boston_housing_project

Isabel Adelhardt

2024-09-05

- exploring multivariate relationships
- running multiple linear regression.
- manually create a joint confidence region
- run ANOVA and analyze the output

Load in Data

Load in the data:

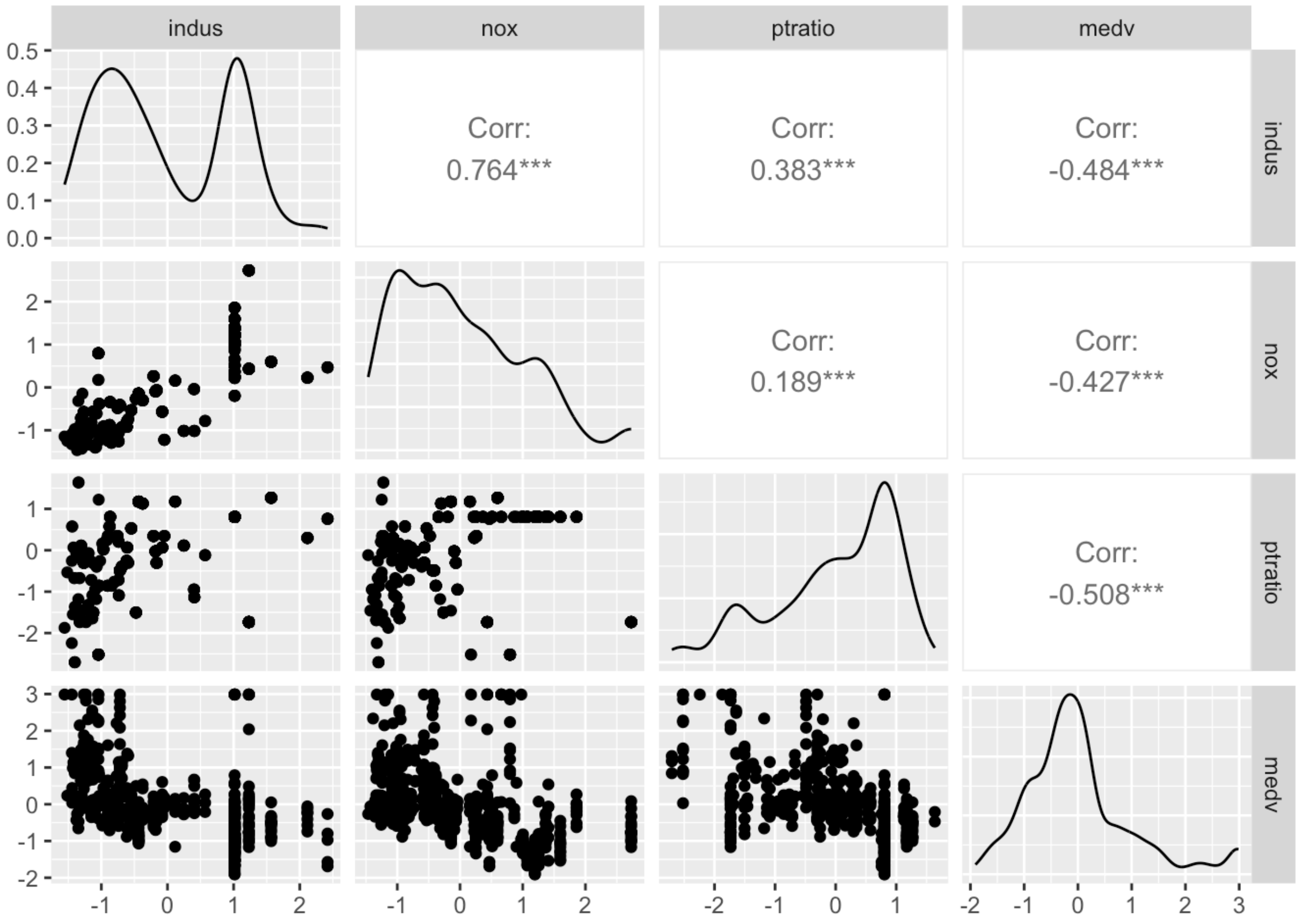
```
boston_housing <- read.table("BostonHousing.txt", header = TRUE, sep = ",") %>%
    scale() %>% as.data.frame() %>% dplyr::select(-chas)
```

Recall our data dictionary and some scatter plot of some variables of interest:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- PWORK - percentage of population working class
- MEDV - Median value of homes (in $1000’s)

Exploratory Data Analysis

```
ggpairs(select(boston_housing, indus, nox, ptratio, medv))
```



`NOX` and `PTRATIO` have a curved association. `INDUS` and `PTRATIO` have a non-linear (curved) association. `INDUS` has a bimodial distribution. `NOX` and `PTRATIO` are skewed right and left respectfully.

