Homework 3

David Lewis

March 13, 2025

0. Topic for Presentation

I would like to do my project on Diffbind, an R package that predicts differentially bound sites where proteins bind (using ChIP-seq data).

https://bioconductor.org/packages/release/bioc/html/DiffBind.html

Import Data_mixmodel.txt

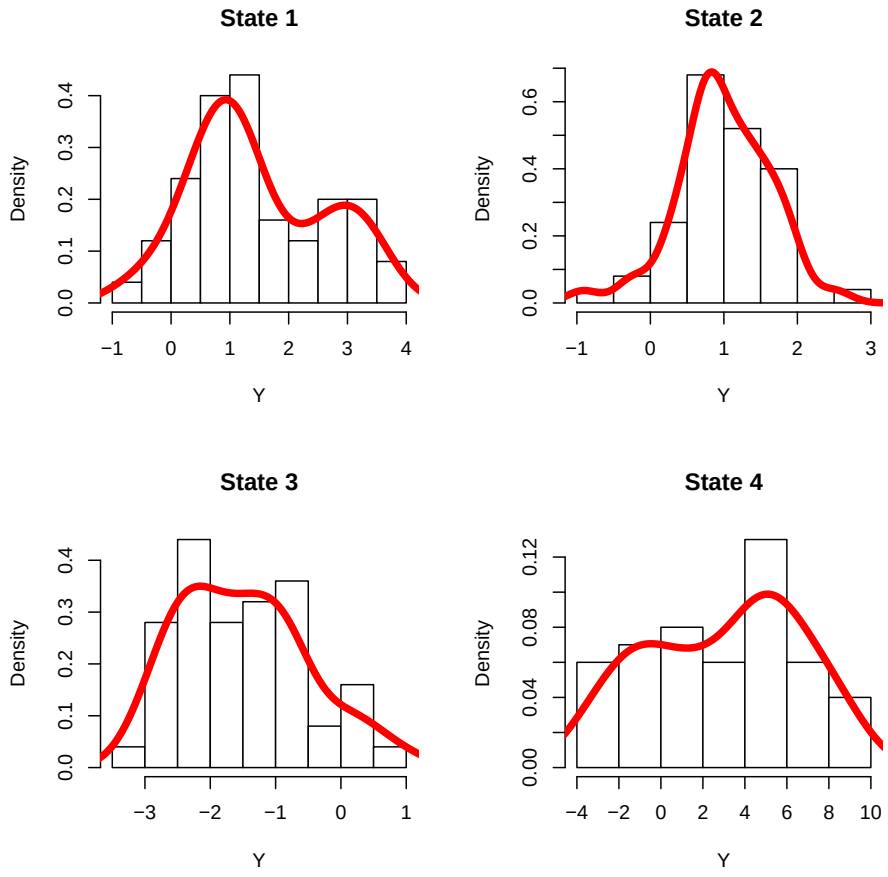
```
data <- read.table("data_mixmodel.txt", stringsAsFactors = TRUE)
```

Utility function for plotting

```
apply_plotting_func <- function(FUN) {
    for (i in seq(unique(data$state))) {
        d <- data[data$state == i, ]
        FUN(d, i)
    }
}
```

