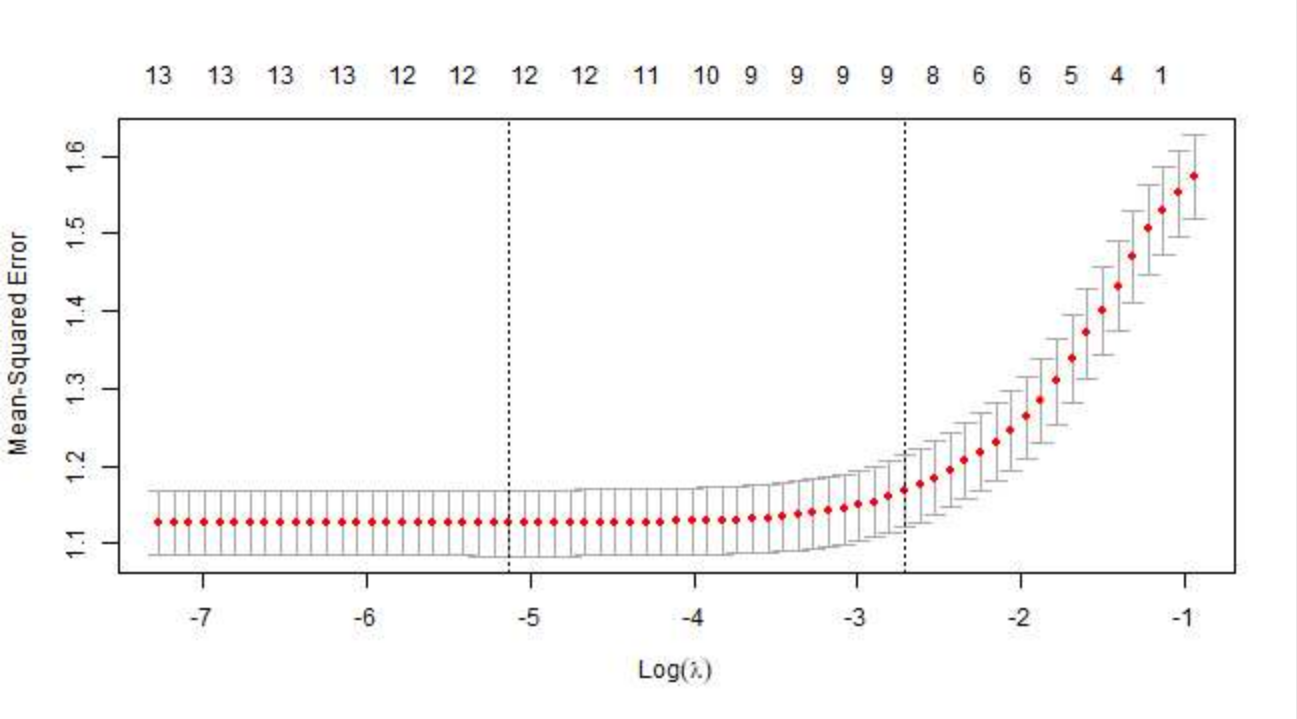
LASSO

```
plot(lassFit)
```



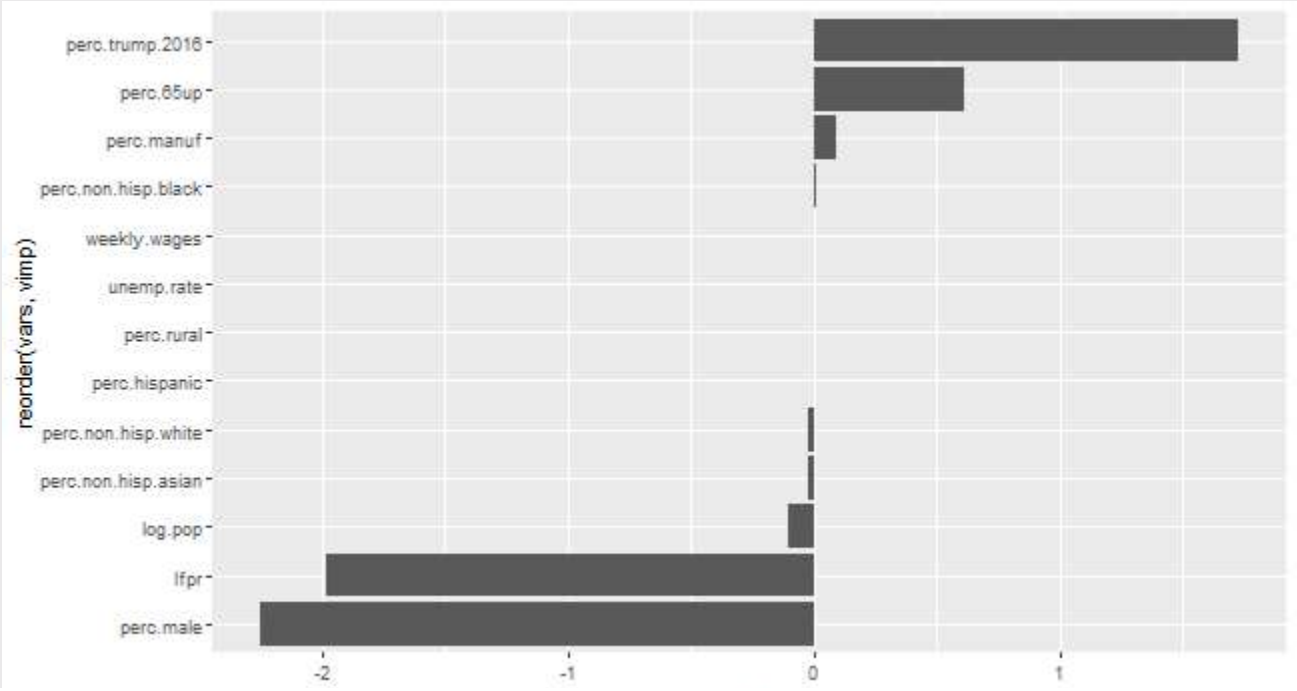
Has its own CV!

```
cv.lassFit <- cv.glmnet(x = as.matrix(X),y = as.matrix(Y))

plot(cv.lassFit)
```



