Homework Assignment 1

Greg Forkutza
Student ID: 400277514

24 October, 2023

Contents

BMB: table of contents is a little bit of overkill?

1

`dataset::longley` is a macroeconomic data set with 7 economical variables, observed yearly from 1947 to 1962.

```
head(longley)
```

```
##      GNP.deflator     GNP Unemployed Armed.Forces Population Year Employed
## 1947         83.0 234.289      235.6        159.0    107.608 1947   60.323
## 1948         88.5 259.426      232.5        145.6    108.632 1948   61.122
## 1949         88.2 258.054      368.2        161.6    109.773 1949   60.171
## 1950         89.5 284.599      335.1        165.0    110.929 1950   61.187
## 1951         96.2 328.975      209.9        309.9    112.075 1951   63.221
## 1952         98.1 346.999      193.2        359.4    113.270 1952   63.639
```

(a)

We first run a linear model, using the employment numbers as the response on all the rest of the variables. Then we show using VIF that there is multi-collineairty.

```
# Liner model for original data
m1 <- lm(Employed ~ . , data = longley)
summary(m1)
```

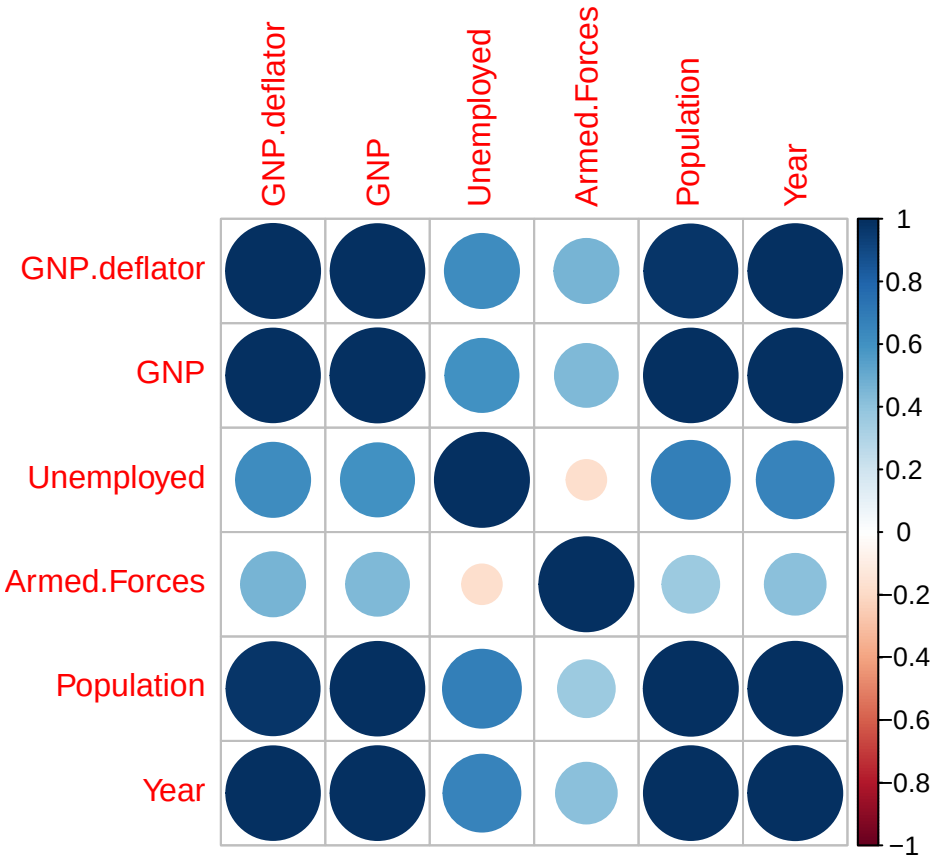
```
##
## Call:
## lm(formula = Employed ~ ., data = longley)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41011 -0.15767 -0.02816  0.10155  0.45539
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 **
## GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141
## GNP          -3.582e-02  3.349e-02  -1.070 0.312681
## Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 **
## Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
## Population   -5.110e-02  2.261e-01  -0.226 0.826212
## Year          1.829e+00  4.555e-01   4.016 0.003037 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3049 on 9 degrees of freedom
## Multiple R-squared:  0.9955, Adjusted R-squared:  0.9925
## F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

It is suspicious that all of the coefficients are quite small. (**BMB**: why? doesn't that depend on the units of the predictors and response variable?) Furthermore the t values are quite small for some of the variables. If we look at the correlation plot for predictors we see

```
data <- longley %>%
        select(-Employed)
```

```r
# Define function to convert covariance matrix to correlation matrix.

cov_to_cor <- function(covmat) {

   # Extract std dev's and name
     SD <- diag(covmat) %>%
           sqrt()
     names(SD) <- colnames(covmat)

  # Extract dimensions of covariance matrix
  nrow <- dim(covmat)[1]
  ncol <- dim(covmat)[2]

  # Scale the covariance matrix
  for (i in 1: nrow ) {
    for (j in 1:ncol) {
      scale <- SD[i] * SD[j]
      covmat[i,j] <- covmat[i,j] / scale
  }
  }
  return(covmat)
}
```

```r
corrplot::corrplot(cov_to_cor(cov(data)))
```



We observe that GNP.deflator, GNP and Population are very correlated in the correlation plot and also have small t-values for their corresponding coefficients in the linear model.

Now us compute the Variance Inflation Factor (VIF) for the predictors. It is a useful measure of multi-collinearity in a data set. The VIF is the ratio of the variance of estimating some parameter in a model that includes multiple other predictors by the variance of a model constructed using only one term. It is defined as

$$VIF = \frac{1}{1 - R^2},$$

where R^2 is the coefficient of determination.

```
compute_vif <- function(data, response_col) {
  # Drop the response variable to get only predictors
  predictors <- data %>%
                select(-response_col)

  # Initialize an empty vector to store VIF values
  vif_values <- numeric(ncol(predictors))
```

```r
  # Loop through each predictor to calculate its VIF
  for (i in seq_along(predictors)) {

    # Isolate the predictor of interest
    predictor_of_interest <- names(predictors)[i]

    # Create a formula to regress the predictor of interest on all other predictors
    formula <- as.formula(paste(predictor_of_interest, "~ ."))

