PVS analysis

Nkosi Sampson

2024-08-19

```r
library(readxl)
library(ggplot2)
library(reshape2)
library(ggpubr)
library(dplyr)
library(corrplot)
library(Hmisc)
```

```r
t_lm <- readxl::read_excel("LASI_DAD_neuroimaging_withIDs.xlsx")
wmpv<-  readr::read_csv("wmPVSnew.csv")
bgpv<-  readr::read_csv("bgPVSnew.csv")
```

```r
pvsNew <- left_join(wmpv, bgpv, "Subject")
pvsNew$wm_vol = pvsNew$wm_volume
pvsNew$bg_vol = pvsNew$bg_volume
pvsNew$pvs_wm_vol = pvsNew$wm_pvs_volume
pvsNew$pvs_bg_vol = pvsNew$bg_pvs_volume
```

Data Cleaning

```r
# Calculate the mean for each variable for each BaseID
vars_to_average <- c('pvs_wm_vol', 'wm_vol', 'pvs_bg_vol', 'bg_vol')

# Extract the base ID (removing the '_1' or '_2' suffix)
pvsNew <- pvsNew %>%
  mutate(BaseID = sub("_[12]$", "", Subject)) %>%
  group_by(BaseID) %>%
  summarise(across(all_of(vars_to_average), mean, na.rm = TRUE))
```

```
## Warning: There was 1 warning in ‘summarise()‘.
## i In argument: ‘across(all_of(vars_to_average), mean, na.rm = TRUE)‘.
## i In group 1: ‘BaseID = "2YZJ5G"‘.
## Caused by warning:
## ! The ‘...‘ argument of ‘across()‘ is deprecated as of dplyr 1.1.0.
## Supply arguments directly to ‘.fns‘ through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
```

```
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

There are 126 observations in the t_lm dataste and 118 in the WMPav datset. Which ones are in the one and not the other?

```r
# Identify which BaseIDs in t_lm are not in WMPVav
missing_ids1 <- setdiff(t_lm$SubjectID, pvsNew$BaseID)
print(missing_ids1)
```

```
##  [1] "5JP3BN"   "82GNEE"   "B5GZ5Z"   "BMR3PH"   "DOFXYN"   "ECX6X7"
##  [7] "KA8PQN"   "PZDMS8"   "RJP7L"    "Scan_003" "T2DQ2J"   "YGZBGR"
## [13] "ZAJXLD"   "ZJSF2H"
```

It looks like there are also some observations in the WMPVav dataset that aren't in the t_lm dataset. These are:

```r
# Identify which BaseIDs in t_lm are not in WMPVav
missing_ids2 <- setdiff( pvsNew$BaseID, t_lm$SubjectID)
print(missing_ids2)
```

```
## [1] "DKZ36X" "HJHMMN" "HWABB4" "JBYGHJ" "XZ95WG"
```

```r
# Filter out the scans from t_lm that are not in WMPVav
filtered_t_lm <- t_lm %>%
  filter(!SubjectID %in% missing_ids1)

# Display the resulting filtered dataset
print(filtered_t_lm)
```

```
## # A tibble: 112 x 43
##    SubjectID ImageID prim_key   Age   Sex EducationYears hmse_score at_risk
##    <chr>       <dbl>    <dbl> <dbl> <dbl>          <dbl>      <dbl>   <dbl>
##  1 2YZJ5G    1130396  1.27e14    63     0             11         27       1
##  2 42NSBC    1017575  1.29e14    69     1              0         15       1
##  3 4C5G6B    1257305  1.19e14    70     1              0         18       0
##  4 5CQ5AE    1026211  1.29e14    72     0              5         26       1
##  5 6SFEH8    1287417  1.19e14    61     0              3         29       1
##  6 7BECRF    1007458  1.29e14    62     0             12         28       0
##  7 7XH95C    1283511  1.19e14    76     1              7         26       1
##  8 89PEPN    1168591  1.27e14    68     1             11         28       0
##  9 8YFJ5J    1030788  1.29e14    80     1              4         26       1
## 10 98LMPT    1010523  1.29e14    70     0              9         29       1
## # i 102 more rows
## # i 35 more variables: lasi_score <dbl>, Literate <dbl>, Urban <dbl>,
## #   Rural <dbl>, BMI <dbl>, WMH <dbl>, eTIV <dbl>, vol_hippocampus <dbl>,
## #   vol_cortex <dbl>, AirPollution_2016 <dbl>, UsesUncleanCookingFuel <dbl>,
## #   UsesUncleanHouseholdFuel <dbl>, Hypertension <dbl>,
## #   Depressed_TimesPerWeek <dbl>, Lonely_TimesPerWeek <dbl>,
## #   WalkingFrequency <dbl>, ExerciseFrequency <dbl>, WorkFrequency <dbl>, ...
```

```r
# Filter out the scans from t_lm that are not in WMPVav
filtered_pvsNew <- pvsNew %>%
  filter(!BaseID %in% missing_ids2)

# Display the resulting filtered dataset
print(filtered_pvsNew)
```

```
## # A tibble: 112 x 5
##    BaseID pvs_wm_vol  wm_vol pvs_bg_vol bg_vol
##    <chr>       <dbl>   <dbl>      <dbl>  <dbl>
##  1 2YZJ5G      3035  395905         219  38871
##  2 42NSBC       576. 331627          81  28472.
##  3 4C5G6B      2370. 318324         273  33478.
##  4 5CQ5AE      2148  438082.        250  39932.
##  5 6SFEH8      2239  340448.        108. 33078.
##  6 7BECRF       352  430546.        141  40498
##  7 7XH95C      4466. 316908.        258. 34268
##  8 89PEPN      2305  402424         474  33957
##  9 8YFJ5J      1707  289364.        214. 31158.
## 10 98LMPT      1493  375424         175  33484
## # i 102 more rows
```

```r
df <- left_join(filtered_t_lm, filtered_pvsNew, by = join_by(SubjectID == BaseID))

print(df)
```

```
## # A tibble: 112 x 47
##    SubjectID ImageID prim_key   Age   Sex EducationYears hmse_score at_risk
##    <chr>       <dbl>    <dbl> <dbl> <dbl>          <dbl>      <dbl>   <dbl>
##  1 2YZJ5G    1130396  1.27e14    63     0             11         27       1
##  2 42NSBC    1017575  1.29e14    69     1              0         15       1
##  3 4C5G6B    1257305  1.19e14    70     1              0         18       0
##  4 5CQ5AE    1026211  1.29e14    72     0              5         26       1
##  5 6SFEH8    1287417  1.19e14    61     0              3         29       1
##  6 7BECRF    1007458  1.29e14    62     0             12         28       0
##  7 7XH95C    1283511  1.19e14    76     1              7         26       1
##  8 89PEPN    1168591  1.27e14    68     1             11         28       0
##  9 8YFJ5J    1030788  1.29e14    80     1              4         26       1
## 10 98LMPT    1010523  1.29e14    70     0              9         29       1
## # i 102 more rows
## # i 39 more variables: lasi_score <dbl>, Literate <dbl>, Urban <dbl>,
## #   Rural <dbl>, BMI <dbl>, WMH <dbl>, eTIV <dbl>, vol_hippocampus <dbl>,
## #   vol_cortex <dbl>, AirPollution_2016 <dbl>, UsesUncleanCookingFuel <dbl>,
## #   UsesUncleanHouseholdFuel <dbl>, Hypertension <dbl>,
## #   Depressed_TimesPerWeek <dbl>, Lonely_TimesPerWeek <dbl>,
## #   WalkingFrequency <dbl>, ExerciseFrequency <dbl>, WorkFrequency <dbl>, ...
```