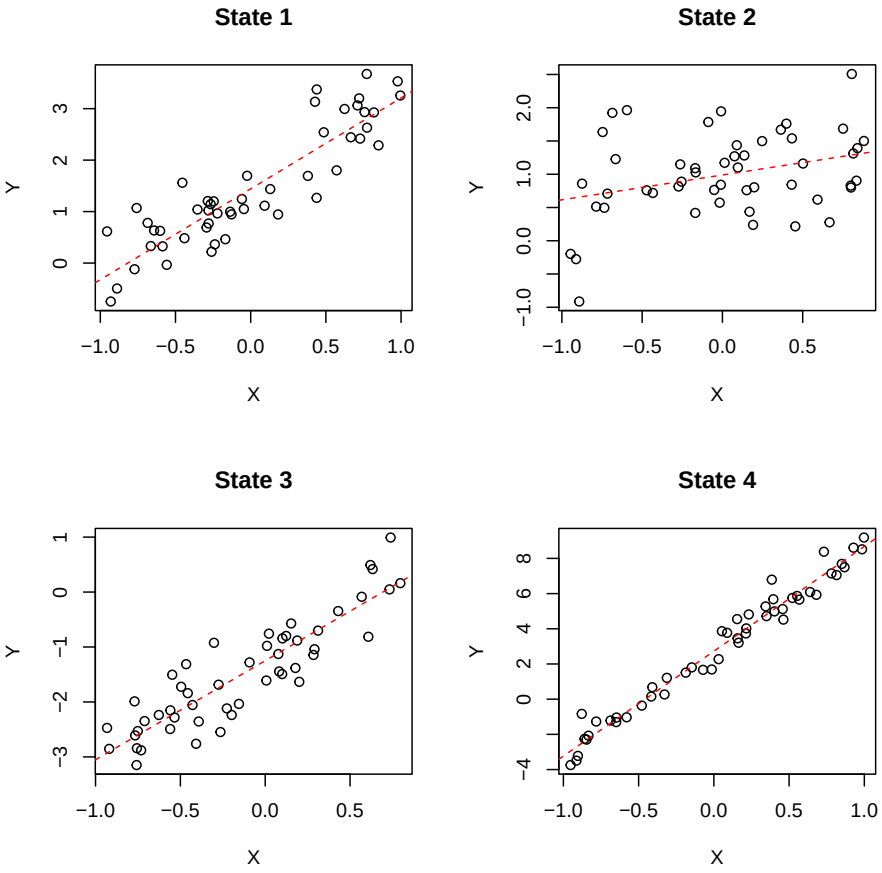
1. Replicate the plot

```r
histogram_density <- function(data, state) {
    v <- data$y
    hist(v, prob = TRUE, col = "white", main = sprintf("State %d", state),
     ↪  xlab = "Y")
    lines(density(v), col = "red", lwd = 5)
}

par(mfrow = c(2, 2))
apply_plotting_func(histogram_density)
```



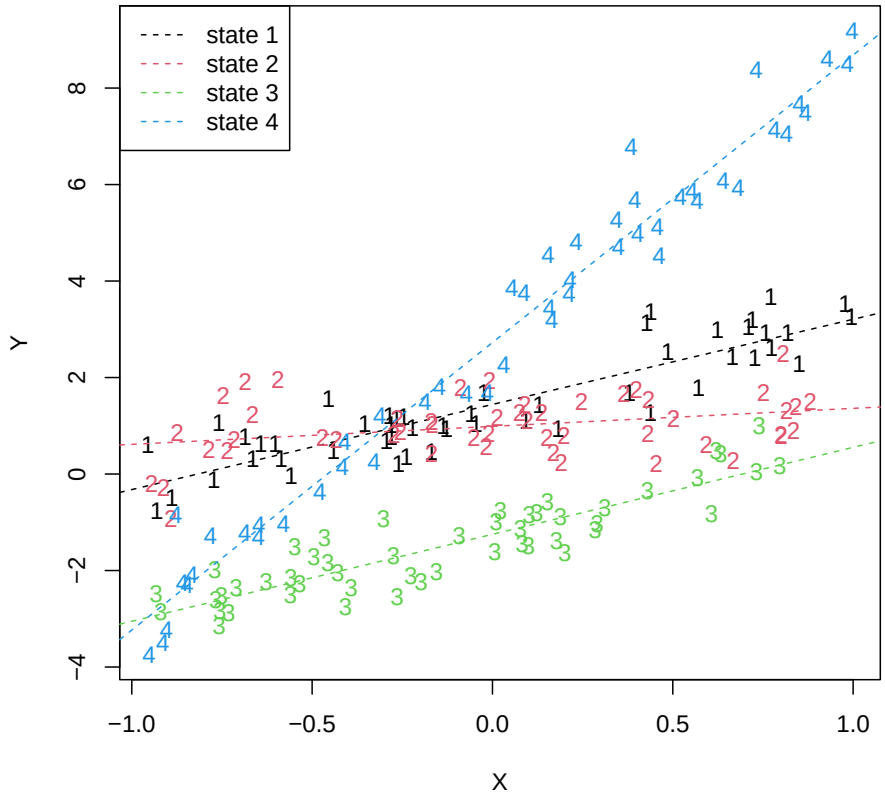
2. Replicate the plot

```r
scatter_lm <- function(data, state) {
    plot(data$x, data$y, xlab = "X", ylab = "Y", main = sprintf("State
    ↪  %d", state))
    abline(lm(y ~ x, data = data), lty = "dashed", col = "red")
}

par(mfrow = c(2, 2))
apply_plotting_func(scatter_lm)
```