    # Fit the linear model
    lm_model <- lm(formula, data = predictors)

    # Compute R-squared
    r_squared <- summary(lm_model)$r.squared

    # Compute VIF
    vif_values[i] <- 1 / (1 - r_squared)
  }

  # Add names
  names(vif_values) <- names(predictors)

  return(vif_values)
}

# Compute VIF values
vif_results <- compute_vif(longley, "Employed")
print(vif_results)
```

```
## GNP.deflator          GNP   Unemployed Armed.Forces   Population         Year
##    135.53244   1788.51348     33.61889      3.58893    399.15102    758.98060
```

The rule of thumb is that a VIF value above 5-10 indicates multicollinearity between that predictor and the other predictors. We observe that all variables other than Armed.Forces are »10.

Since the predictors are correlated and that makes our estimates of coefficients correlated, we can remove the multi-collinearity by using principal components as the predictors in our linear model. Therefore we will use all the predictors. After we compute the principal components we will interpret them in terms of the

Table 1: Proportions of Variance Explained by Components (Covariance Matrix)

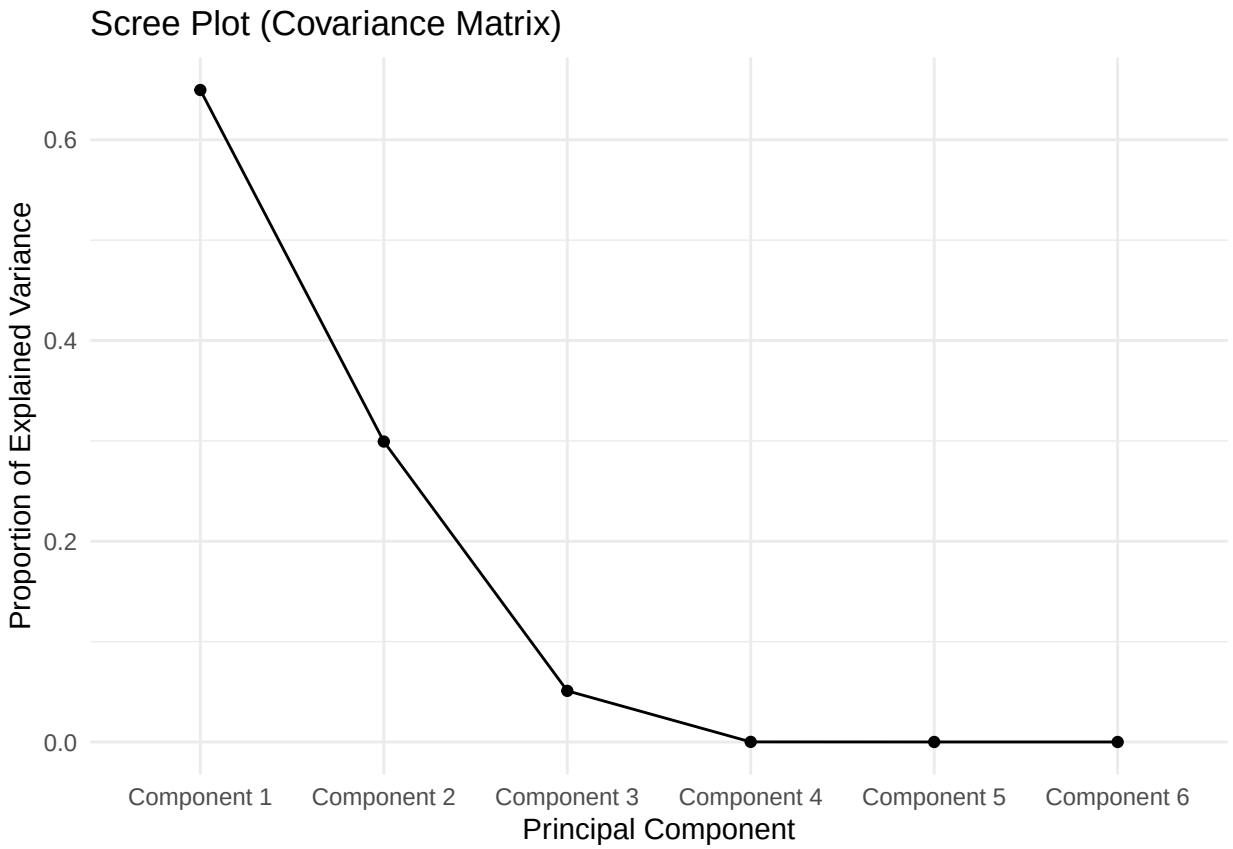
Component	Proportion
Component 1	0.65
Component 2	0.30
Component 3	0.05
Component 4	0.00
Component 5	0.00
Component 6	0.00

Table 2: Proportions of Variance Explained by Components (Correlation Matrix)

Component	Proportion
Component 1	0.51
Component 2	0.19
Component 3	0.14
Component 4	0.07
Component 5	0.06
Component 6	0.03

original predictors. We will do PCA using both the covariance and the correlation matrix, so they can be compared downstream when fitting the linear model.

```
# Define function to compute proportion of total population variation due to the
# k'th principal component.
prop_pc <- function(mat) {
  ei <- eigen(mat)
  evalues <- ei$values
  total <- sum(evalues)
  dim <- length(diag(mat))
  vec <- rep(NA, dim)

  for (i in 1:dim) {
  vec[i] <- evalues[i]/total
  }
  return(vec)

}
```


Scree Plot (Covariance Matrix)
Proportion of Explained Variance
0.6
0.4
0.2
0.0
Component 1
Component 2
Component 3
Component 4
Component 5
Component 6
Principal Component

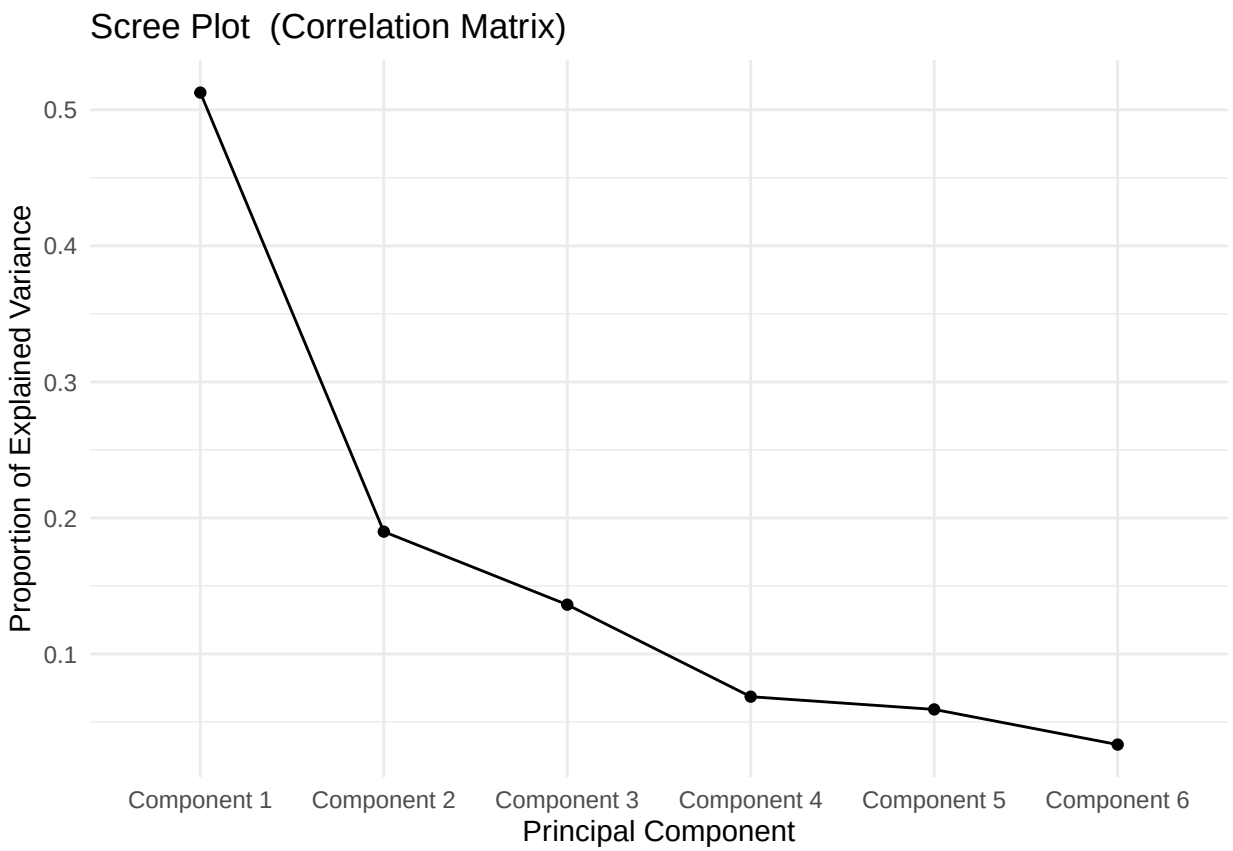

Scree Plot (Correlation Matrix)
Proportion of Explained Variance
0.5
0.4
0.3
0.2
0.1
Component 1
Component 2
Component 3
Component 4
Component 5
Component 6
Principal Component

Table 3: Eigenvectors for the First Two Principal Components (Covariance)

	Component 1	Component 2
GNP.deflator	-0.082	0.034
GNP	-0.756	0.319
Unemployed	-0.626	-0.580
Armed.Forces	-0.158	0.748
Population	-0.054	0.013
Year	-0.037	0.012

Table 4: Eigenvectors for the First Four Principal Components (Correlation)

	Component 1	Component 2	Component 3
GNP.deflator	-0.471	0.003	0.073
GNP	-0.358	0.408	0.593
Unemployed	-0.434	0.404	0.064
Armed.Forces	-0.288	0.403	-0.794
Population	-0.440	-0.507	-0.014
Year	-0.430	-0.501	-0.090

Method	Component	Interpretation
Covariance-based Eigenvectors	Component 1	Strong negative loadings on GNP and Unemployed. This component might capture general economic factors related to both the gross national product and unemployment.
Covariance-based Eigenvectors	Component 2	Strong positive loadings on Armed.Forces,moderate positive loading on GNP and moderate negative loading on Unemployed. This component may capture factors related to the military and the economy.
Correlation-based Eigenvectors	Component 1	Strong loadings on almost all variables except for Armed.Forces. This could be a "general factor" capturing broad economic and demographic aspects.
Correlation-based Eigenvectors	Component 2	Strong negative loadings on Population and Year, and positive loadings on GNP, Unemployed, and Armed.Forces. This might capture more specific year-over-year changes or economic factors separate from demographic growth.

Method	Component	Interpretation
Correlation-based Eigenvectors	Component 3	Strong negative loading on Armed.Forces and strong positive loading on GNP. This could represent aspects of the economy not associated with military activity.

(b)

Table 6: Description of Variables in the Longley Dataset

Variable	Description	Unit	Interpretation	Threshold
GNP.deflator	Gross National Product implicit price deflator	Index (year 1954 = 100)	Measure of level of prices	1 index point
GNP	Gross National Product	Billions of USD	Total value of goods and services	0.5 Billion USD
Unemployed	Number of unemployed	Thousands of people	People looking for a job	1K people
Armed.Forces	Number of people in the armed forces	Thousands of people	People in the military	1K people
Population	Non-institutionalized population	Thousands of people	People outside institutions	1K people
Year	Year of the observation	Years	Year recorded	1 Year
Employed	Number of people employed	Thousands of people	People who have jobs	1K people

(c)