```r
cdr <- readxl::read_excel("H_DAD_mri_cdr_v2.xlsx")
```

```r
df <- left_join(df, cdr, by = join_by(prim_key == prim_key))
print(df)
```

```
## # A tibble: 112 x 83
##    SubjectID ImageID prim_key   Age   Sex EducationYears hmse_score at_risk
##    <chr>       <dbl>    <dbl> <dbl> <dbl>          <dbl>      <dbl>   <dbl>
##  1 2YZJ5G    1130396  1.27e14    63     0             11         27       1
##  2 42NSBC    1017575  1.29e14    69     1              0         15       1
##  3 4C5G6B    1257305  1.19e14    70     1              0         18       0
##  4 5CQ5AE    1026211  1.29e14    72     0              5         26       1
##  5 6SFEH8    1287417  1.19e14    61     0              3         29       1
##  6 7BECRF    1007458  1.29e14    62     0             12         28       0
##  7 7XH95C    1283511  1.19e14    76     1              7         26       1
##  8 89PEPN    1168591  1.27e14    68     1             11         28       0
##  9 8YFJ5J    1030788  1.29e14    80     1              4         26       1
## 10 98LMPT    1010523  1.29e14    70     0              9         29       1
## # i 102 more rows
## # i 75 more variables: lasi_score <dbl>, Literate <dbl>, Urban <dbl>,
## #   Rural <dbl>, BMI <dbl>, WMH <dbl>, eTIV <dbl>, vol_hippocampus <dbl>,
## #   vol_cortex <dbl>, AirPollution_2016 <dbl>, UsesUncleanCookingFuel <dbl>,
## #   UsesUncleanHouseholdFuel <dbl>, Hypertension <dbl>,
## #   Depressed_TimesPerWeek <dbl>, Lonely_TimesPerWeek <dbl>,
## #   WalkingFrequency <dbl>, ExerciseFrequency <dbl>, WorkFrequency <dbl>, ...
```

```
df$cdrnew = ifelse(df$cdr == "0", 0, 1)
df$LOGpvs_bg_vol = log(df$pvs_bg_vol)
df$LOGpvs_wm_vol = log(df$pvs_wm_vol)
df$LOGAirPollution_2016 = log(df$AirPollution_2016)
df$vol_cortex = log(df$vol_cortex)
```

```
#df <- filter(df, cdrnew == "0")
```

```
# List of all fields including the response variable
all_fields <- c('Age', 'Sex', 'EducationYears', 'wm_vol', 'eTIV', 'LOGpvs_bg_vol', 'bg_vol', 'AirPollut

dont_standardize <- c('Urban', 'Literate', 'PreparesHotMeal', 'UsesPublicTransport', 'UsesUncleanCookin


# Custom standardization functionz
nanzscore <- function(x) {
  return((x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE))
}

# Standardize the data
df <- df %>%
  mutate(across(all_fields[!all_fields %in% dont_standardize], nanzscore))
```

Check Assumptions of linear regression

```
library(ggplot2)

# List of all fields excluding the response variable
all_fields <- c('EducationYears', 'BMI', 'DiastolicBP',
```

```r
                'Nap_HoursPerDay',
                'Chores_HoursPerDay', 'EducationYears',
                'HearingTest_l', 'HearingTest_r', 'Urban', 'SystolicBP','Chores_HoursPerDay', 'StoreFre
                'Lonely_TimesPerWeek', 'TV_HoursPerDay', 'Reading_HoursPerDay',
                'WalkingFrequency', 'Computer_HoursPerDay', 'smoking_per_day', 'AlcoholFrequency')

# Function to create partial residual plots for a given variable
create_partial_residual_plot <- function(variable, full_model_formula, df) {

  # Skip the "Sex" variable
  if (variable == "Sex" ) {
    print("Skipping variable 'Sex'")
    return(NULL)
  }

  # Define the reduced model formula by removing the current variable
  reduced_model_formula <- as.formula(paste("LOGpvs_bg_vol ~", paste(setdiff(all.vars(full_model_formula

  # Use complete cases to ensure no missing values
  complete_cases <- complete.cases(df[all.vars(full_model_formula)])
  df_complete <- df[complete_cases, ]

  # Fit the models and handle errors
  full_model <- tryCatch(lm(full_model_formula, data = df_complete), error = function(e) NULL)
  reduced_model <- tryCatch(lm(reduced_model_formula, data = df_complete), error = function(e) NULL)

  # If the models cannot be fitted, skip this variable
  if (is.null(full_model) || is.null(reduced_model)) {
    print(paste("Skipping variable due to singularity or other issues:", variable))
    return(NULL)
  }

  # Calculate residuals from the reduced model
  residuals_reduced <- resid(reduced_model)

  # Extract the coefficient for the predictor of interest from the full model
  beta_X <- tryCatch(coef(full_model)[variable], error = function(e) NULL)

  # Check if the coefficient could be extracted
  if (is.null(beta_X)) {
    print(paste("Skipping variable due to coefficient extraction issue:", variable))
    return(NULL)
  }

  # Extract the values of the predictor of interest
  X_values <- df_complete[[variable]]

  # Calculate partial residuals
  partial_residuals <- residuals_reduced + beta_X * X_values

  # Create the plot using ggplot2
  p <- ggplot(data.frame(X_values, partial_residuals), aes(x = X_values, y = partial_residuals)) +
    geom_point() +
```

```r
    geom_smooth(method = "lm", col = "red") +
    labs(x = variable, y = "Partial Residuals", title = paste("Partial Residual Plot for", variable))

  return(p)
}

# Store all plots in a list
plot_list <- list()

# Outer loop through each variable in the list to build the full model
for (full_model_variable in all_fields) {
  full_model_formula <- as.formula(paste("LOGpvs_bg_vol ~ Age + Sex + eTIV + bg_vol +", full_model_varia

  # Inner loop to create partial residual plots for each variable in the full model
  for (variable in c('Age', 'Sex',  'eTIV', 'bg_vol', full_model_variable)) {
    plot <- create_partial_residual_plot(variable, full_model_formula, df)
    if (!is.null(plot)) {
      plot_list <- c(plot_list, list(plot))
    }
  }
}

# Save all plots to a multi-page PDF file
pdf("partial_residual_plots.pdf", width = 8.5, height = 11) # Standard page size, adjust if needed
for (p in plot_list) {
  print(p)
}
dev.off()
```

```r
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
# List of all fields
all_fields <- c( 'EducationYears', 'Age', 'Sex', 'eTIV', 'bg_vol', 'logWMHvol', 'SystolicBP', 'BMI', 'C
                'UsesPublicTransport', 'WorkFrequency', 'Depressed_TimesPerWeek', 'Literate',
                'PreparesHotMeal', 'AirPollution_2016', 'Nap_HoursPerDay', 'ExerciseFrequency',
                'Lonely_TimesPerWeek', 'UsesUncleanHouseholdFuel',  'TV_HoursPerDay', 'Reading_HoursPer
                'WalkingFrequency', 'EducationYears', 'UsesUncleanCookingFuel',
                'smoking_per_day', 'HearingTest', 'AlcoholFrequency', 'Urban', 'Literate', 'PreparesHot