3. Replicate the plot

```
lm_all <- function(data, state) {
    text(data$x, data$y, labels = state, col = state)
    abline(lm(y ~ x, data = data), lty = "dashed", col = state)
}

par(mfrow = c(1, 1))
plot(data$x, data$y, type = "n", xlab = "X", ylab = "Y")
apply_plotting_func(lm_all)
legend("topleft", legend = sapply(unique(data$state), FUN = function(x) {
    sprintf("state %d", x)
}), col = unique(data$state), lty = "dashed")
```



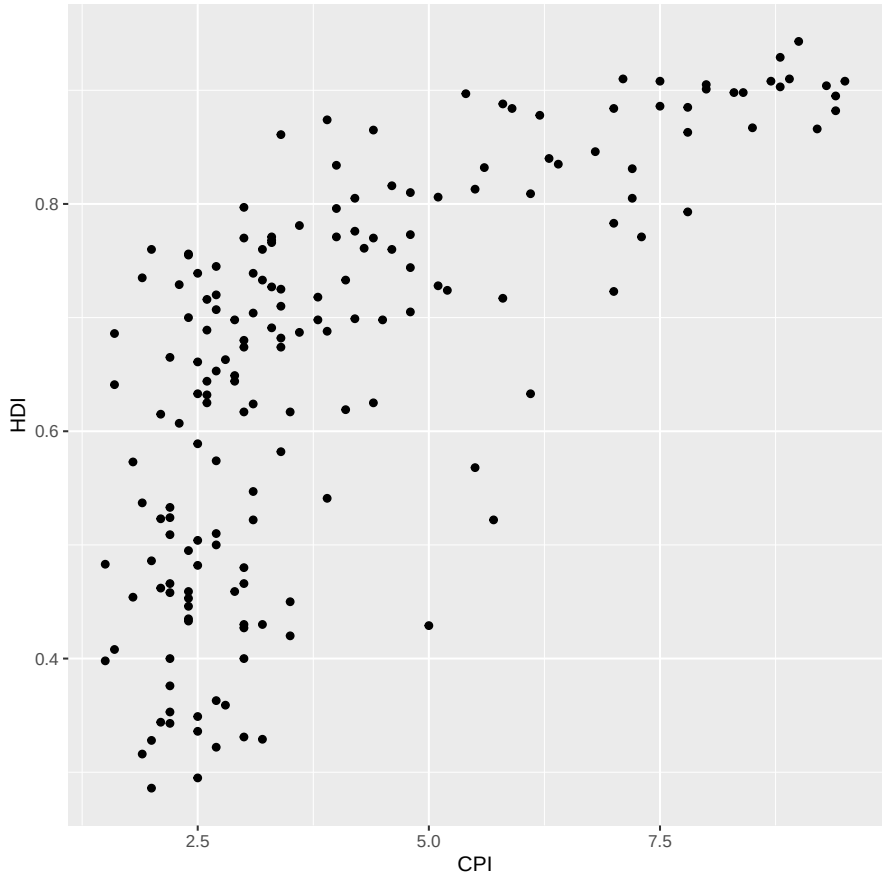
The `type="n"` argument creates the plot with the correct dimensions as to show the points, but does not show the points.

Load EconomistData.csv and load ggplot2

```
library(ggplot2)
data <- read.csv("EconomistData.csv", stringsAsFactors = TRUE)
```