```
# Center and Scale data
data_sc <- scale(data)

# Transform data with eigenvectors for covariation matrix
pc_scores_cov <- data_sc %*% eigenvectors_cov

# Transform data with eigenvectors for correlation matrix
pc_scores_cor <- data_sc %*% eigenvectors

# Fit new lm
response <- longley$Employed

mod_pc_cov <- lm(response ~ pc_scores_cov)
mod_pc_cor  <- lm(response ~ pc_scores_cor)
summary(mod_pc_cov)
```

```
##
## Call:
## lm(formula = response ~ pc_scores_cov)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.44203 -0.67573  0.02242  0.86219  1.77752
```

```
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                65.3170     0.3154 207.097  < 2e-16 ***
## pc_scores_covComponent 1  -2.1029     0.2272  -9.256 4.38e-07 ***
## pc_scores_covComponent 2   0.9253     0.3025   3.059  0.00915 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.262 on 13 degrees of freedom
## Multiple R-squared:  0.8882, Adjusted R-squared:  0.871
## F-statistic: 51.62 on 2 and 13 DF,  p-value: 6.542e-07
```

```
summary(mod_pc_cor)
```

```
##
## Call:
## lm(formula = response ~ pc_scores_cor)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99469 -0.19427 -0.04553  0.17479  1.03934
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               65.31700    0.12297 531.180  < 2e-16 ***
## pc_scores_corComponent 1 -1.43795    0.06595 -21.805 5.07e-11 ***
## pc_scores_corComponent 2 -5.18685    0.63432  -8.177 3.00e-06 ***
## pc_scores_corComponent 3 -1.63049    0.25602  -6.369 3.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4919 on 12 degrees of freedom
## Multiple R-squared:  0.9843, Adjusted R-squared:  0.9804
## F-statistic: 250.9 on 3 and 12 DF,  p-value: 4.349e-11
```

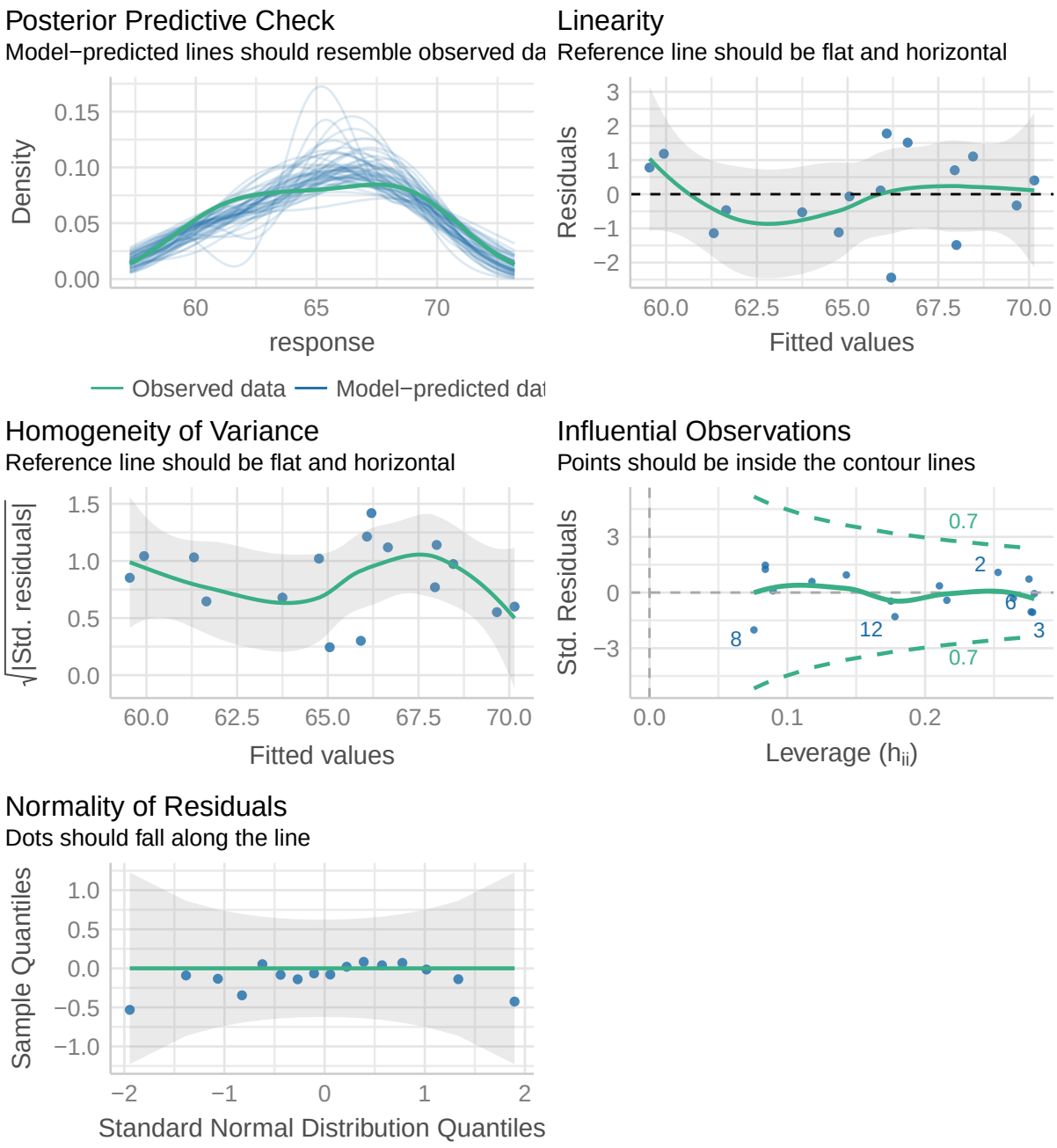
Both models seem to fit the data well. The R^2 value is high for both models but for the correlation model $R^2 = 99.85$ is very high. Both models show significant t-values for their coefficients, indicating that the

selected principal components are important predictors for the employment rate in both cases.

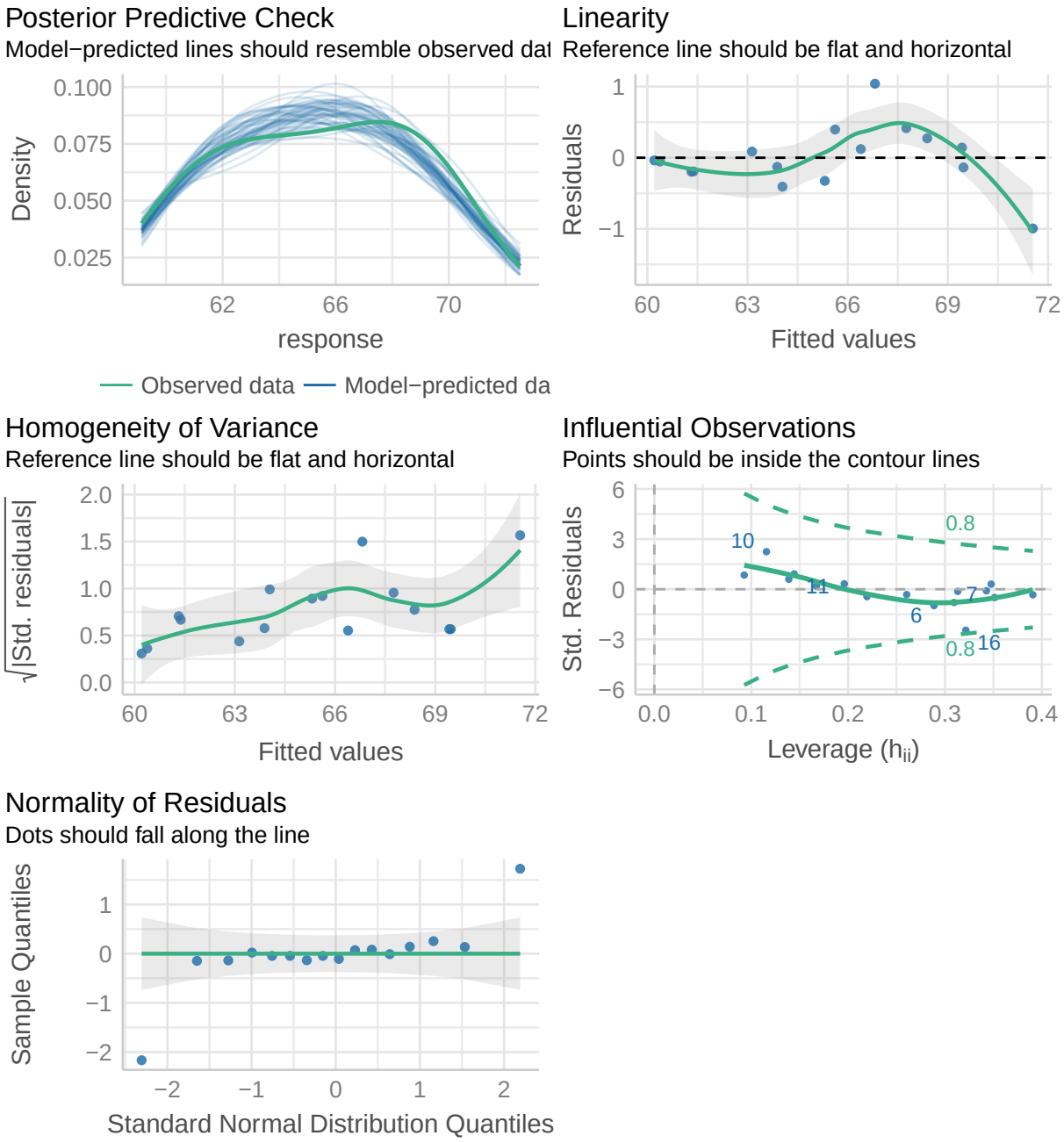
BMB: I would have said that with 15 observations and 6 potential predictors, you should have decided immediately that you needed to reduce the number of covariates …

(d)