# Loop through each variable in the list
for (variable in all_fields) {
  # Define the formula for the linear model
  formula <- as.formula(paste("LOGpvs_wm_vol ~ Age + Sex +  eTIV + wm_vol +", variable))
```

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

  #plot(resid(lm_model))

  # Perform the Kolmogorov-Smirnov test
  ks_test <- ks.test(resid(lm_model), "pnorm", mean = mean(resid(lm_model)), sd = sd(resid(lm_model)))

  # Calculate the VIF values
  vif_values <- vif(lm_model)

  # Check for p-value < 0.05
  if (ks_test$p.value < 0.05) {
    print(paste("Warning: p-value < 0.05 for variable:", variable))
  }

  # Check for VIF > 5
  if (any(vif_values > 5)) {
    print(paste("Warning: VIF > 5 for variable:", variable))
  }
}
```

```
## [1] "Warning: p-value < 0.05 for variable: DiastolicBP"
```

This process was repeated for the model with LOGpvs_bg_vol as the response variable

Additional R squared analaysis

```
# List of all fields including the response variable
all_fields <- c('EducationYears', 'AirPollution_2016','Age', 'Sex', 'wm_vol', 'SystolicBP', 'BMI', 'Cho
                'UsesPublicTransport', 'WorkFrequency', 'Depressed_TimesPerWeek', 'Literate',
                'PreparesHotMeal', 'Nap_HoursPerDay', 'ExerciseFrequency',
                'Lonely_TimesPerWeek', 'UsesUncleanHouseholdFuel', 'TV_HoursPerDay', 'Reading_HoursPerDa
                'WalkingFrequency', 'EducationYears', 'Computer_HoursPerDay', 'UsesUncleanCookingFuel',
                'smoking_per_day', 'HearingTest', 'AlcoholFrequency')

dont_standardize <- c('Urban', 'Literate', 'PreparesHotMeal', 'UsesPublicTransport', 'UsesUncleanCookin


# Custom standardization function
nanzscore <- function(x) {
  return((x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE))
}

# Standardize the data
df_std <- df %>%
  mutate(across(all_fields[!all_fields %in% dont_standardize], nanzscore))


# Baseline model
```

```r
mdl_baseline <- lm(LOGpvs_wm_vol ~ Age + Sex + eTIV + wm_vol + logWMHvol, data = df_std)
summary(mdl_baseline)
```

```
##
## Call:
## lm(formula = LOGpvs_wm_vol ~ Age + Sex + eTIV + wm_vol + logWMHvol,
##     data = df_std)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7898 -0.3955  0.1233  0.4260  1.3984
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.57770    0.65359   7.004 2.37e-10 ***
## Age          0.03700    0.07099   0.521   0.6033
## Sex          0.15898    0.07876   2.019   0.0461 *
## eTIV         0.01743    0.10862   0.160   0.8728
## wm_vol       0.21022    0.10517   1.999   0.0482 *
## logWMHvol    0.79269    0.16544   4.791 5.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6504 on 106 degrees of freedom
## Multiple R-squared:  0.2657, Adjusted R-squared:  0.2311
## F-statistic: 7.672 on 5 and 106 DF,  p-value: 3.449e-06
```

```r
# Calculate baseline R-squared
explain_variance_baseline <- summary(mdl_baseline)$r.squared
```

```r
all_fields_predictors <- all_fields

# Initialize output dataframe
pvsWM_out_all <- data.frame(field_name = all_fields_predictors,
                            add_Rsquared = numeric(length(all_fields_predictors)),
                            t = numeric(length(all_fields_predictors)),
                            p = numeric(length(all_fields_predictors)))

# Analyze each field
for (i in seq_along(all_fields_predictors)) {
  formula_str <- paste('LOGpvs_wm_vol ~ Age + Sex + eTIV + wm_vol + logWMHvol +', all_fields_predictors
  mdl_i <- lm(as.formula(formula_str), data = df_std)
  summary_i <- summary(mdl_i)

  # Extract coefficient index correctly (depends on how many predictors you have)
  coeff_index <- which(names(coef(mdl_i)) == all_fields_predictors[i])

  pvsWM_out_all$t[i] <- coef(summary_i)[coeff_index, "t value"]
  pvsWM_out_all$p[i] <- coef(summary_i)[coeff_index, "Pr(>|t|)"]
  pvsWM_out_all$add_Rsquared[i] <- round((summary_i$r.squared - explain_variance_baseline) * 100, 1)
}

# Adjust p-values using Benjamini-Hochberg method
```

```r
pvsWM_out_all$p_wo_rural_FDR <- p.adjust(pvsWM_out_all$p, method = "BH")

# Sort by additional R-squared
pvsWM_out_all <- pvsWM_out_all[order(-pvsWM_out_all$add_Rsquared), ]

# Display the additional R-squared table
print(pvsWM_out_all)
```

```
##                   field_name add_Rsquared           t            p
## 2         AirPollution_2016         28.4  8.13707074 8.755791e-13
## 9                     Urban         11.7  4.46337746 2.035329e-05
## 17          Nap_HoursPerDay          4.0  2.46480638 1.533047e-02
## 8        Chores_HoursPerDay          3.6 -2.32412675 2.204413e-02
## 20 UsesUncleanHouseholdFuel          2.9 -2.07536674 4.039622e-02
## 11           StoreFrequency          2.4  1.90083287 6.006599e-02
## 28              HearingTest          1.9 -0.67668994 5.001196e-01
## 14   Depressed_TimesPerWeek          1.8  1.61257841 1.098379e-01
## 7                       BMI          1.3 -1.13386164 2.594847e-01
## 15                 Literate          1.3  1.35736473 1.775757e-01
## 18        ExerciseFrequency          1.1  1.23855103 2.182736e-01
## 26   UsesUncleanCookingFuel          1.1 -1.23638956 2.190723e-01
## 13            WorkFrequency          0.9  1.15694220 2.499213e-01
## 22      Reading_HoursPerDay          0.8 -1.05856427 2.922274e-01
## 1            EducationYears          0.7  0.99227790 3.233432e-01
## 24           EducationYears          0.7  0.99227790 3.233432e-01
## 12      UsesPublicTransport          0.5 -0.81189233 4.186902e-01
## 25     Computer_HoursPerDay          0.5  0.83020178 4.083084e-01
## 21           TV_HoursPerDay          0.4  0.72389892 4.707376e-01
## 29         AlcoholFrequency          0.3  1.45513168 1.486446e-01
## 23         WalkingFrequency          0.2  0.56578838 5.727448e-01
## 10              DiastolicBP          0.1  0.32115297 7.487328e-01
## 3                       Age          0.0  0.52123871 6.032878e-01
## 4                       Sex          0.0  2.01860033 4.605386e-02
## 5                    wm_vol          0.0  1.99874618 4.819577e-02
## 6               SystolicBP          0.0 -0.20859737 8.351667e-01
## 16          PreparesHotMeal          0.0 -0.02064381 9.835690e-01
## 19       Lonely_TimesPerWeek          0.0  0.24366681 8.079643e-01
## 27           smoking_per_day         -0.1  1.25785056 2.112627e-01
##    p_wo_rural_FDR
## 2    2.539179e-11
## 9    2.951227e-04
## 17   1.481945e-01
## 8    1.598199e-01
## 20   1.996682e-01
## 11   2.177392e-01
## 28   6.305855e-01
## 14   3.539220e-01
## 7    4.703161e-01
## 15   4.537927e-01
## 18   4.537927e-01
## 26   4.537927e-01
## 13   4.703161e-01
## 22   4.935238e-01
```

```
## 1    4.935238e-01
## 24   4.935238e-01
## 12   5.781912e-01
## 25   5.781912e-01
## 21   6.205177e-01
## 29   4.310694e-01
## 23   6.920666e-01
## 10   8.351251e-01
## 3    6.998138e-01
## 4    1.996682e-01
## 5    1.996682e-01
## 6    8.649941e-01
## 16   9.835690e-01
## 19   8.649941e-01
## 27   4.537927e-01
```

```r
summary(lm(LOGpvs_wm_vol ~ Age + Sex +  eTIV + wm_vol + logWMHvol, df_std))
```

```
##
## Call:
## lm(formula = LOGpvs_wm_vol ~ Age + Sex + eTIV + wm_vol + logWMHvol,
##     data = df_std)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7898 -0.3955  0.1233  0.4260  1.3984
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.57770    0.65359   7.004 2.37e-10 ***