Variable Importance

```
best <- cv.lassFit$glmnet.fit$beta[,cv.lassFit$index[2,]]
vimpLass <- data.frame(vimp = best,
                       vars = names(best))
vimpLass %>%
  ggplot(aes(x = vimp,y = reorder(vars,vimp))) +
  geom_bar(stat = 'identity')
```



Maps

- Install `maps`

```
require(tidyverse)
require(maps)
require(mapproj)

# Load dataset included in maps package
states48 <- map_data('state')

states48 %>%
  as_tibble()
```

```
## # A tibble: 15,537 × 6
##     long   lat group order region  subregion
##    <dbl> <dbl> <dbl> <int> <chr>   <chr>
##  1 -87.5  30.4     1     1 alabama <NA>
##  2 -87.5  30.4     1     2 alabama <NA>
##  3 -87.5  30.4     1     3 alabama <NA>
##  4 -87.5  30.3     1     4 alabama <NA>
##  5 -87.6  30.3     1     5 alabama <NA>
##  6 -87.6  30.3     1     6 alabama <NA>
##  7 -87.6  30.3     1     7 alabama <NA>
```

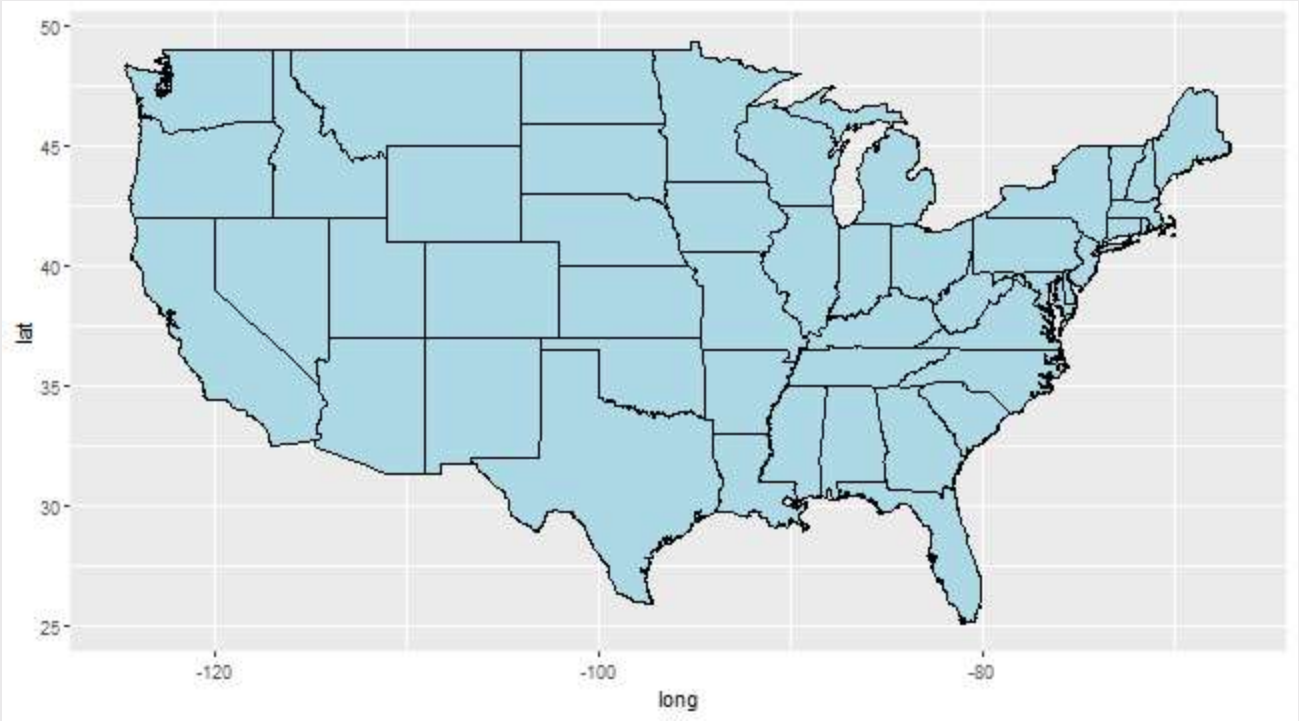
Maps

- `maps` data are prepared to run with `ggplot()`
    - Specifically, want to use `geom_polygon()`

```
p <- states48 %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group),
               color = 'black',
               fill = 'lightblue')
```

