

Exploratory Data Analysis on NCES Data

Developed by Yuqi Liao, Paul Bailey, and Ting Zhang*[†]

May 10, 2018

Vignette Outline

This vignette provides examples of conducting exploratory data analysis (EDA) on NAEP data, organized by the following plot types:

- Bar charts
- Histograms
- Box plots
- Contour plots
- Residual plots

The `EdSurvey` package gives users functions to efficiently analyze education survey and assessment data, taking into account complex sample survey designs and the use of plausible values. The `getData()` function in the `EdSurvey` package generates a `data.frame` that allows for EDA on the requested NAEP data. For more details on setting up the environment for analysis and for using the `getData()` function, please see *Using the getData Function in EdSurvey to Manipulate the NAEP Primer Data*.

Preparation of the EDA

The first step is to set up the environment and read in data for the EDA. This vignette uses data from the `NAEPprimer` package. Please see *Using EdSurvey to Analyze NAEP Data: An Illustration of Analyzing NAEP Primer* for more details.

```
# R code for loading the EdSurvey package and reading in NAEP data
library(EdSurvey)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))
```

Generate the Data Frame of Interest

The `sdf` variable is a pointer to the data, but no actual data have yet been read in. This allows `EdSurvey` to conserve memory usage, reading in only necessary variables. Before we can do EDA, we will need a `data.frame` with the actual data on it. Users can use `getData()` to generate a `data.frame` for further analysis.

```
# R code for generating a data.frame for further analysis
gddat <- getData(sdf,
  c('dsex', 'sdracem', 'pared', 'b017451',
    'composite', 'geometry', 'origwt'),
  omittedLevels = FALSE)
```

In this example, `getData` extracts the following:

*This publication was prepared for NCES under Contract No. ED-IES-12-D-0002 with the American Institutes for Research. Mention of trade names, commercial products, or organizations does not imply endorsement by the U.S. Government.

[†]The authors would like to thank Dan Sherman and Claire Kelley for reviewing this document.

- four variables: `dsex` (Gender), `sdracem` (Race/ethnicity (from school records)), `pared` (Parental education level (from 2 questions)), and `b017451` (Talk about studies at home)
- five plausible values associated with `composite` and five others with `geometry`
- the weights for this data frame: `origwt` and 62 replicate weights (`srwt01` to `srwt62`)

`omittedLevels` is set to `FALSE` so that variables with special values (such as multiple entries or NAs) can still be returned by `getData` and manipulated by the user. The default setting (i.e., `omittedLevels = TRUE`) removes these values from factors that are not typically included in regression analysis and cross-tabulation.

`origwt` is the default student weight in the `gddat`. To find the default weight for any `edsurvey.data.frame`, use the following code.

```
# R code to display the default weight of an edsurvey.data.frame
showWeights(sdf)
```

```
## There are 1 full sample weight(s) in this edsurvey.data.frame
## 'origwt' with 62 JK replicate weights (the default).
```

On the NAEP Primer, there is only one set of weights, but the default weights will always be followed by the parenthetical text, “the default,” as `origwt` is here.

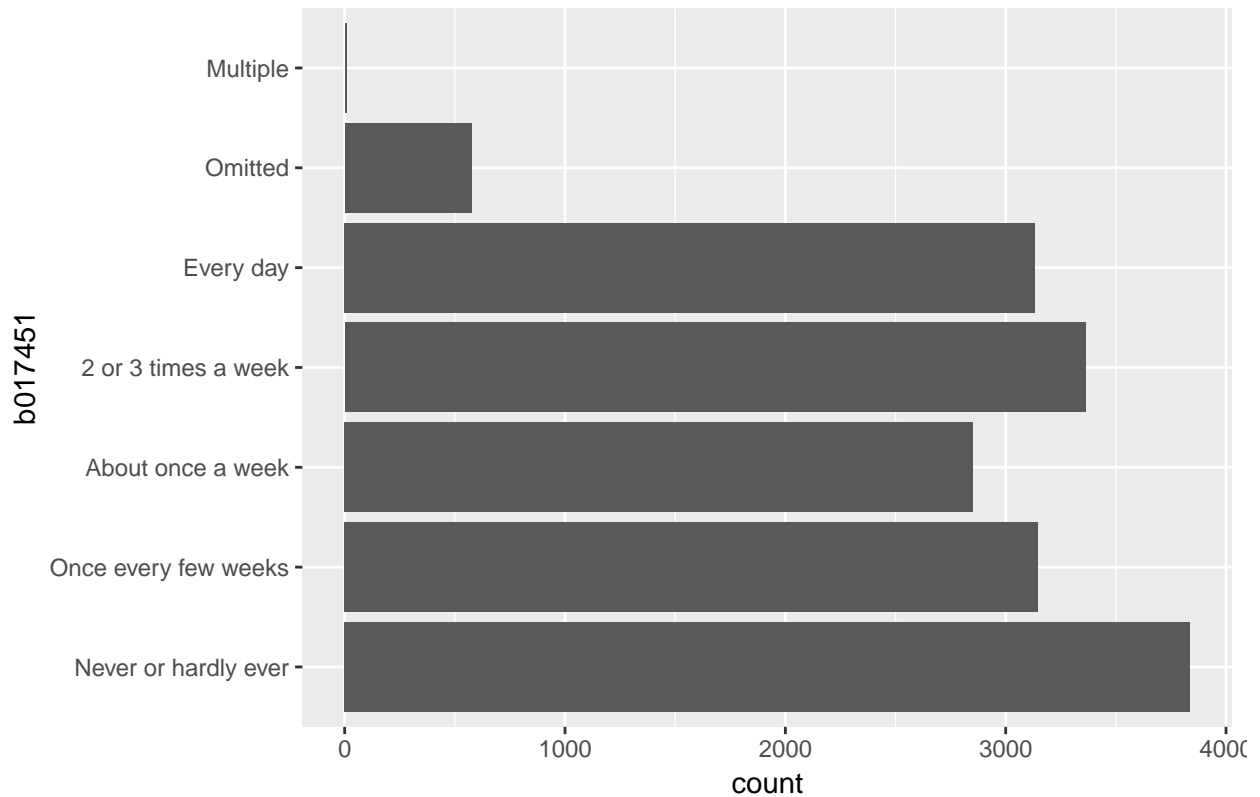
Because doing EDA is about exploring the data, the visuals created during the EDA process are usually preliminary. In the same spirit, the charts presented in the main body of this vignette all adopt a “quick and dirty” approach—unless otherwise noted, they have not had weights applied and, in the case where plausible values are visualized, only one set of plausible values is used.