## Age          0.03700    0.07099   0.521   0.6033
## Sex          0.15898    0.07876   2.019   0.0461 *
## eTIV         0.01743    0.10862   0.160   0.8728
## wm_vol       0.21022    0.10517   1.999   0.0482 *
## logWMHvol    0.79269    0.16544   4.791 5.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6504 on 106 degrees of freedom
## Multiple R-squared:  0.2657, Adjusted R-squared:  0.2311
## F-statistic: 7.672 on 5 and 106 DF,  p-value: 3.449e-06
```

```r
# List of all fields including the response variable
all_fields <- c('EducationYears', 'AirPollution_2016','Age', 'Sex', 'wm_vol', 'SystolicBP', 'BMI', 'Cho
                'UsesPublicTransport', 'WorkFrequency', 'Depressed_TimesPerWeek', 'Literate',
                'PreparesHotMeal', 'Nap_HoursPerDay', 'ExerciseFrequency',
                'Lonely_TimesPerWeek', 'UsesUncleanHouseholdFuel', 'TV_HoursPerDay', 'Reading_HoursPerDa
                'WalkingFrequency', 'EducationYears', 'Computer_HoursPerDay', 'UsesUncleanCookingFuel',
                'smoking_per_day', 'HearingTest', 'AlcoholFrequency')

dont_standardize <- c('Urban', 'Literate', 'PreparesHotMeal', 'UsesPublicTransport', 'UsesUncleanCookin
```

```r
# Custom standardization function
nanzscore <- function(x) {
  return((x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE))
}

# Standardize the data
df_std <- df %>%
  mutate(across(all_fields[!all_fields %in% dont_standardize], nanzscore))


# Baseline model
mdl_baseline <- lm(LOGpvs_bg_vol ~ Age + Sex + eTIV + bg_vol, data = df_std)
summary(mdl_baseline)
```

```
##
## Call:
## lm(formula = LOGpvs_bg_vol ~ Age + Sex + eTIV + bg_vol, data = df_std)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.29204 -0.59989 -0.06056  0.46771  2.61631
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.327e-16  8.419e-02   0.000 1.000000
## Age          3.389e-01  9.120e-02   3.716 0.000324 ***
## Sex          1.423e-01  1.109e-01   1.283 0.202273
## eTIV         1.863e-02  1.104e-01   0.169 0.866358
## bg_vol       5.214e-01  1.078e-01   4.838 4.43e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.891 on 107 degrees of freedom
## Multiple R-squared:  0.2348, Adjusted R-squared:  0.2062
## F-statistic: 8.207 on 4 and 107 DF,  p-value: 8.228e-06
```

```r
# Calculate baseline R-squared
explain_variance_baseline <- summary(mdl_baseline)$r.squared
```

```r
all_fields_predictors <- all_fields

# Initialize output dataframe
pvsWM_out_all <- data.frame(field_name = all_fields_predictors,
                            add_Rsquared = numeric(length(all_fields_predictors)),
                            t = numeric(length(all_fields_predictors)),
                            p = numeric(length(all_fields_predictors)))

# Analyze each field
for (i in seq_along(all_fields_predictors)) {
  formula_str <- paste('LOGpvs_bg_vol ~ Age + Sex + eTIV + bg_vol +', all_fields_predictors[i])
  mdl_i <- lm(as.formula(formula_str), data = df_std)
  summary_i <- summary(mdl_i)
```

```r
  # Extract coefficient index correctly (depends on how many predictors you have)
  coeff_index <- which(names(coef(mdl_i)) == all_fields_predictors[i])

  pvsWM_out_all$t[i] <- coef(summary_i)[coeff_index, "t value"]
  pvsWM_out_all$p[i] <- coef(summary_i)[coeff_index, "Pr(>|t|)"]
  pvsWM_out_all$add_Rsquared[i] <- round((summary_i$r.squared - explain_variance_baseline) * 100, 1)
}

# Adjust p-values using Benjamini-Hochberg method
pvsWM_out_all$p_wo_rural_FDR <- p.adjust(pvsWM_out_all$p, method = "BH")

# Sort by additional R-squared
pvsWM_out_all <- pvsWM_out_all[order(-pvsWM_out_all$add_Rsquared), ]

# Display the additional R-squared table
print(pvsWM_out_all)
```

```
##                  field_name add_Rsquared           t            p
## 2          AirPollution_2016         12.1  4.46689485 1.991793e-05
## 6                 SystolicBP         10.5  4.10869904 7.858881e-05
## 9                      Urban         10.2  4.04040908 1.012630e-04
## 5                     wm_vol          4.1 -2.45640394 1.565740e-02
## 10               DiastolicBP          3.1  2.12663589 3.577093e-02
## 12       UsesPublicTransport          2.9 -2.03265828 4.458689e-02
## 8         Chores_HoursPerDay          2.2 -1.76785032 7.996324e-02
## 18         ExerciseFrequency          1.8  1.57696759 1.177828e-01
## 15                  Literate          1.3  1.33895332 1.834502e-01
## 17           Nap_HoursPerDay          1.3  1.34869774 1.803085e-01
## 14     Depressed_TimesPerWeek          1.2  1.31776476 1.904233e-01
## 20 UsesUncleanHouseholdFuel          0.9 -1.09526328 2.758837e-01
## 7                        BMI          0.6  0.76927678 4.434726e-01
## 11            StoreFrequency          0.5 -0.81714169 4.156803e-01
## 26    UsesUncleanCookingFuel          0.5 -0.80717077 4.213756e-01
## 23          WalkingFrequency          0.4  0.72954875 4.672751e-01
## 21            TV_HoursPerDay          0.3 -0.60990419 5.432305e-01
## 1             EducationYears          0.2  0.58191593 5.618589e-01
## 13             WorkFrequency          0.2  0.57493430 5.665540e-01
## 19       Lonely_TimesPerWeek          0.2  0.49957238 6.184116e-01
## 22       Reading_HoursPerDay          0.2 -0.52460881 6.009506e-01
## 24            EducationYears          0.2  0.58191593 5.618589e-01
## 16           PreparesHotMeal          0.1 -0.37659918 7.072243e-01
## 25      Computer_HoursPerDay          0.1  0.45374264 6.509417e-01
## 3                        Age          0.0  3.71578640 3.240095e-04
## 4                        Sex          0.0  1.28297441 2.022731e-01
## 29          AlcoholFrequency          0.0  0.48853065 6.261928e-01
## 27           smoking_per_day         -0.2 -0.08007193 9.363325e-01
## 28               HearingTest         -0.5 -1.53578387 1.276282e-01
##    p_wo_rural_FDR
## 2    0.0005776200
## 6    0.0009788757
## 9    0.0009788757
## 5    0.0908129221
## 10   0.1728928461
```

```
## 12   0.1847171356
## 8    0.2898667590
## 18   0.3701217455
## 15   0.4189942559
## 17   0.4189942559
## 14   0.4189942559
## 20   0.5333751064
## 7    0.6984457927
## 11   0.6984457927
## 26   0.6984457927
## 23   0.6984457927
## 21   0.6984457927
## 1    0.6984457927
## 13   0.6984457927
## 19   0.6984457927
## 22   0.6984457927
## 24   0.6984457927
## 16   0.7324823039
## 25   0.6991595941
## 3    0.0023490686
## 4    0.4189942559
## 29   0.6984457927
## 27   0.9363324667
## 28   0.3701217455
```

```
summary(lm(LOGpvs_bg_vol ~ Age + Sex +  eTIV + bg_vol, df_std))
```

```
##
## Call:
## lm(formula = LOGpvs_bg_vol ~ Age + Sex + eTIV + bg_vol, data = df_std)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.29204 -0.59989 -0.06056  0.46771  2.61631
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.327e-16  8.419e-02   0.000 1.000000
## Age          3.389e-01  9.120e-02   3.716 0.000324 ***
## Sex          1.423e-01  1.109e-01   1.283 0.202273
## eTIV         1.863e-02  1.104e-01   0.169 0.866358
## bg_vol       5.214e-01  1.078e-01   4.838 4.43e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.891 on 107 degrees of freedom
## Multiple R-squared:  0.2348, Adjusted R-squared:  0.2062
## F-statistic: 8.207 on 4 and 107 DF,  p-value: 8.228e-06
```