Maps

```
p
```



Maps

- Can quickly improve with pre-made `theme()`s and `coord()`s

```
p +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

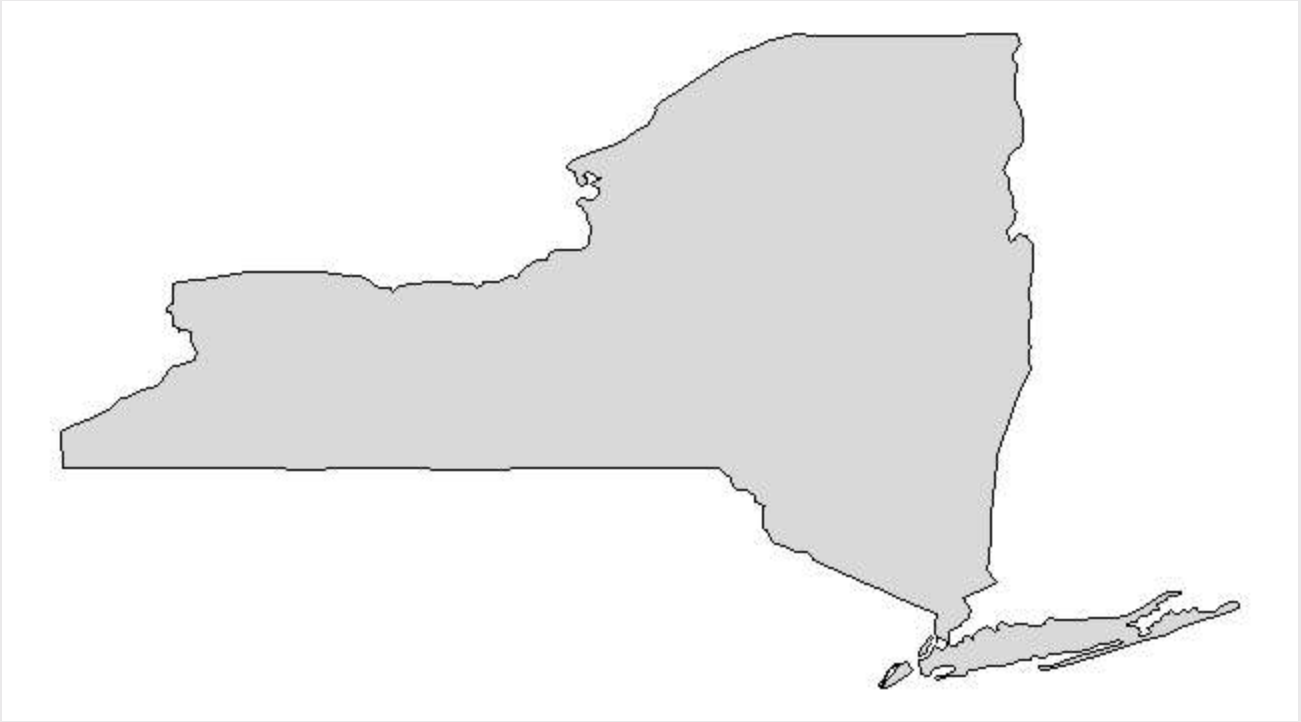
Maps

- Can zoom in with `filter()`

```
ny <- states48 %>%
  filter(region == 'new york')

pNY <- ny %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group),
               color = 'black',
               fill = 'grey85') +
  theme_void()
```

Maps

```
pNY
```



Maps

```
pNY +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

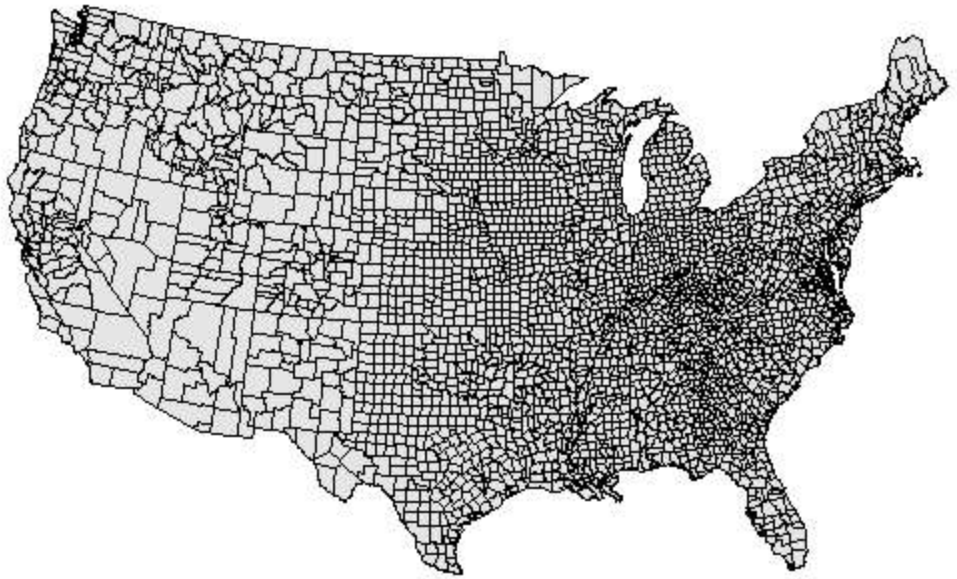
Maps

- Can also get counties prepared!

```
counties <- map_data('county')

pCty <- counties %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group),
               color = 'black',
               fill = 'grey90') +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