Note: Ideally, to compare relationships across scatter plots, we should normalize the variables, i.e., substract the mean and divide by the SD. This standardization does not impact the correlation. It puts all the measurements in standard units so scatter plots are more easily compared. The automatic scaling of the pairs of scatter plots in the `ggpairs()` output effectively do this for us.

```
model = lm(medv ~ indus + nox, data = boston_housing)
summary(model)
```

```
## 
## Call:
## lm(formula = medv ~ indus + nox, data = boston_housing)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -1.357 -0.530 -0.175  0.328  3.512 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.81e-16   3.88e-02    0.00    1.000    
## indus       -3.78e-01   6.01e-02   -6.28  7.3e-10 ***
## nox         -1.39e-01   6.01e-02   -2.31    0.021 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.872 on 503 degrees of freedom
## Multiple R-squared:  0.242,  Adjusted R-squared:  0.239 
## F-statistic: 80.3 on 2 and 503 DF,  p-value: <2e-16
```

The coefficient for non-retail proportion is interpreted in the presence of pollution. For example, if we choose a value for pollution of 0.5 and plug this value into the fitted model, then the relationsip between `INDUS` and `MEDV` is:

$$34.2883 + 0.5 * (-11.0291)) + indus * (-0.5062)$$

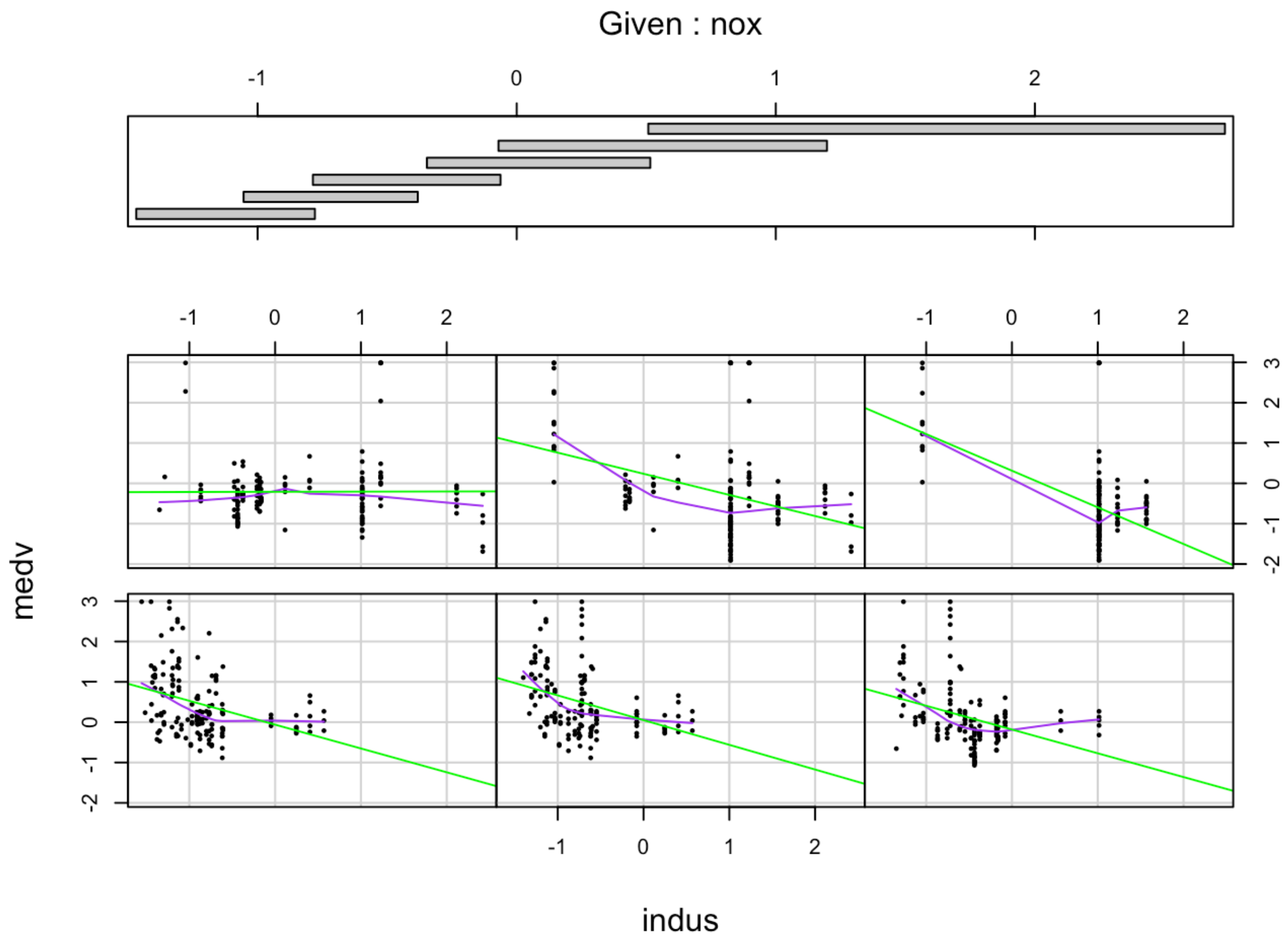
For pollution of 0.4, the relationship is:

$$34.2883 + 0.4 * (-11.0291)) + indus * (-0.5062)$$

Notice that the intercept changes, but the slope of these two lines remains the same. The implication is that the linear relationship remains the same for all values of `NOX` , i.e., the slope does not change, regardless of pollution levels.

We can examine this property with a coplot that conditions on various ranges of `NOX` . In short, the data is split-up into small ranges of `NOX` , and a corresponding scatter plot and regression line of `INDUS` on `MEDV` is visualized for each range.

```
coplot(medv ~ indus | nox, data = boston_housing,
       panel=function(x,y,...) {
         panel.smooth(x,y,span=.8,iter=5, col.smooth = "purple", ...)
         abline(lm(y ~ x), col="green")
       } ,
       pch = 19, cex = 0.3,
       ylab = "medv")
```



The green line represents the best linear regression fit line conditioning on a specific `NOX` value. The purple lines are a non-linear fit between `INDUS` and `MEDV` conditioning on a specific `NOX` value. The slopes are different in each plot, meaning the assumptions are violated. We might want to add an interaction term INDUS:NOX.

```
model2 = lm(medv ~ indus + nox + ptratio, data = boston_housing)
summary(model2)
```

```
##
## Call:
## lm(formula = medv ~ indus + nox + ptratio, data = boston_housing)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.564 -0.492 -0.120  0.301  3.696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.79e-16   3.51e-02    0.00    1.000
## indus       -1.44e-01   5.88e-02   -2.45    0.015 *
## nox         -2.40e-01   5.53e-02   -4.34  1.7e-05 ***
## ptratio     -4.07e-01   3.87e-02  -10.53  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.79 on 502 degrees of freedom
## Multiple R-squared:  0.379,  Adjusted R-squared:  0.375
## F-statistic:  102 on 3 and 502 DF,  p-value: <2e-16
```

When we compare the significance of the coefficients on `INDUS` and `NOX` between this regression and the 2-variate regression from earlier, we see that `INDUS` becomess less significant while `NOX` becomes more significant with the addition of `ptratio` in or linear model. In order to asses the implications this result may have for a casual study of the effect of non-retail industry (`INDUS`) on home value in Boston, we would need to make sure all confounding variables are added before determining a causal relationship between `INDUS` , `NOX` , and `MEDV` .