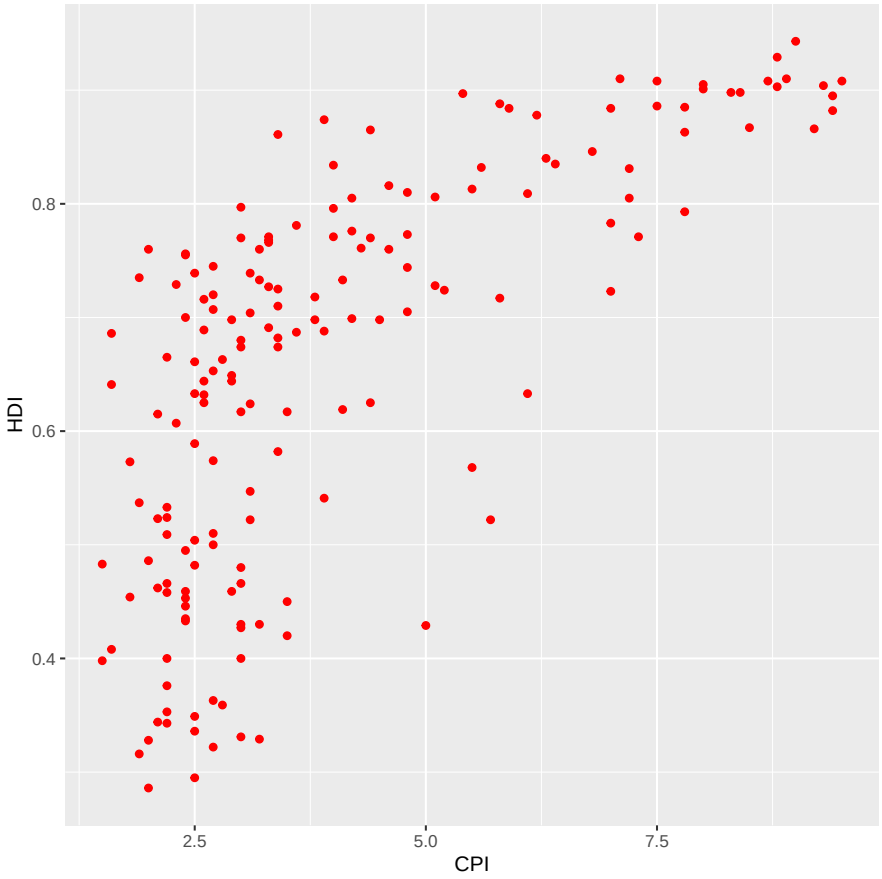
4. Create a scatter plot with CPI on the x axis and HDI on the y axis (2 points)

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point()
```



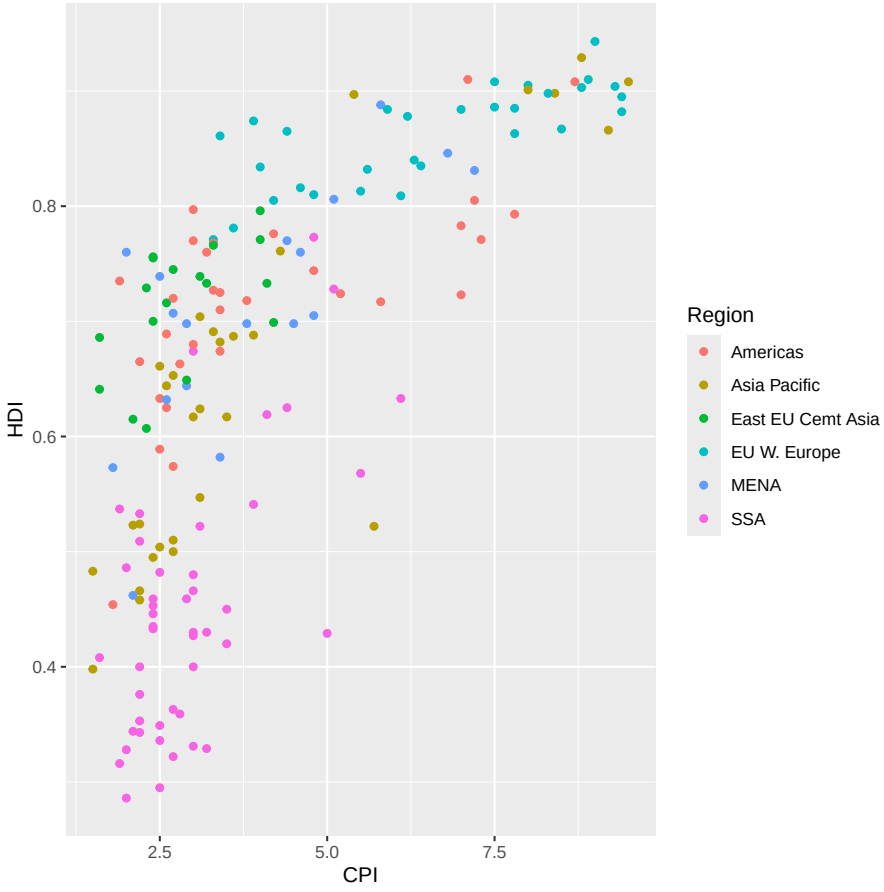
5. Make the points red

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(col = "red")
```