```
pCty
```

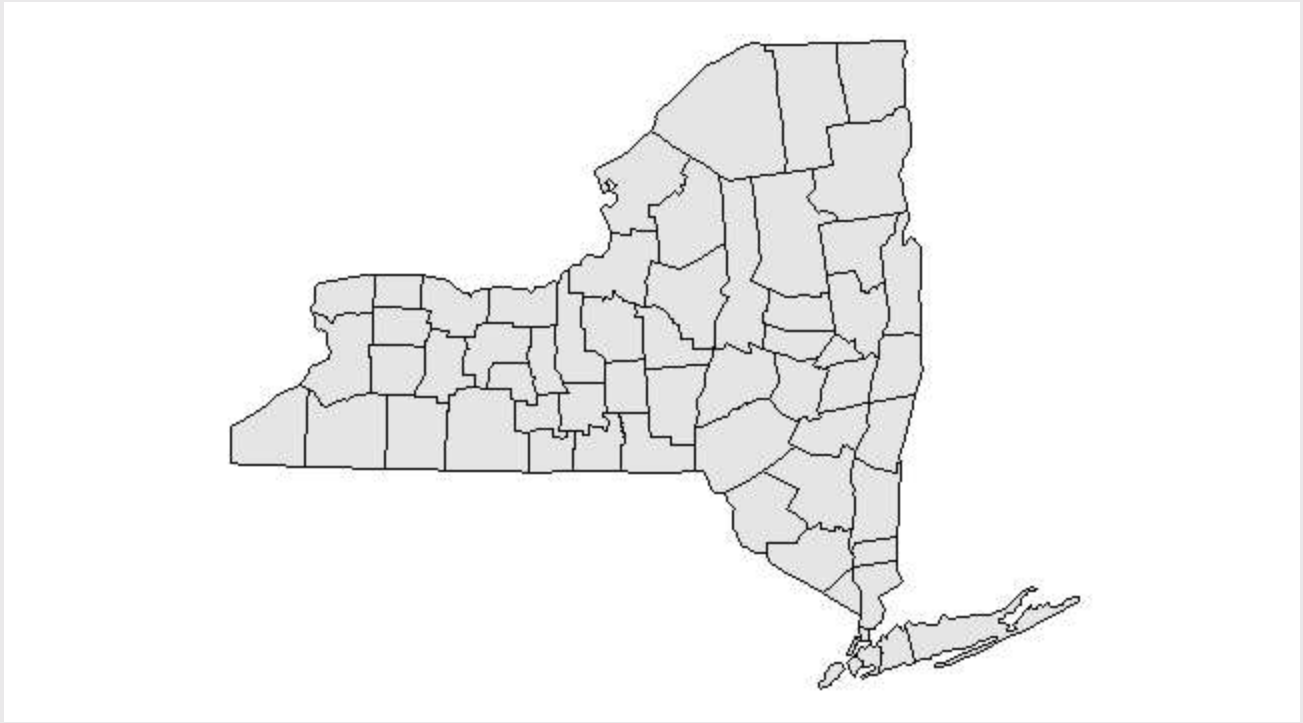


Maps

```
pNYCty <- counties %>%
  filter(region == 'new york') %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group),
               color = 'black',
               fill = 'grey90') +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

```
pNYCty
```



Maps

- Want to visualize data!
- Put the `fill` inside the `aes`
- But we need data first

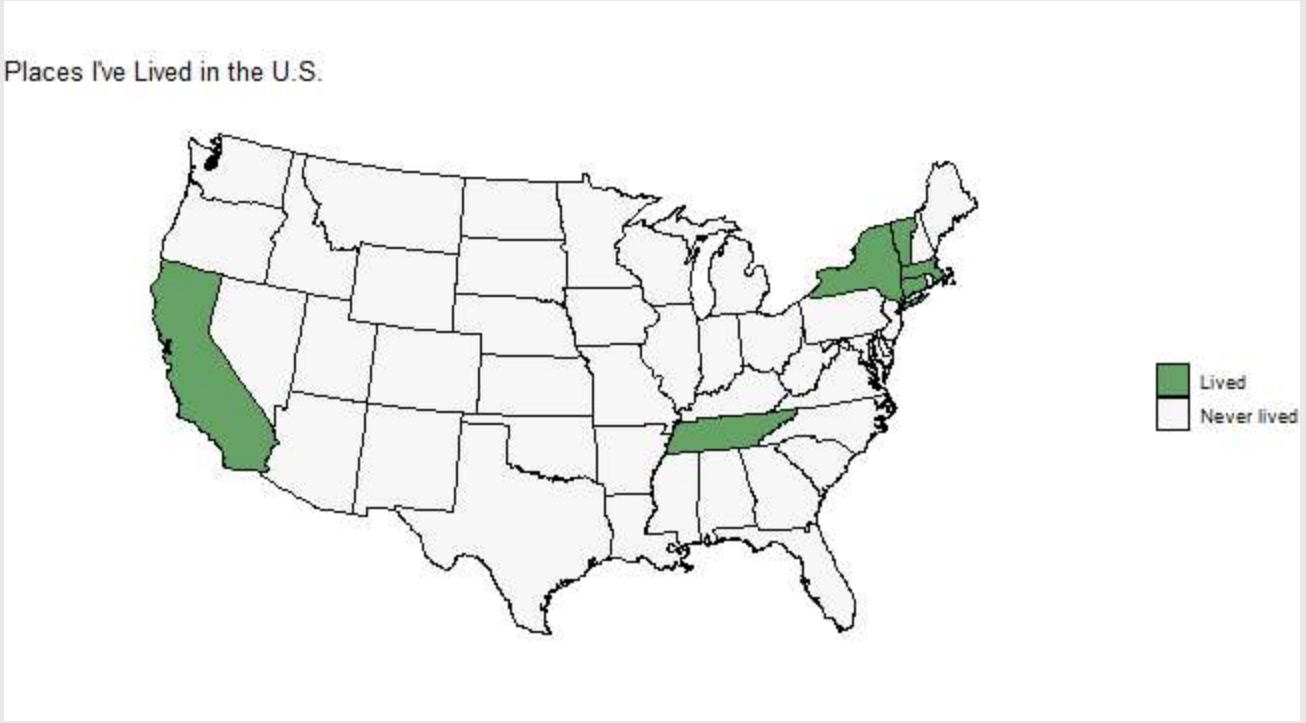
```
JBStates <- c('new
york','california','vermont','massachusetts','district of
columbia','tennessee','connecticut')

states48 <- states48 %>%
  mutate(jbLived = ifelse(region %in% JBStates,'Lived','Never
lived'))
```

Maps

```
p <- states48 %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group,fill = jbLived),
               color = 'black',alpha = .6) +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