Mediation Analysis

Assumptions of linear regression were checked with the same methods as before, for each path of the mediation analysis.

```r
# List of all fields including the response variable
all_fields <- c('Age', 'Sex', 'EducationYears', 'wm_vol', 'eTIV', 'LOGpvs_wm_vol', 'bg_vol', 'AirPolluti

dont_standardize <- c('Urban', 'Literate', 'PreparesHotMeal', 'UsesPublicTransport', 'UsesUncleanCookin


# Custom standardization functionz
nanzscore <- function(x) {
  return((x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE))
}

# Standardize the data
df_std <- df %>%
  mutate(across(all_fields[!all_fields %in% dont_standardize], nanzscore))
```

```r
library(lavaan)
```

```
## This is lavaan 0.6-18
## lavaan is FREE software! Please report any bugs.
```

```r
set.seed(01)

model1 <- "
# Path c' (direct effect)
LOGpvs_wm_vol ~ c*Urban + Age + Sex + EducationYears + eTIV  + wm_vol + logWMHvol

#Path a
AirPollution_2016 ~ a*Urban + Age + Sex + EducationYears + eTIV + wm_vol + logWMHvol

#Path b
LOGpvs_wm_vol ~ b*AirPollution_2016 + Age + Sex + EducationYears + eTIV + wm_vol + logWMHvol

#Indirect Effect (a*b)
ab := a*b
"

# Fit the model with bootstrap resampling
fitmod1 <- sem(model1, data = df_std)

# Summarize the results with BCA confidence intervals
summary(fitmod1, fit.measures = TRUE, rsquare = TRUE, ci = TRUE)
```

```
## lavaan 0.6-18 ended normally after 1 iteration
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        17
##
##   Number of observations                           112
##
## Model Test User Model:
##
```

```
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Model Test Baseline Model:
##
##   Test statistic                               149.269
##   Degrees of freedom                                15
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    1.000
##   Tucker-Lewis Index (TLI)                       1.000
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -242.203
##   Loglikelihood unrestricted model (H1)       -242.203
##
##   Akaike (AIC)                                 518.407
##   Bayesian (BIC)                               564.621
##   Sample-size adjusted Bayesian (SABIC)        510.895
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.000
##   90 Percent confidence interval - lower         0.000
##   90 Percent confidence interval - upper         0.000
##   P-value H_0: RMSEA <= 0.050                       NA
##   P-value H_0: RMSEA >= 0.080                       NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.000
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                       Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##   LOGpvs_wm_vol ~
##     Urban      (c)       0.124    0.187    0.665    0.506   -0.242    0.490
##     Age                  0.190    0.075    2.528    0.011    0.043    0.338
##     Sex                  0.143    0.080    1.787    0.074   -0.014    0.299
##     EducatnYrs           0.162    0.069    2.343    0.019    0.026    0.297
##     eTIV                -0.067    0.109   -0.612    0.541   -0.280    0.147
##     wm_vol               0.261    0.107    2.428    0.015    0.050    0.472
##     logWMHvol            0.762    0.172    4.442    0.000    0.426    1.098
##   AirPollution_2016 ~
##     Urban      (a)       1.322    0.188    7.041    0.000    0.954    1.689
##     Age                  0.002    0.091    0.023    0.982   -0.176    0.180
```

```
##     Sex                  0.057    0.096    0.591    0.554   -0.132    0.246
##     EducatnYrs          -0.170    0.082   -2.085    0.037   -0.330   -0.010
##     eTIV                 0.076    0.131    0.582    0.560   -0.181    0.334
##     wm_vol               0.073    0.130    0.561    0.575   -0.181    0.327
##     logWMHvol            0.095    0.207    0.459    0.646   -0.311    0.500
##   LOGpvs_wm_vol ~
##     ArPll_2016 (b)       0.551    0.078    7.037    0.000    0.397    0.704
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##    .LOGpvs_wm_vol     0.422    0.056    7.483    0.000    0.311    0.532
##    .AirPolltn_2016    0.615    0.082    7.483    0.000    0.454    0.776
##
## R-Square:
##                    Estimate
##     LOGpvs_wm_vol     0.575
##     AirPolltn_2016    0.380
##
## Defined Parameters:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##     ab                0.728    0.146    4.977    0.000    0.441    1.014
```

```r
# Fit separate models for paths a and b
fit_a <- lm(AirPollution_2016 ~ Urban + Age + Sex + EducationYears + eTIV + wm_vol + logWMHvol, data = d
fit_b <- lm(LOGpvs_wm_vol ~ AirPollution_2016 + Age + Sex + EducationYears + eTIV + wm_vol + logWMHvol,

# Extract residuals
residuals_a <- residuals(fit_a)
residuals_b <- residuals(fit_b)

# Calculate correlation between residuals
correlation <- cor(residuals_a, residuals_b)

# Print the correlation
print(correlation)
```

```
## [1] -0.03475169
```

```r
library(RMediation)
```

```
## Loading required package: e1071
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:Hmisc':
##
##     impute
```

```
## Loading required package: OpenMx
```

```
## OpenMx may run faster if it is compiled to take advantage of multiple cores.
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
medci(mu.x = 18.76, mu.y = .03028, se.x = 2.765, se.y = 0.003539, rho =  -0.03475169, alpha = 0.05, typ
```

```
## $`95% CI`
## [1] 0.3742413 0.7870480
##
## $Estimate
## [1] 0.5677127
##
## $SE
## [1] 0.1054852
```

```r
# Assuming your dataframe is named df

# Fit the linear model
model <- lm(LOGpvs_wm_vol ~ Urban + Age + Sex + EducationYears + eTIV + wm_vol + logWMHvol, data = df)

# Calculate the confidence intervals for the coefficients
conf_intervals <- confint(model)

# Extract the confidence interval for Urban
conf_intervals["Urban", ]
```

```
##     2.5 %    97.5 %
## 0.3469715 0.9167652
```

```r
# Load necessary package
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit
```

```r
# Fit the models
model_c <- lm(LOGpvs_wm_vol ~ Urban + Age + Sex +  eTIV + wm_vol + logWMHvol, data = df)
model_a <- lm(AirPollution_2016 ~ Urban + Age + Sex + eTIV + wm_vol + logWMHvol, data = df)
model_b <- lm(LOGpvs_wm_vol ~ AirPollution_2016 + Urban + Age + Sex + eTIV + wm_vol + logWMHvol, data =

# Extract coefficients
a <- coef(model_a)["Urban"]
```

```r
b <- coef(model_b)["AirPollution_2016"]
c <- coef(model_c)["Urban"]
c_prime <- coef(model_b)["Urban"]

# Compute effects
indirect_effect <- a * b
direct_effect <- c_prime
total_effect <-  abs(direct_effect) + abs(indirect_effect)
proportion_mediated <- (abs(total_effect) - abs(direct_effect)) / abs(total_effect)

# Define a function to compute the effects for bootstrapping
boot_function <- function(data, indices) {
  d <- data[indices, ]  # Resample with replacement
  model_a <- lm(AirPollution_2016 ~ Urban + Age + Sex + eTIV + wm_vol + logWMHvol, data = d)
  model_b <- lm(LOGpvs_wm_vol ~ AirPollution_2016 + Urban + Age + Sex + eTIV + wm_vol + logWMHvol, data
  model_c <- lm(LOGpvs_wm_vol ~ Urban + Age + Sex + eTIV + wm_vol + logWMHvol, data = d)

  a <- coef(model_a)["Urban"]
  b <- coef(model_b)["AirPollution_2016"]
  c <- coef(model_c)["Urban"]
  c_prime <- coef(model_b)["Urban"]

  indirect_effect <- a * b
  direct_effect <- c_prime
  total_effect <-  abs(direct_effect) + abs(indirect_effect)
  proportion_mediated <- (abs(total_effect) - abs(direct_effect)) / abs(total_effect)

  return(c(indirect_effect, direct_effect, total_effect, proportion_mediated))
}

# Perform bootstrapping
set.seed(123)
boot_results <- boot(data = df, statistic = boot_function, R = 10000)

# Extract bootstrap estimates
bootstrap_estimates <- boot_results$t

# Calculate 95% confidence intervals
a_ci <- quantile(bootstrap_estimates[,1], c(0.025, 0.975))
b_ci <- quantile(bootstrap_estimates[,2], c(0.025, 0.975))
indirect_effect_ci <- quantile(bootstrap_estimates[,1], c(0.025, 0.975))
direct_effect_ci <- quantile(bootstrap_estimates[,2], c(0.025, 0.975))
total_effect_ci <- quantile(bootstrap_estimates[,3], c(0.025, 0.975))
proportion_mediated_ci <- quantile(bootstrap_estimates[,4], c(0.025, 0.975))

# Display results
print("Path a (Urban -> AirPollution_2016):")
```

```
## [1] "Path a (Urban -> AirPollution_2016):"
```

```r
print(a)
```

```
##    Urban
```

```
## 1.297513
```

```r
print("95% CI for Path a:")
```

```
## [1] "95% CI for Path a:"
```

```r
print(a_ci)
```

```
##      2.5%     97.5%
## 0.3179303 0.6961586
```

```r
print("Path b (AirPollution_2016 -> LOGpvs_bg_vol):")
```

```
## [1] "Path b (AirPollution_2016 -> LOGpvs_bg_vol):"
```

```r
print(b)
```

```
## AirPollution_2016
##         0.3821507
```

```r
print("95% CI for Path b:")
```

```
## [1] "95% CI for Path b:"
```

```r
print(b_ci)
```

```
##       2.5%      97.5%
## -0.1472653  0.4899954
```

```r
# Display results
print("Indirect Effect (a*b):")
```

```
## [1] "Indirect Effect (a*b):"
```

```r
print(indirect_effect)
```

```
##     Urban
## 0.4958457
```

```r
print("95% CI for Indirect Effect:")
```

```
## [1] "95% CI for Indirect Effect:"
```

```r
print(indirect_effect_ci)
```

```
##      2.5%     97.5%
## 0.3179303 0.6961586
```

```r
print("Direct Effect (c'):")
```

```
## [1] "Direct Effect (c’):"
```

```r
print(direct_effect)
```

```
##     Urban
## 0.1431056
```

```r
print("95% CI for Direct Effect:")
```

```
## [1] "95% CI for Direct Effect:"
```

```r
print(direct_effect_ci)
```

```
##       2.5%      97.5%
## -0.1472653  0.4899954
```

```r
print("Total Effect (c):")
```

```
## [1] "Total Effect (c):"
```

```r
print(total_effect)
```

```
##     Urban
## 0.6389513
```

```r
print("95% CI for Total Effect:")
```

```
## [1] "95% CI for Total Effect:"
```

```r
print(total_effect_ci)
```

```
##      2.5%     97.5%
## 0.4137942 0.9777387
```

```r
print("Proportion Mediated (c-c`/c):")
```

```
## [1] "Proportion Mediated (c-c‘/c):"
```

```r
print(proportion_mediated)
```

```
##     Urban
## 0.7760304
```

```r
print("95% CI for Proportion Mediated:")
```