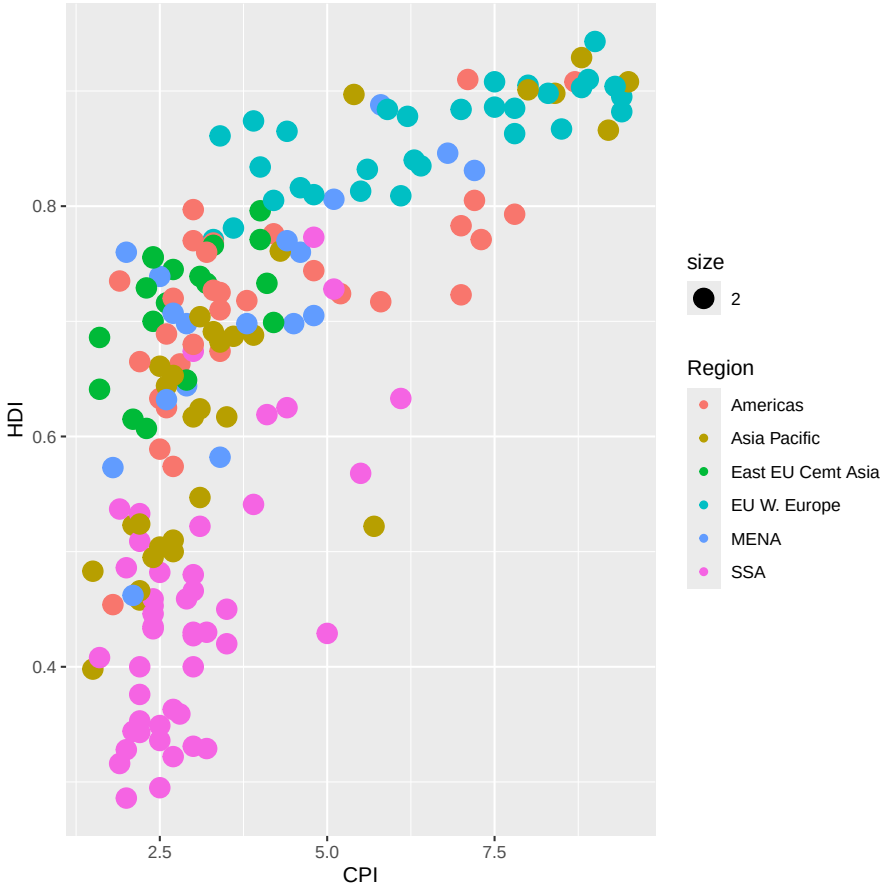
6. Map the color of the points to region

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(aes(col = Region))
```



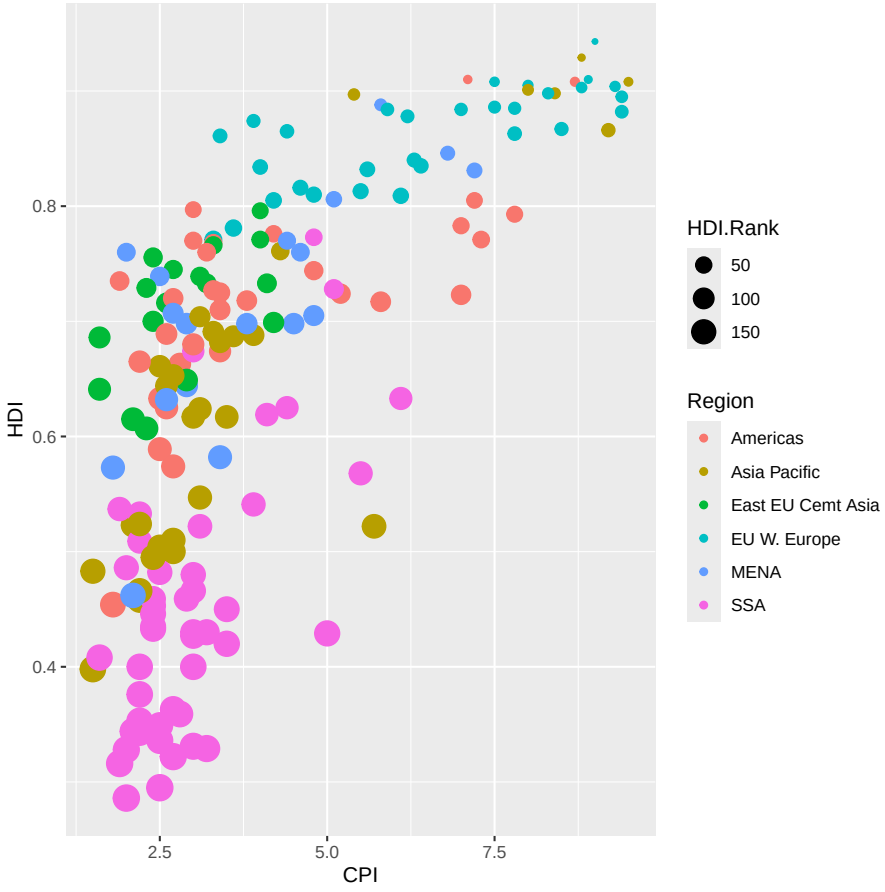
7. Make the points bigger by setting size to 2

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(aes(size = 2, col = Region))
```