```
p +
  scale_fill_manual(name = '',values = c('Lived' = 'darkgreen','Never lived' = 'grey95')) +
  labs(title = "Places I've Lived in the U.S.")
```



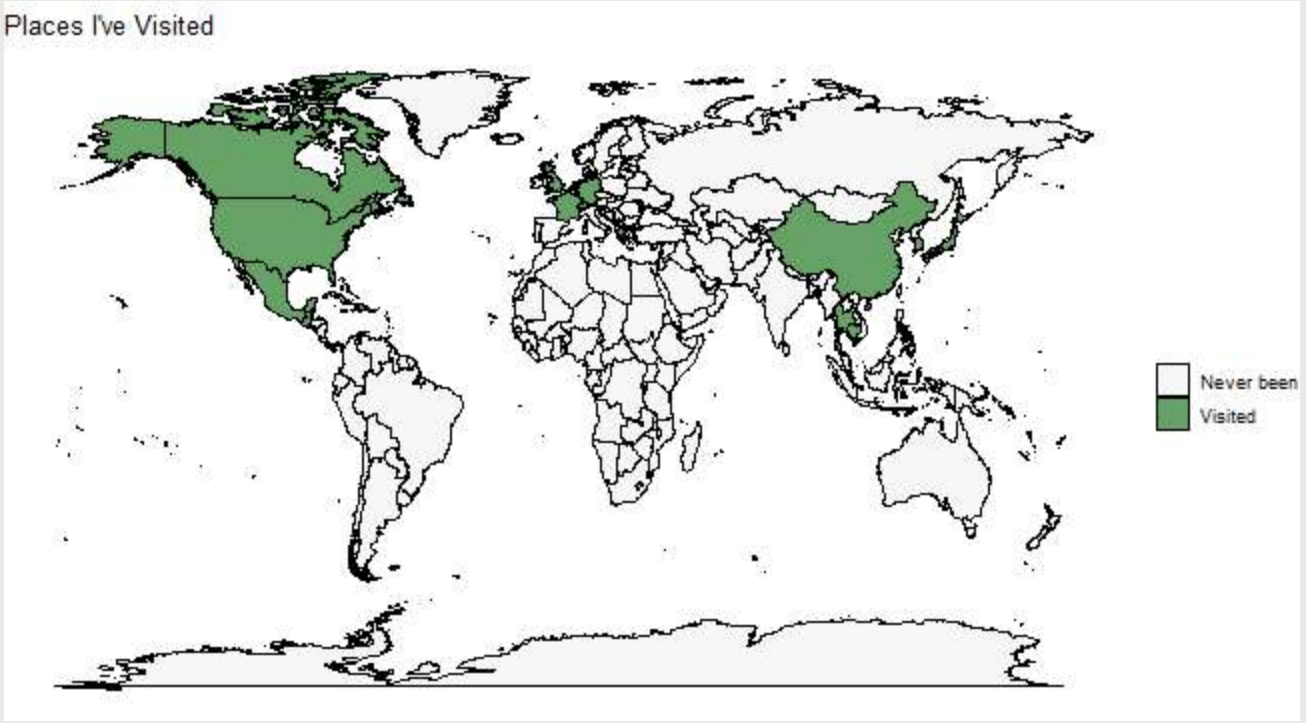
Maps

- Also have world maps!

```
JBCountries <- c('USA','South Korea','France','UK',
                 'Germany','Switzerland','Greece',
                 'Dominican Republic','Saint Lucia',
                 'China','Thailand','Cambodia','Japan',
                 'Guatemala','Aruba','Canada','Belize','Mexico')
world <- map_data('world')

world <- world %>%
  mutate(jbVisit = ifelse(region %in% JBCountries,'Visited','Never
been'))

p <- world %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group,fill = jbVisit),
               color = 'black',alpha = .6) +
  theme_void()
```

Maps

```
p +
  scale_fill_manual(name = '',values = c('Visited' =
'darkgreen','Never been' = 'grey95')) +
  labs(title = "Places I've Visited")
```



Maps

- More interesting data?

```
PollDat <- readRDS('../data/PresStatePolls04to20.Rds') %>%
  as_tibble() %>%
  rename(region = state.name)

PollDat %>% head()
```

```
## # A tibble: 6 × 11
##   state.postal  year sampl…¹ dem.p…² rep.p…³ dem.v…⁴ rep.v…⁵
##   <chr>        <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 AK            2016     500    30      38      37.7    52.9
## 2 AK            2016     201    37.4    37.7    37.7    52.9
## 3 AK            2016     660    30.6    36.1    37.7    52.9
## 4 AK            2008     500    42      53      37.9    59.4
## 5 AK            2008     500    41      57      37.9    59.4
## 6 AK            2016     409    31      48      37.7    52.9
## # … with 4 more variables: days.until.election <dbl>,
## #   days.in.field <dbl>, EV <int>, region <chr>, and
## #   abbreviated variable names ¹samplesize, ²dem.poll,
## #   ³rep.poll, ⁴dem.vote, ⁵rep.vote
```

Maps

- Collapse to state-by-year

```
PollDat <- PollDat %>%
  group_by(year,region) %>%
  summarise(DemPct = mean(dem.poll,na.rm=T),
            RepPct = mean(rep.poll,na.rm=T),
            DemVote = first(dem.vote),
            RepVote = first(rep.vote)) %>%
  ungroup()
```

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

Maps

- Merge with map data

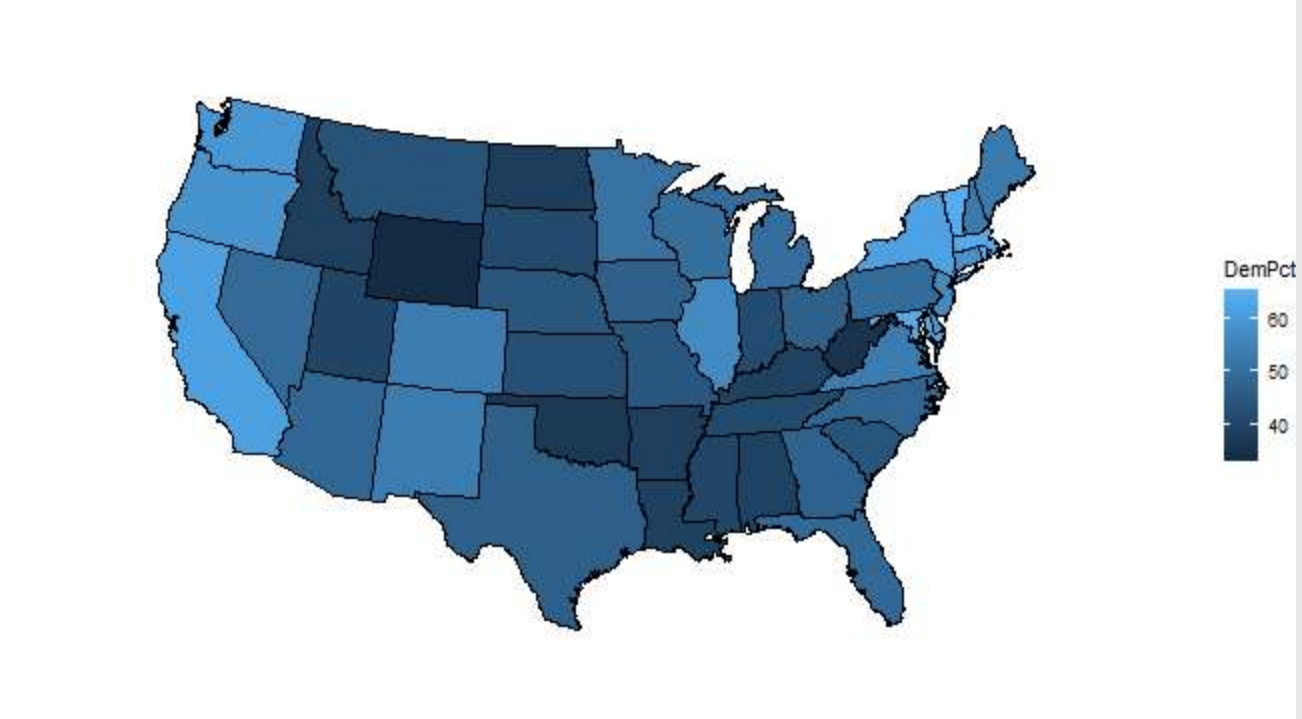
```
toplot <- states48 %>%
  left_join(PollDat %>%
              filter(year == 2020))
```