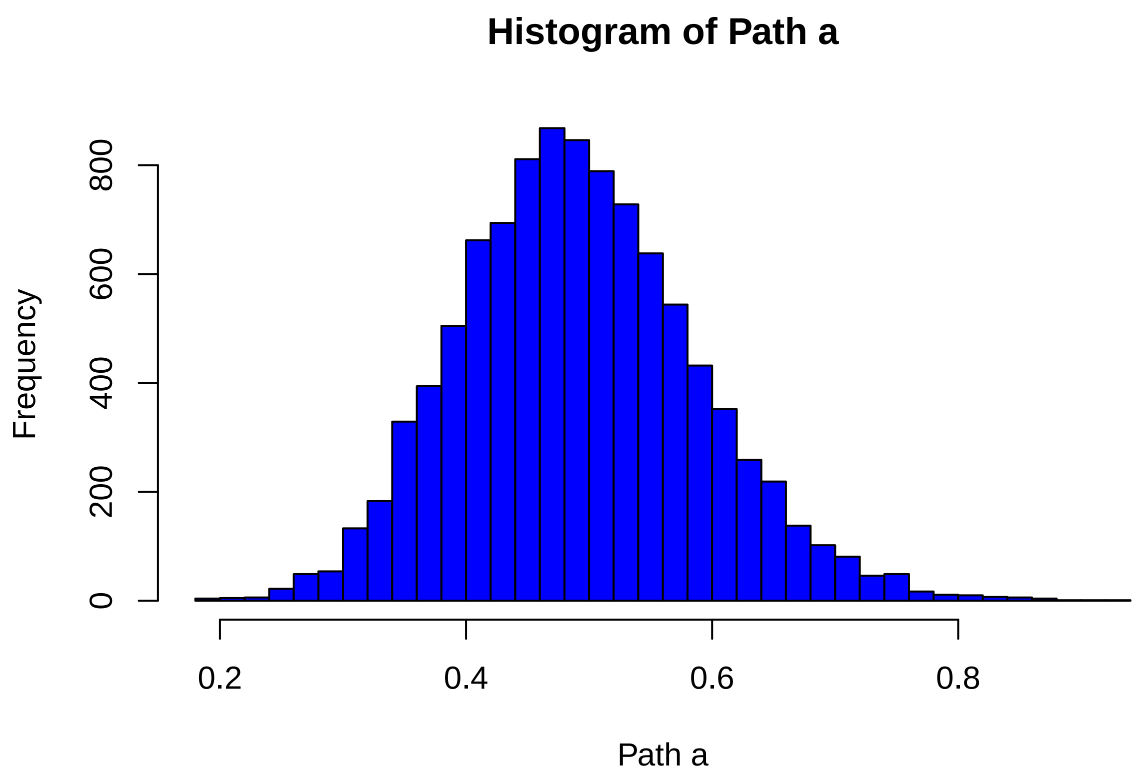
```
## [1] "95% CI for Proportion Mediated:"
```

```r
print(proportion_mediated_ci)
```