```
check_cov <- performance::check_model(mod_pc_cov)
par(mfrow= c(2,2))
plot(check_cov, title = "PCR (Covariance")
```




```
check_cor <- performance::check_model(mod_pc_cor)
par(mfrow= c(2,2))
plot(check_cor, title = "PCR (Correlation)")
```



The posterior check looks much better for the correlation PCR (PCRcor) then for the covariance PCR (PCRcov). The linearity for PCRcov has wider CI's but it is more rectangular then PCRcor which appears fairly non-linear after in the upper half. A similar argument holds for the homogeneity of variance. The PCRcor model appears to have a borderline influential observation for the 16th data point. The PCRcov model does not appear to any influential observation. The normality of residuals is very good for both model except that PCRcor has an extreme looking point on both side of the 2nd quantile.

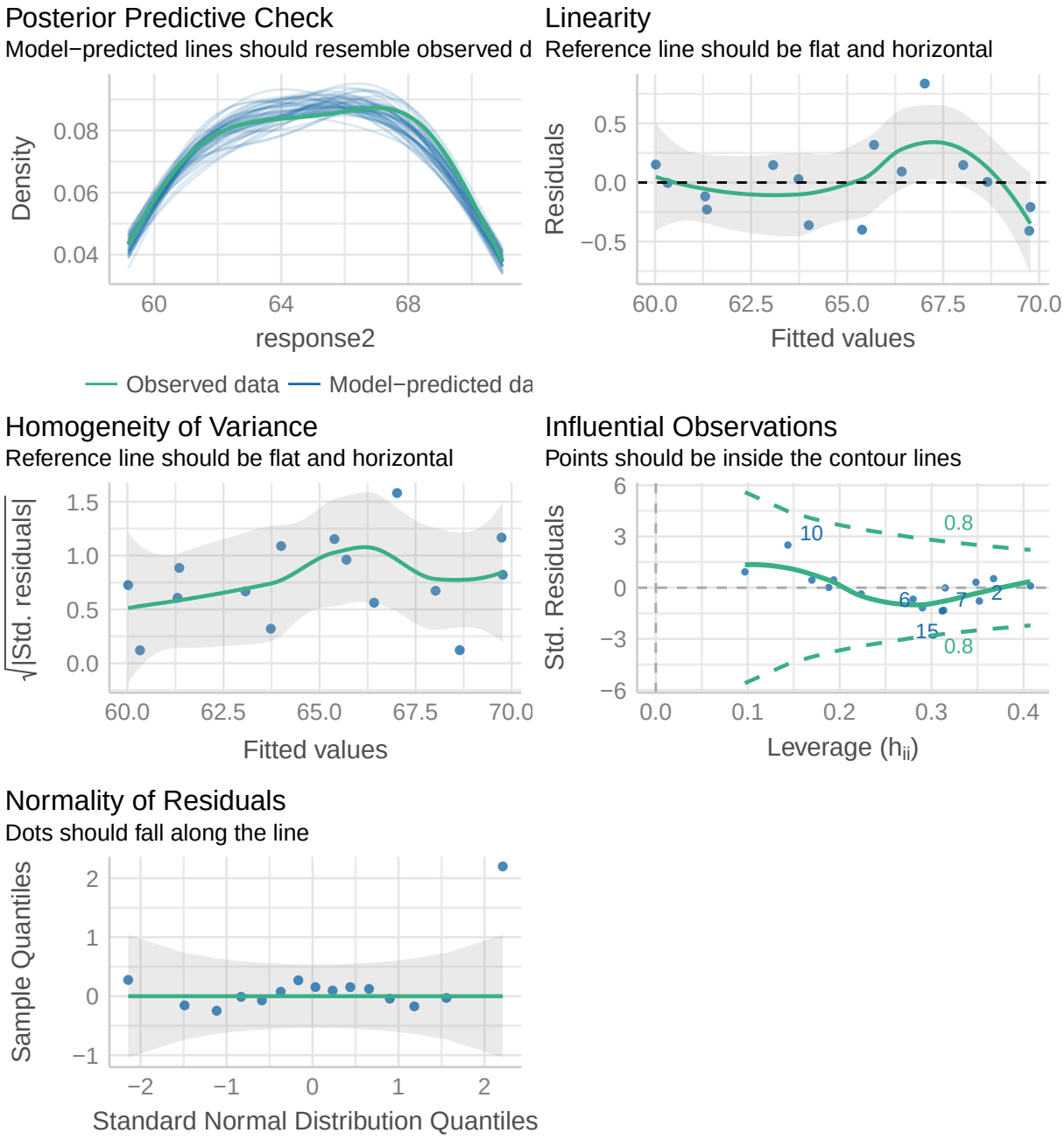
(e)

We will not drop any observations but for curiosity sake lets drop the possible influential observation for PCRcor and refit the model.