8. Map the size of the points to HDI.Rank

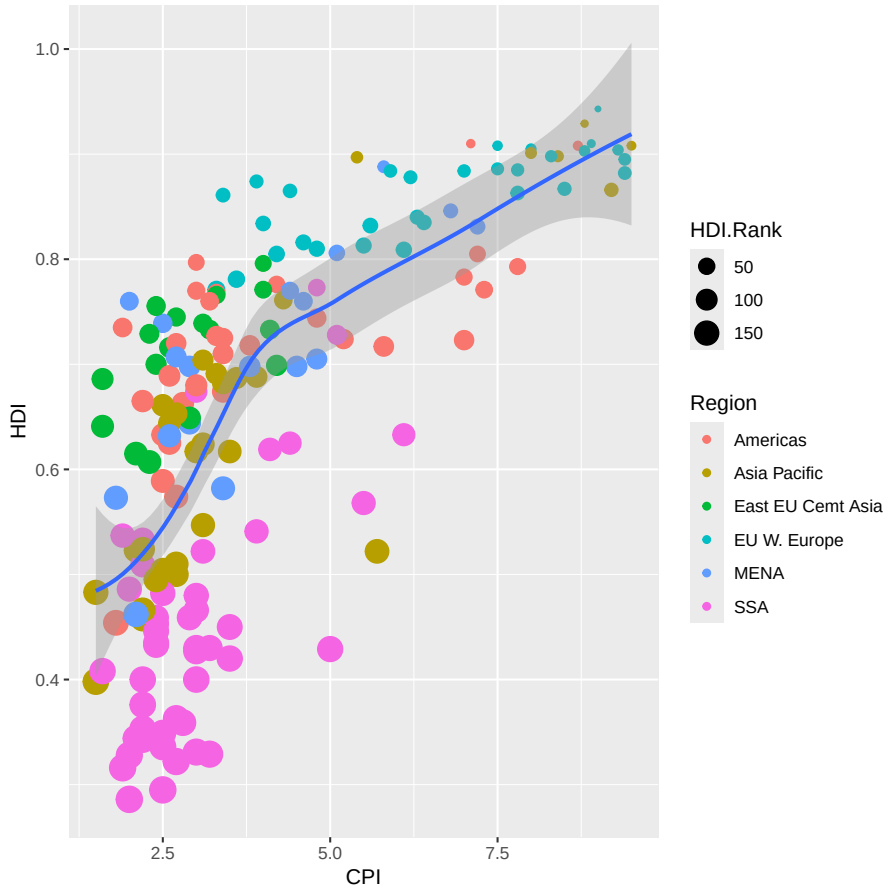
```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(aes(col = Region, size = HDI.Rank))
```