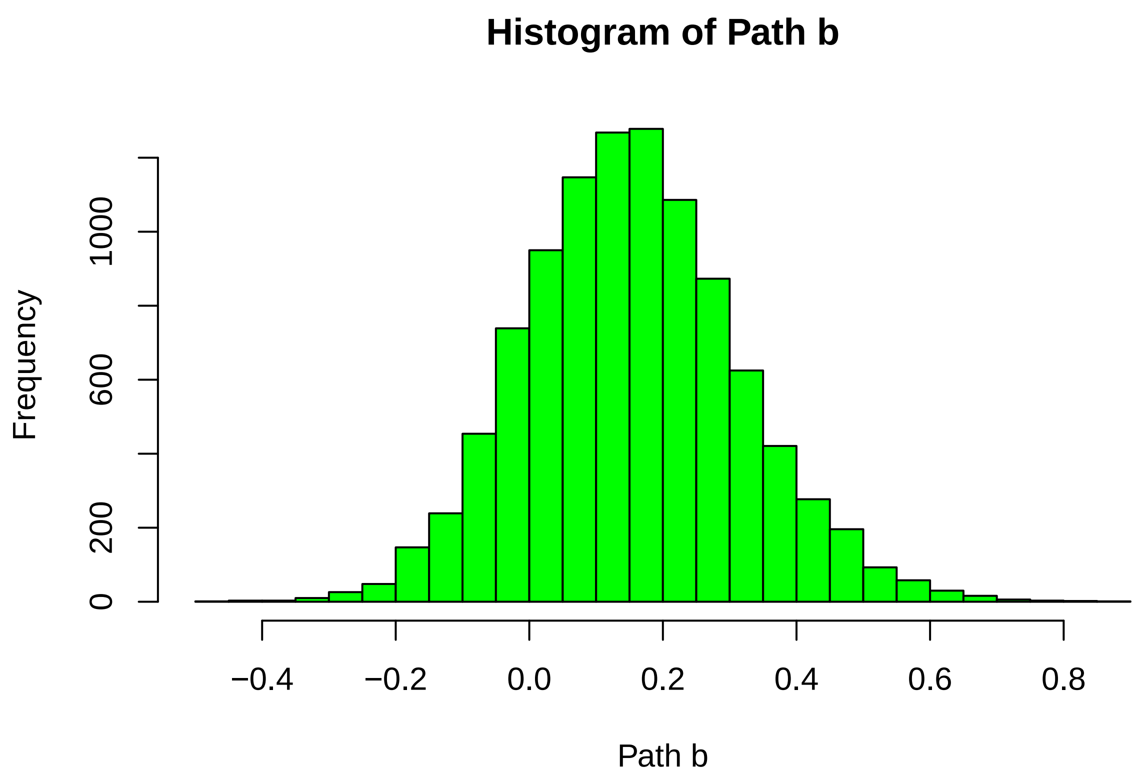
```
##      2.5%     97.5%
## 0.4493164 0.9862231
```

```r
# Extract paths a and b from the bootstrap results
a_path <- bootstrap_estimates[, 1]
b_path <- bootstrap_estimates[, 2]

# Plot histogram for path a
hist(a_path, main="Histogram of Path a", xlab="Path a", col="blue", breaks=30)
```




```r
# Plot histogram for path b
hist(b_path, main="Histogram of Path b", xlab="Path b", col="green", breaks=30)
```




```r
# Load necessary package
library(boot)

# Fit the models
model_c <- lm(LOGpvs_bg_vol ~ Urban + Age + Sex +  eTIV + bg_vol, data = df)
model_a <- lm(AirPollution_2016 ~ Urban + Age + Sex + eTIV + bg_vol, data = df)
model_b <- lm(LOGpvs_bg_vol ~ AirPollution_2016 + Urban + Age + Sex + eTIV + bg_vol, data = df)

# Extract coefficients
a <- coef(model_a)["Urban"]
b <- coef(model_b)["AirPollution_2016"]
c <- coef(model_c)["Urban"]
c_prime <- coef(model_b)["Urban"]

# Compute effects
indirect_effect <- a * b
direct_effect <- c_prime
total_effect <- abs(direct_effect) + abs(indirect_effect)
proportion_mediated <- (abs(total_effect) - abs(direct_effect)) / abs(total_effect)

# Define a function to compute the effects for bootstrapping
boot_function <- function(data, indices) {
  d <- data[indices, ]  # Resample with replacement
  model_a <- lm(AirPollution_2016 ~ Urban + Age + Sex + EducationYears + eTIV + bg_vol, data = d)
  model_b <- lm(LOGpvs_bg_vol ~ AirPollution_2016 + Urban + Age + Sex + EducationYears + eTIV + bg_vol,
  model_c <- lm(LOGpvs_bg_vol ~ Urban + Age + Sex + EducationYears + eTIV + bg_vol, data = d)
```

```r
  a <- coef(model_a)["Urban"]
  b <- coef(model_b)["AirPollution_2016"]
  c <- coef(model_c)["Urban"]
  c_prime <- coef(model_b)["Urban"]

  indirect_effect <- a * b
  direct_effect <- c_prime
  total_effect <-  abs(direct_effect) + abs(indirect_effect)
  proportion_mediated <- (abs(total_effect) - abs(direct_effect)) / abs(total_effect)

  return(c(indirect_effect, direct_effect, total_effect, proportion_mediated))
}

# Perform bootstrapping
set.seed(123)
boot_results <- boot(data = df, statistic = boot_function, R = 10000)

# Extract bootstrap estimates
bootstrap_estimates <- boot_results$t

# Calculate 95% confidence intervals
a_ci <- quantile(bootstrap_estimates[,1], c(0.025, 0.975))
b_ci <- quantile(bootstrap_estimates[,2], c(0.025, 0.975))
indirect_effect_ci <- quantile(bootstrap_estimates[,1], c(0.025, 0.975))
direct_effect_ci <- quantile(bootstrap_estimates[,2], c(0.025, 0.975))
total_effect_ci <- quantile(bootstrap_estimates[,3], c(0.025, 0.975))
proportion_mediated_ci <- quantile(bootstrap_estimates[,4], c(0.025, 0.975))

# Display results
print("Path a (Urban -> AirPollution_2016):")
```

```
## [1] "Path a (Urban -> AirPollution_2016):"
```

```r
print(a)
```

```
##    Urban
## 1.325672
```

```r
print("95% CI for Path a:")
```

```
## [1] "95% CI for Path a:"
```

```r
print(a_ci)
```

```
##       2.5%      97.5%
## 0.09030773 0.70395092
```

```r
print("Path b (AirPollution_2016 -> LOGpvs_bg_vol):")
```

```
## [1] "Path b (AirPollution_2016 -> LOGpvs_bg_vol):"
```

```r
print(b)
```

```
## AirPollution_2016
##         0.2557909
```

```r
print("95% CI for Path b:")
```

```
## [1] "95% CI for Path b:"
```

```r
print(b_ci)
```

```
##       2.5%      97.5%
## -0.1080695  0.8970221
```

```r
# Display results
print("Indirect Effect (c-c`):")
```

```
## [1] "Indirect Effect (c-c‘):"
```

```r
print(indirect_effect)
```

```
##     Urban
## 0.3390948
```

```r
print("95% CI for Indirect Effect:")
```

```
## [1] "95% CI for Indirect Effect:"
```

```r
print(indirect_effect_ci)
```

```
##       2.5%      97.5%
## 0.09030773 0.70395092
```

```r
print("Direct Effect (c'):")
```

```
## [1] "Direct Effect (c’):"
```

```r
print(direct_effect)
```

```
##     Urban
## 0.4229796
```

```r
print("95% CI for Direct Effect:")
```

```
## [1] "95% CI for Direct Effect:"
```

```r
print(direct_effect_ci)
```

```
##       2.5%      97.5%
## -0.1080695  0.8970221
```

```r
print("Total Effect (c):")
```

```
## [1] "Total Effect (c):"
```

```r
print(total_effect)
```

```
##     Urban
## 0.7620744
```

```r
print("95% CI for Total Effect:")
```

```
## [1] "95% CI for Total Effect:"
```

```r
print(total_effect_ci)
```

```
##      2.5%     97.5%
## 0.4000679 1.1663594
```

```r
print("Proportion Mediated (a*b/c):")
```

```
## [1] "Proportion Mediated (a*b/c):"
```

```r
print(proportion_mediated)
```

```
##     Urban
## 0.4449628
```

```r
print("95% CI for Proportion Mediated:")
```