Bar Charts

Figure 1 shows a bar chart with counts of the variable `b017451` in each category.

```
# R code for generating Figure 1
require(ggplot2)
bar1 <- ggplot(data=gddat, aes(x=b017451)) +
  geom_bar() +
  coord_flip() +
  labs(title = "Figure 1")
bar1
```

Figure 1

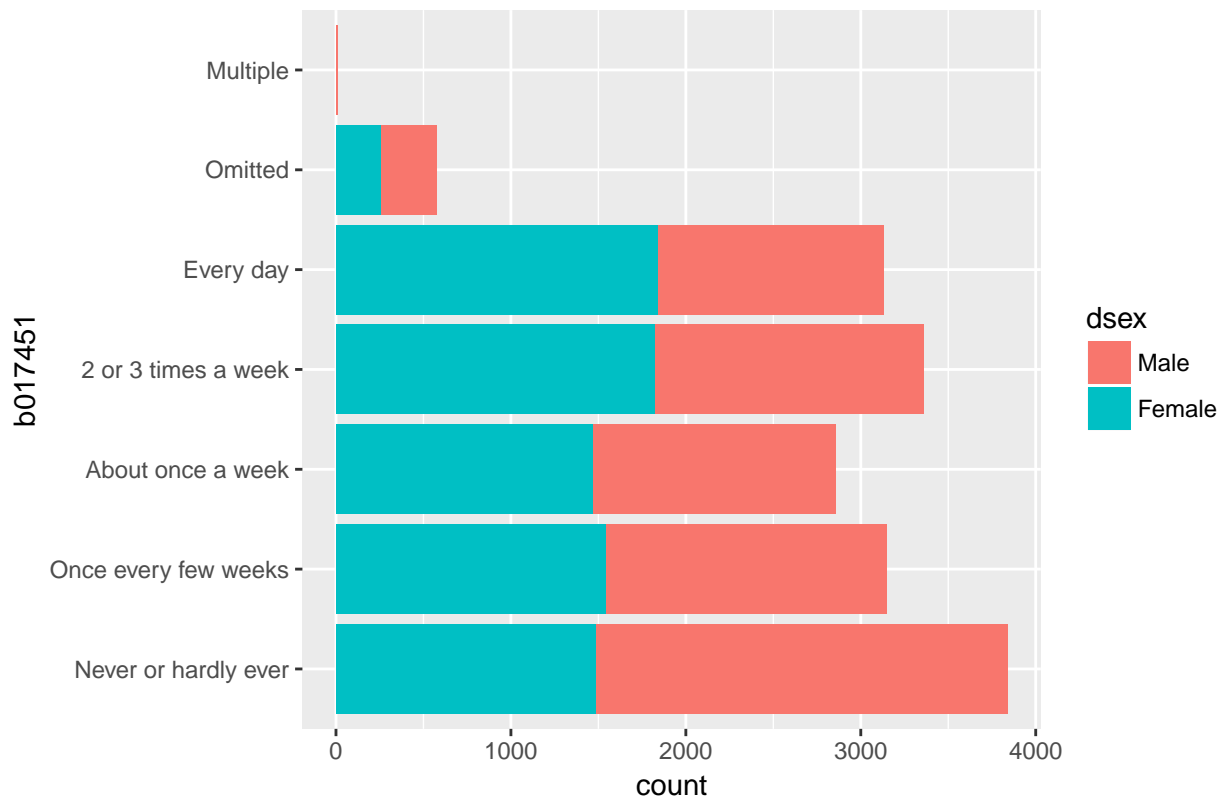


Note that these counts are unweighted and represent only the counts of survey respondents. Because this vignette examines EDA, the proper use of weights is not covered.

We can divide the bar chart further by another variable. Figure 2 shows the counts for each level of b017451 broken down by dsex.

```
# R code for generating Figure 2
bar2 <- ggplot(data=gddat, aes(x=b017451, fill=dsex)) +
  geom_bar() +
  coord_flip() +
  labs(title = "Figure 2")
bar2
```

Figure 2



Bar charts can be customized in many ways, including changing its color palette. We can do it manually or automatically, as demonstrated below. Users may find it helpful to read tutorials online that are about *Colors in R* and *How to Change Colors Automatically and Manually*.

```
# R code for changing the color palette manually  
bar2 + scale_fill_manual(values=c("#E69F00", "#56B4E9"))
```

```
# R code for changing the color palette automatically  
bar2 + scale_fill_brewer(palette="Dark2")
```

The `ggplot2` R package is a powerful plotting platform. Here we use just a few features, although users also could customize the plots extensively.¹ R was built on the ability to make customizable graphics, so base R plotting functions also are more than capable of generating all the plots shown in this document.