9. add a layer with a smooting line on top of the scatter plot in 8

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(aes(col = Region, size = HDI.Rank)) +
    geom_smooth()
```

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

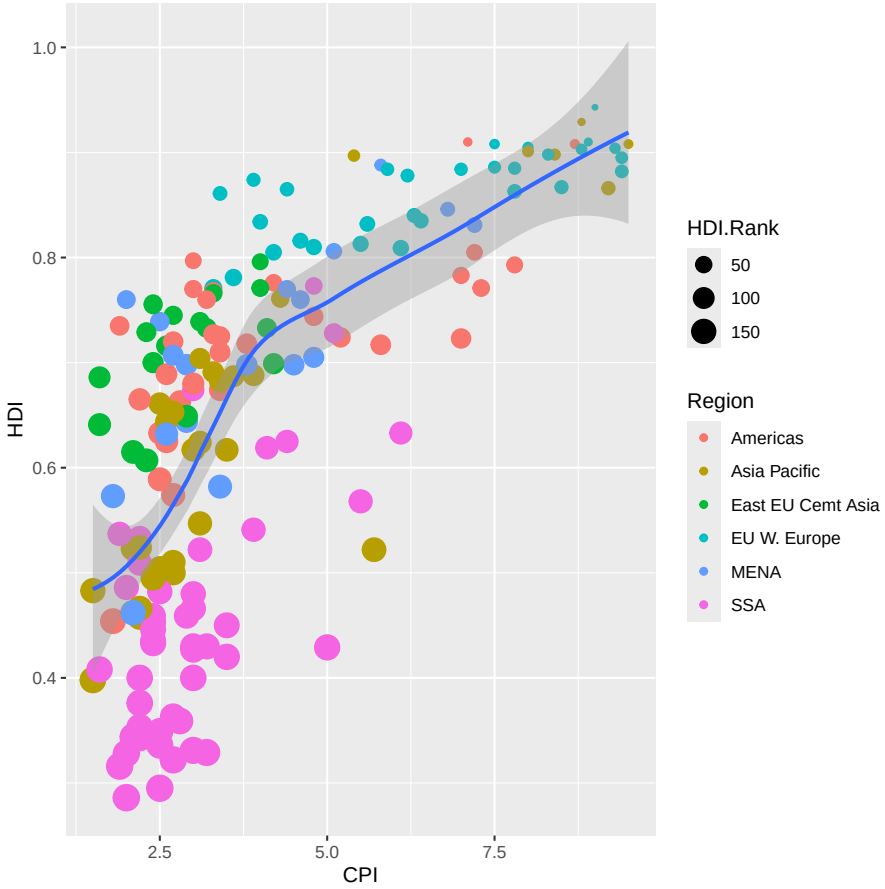


The default method is loess when there are less than 1000 observations. There are less than 1000 observations in EconomistData.csv, so the method is loess.

10. Change the default method

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(aes(size = HDI.Rank, col = Region)) +
    geom_smooth(method = "loess")
```

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

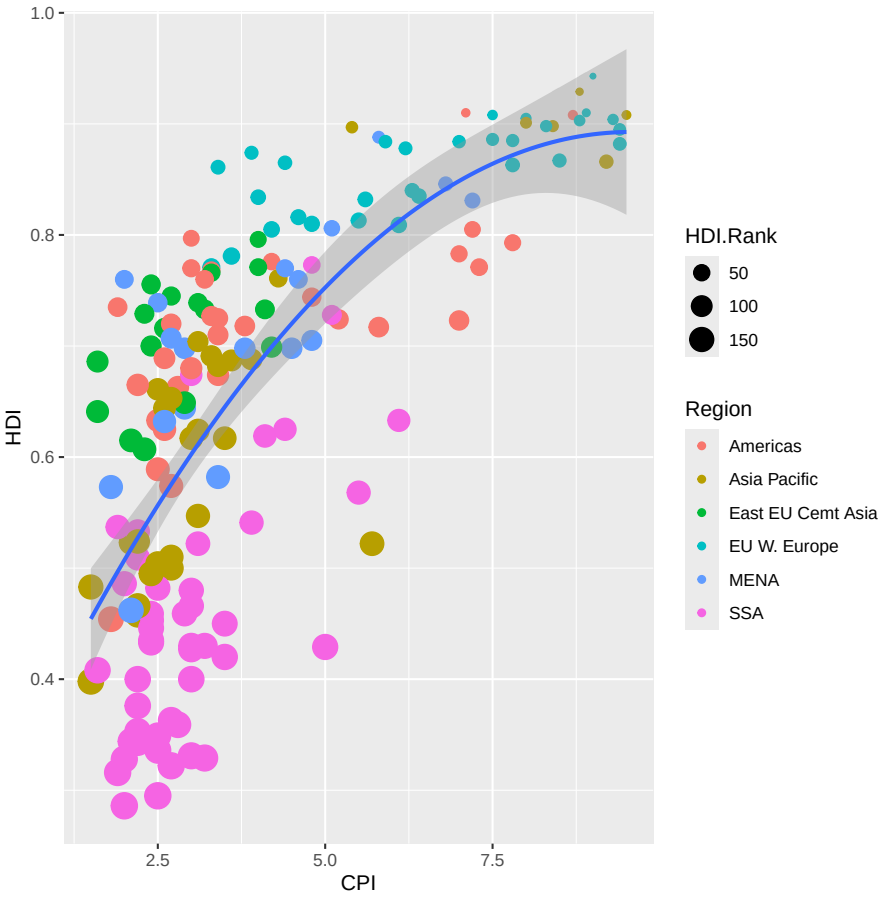


The default method is loess in this case, so just setting the method to loess should do nothing. We can change the smoothness using the span parameter.