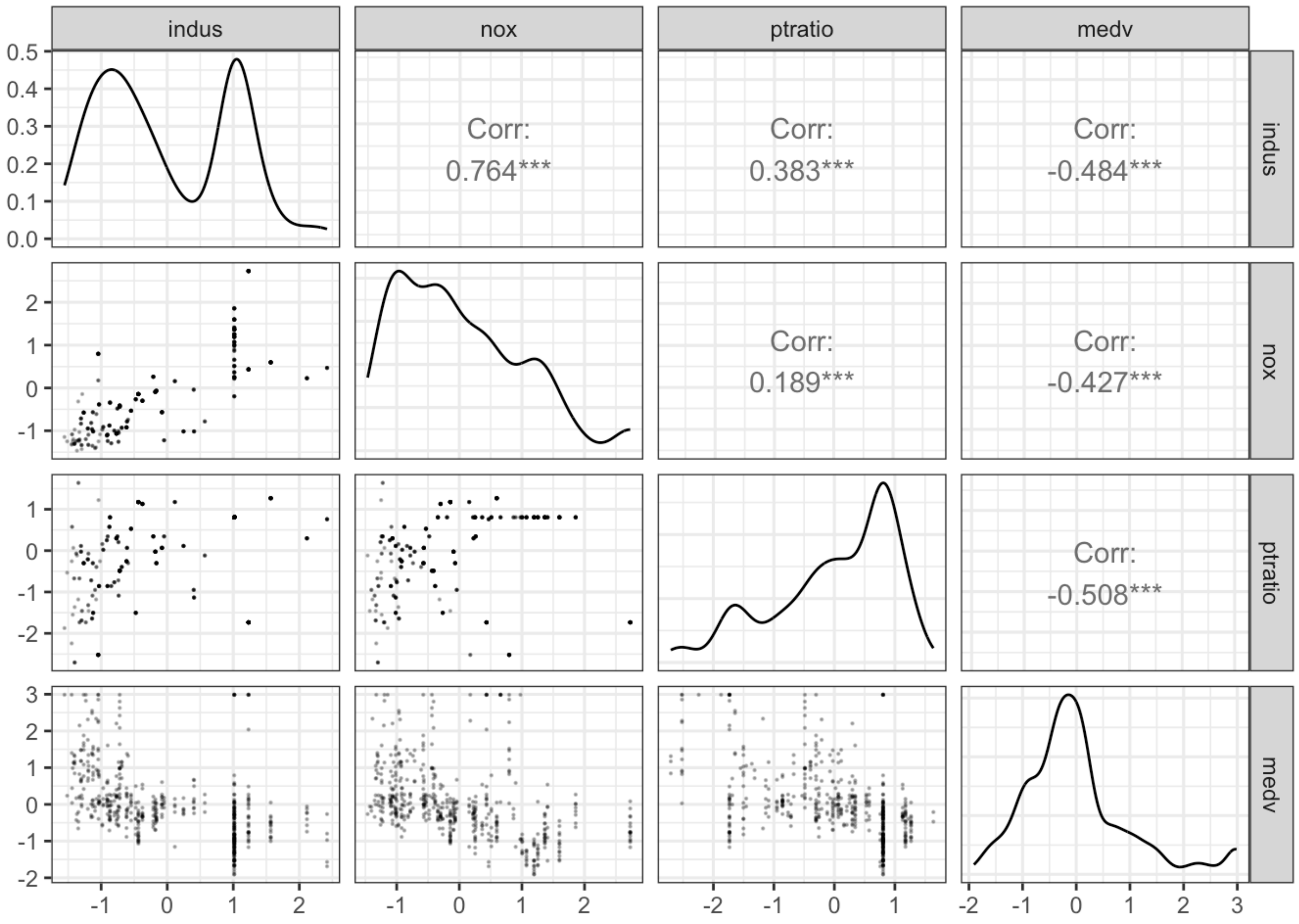
With most covariates in this dataset, you can find regression specifications for which the sign on the estimate will even flip. As we saw before, the slope of `INDUS` on `MEDV` may decrease conditional on the value of `NOX` . One way that we may encode for this in a regression model is via **interaction** functions of `INDUS` and `NOX` .

```
model3 = lm(medv ~ indus + nox + indus*nox,
            data = boston_housing)
summary(model3)
```

```
##
## Call:
## lm(formula = medv ~ indus + nox + indus * nox, data = boston_housing)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.354 -0.535 -0.177  0.335  3.506
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0265     0.0579    0.46    0.647
## indus        -0.3850     0.0614   -6.28  7.5e-10 ***
## nox          -0.1226     0.0657   -1.87    0.063 .
## indus:nox    -0.0348     0.0563   -0.62    0.537
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.873 on 502 degrees of freedom
## Multiple R-squared:  0.243,  Adjusted R-squared:  0.238
## F-statistic: 53.6 on 3 and 502 DF,  p-value: <2e-16
```

We find that the interaction term has a negative coefficient (though all terms are now insignificant), indicating that the slope of `INDUS` is decreasing by approximately -0.04 for each increase in `NOX` of 0.1.

```
boston_housing %>%
    dplyr::select(indus, nox, ptratio, medv) %>%
    GGally::ggpairs(
        ## optional:
        lower = list(continuous = wrap("points", alpha = 0.3, size=0.1))) +
    theme_bw()
```



NOX (nitric oxides concentration) in a neighborhood is highly *positively* correlated with proportion of non-retail business acres, so we would imagine that it is important to consider their confidence intervals jointly if we are interested in both.