```
## Joining, by = "region"
```

```
p <- toplot %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group,fill = DemPct),
               color = 'black') +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

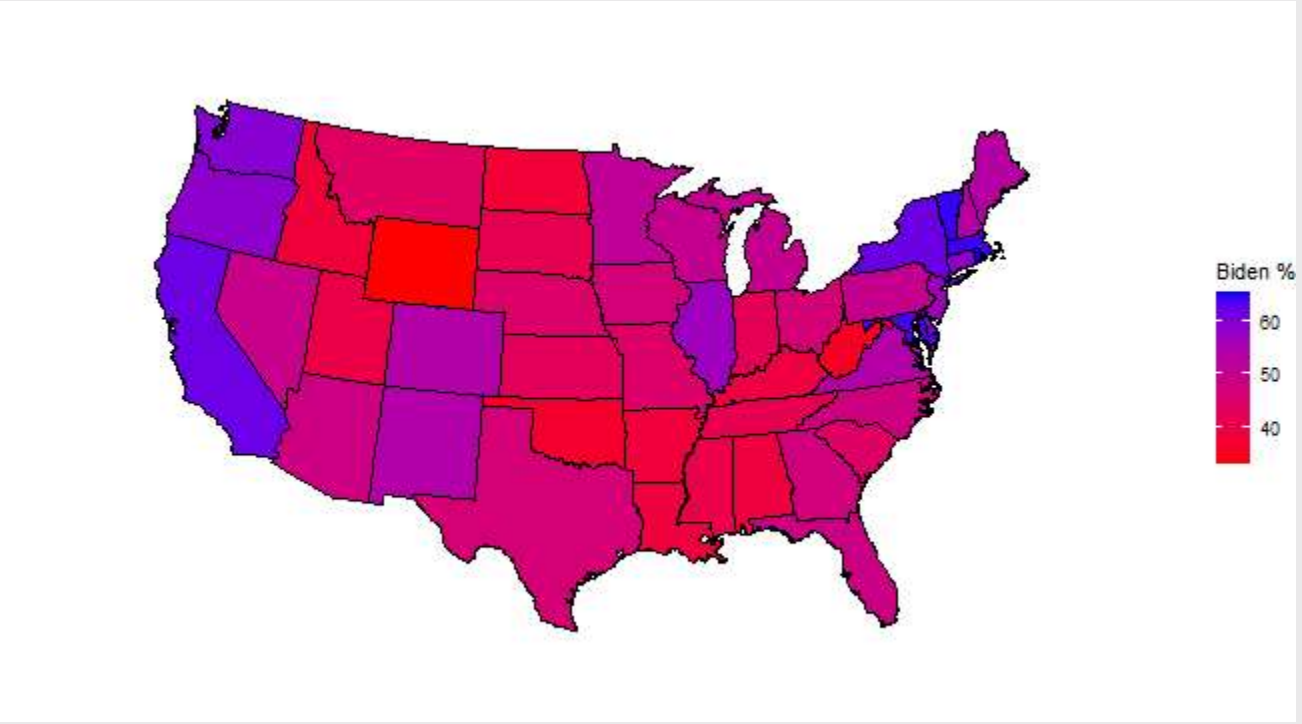
```
p
```



Maps

- Let's adjust with `scale_fill_continuous()`

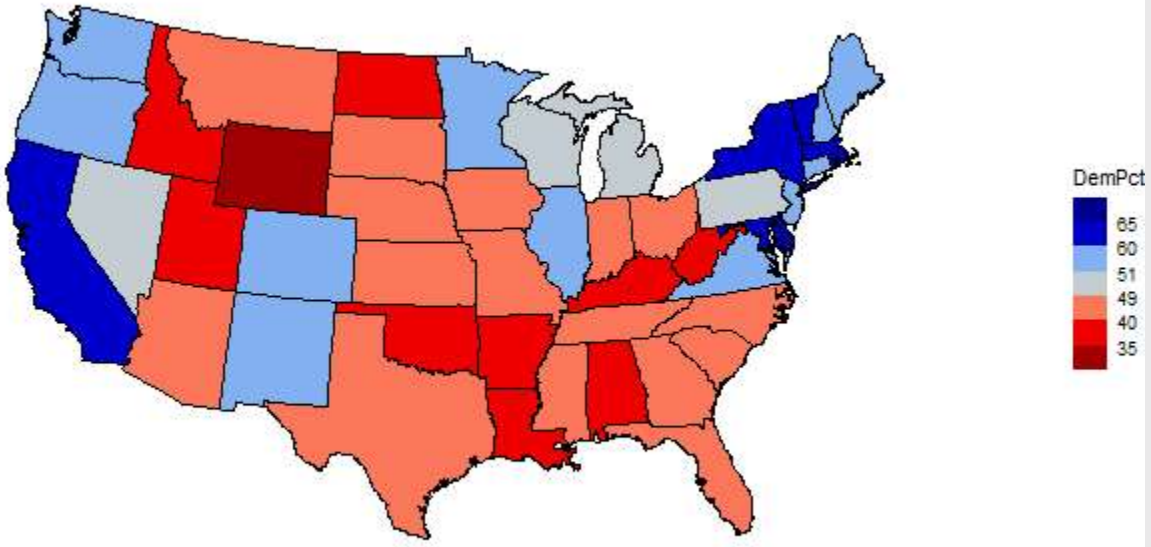
```
p +
  scale_fill_continuous(name = 'Biden %',low = 'red',high = 'blue')
```