To increase the smoothness, we can set the span parameter to something like `10`

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(aes(size = HDI.Rank, col = Region)) +
    geom_smooth(method = "loess", span = 10)
```

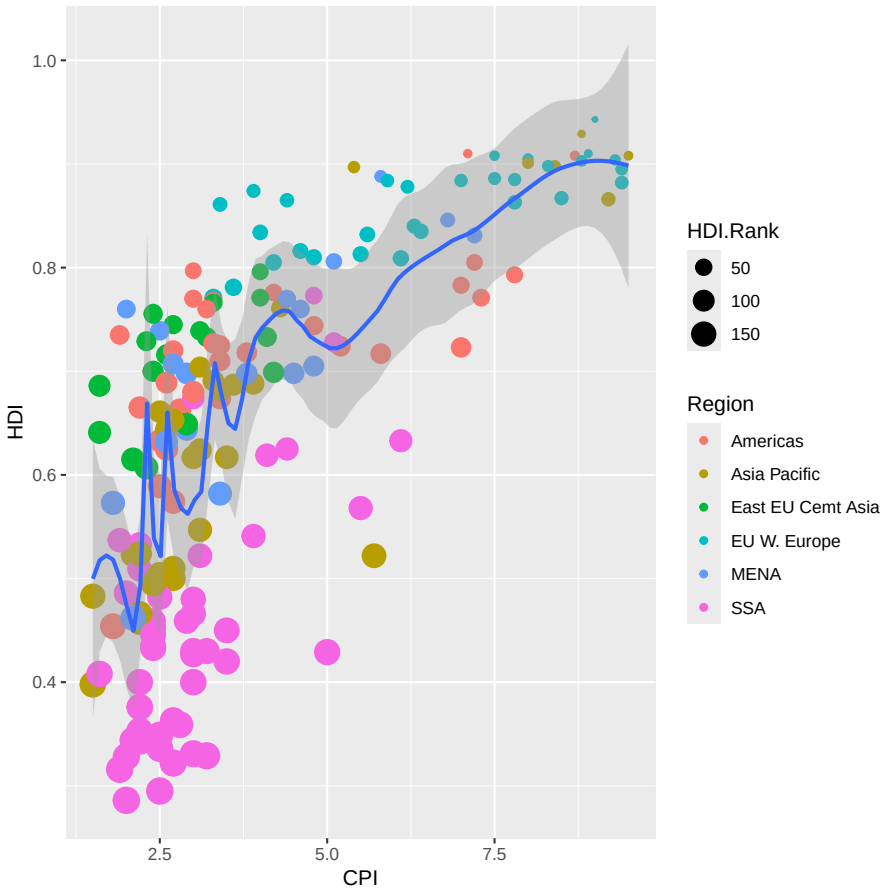
```
## `geom_smooth()` using formula = 'y ~ x'
```



To decrease the smoothness, we can set the span parameter to something like `0.2`

```
ggplot(data, aes(x = CPI, y = HDI)) +
    geom_point(aes(size = HDI.Rank, col = Region)) +
    geom_smooth(method = "loess", span = 0.2)
```

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



If the goal of the analysis is to find a line that can be used for prediction, then to little smoothing will result in an overfit, while too much smoothing would result in an underfit both of which would decrease the usefulness of the model. To select a smoothness that is ideal, one would try to find a smoothness that fits the overall pattern of the data, but does not exhibit the noise inherent in the data. The only way to find the ideal smoothing using only the plot as feedback would be go based on intuition or "vibes".

For a better method, one might attempt to maximize or minimize some statistic by cross validating the span parameter. The span parameter is set to 0.75 by default, so there is no "auto" option in this case. Instead, one might use cross validation to find the span parameter that minimizes the Root Mean Square Error of Prediction. Attempting to optimize the smoothing parameter is only really useful if the goal is to predict something using the model.