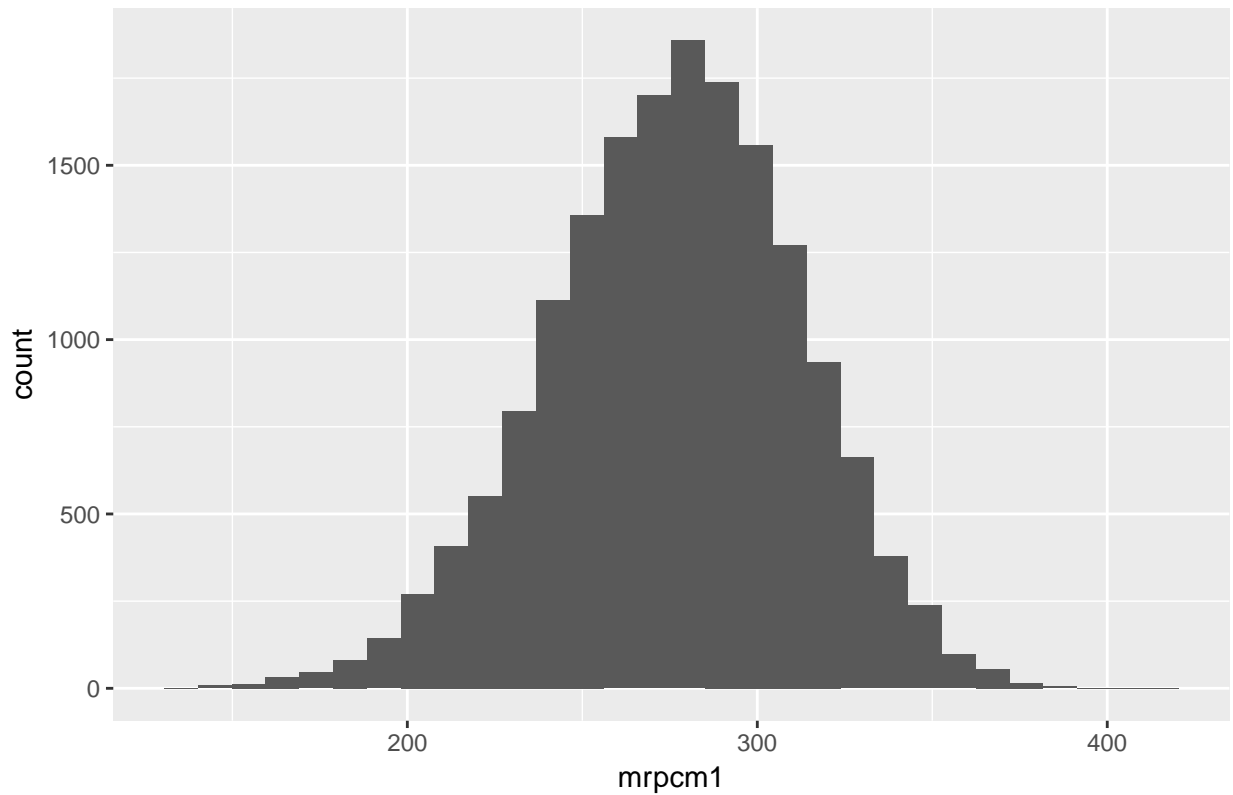
Histograms

The histogram is a good way to visualize the distribution of continuous variables. Figure 3 is a basic histogram that uses the first plausible value of `composite`. Although all plausible values could be used, generating a histogram with a single plausible gives an unbiased (but unweighted) estimate of the frequencies in each bin.

```
# R code for generating Figure 3  
hist1 <- ggplot(gddat, aes(x=mrpcm1)) +  
  geom_histogram() +  
  labs(title = "Figure 3")  
hist1
```

¹See <http://ggplot2.tidyverse.org/index.html> for an overview of `ggplot2` and useful resources to learn the package.