Maps

- Could also `cut` to create bins

```
p +
  scale_fill_stepsn(colors =
c('darkred','red','tomato','grey80','skyblue','blue','darkblue'),
                    breaks = c(30,35,40,49,51,60,65,70))
```



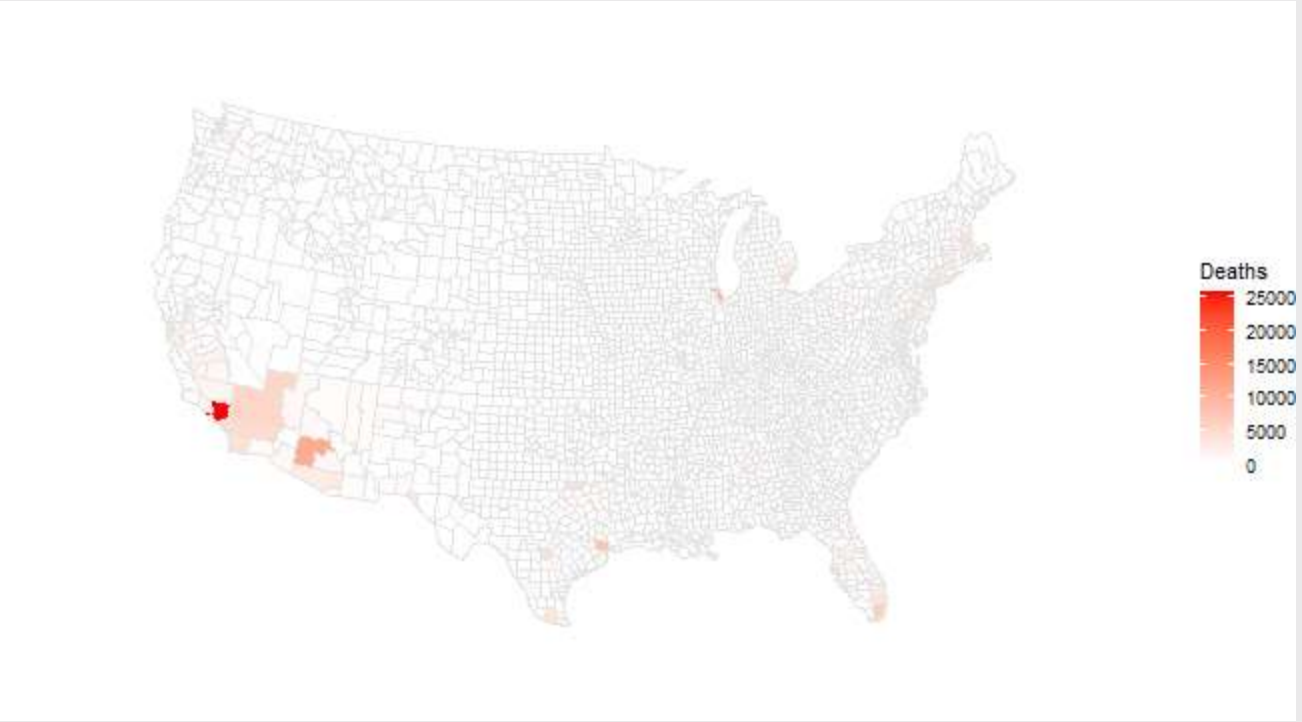
Maps

- Different geographies & different data

```
countycovid <- readRDS('../data/countycovid.Rds') # Already prepared!

p <- countycovid %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group,fill = deaths),
               color = 'grey90') +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

```
p +
  scale_fill_continuous(name = 'Deaths',low = 'white',high = 'red')
```



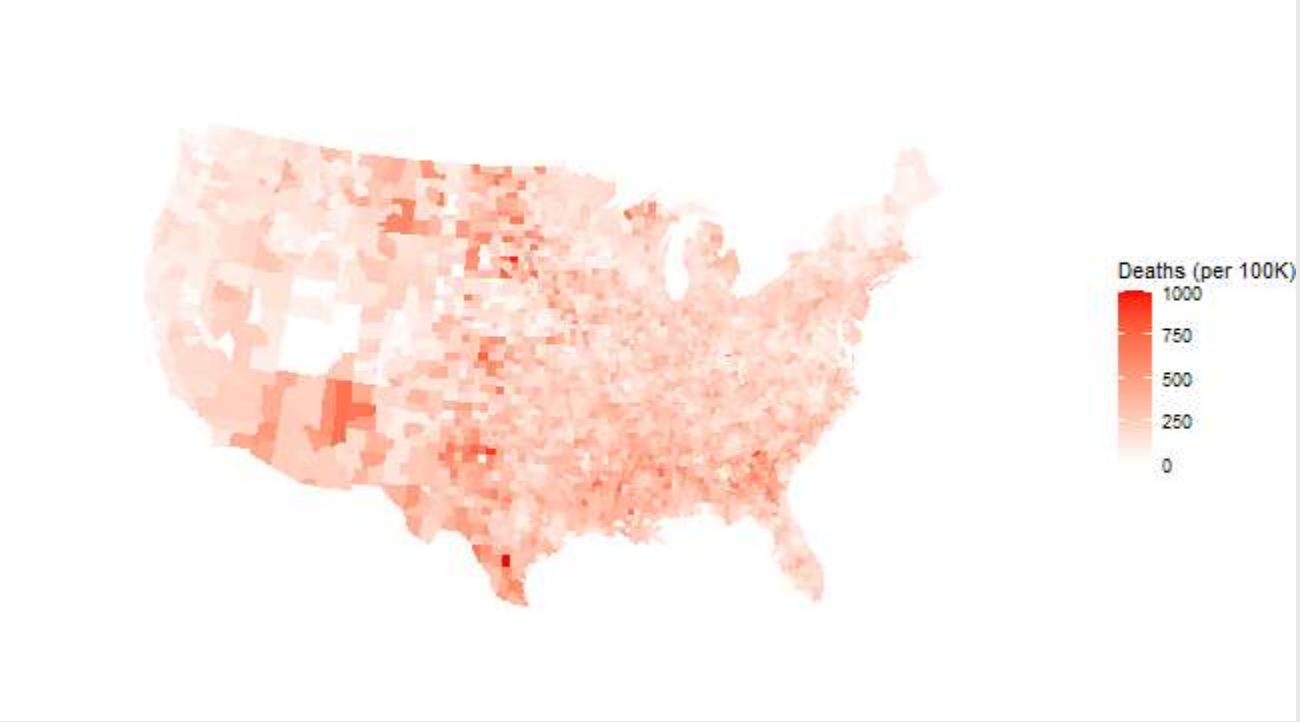
Maps

- Raw deaths are a bad measure...why?
- Want to normalize by population!