```
data2 <- data_sc[1:15,]
pc_scores_cor2 <- data2 %*% eigenvectors
response2<- response[1:15]
mod_pc_cor2  <- lm(response2 ~ pc_scores_cor2)
summary(mod_pc_cor)
```

```
##
## Call:
## lm(formula = response ~ pc_scores_cor)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99469 -0.19427 -0.04553  0.17479  1.03934
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               65.31700    0.12297 531.180  < 2e-16 ***
## pc_scores_corComponent 1 -1.43795    0.06595 -21.805 5.07e-11 ***
## pc_scores_corComponent 2 -5.18685    0.63432  -8.177 3.00e-06 ***
## pc_scores_corComponent 3 -1.63049    0.25602  -6.369 3.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4919 on 12 degrees of freedom
## Multiple R-squared:  0.9843, Adjusted R-squared:  0.9804
## F-statistic: 250.9 on 3 and 12 DF,  p-value: 4.349e-11
```

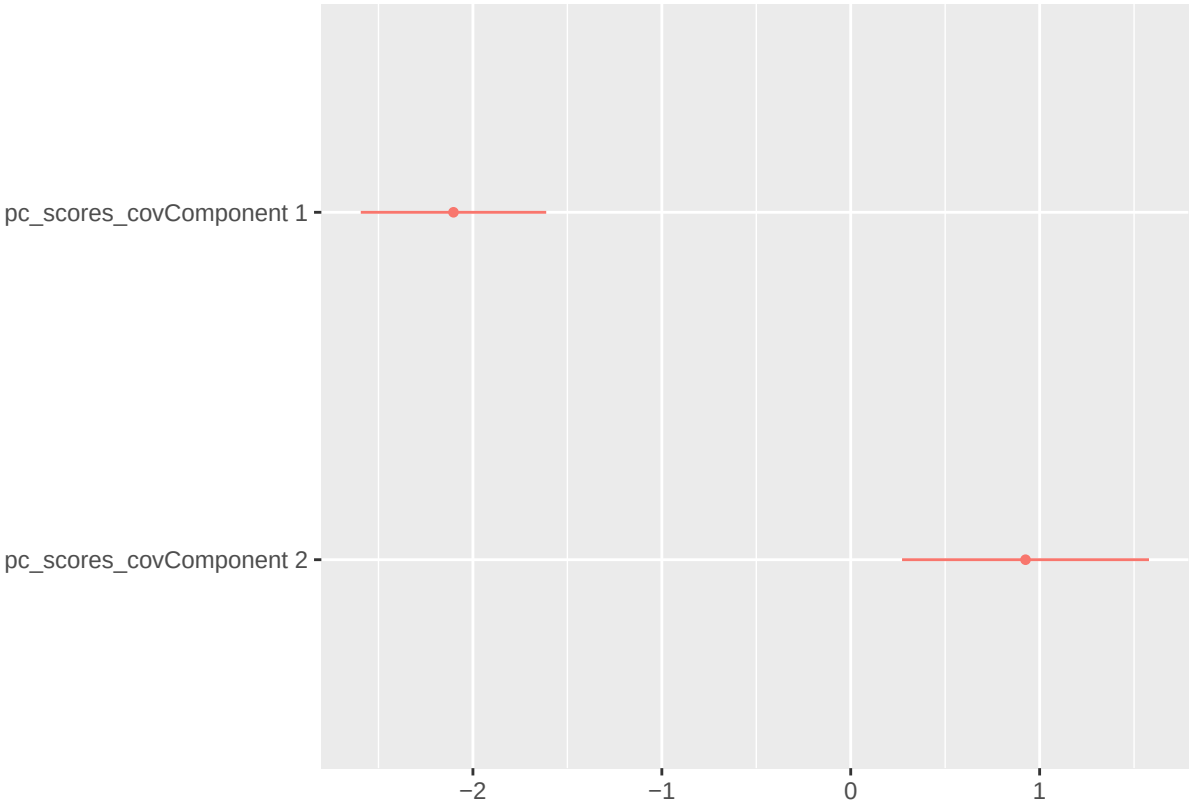
```
check_cor2 <- performance::check_model(mod_pc_cor2)
par(mfrow= c(2,2))
plot(check_cor2, title = "PCR (Correlation)")
```

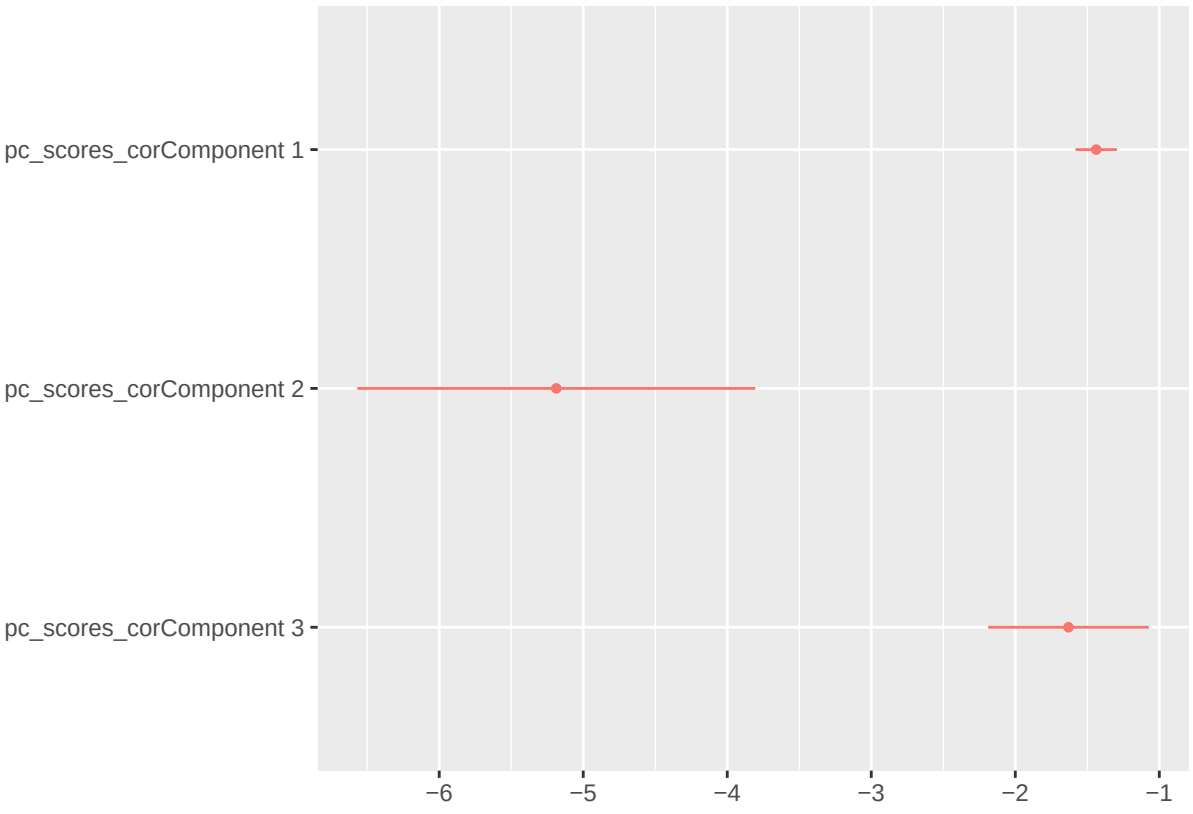


We see that the linearity and the homogeneity of variance both looks better now.

(f)

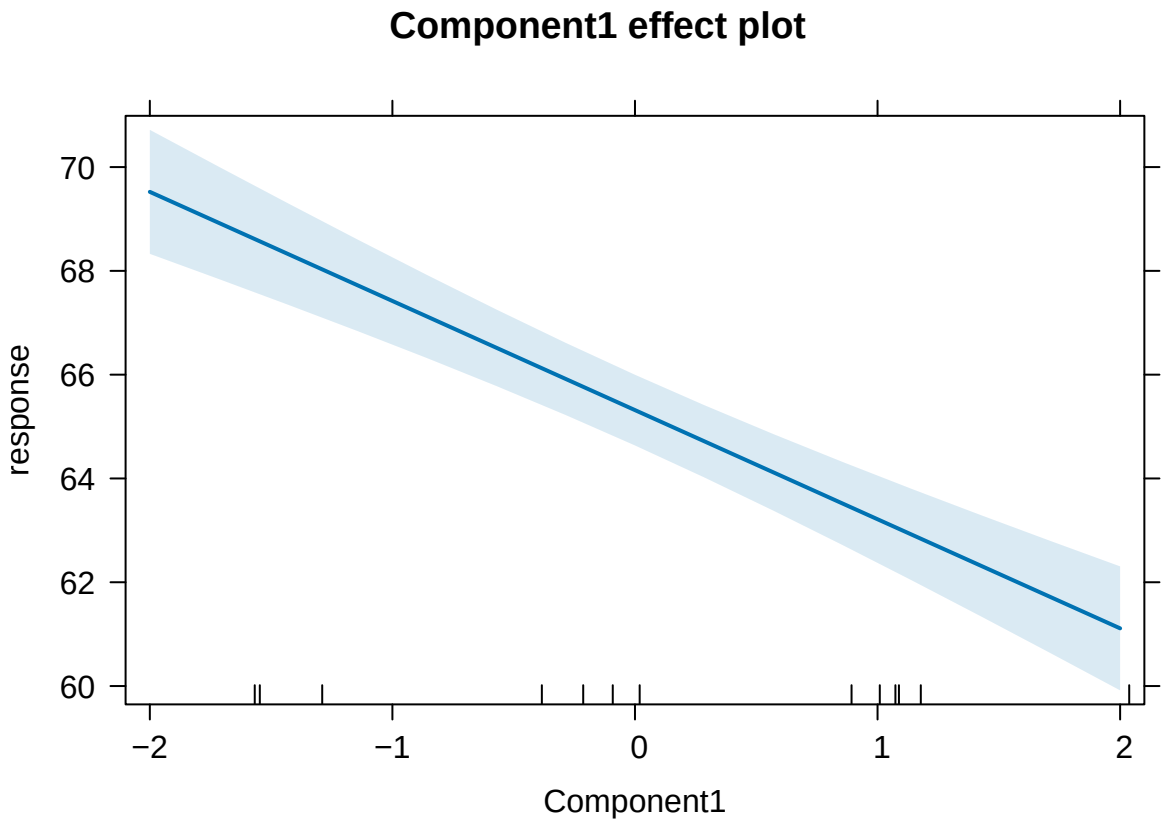
The data is scaled and centered as is custom when working with principal components.



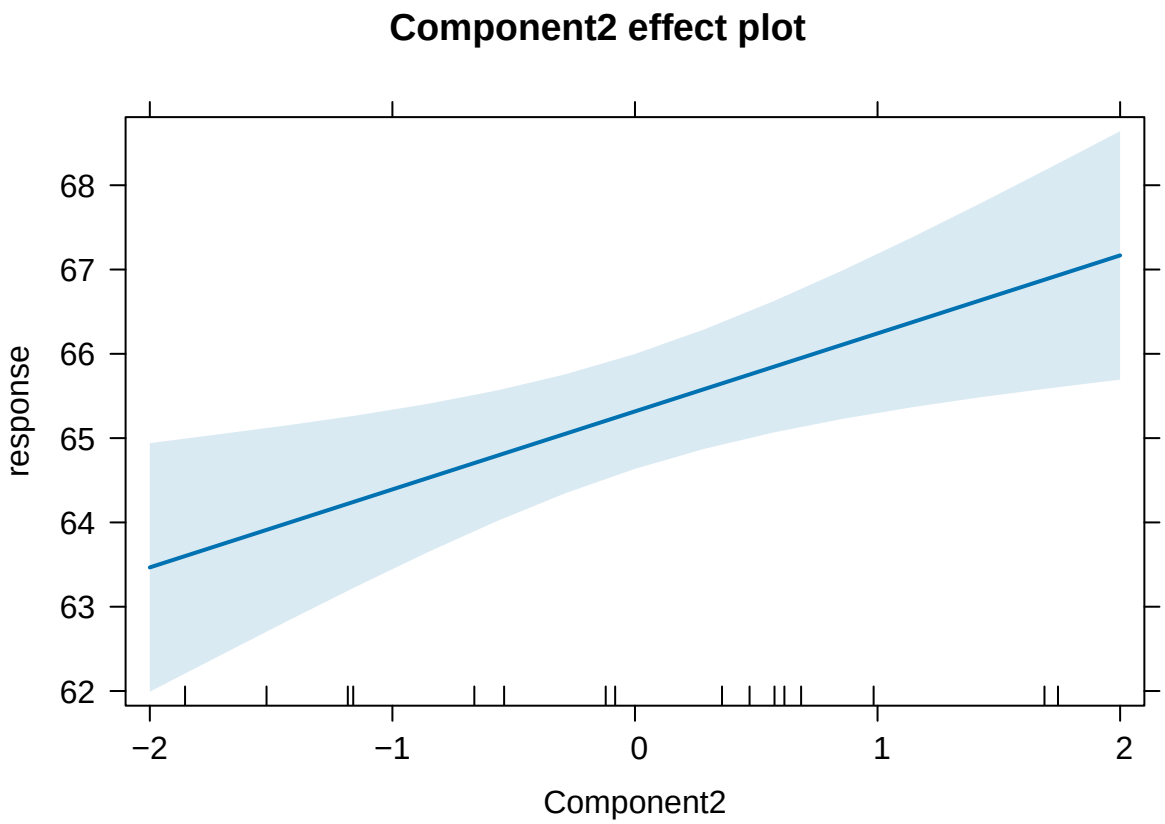

pc_scores_corComponent 1
pc_scores_corComponent 2
pc_scores_corComponent 3
−6
−5
−4
−3
−2
−1

(g)

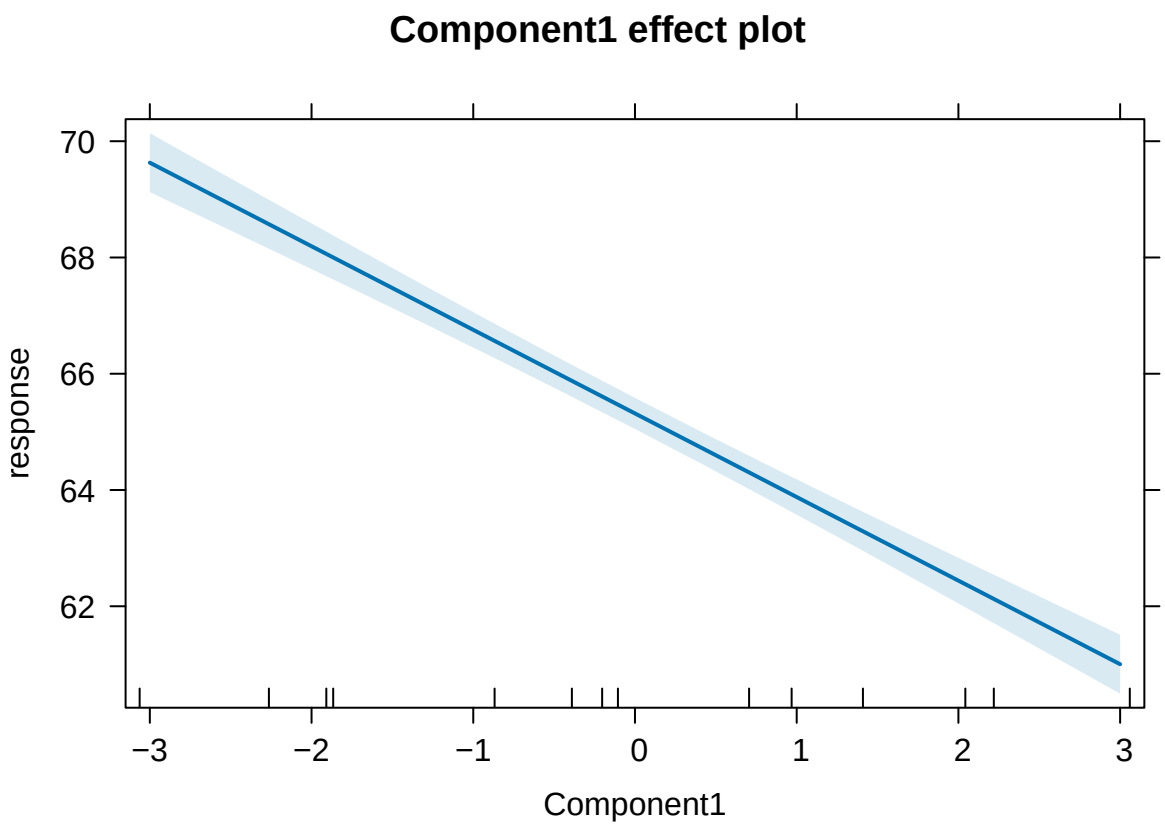
```
#effect() was very picky so had to make some changes below to make it happy.