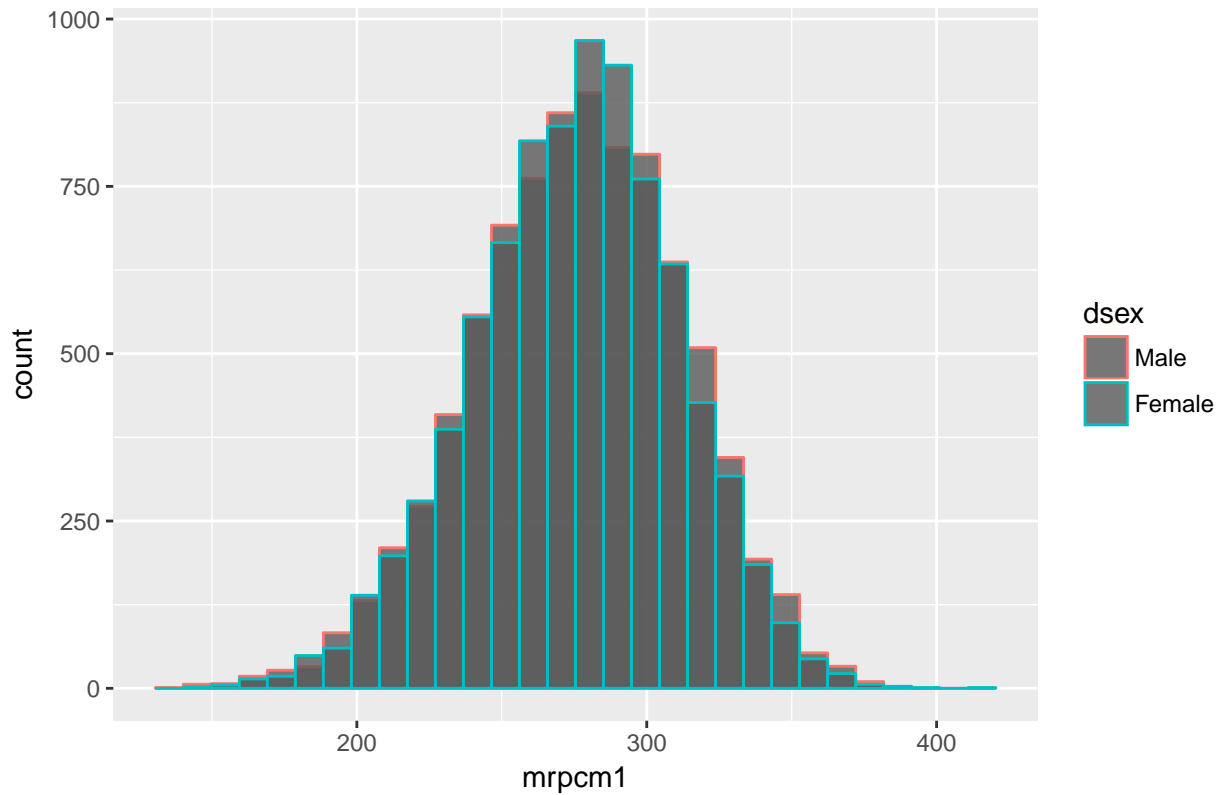
Figure 3



If we add a categorical variable `dsex`, the output will be two histograms that share the x and y axes, as shown in Figure 4.

```
# R code for generating Figure 4  
hist2 <- ggplot(gddat, aes(x=mrpcm1, color = dsex)) +  
  geom_histogram(position="identity", alpha = 0.8) +  
  labs(title = "Figure 4")  
hist2
```

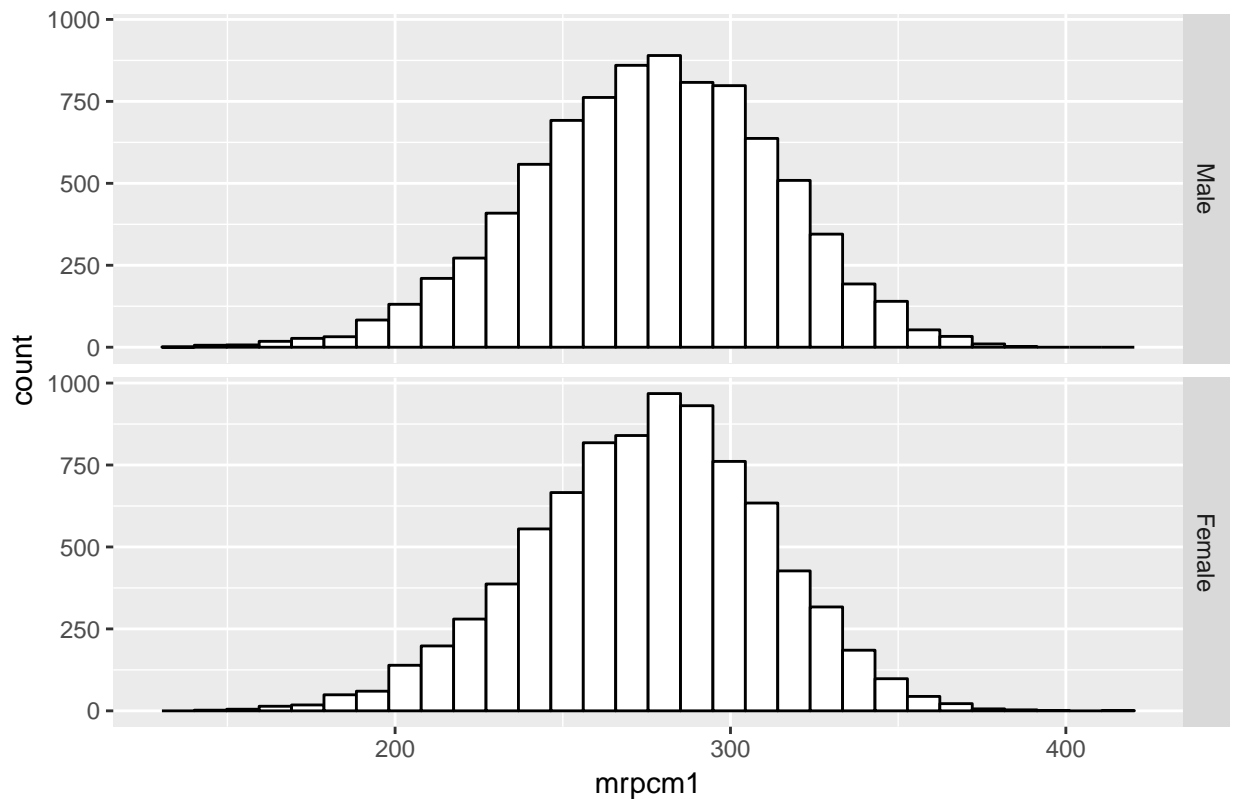
Figure 4



We can view these multiple histograms separately, using facets. In Figure 5, `mrpcm1` for male and female both start at zero on the y axis.

```
# R code for generating Figure 5
hist3 <- ggplot(gddat, aes(x=mrpcm1))+
  geom_histogram(color="black", fill="white")+
  facet_grid(dsex ~ .) +
  labs(title = "Figure 5")
hist3
```