We will consider the following two models (one controlling for several other covariates, the other is not):

Joint confidence region

```
lm_only_inference_covars <- lm(medv ~ indus + nox, data = boston_housing)
lm_include_confounders <- lm(medv ~ indus + nox + ., data = boston_housing)

summary(lm_only_inference_covars)
```

```
##
## Call:
## lm(formula = medv ~ indus + nox, data = boston_housing)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.357 -0.530 -0.175  0.328  3.512
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.81e-16   3.88e-02    0.00    1.000
## indus       -3.78e-01   6.01e-02   -6.28  7.3e-10 ***
## nox         -1.39e-01   6.01e-02   -2.31    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.872 on 503 degrees of freedom
## Multiple R-squared:  0.242,  Adjusted R-squared:  0.239
## F-statistic: 80.3 on 2 and 503 DF,  p-value: <2e-16
```

```
summary(lm_include_confounders)
```

```
##
## Call:
## lm(formula = medv ~ indus + nox + ., data = boston_housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3894 -0.3046 -0.0672  0.2274  2.8881
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.92e-17   2.34e-02    0.00  1.00000
## indus        2.54e-02   4.66e-02    0.55  0.58563
## nox         -2.32e-01   4.90e-02   -4.73  2.9e-06 ***
## crim        -1.19e-01   3.11e-02   -3.83  0.00014 ***
## zn           1.21e-01   3.55e-02    3.40  0.00073 ***
## rm           2.82e-01   3.24e-02    8.71  < 2e-16 ***
## age          1.83e-02   4.11e-02    0.44  0.65707
## dis         -3.44e-01   4.66e-02   -7.38  6.8e-13 ***
## rad          2.96e-01   6.36e-02    4.65  4.3e-06 ***
## tax         -2.60e-01   6.98e-02   -3.73  0.00021 ***
## ptratio     -2.30e-01   3.13e-02   -7.35  8.2e-13 ***
## pwork       -4.37e-01   3.96e-02  -11.04  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.527 on 494 degrees of freedom
## Multiple R-squared:  0.729,  Adjusted R-squared:  0.723
## F-statistic:  121 on 11 and 494 DF,  p-value: <2e-16
```

Now let's conduct the F-test inversion procedure to create a joint CI for these models.

Let V_q be the rows/columns of $(X^\top X)^{-1}$ corresponding to the covariates for which we are making the joint confidence region.

In the F-stat computation, we need the inverse of this V_q.

Q2: Complete the following function that takes a vector of variable names and the data frame, and returns V_q^{-1}.

```
getVqNeg1 <- function(variable_names = c("indus", "nox"), data = boston_housing) {

    ## find which indices correspond to our variable names
    covar_indices <- c()

    for(n in variable_names) {
        index <- grep(n, colnames(data))
        covar_indices <- append(covar_indices, index)
    }

    data <- as.matrix(data)
    ## TODO: invert X^\top X and extract the subsection corresponding to covar_indice
s
    ## HINT: inv() take a matrix inverse.
    xtx_inv= inv(t(data) %*% data)
    v_q = xtx_inv[covar_indices,covar_indices]
    v_q_neg1 = inv(v_q)


    return(v_q_neg1)
}


## test:
v_q_neg1 <- getVqNeg1(variable_names = c("indus", "nox"), data = boston_housing)
v_q_neg1
```

```
##       indus   nox
## indus 138.0  34.8
## nox    34.8 119.4
```

Recall if we want to test that some candidate values $\vec{\beta}_q^{(0)}$, are in the joint confidence region, the relevant F-stat to compute is:

$$\frac{(\hat{\beta}_q - \vec{\beta}_q^{(0)})^\top V_q^{-1}(\hat{\beta}_q - \vec{\beta}_q^{(0)}) \,/\, q}{RSS \,/\, (n-p-1)}$$