# Rename columns to remove spaces  and convert matrix to data frame
colnames(pc_scores_cov)<- c("Component1", "Component2")
pc_scores_cov_df <- as.data.frame(pc_scores_cov)

colnames(pc_scores_cor)<- c("Component1", "Component2", "Component3")
pc_scores_cor_df <- as.data.frame(pc_scores_cor)



# Fitting the model using the data frame
mod_pc_cov_updated <- lm(formula = response ~ Component1 + Component2,
                         data = pc_scores_cov_df)
mod_pc_cor_updated <- lm(formula = response ~ Component1 + Component2 +
                           Component3,
                         data = pc_scores_cor_df)

# Creating the effect objects for each component
eff1_cov <- effect("Component1", mod_pc_cov_updated)
eff2_cov <- effect("Component2", mod_pc_cov_updated)

eff1_cor <- effect("Component1", mod_pc_cor_updated)
eff2_cor <- effect("Component2", mod_pc_cor_updated)
eff3_cor <- effect("Component3", mod_pc_cor_updated)

# Plotting the effect for each component
par(mfrow = c(3,2))
plot(eff1_cov)
```



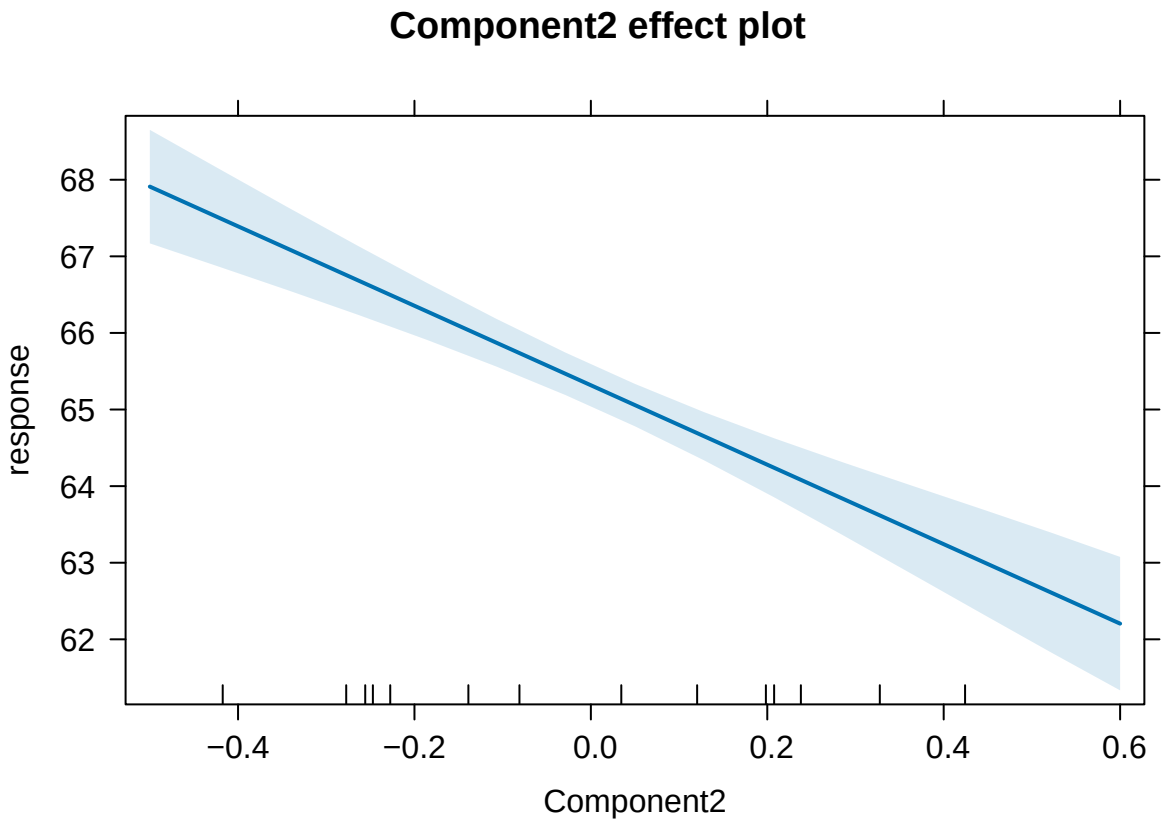

```
plot(eff2_cov)
```



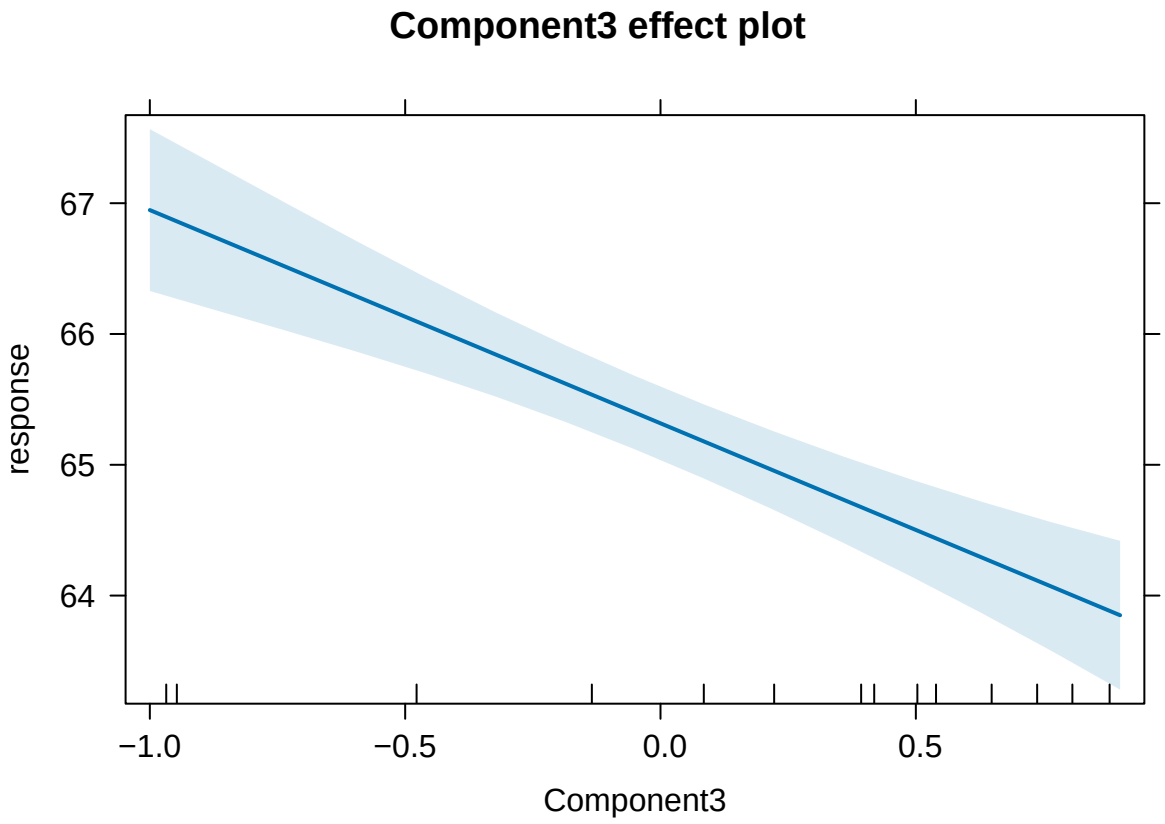
```
plot(eff1_cor)
```




```
plot(eff2_cor)
```




```
plot(eff3_cor)
```



For PCRcov we see that the main effect for the 1st principal component has a large negative magnitude. This makes sense at the corresponding eigenvector had large negative loadings on `Unemploymed` and `GNP`. Therefore when `Unemploymed` and `GNP` are low, employment is high. in this component are associated with higher employment. The 2nd principal component has a large positive loading on `Armed.Forces` and a moderate negative loading on `Unemployed`. Therefore when unemployment is low and enrollment in armed forces is high , this component is associated with low employment.

A similar interpreation can be done for the correlation plot (which I will not do).

2

```r
# Create a data frame with the unique combinations of Period and Treatment
design <- expand.grid(
  Period = factor(c("Before", "After")),
  Treatment = factor(c("Control", "Impact"))
)

# Define the contrasts for Period and Treatment
contrasts(design$Period) <- matrix(c(-1, 1), 2)
contrasts(design$Treatment) <- matrix(c(-1, 1), 2)

# Construct the minimal model matrix
model_matrix_1 <- model.matrix(~ Period * Treatment, design)
print(model_matrix_1)
```