```
p <- countycovid %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group,
                   fill = deaths*100000/population)) +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

```
p +
  scale_fill_continuous(name = 'Deaths (per 100K)',
                        low = 'white',high = 'red')
```



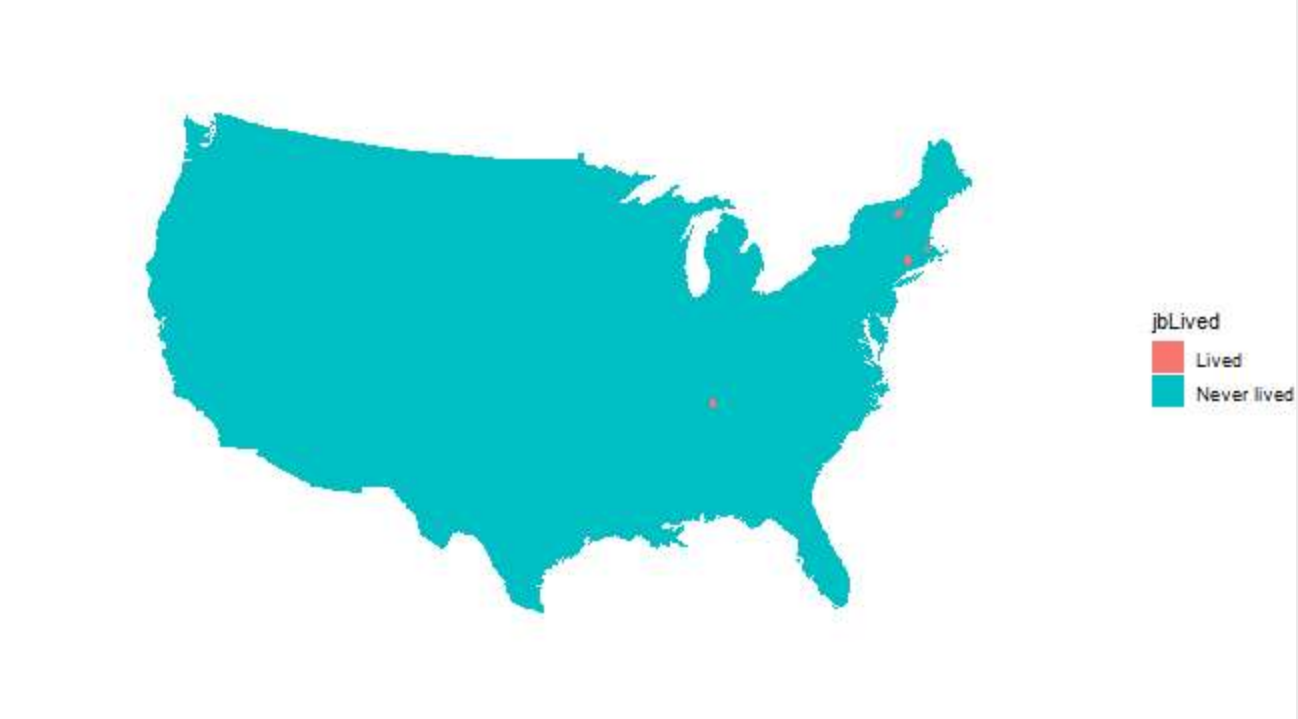
Maps

- What about more precise things?
- I.e., where I've lived by county

```
jbCounties <- c('norfolk:massachusetts','washington:vermont',
                'manhattan:new york','san francisco:california',
                'davidson:tennessee','hartford:connecticut',
                'washington:district of columbia')

counties <- counties %>%
  mutate(combReg = paste0(subregion,":",region)) %>%
  mutate(jbLived = ifelse(combReg %in% jbCounties,'Lived',
                          'Never lived'))

p <- counties %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group,
                   fill = jbLived)) +
  theme_void() +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

```
p
```



Maps

- Can add points instead!

```
jbLivedDF <- data.frame(combReg = jbCounties,
                        years = c(18,18,7,1,.4,3,4))

counties <- counties %>% left_join(jbLivedDF)

p <- counties %>%
  ggplot() +
  geom_polygon(aes(x = long,y = lat,group = group),
               fill = 'grey95',color = 'grey70') +
  geom_point(data = counties %>%
               drop_na(years) %>%
               group_by(group,years) %>%
               summarise(long = mean(long),lat = mean(lat)),
             aes(x = long,y = lat,size = years),shape = 21,color =
'red') +
  theme_void() +
  scale_size_continuous(range = c(2,10),breaks = c(1,5,10)) +
  coord_map('albers',lat0 = 30,lat1 = 40)
```

Maps

```
p
```