Figure 5

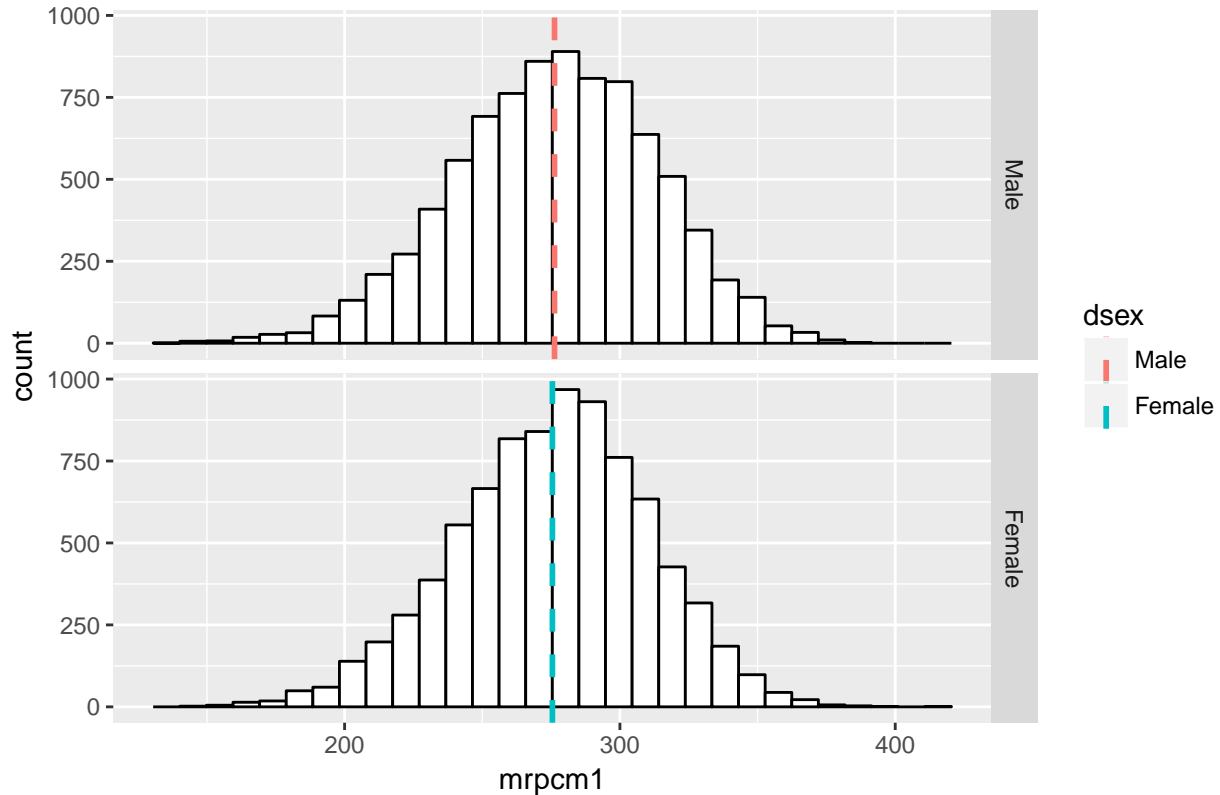


We can use `aggregate()` to calculate more statistics for the EDA process. The following example demonstrates how to add the mean of `mrpcm1` for each gender in the histograms. Figure 6 shows that the means for male and female students are close to each other.

```
# R code for adding mean lines
meanlines <- aggregate(mrpcm1 ~ dsex, gddat, mean, na.rm=TRUE)

# R code for generating Figure 6
hist4 <- ggplot(gddat, aes(x=mrpcm1))+
  geom_histogram(color="black", fill="white") +
  facet_grid(dsex ~ .) +
  geom_vline(data=meanlines, aes(xintercept=mrpcm1, color=dsex),
            linetype="dashed", size=1) +
  labs(title = "Figure 6")
hist4
```