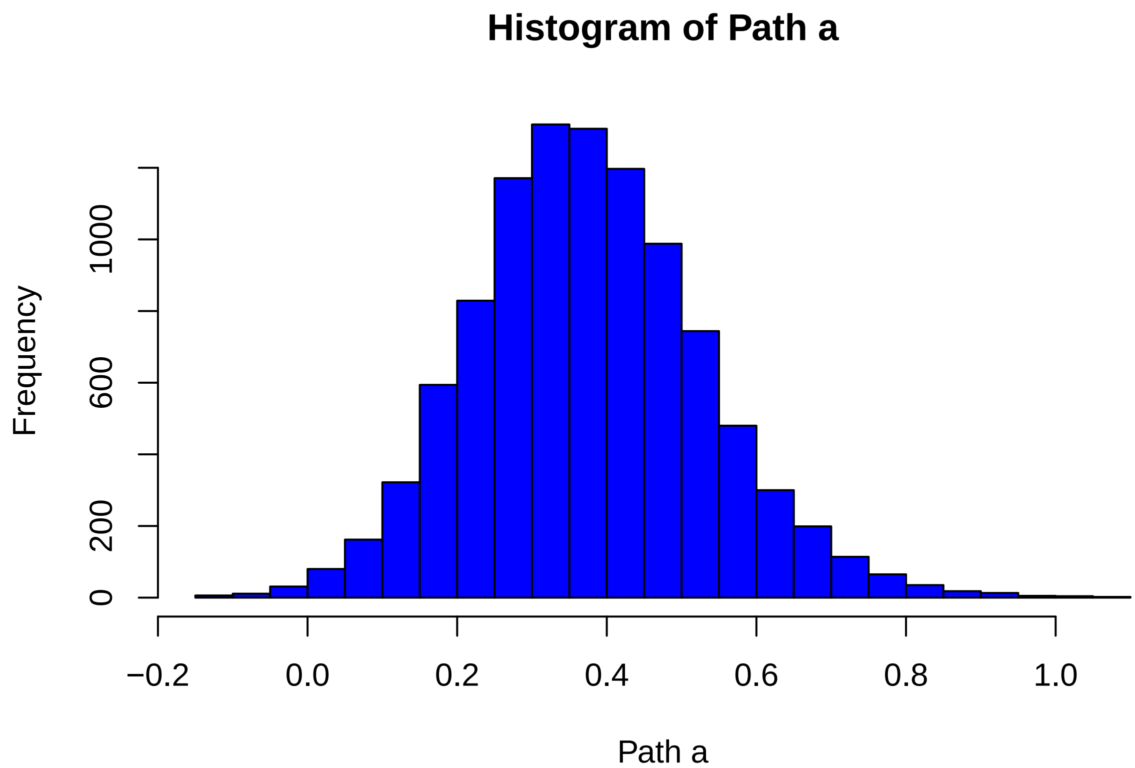
```
## [1] "95% CI for Proportion Mediated:"
```

```r
print(proportion_mediated_ci)
```

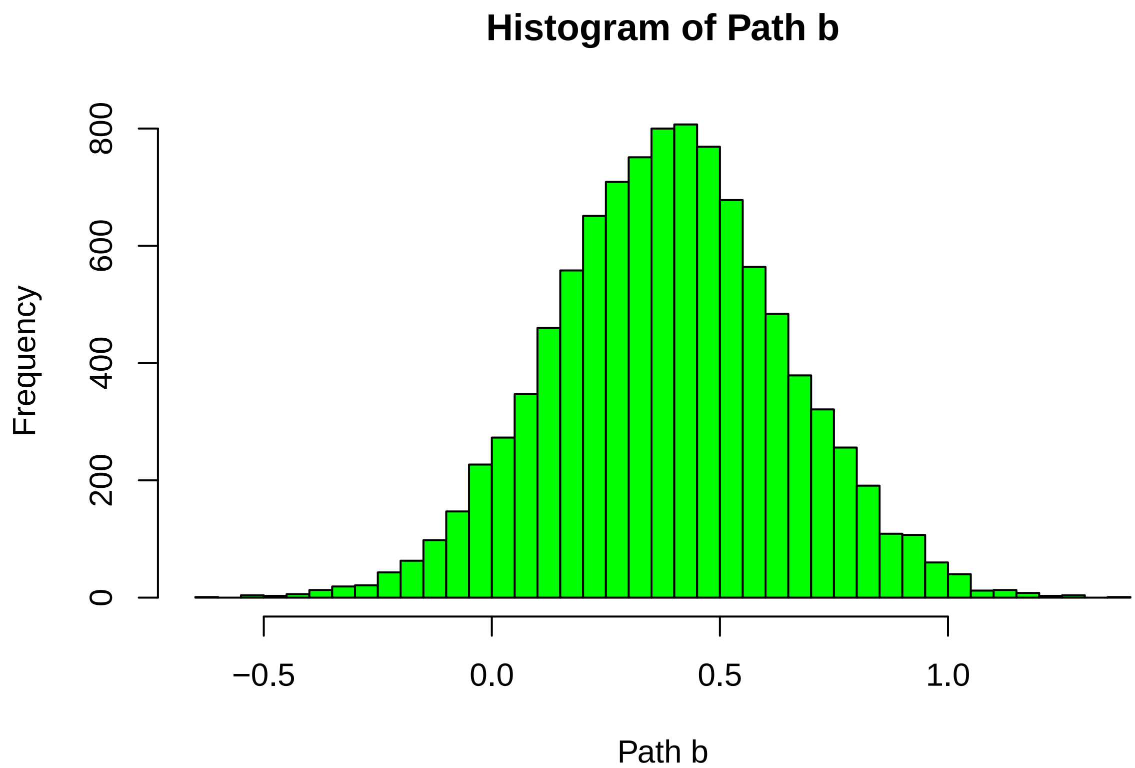
```
##      2.5%     97.5%
## 0.1098081 0.9549832
```

```r
# Extract paths a and b from the bootstrap results
a_path <- bootstrap_estimates[, 1]
b_path <- bootstrap_estimates[, 2]

# Plot histogram for path a
hist(a_path, main="Histogram of Path a", xlab="Path a", col="blue", breaks=30)
```




```
# Plot histogram for path b
hist(b_path, main="Histogram of Path b", xlab="Path b", col="green", breaks=30)
```


Histogram of Path b
Frequency
0
200
400
600
800
−0.5
0.0
0.5
1.0
Path b