where $\hat{\beta}_q$ and RSS are from the unrestricted model.

```
getFstat <- function(variable_names = c("indus", "nox"), beta_test = c(0,0), lm, v_q_
neg1) {
    ## get RSS from unrestricted model.
    rss <- sum(resid(lm)^2)

    ## get estimates of parameters of interest
    lm_sum <- summary(lm)

    ## get coef indices
    coef_inds <- sapply(variable_names, function(n){
        which(rownames(lm_sum$coefficients) == n)
        })

    ## put relevant coefficient estimates into vector:
    beta_hat <- lm_sum$coefficients[coef_inds]



    ## other params:
    q <- length(variable_names)
    p <- nrow(lm_sum$coefficients) - 1
    n <- length(resid(lm))

    ## compute F-statistic
    f_stat_num <- (t(beta_hat - beta_test) %*% v_q_neg1 %*% (beta_hat - beta_test)) /
q
    f_stat_denom <- rss / (n - p - 1)
    f_stat <- f_stat_num / f_stat_denom


        f_stat <- as.numeric(f_stat)
    return(f_stat)
}
getFstat(lm = lm_include_confounders, v_q_neg1 = v_q_neg1)
```

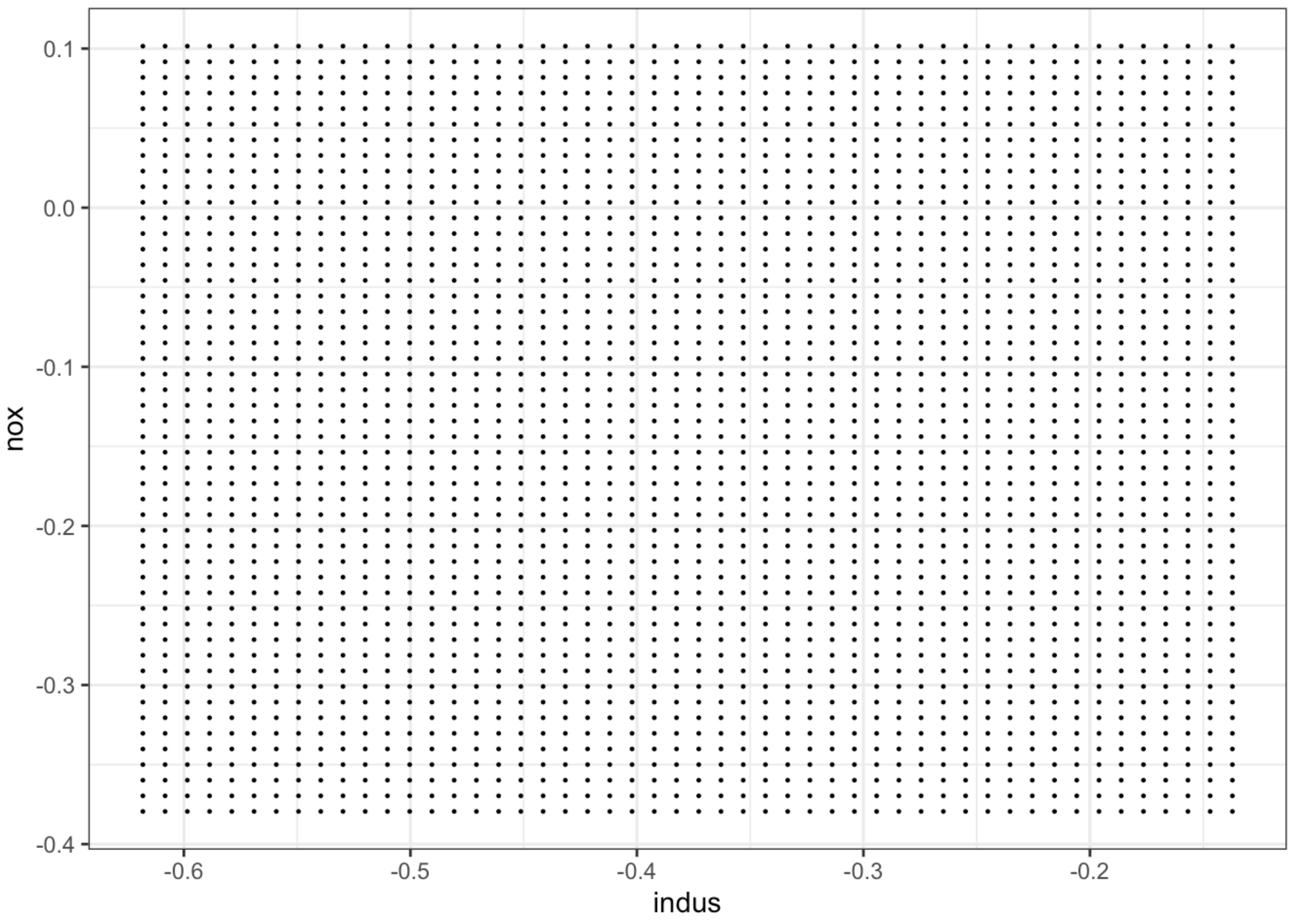
```
## [1] 11
```

The following quick function will get the level-α critical value from relevant F-distribution:

```
getFalpha <- function(alpha = 0.05, df1, df2) {
    return(
        qf(1 - alpha, df1, df2)
    )
}
```