```
##   (Intercept) Period1 Treatment1 Period1:Treatment1
## 1           1       1         -1                -1
## 2           1      -1         -1                 1
## 3           1       1          1                 1
## 4           1      -1          1                -1
## attr(,"assign")
## [1] 0 1 2 3
## attr(,"contrasts")
## attr(,"contrasts")$Period
##        [,1]
## After    -1
## Before    1
##
## attr(,"contrasts")$Treatment
##         [,1]
## Control   -1
## Impact     1
```

First consider `model_matrix_1` for the model `~ Period * Treatment`. The first column of 1s represents the intercept. This is used to estimate the average value of `Control` and `Impact` during the `Before` period. The second column indicates the effect of moving from `Control` to `Impact` within the `Before` period. The

contrast matrix for `Period` is set up to estimate the difference ,$(\bar{\mu}A - \bar{\mu}B)$. The third column indicates the effect of moving from `Before` to `After`, averaged over the `Control` and `Impact`. The contrast matrix for `Period` is set up to estimate the difference, $(\delta(\mu_A) - \delta(\mu_B))$. The fourth column represents the interaction between `Period` and `Treatment`. It estimates the difference in the difference `Control-Impact` between the `Before` and `After` periods. Therefore `model_matrix_1` allows for estimating the intercept, main effects and interaction.

```r
# Create the model matrix for ~ 0 + Period:Treatment
model_matrix_2 <- model.matrix(~ 0 + Period:Treatment, design)
print(model_matrix_2)
```

```
##   PeriodAfter:TreatmentControl PeriodBefore:TreatmentControl
## 1                            0                             1
## 2                            1                             0
## 3                            0                             0
## 4                            0                             0
##   PeriodAfter:TreatmentImpact PeriodBefore:TreatmentImpact
## 1                           0                            0
## 2                           0                            0
## 3                           0                            1
## 4                           1                            0
## attr(,"assign")
## [1] 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Period
##        [,1]
## After    -1
## Before    1
##
## attr(,"contrasts")$Treatment
##         [,1]
## Control   -1
## Impact     1
```

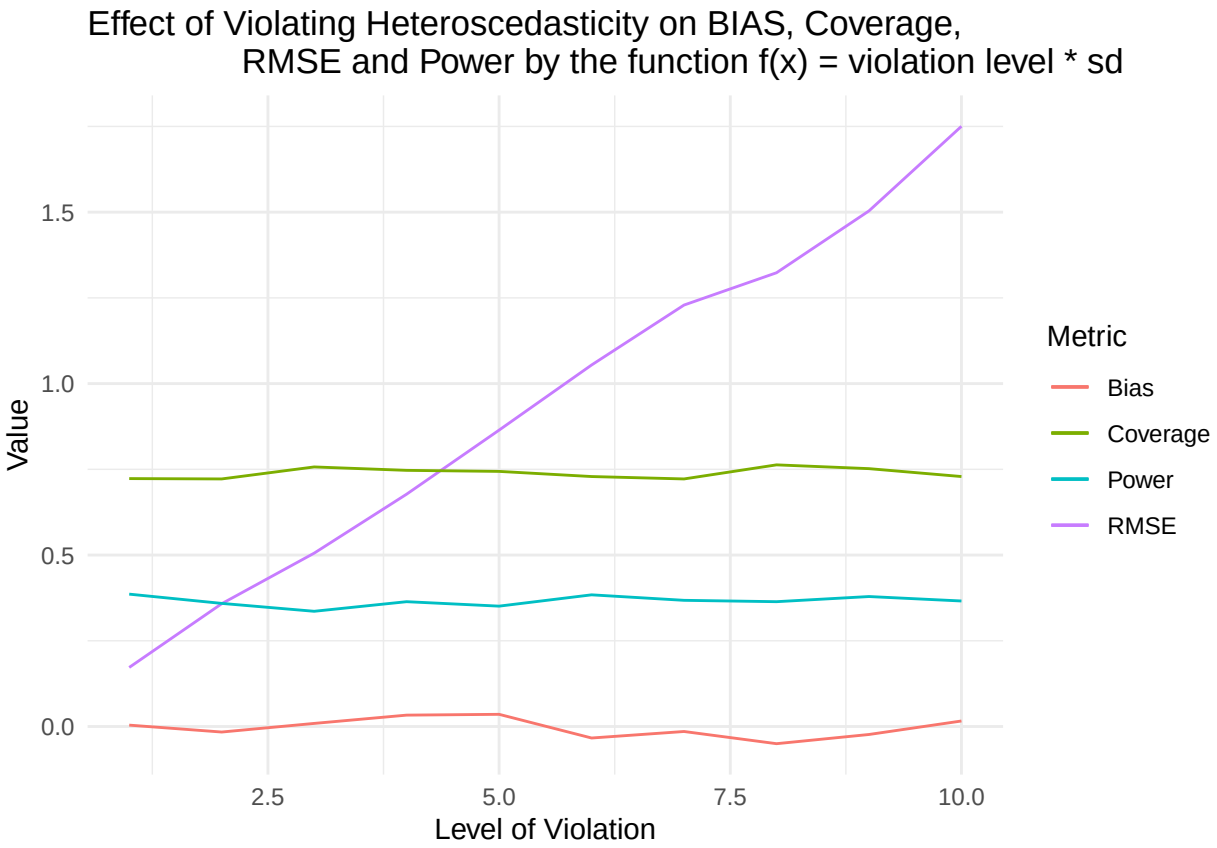
`model_matrix_2` for the model `~ 0 + Period * Treatment` does not have an intercept. Each column includes a single 1 for each of the 4 possible interactions. This means you can only estimate the interaction effect between `Period` and `Treatment`.

3

```r
set.seed(123)

# Function to simulate data with different levels of violation
simulate_data <- function(n=100 , violation_level=1) {
  x <- rnorm(n)
  epsilon <- rnorm(n, mean = 0, sd = abs(violation_level * x))
  y <- 2 * x + 3 + epsilon
  data.frame(x, y)
}

# Simulate
simulate <- function(n_sims=1000, violation_level=1,alpha = 0.05) {
  true_slope <- 2
  slopes <- numeric(n_sims)
  coverages <- numeric(n_sims)
  reject_null <- numeric(n_sims)

  for (i in 1:n_sims) {
    data <- simulate_data(violation_level = violation_level)
    model <- lm(y ~ x, data = data)
    slopes[i] <- coef(model)[2]
    conf_int <- confint(model)[2, ]
    coverages[i] <- conf_int[1] <= true_slope && conf_int[2] >= true_slope
    test <- bptest(model)
    reject_null[i] <- test$p.value < alpha
  }

  bias <- mean(slopes - true_slope)
  rmse <- sqrt(mean((slopes - true_slope)^2))
  coverage <- mean(coverages)
  power <- mean(reject_null)

  list(bias = bias, rmse = rmse, coverage = coverage, power = power)
}
```

```r
# Run the simulation with different levels of violation
results <- data.frame(violation_level = numeric(),
                      bias = numeric(),
                      rmse = numeric(),
                      coverage = numeric(),
                      power = numeric())
for (violation_level in seq(1, 10, by = 1)) {
  result <- simulate(violation_level = violation_level)
  results <- rbind(results, data.frame(violation_level, bias = result$bias,
                                       rmse = result$rmse,
                                       coverage = result$coverage,
                                       power = result$power))
}
print(results)
```

```
##    violation_level         bias      rmse coverage power
## 1                1  0.004054285 0.1721927    0.723 0.386
## 2                2 -0.016098659 0.3580873    0.722 0.359
## 3                3  0.009015272 0.5055129    0.757 0.336
## 4                4  0.033175944 0.6775691    0.747 0.364
## 5                5  0.035515563 0.8643396    0.744 0.351
## 6                6 -0.033579776 1.0542837    0.729 0.384
## 7                7 -0.014555320 1.2290865    0.722 0.368
## 8                8 -0.050130958 1.3233036    0.763 0.364
## 9                9 -0.023257116 1.5036724    0.752 0.379
## 10              10  0.016101423 1.7504208    0.729 0.366
```


Effect of Violating Heteroscedasticity on BIAS, Coverage,
RMSE and Power by the function f(x) = violation level * sd
Value
1.5
1.0
0.5
0.0
2.5
5.0
7.5
10.0
Level of Violation
Metric
Bias
Coverage
Power
RMSE