Figure 6

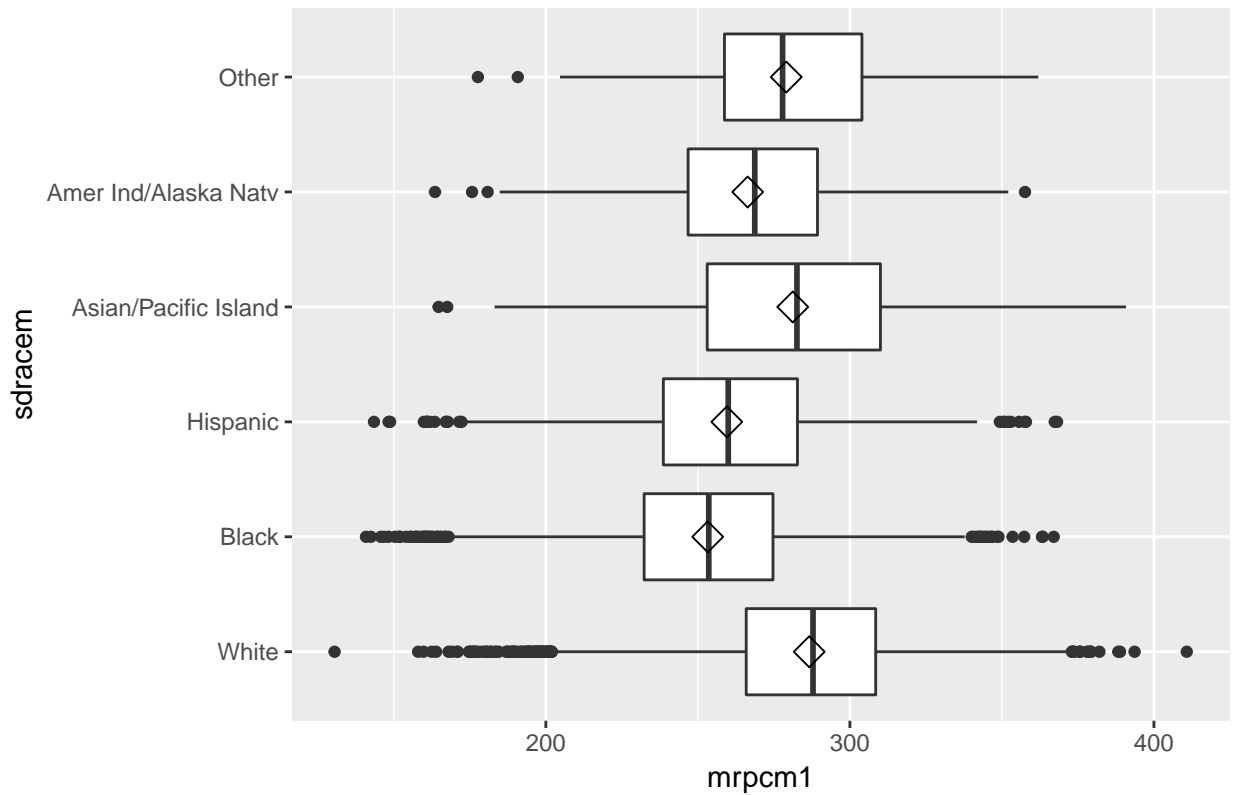


Box Plots

Box plots offer another way to visualize a distribution. A box plot shows the distribution of a single variable through its quartiles. Box plots also may have lines extending vertically from the boxes, called *whiskers*, indicating variability outside the upper and lower quartiles, hence the term *box-and-whisker plot*. Outliers may be plotted as individual points. In Figure 7, we create a basic box plot by using `geom_boxplot()`, which shows us five statistics of the distribution of `mrpcm1` by `sdracem`. At the far left and far right sides of the graph are the minimum and the maximum, respectively. The left and right sides of the box are the first and the third quartiles, respectively. The vertical bar in the box indicates the median. Using `stat_summary()`, we can add another statistic on top of the basic box plot: the mean of `mrpcm1` by `sdracem`. This mean is represented by the diamond-shaped symbol, as indicated by `shape=23`. For a comprehensive list of symbols, see [this reference page](#).

```
# R code for generating Figure 7
box1 <- ggplot(gddat, aes(x=sdracem, y=mrpcm1)) +
  geom_boxplot() +
  stat_summary(fun.y=mean, geom="point", shape=23, size=4) +
  coord_flip() +
  labs(title = "Figure 7")
box1
```