We will test a grid of values for the inversion procedure. Please briefly review the following function that will get a grid of candidate values within 3 standard errors of the 1D parameter estimates:

```r
getCandidatesGrid <- function(lm, variable_names = c("indus", "nox"), n_points = 50)
{
    lm_sum <- summary(lm)

    ## get coef indices
    coef_inds <- sapply(variable_names, function(n){
        which(rownames(lm_sum$coefficients) == n)
        })

    coef_ests <- lm_sum$coefficients[coef_inds, 1]
    coef_std_errors <- lm_sum$coefficients[coef_inds, 2]

    grid_max <- coef_ests + 4 * coef_std_errors
    grid_min <- coef_ests - 4 * coef_std_errors

    p <- length(variable_names)
    ## create grid:
    grids_1d <- lapply(1:p, function(i) {pracma::linspace(
        grid_min[i], grid_max[i], n = n_points)} )

    grid_2d <- expand.grid(grids_1d) %>% as.data.frame()

    names(grid_2d) <- variable_names

    return(grid_2d)
}

### plot the example outputs of this function:
grid_example <-
    getCandidatesGrid(lm = lm_only_inference_covars, variable_names = c("indus", "nox
"))


grid_example %>% ggplot(aes(x=indus, y=nox)) + geom_point(size = 0.2) + theme_bw()
```



The above are the “candidates” we will test.

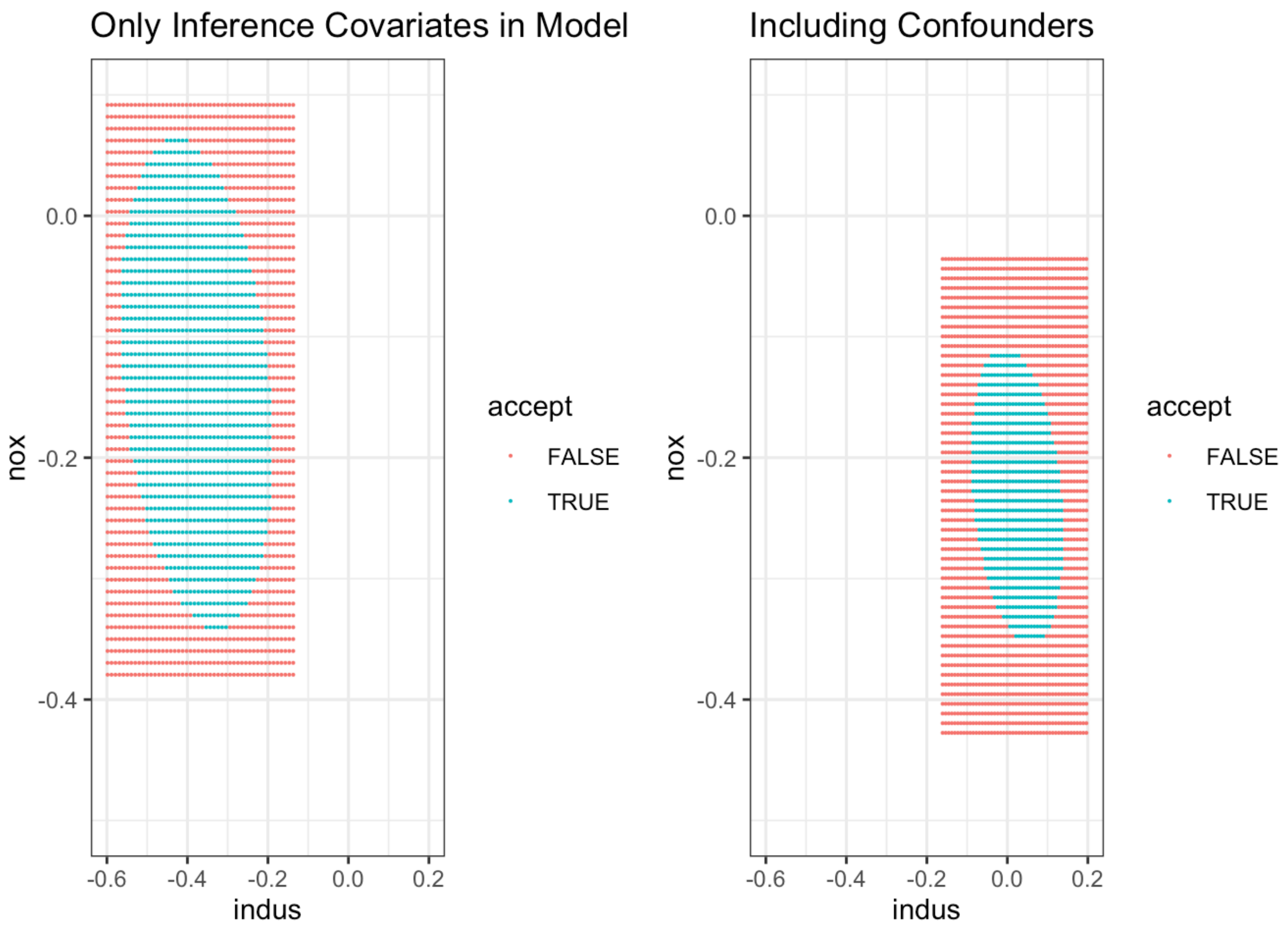
```
invertFtest <- function(
    variable_names = c("indus", "nox"),
    lm = lm_include_confounders,
    data = boston_housing,
    alpha = 0.05) {

    ## get constants of interest
    n <- nrow(boston_housing)
    q <- length(variable_names)
    p <- nrow(summary(lm)$coefficients) - 1

    ## get F_alpha
    f_alpha <- getFalpha(.05, q, n-p-1)



    ## get vq_neg1
    v_q_neg1 <- getVqNeg1(variable_names, data)

    ## get grid of candidate values:
    candidate_grid <-
        getCandidatesGrid(lm, variable_names)

    acceptance_vec <- c()

    for(i in 1:nrow(candidate_grid)) {

        ## need this as a vector for our function
        candidate_i_as_vec <- candidate_grid[i, ] %>% t() %>% c()
        ## compute f_stat for row_i
        f_stat_i <- getFstat(variable_names, beta_test = candidate_i_as_vec, lm, v_q_
neg1)



        ## compute if the f_stat_i is accepted into the confidence region
        accept_i <- (f_stat_i <= f_alpha)



        acceptance_vec <- append(acceptance_vec, accept_i)
    }

    return(candidate_grid %>% mutate(accept = acceptance_vec))
}
```