Figure 7



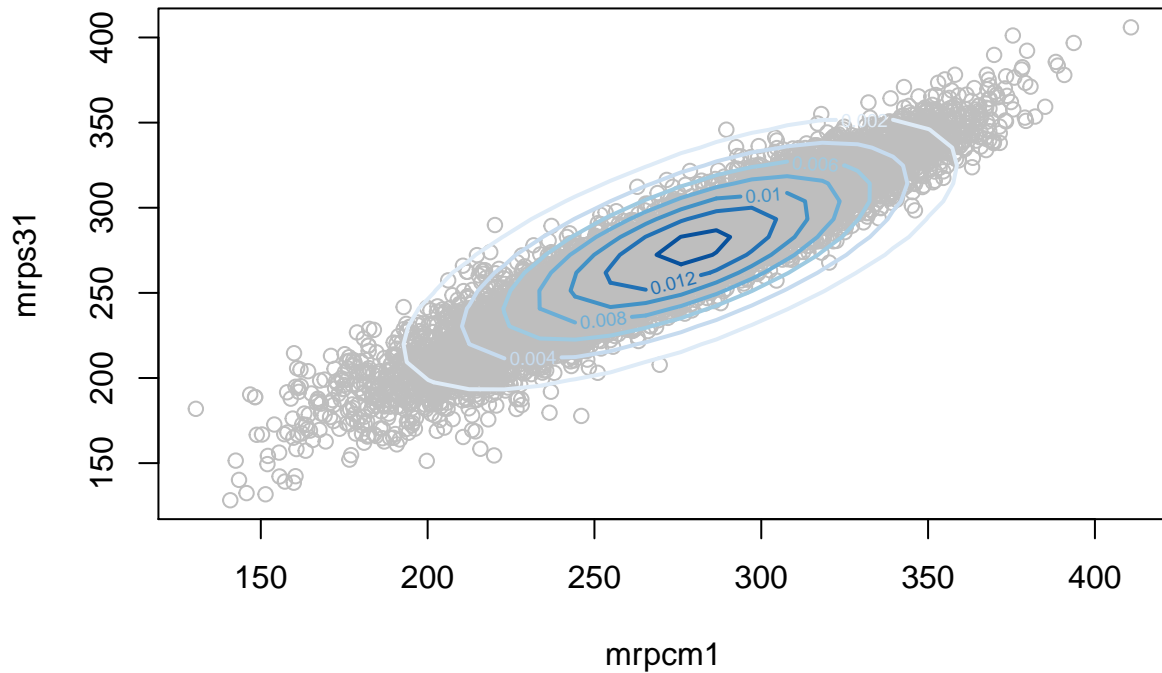
Contour Plots

Contour plots can be used to visualize the relationship between two variables. We used a built-in plotting function `contourPlot` to generate a base contour plot,² as shown in Figure 8. The figure shows the distribution of `mrpcm1` (the first plausible value of `composite`) in relation to `mrps31` (the first plausible value of `geometry`), as well as the joint density of these two variables.

```
# R code for generating Figure 8  
contourPlot(x = gddat$mrpcm1, y = gddat$mrps31, xlab = "mrpcm1",  
            ylab = "mrps31", main="Figure 8")
```

²Use `?contourPlot` in the R console to see more.

Figure 8



Residual Plots

To explore the relationship between multiple variables, regression analyses are usually conducted. The built-in regression function `lm.sdf` fits a linear regression model that fully accounts for the complex sample design used for the NAEP data. In this example, we begin by estimating a linear regression model in which we regress `mrpcm1` on `pared`.

```
# R code for estimating the linear regression model  
summary(lm1 <- lm.sdf(composite ~ pared, sdf))
```

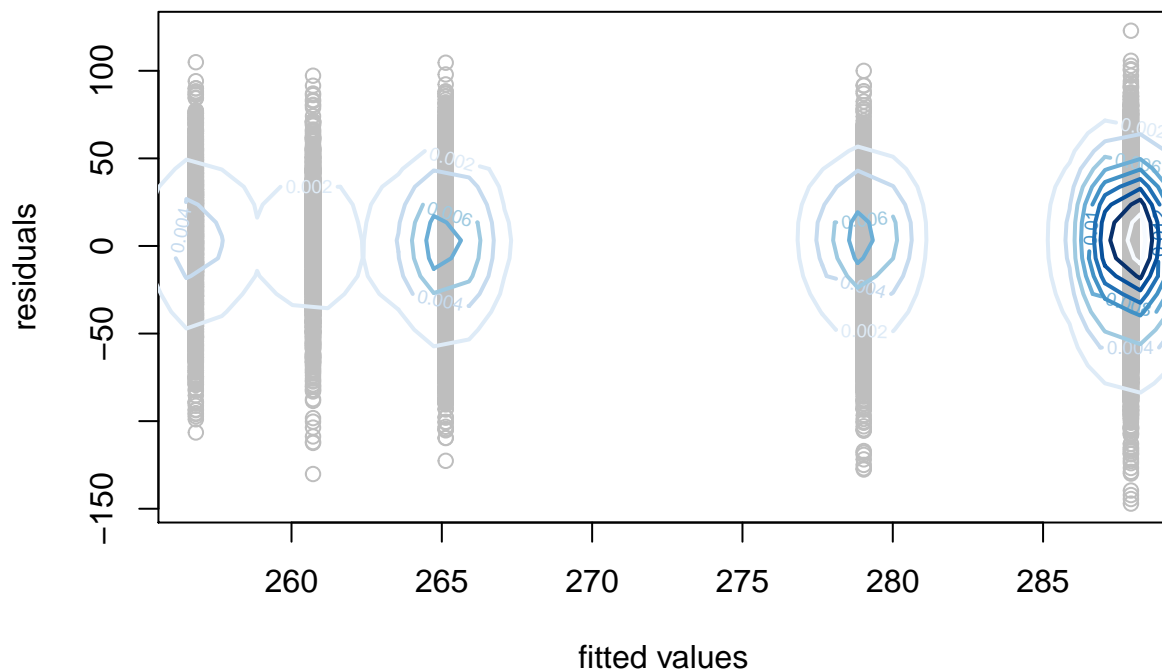
```
##  
## Formula: composite ~ pared  
##  
## jrrIMax: 1  
## Weight variable: 'origwt'  
## Variance method: jackknife  
## JK replicates: 62  
## full data n: 17606  
## n used: 16328  
##  
## Coefficients:  
##           coef          se         t   dof  Pr(>|t|)  
## (Intercept)    260.7158    1.3274 196.4044 41.703 < 2.2e-16 ***  
## paredGraduated H.S.      4.4132    1.5935   2.7695 62.168 0.007393 **  
## paredSome ed after H.S. 18.3193    1.3558 13.5118 55.724 < 2.2e-16 ***  
## paredGraduated college  27.2069    1.6424 16.5652 45.714 < 2.2e-16 ***
```

```
## paredI Don't Know      -3.8982   1.6964  -2.2979  42.210  0.026592 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.1114
```