The following code chunk will now use our function to produce the joint confidence region for the `nox` , `indus` only model and the model that includes our potential confounders. We also go ahead and plot the regions on the same range:

```
confidence_region1 <- invertFtest(
    variable_names = c("indus", "nox"),
    lm = lm_only_inference_covars,
    data = boston_housing,
    alpha = 0.05)

confidence_region2 <- invertFtest(
    variable_names = c("indus", "nox"),
    lm = lm_include_confounders,
    data = boston_housing,
    alpha = 0.05)

p1 <- confidence_region1 %>%
    ggplot(aes(x = indus, y = nox, color = accept)) + geom_point(size=0.05) +
    ggtitle("Only Inference Covariates in Model") +
    xlim(-0.6, 0.2) +
    ylim(-0.5, 0.1) + theme_bw()

p2 <- confidence_region2 %>%
    ggplot(aes(x = indus, y = nox, color = accept)) + geom_point(size=0.05) +
    ggtitle("Including Confounders") +
    xlim(-0.6, 0.2) +
    ylim(-0.5, 0.1) + theme_bw()

ggarrange(p1, p2, nrow = 1)
```

```
## Warning: Removed 148 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 100 rows containing missing values (`geom_point()`).
```



The 2 confidence intervals are in two different locations the overall size of our confidence interval decreases when we add confounders to our model. The ellipse is negaitvely sloped because indus and nox are both positively correlated, when we give more weight to one covariate we take away some weight from the other.

While we have gone through the process of computing these regions one point at a time, note that one could more directly get the region by plugging the equation:

$$F_0(\vec{\beta}_q^{(0)}) \leq F_\alpha$$

into an ellipse solver. This would help us more quickly find the region for many covariates, for which the size of the grid we would have to examine would be very very large (i.e. for two covariates, our grid size is 50×50. for p covariates, the grid size is 50^p).