Given the regression results (please note that weights and all sets of plausible values of `composite` are used in the previous regression), we can draw a *residual* plot to see if residuals of the regression have nonlinear patterns. The residual plot, as shown in Figure 9, is in the form of a contour plot. On the *x* axis are the fitted values, and on the *y* axis are the residuals. The contour lines reflect the probability density of the data. If residuals of a regression are spread equally across fitted values, it indicates that the regression is unlikely to have nonlinear relationships.

```
# R code for generating Figure 9
contourPlot(x=lm1$fitted.values, y=lm1$residuals[,1],
            xlab="fitted values", ylab="residuals", main="Figure 9")
```

Figure 9



As an example, we also want to show a regression with a nonlinear relationship between the dependent and the key independent variables. To do this, we generate a variable that has a nonlinear relationship with the dependent variable. First, we read in `gddat` again but add the argument `addAttributes` set to `TRUE`. `addAttributes` is set to `TRUE` so that the object (`gddat`) returned by this call to `getData` can be passed to other `EdSurvey` package functions (in this case, `lm.sdf`) if needed. For more details on this, see the vignette titled *Using the getData Function in EdSurvey to Manipulate the NAEP Primer Data*.

```
# R code for generating a data.frame with addAttributes set to TRUE
ledat <- getData(sdf,
                c('dsex', 'sdracem', 'pared', 'b017451',
                  'composite', 'geometry', 'origwt'),
```

```
addAttributes = TRUE, omittedLevels = FALSE)
```

```
# R code for creating a quadratic variable to be used as the key independent variable  
# in the lm.sdf regression  
ledat$quardVar <- sqrt(ledat$mrpcm1-rnorm(length(ledat$mrpcm1)))
```

Figure 10 shows the residual plot for the nonlinear relationship between the new variable (quardVar), intentionally generated to be nonlinear, and the dependent variable mrpcm1.

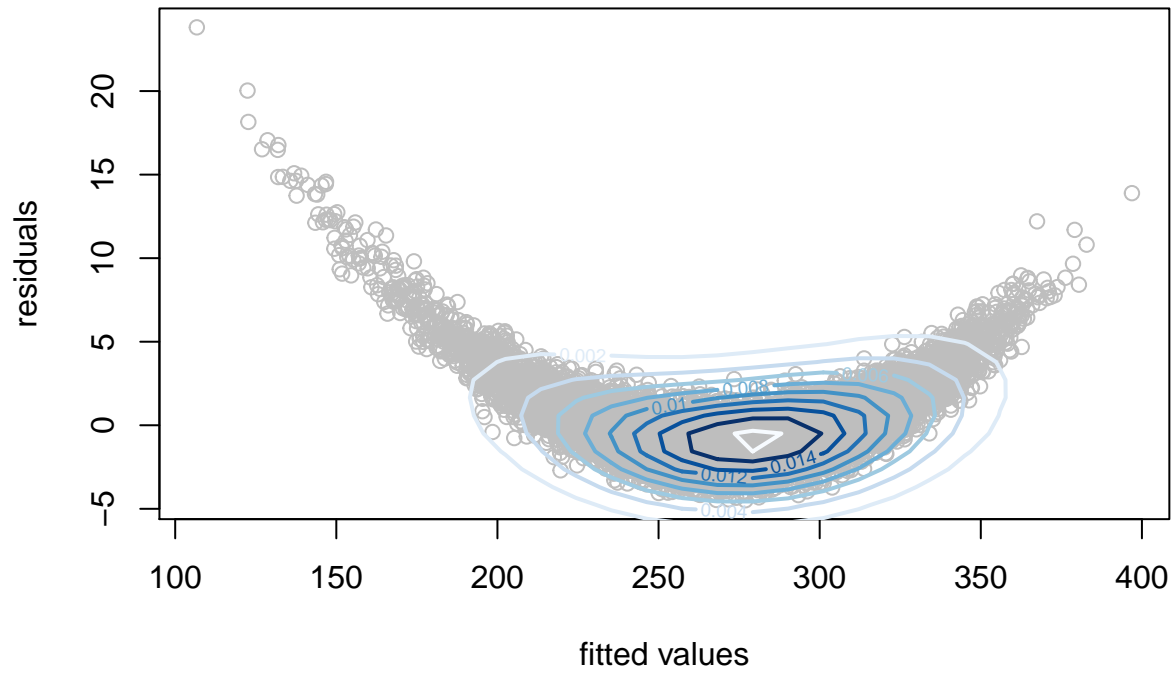
```
# R code for estimating the linear regression model  
summary(lm2 <- lm.sdf(mrpcm1 ~ quardVar, data=ledat))
```

```
##  
## Formula: mrpcm1 ~ quardVar  
##  
## Weight variable: 'origwt'  
## Variance method: jackknife  
## JK replicates: 62  
## full data n: 17606  
## n used: 16915  
##  
## Coefficients:  
##           coef           se           t    dof Pr(>|t|)  
## (Intercept) -266.202813    0.957149  -278.120609 41.931 < 2.2e-16 ***  
## quardVar      32.711198    0.057548  568.419505 42.196 < 2.2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Multiple R-squared:  0.9969
```

In Figure 10, the residuals are not spread equally across fitted values. It thus indicates that the regression is likely nonlinear.

```
# R code for generating Figure 10  
contourPlot(x=lm2$fitted.values, y=lm2$residuals[,1],  
            xlab="fitted values", ylab="residuals", main="Figure 10")
```

Figure 10



Notes

For a full list of EdSurvey functions and documentation, use the R help viewer:

```
# R code for viewing EdSurvey documentation  
help(package="EdSurvey")
```