Handout 2

Put your name here!

In this handout, we will learn how to subset data, a crucial step in adjusting for confounders. We will also learn how to plot points and lines, as well as how to create density plots.

Topics and Concepts Covered

- Subsetting vectors
- Subsetting data frames
- Creating tables
- Producing density plots with multiple densities in the same figure

R Commands Covered

- Subsetting a vector using `[` and `]`
- Boolean operators for subsetting: `>`, `>=`, `<`, `<=`, and `==`
- Creating multiple conditions through using `&` and `|`
- Learning the class of a vector (`factor`, `numeric`, `character`) with `class`
- Converting numeric vectors to factors with `as.factor`
- Setting the levels of a factor with `levels`
- Using `barchart` to create bar charts
- Using `plot` and `lines` to create and add lines to a figure
- Using `density` to make density plots
- Using `legend` to add a legend to a plot

Before beginning this handout do not forget to make a new folder for this assignment and set your working directory!

Boolean (Logical) Operators Commonly Used for Subsetting

- `>`, `>=`. Greater than, greater than or equal to
- `<`, `<=`. Less than, less than or equal to
- `==`. Exactly equal to
- `&`. And
- `|`. Or

Examples

Return all elements of `y` for which `x` is less than (less than or equal to) 2

```
y[ x < 2]
y[x <= 2]
```

Return all elements of `y` for which `x` is exactly equal to 2

```
y[ x == 2]
```

Return all elements of `y` for which `x` is exactly equal to `z`

```
y[ x == z]
```

Return all elements of `y` for which `x` is greater than 2 *and* `x` is less than 5

```
y[(x > 2) & (x < 5)]
```

Return all elements of y for which x is less than or equal to 2 *or* x is greater than 5

```
y[(x <= 2) | (x > 5)]
```

Summary of Options Used in Figures

Here are some parameters with example values you can use with `plot`, `lines`, `points`:

- `main = "Distribution of Wealth"`. Set the main figure title.
- `xlab = "Time"`. Set the x-axis label.
- `ylab = "Density"`. Set the y-axis label.
- `xlim = c(-1, 2.4)`. Constrain the x-axis to start at -1 and end at 2.4.
- `ylim = c(0, 1)`. Constrain the y-axis to start at 0 and end at 1.
- `lty = 2`. Change the line type. 1 is solid, 2 is dashed, and 3–5 are different types of dashed lines.
- `pch = 19`. Set the plotting character of points. 1 is an unfilled circle. See the help page for `points` for the complete list.
- `col = "red"`. Set the line or point color. You can provide more than one!

Legends

`legend` adds a legend to your plot. It has the following arguments:

- The first argument is the legend's location. Choose one of 'topleft', 'bottomleft', 'topright', or 'bottom right'.
- `legend` is a vector of character strings, indicating what the legend should contain.
- `col`. A vector of colors, corresponding to the elements of `legend`.
- `lty`. A vector of line types, corresponding with the elements of `legend`.
- `bg = "grey"`. Turns the background of the legend from the default background color, usually white, to grey.

Subsetting Data

Subsetting data is going to be a crucial component of this course. We are going to explore how the relationship between two variables changes as we move from one subset of the data to another, and use this information to draw inferences.

Numeric, Factor, and Character Variables

In order to learn how to select subsets of data, we are going to begin with subsetting a single vector. When we *subset* a vector, we are looking only at a portion of the vector that satisfies certain conditions. To motivate this example, we are going to consider the "Stop-and-Frisk" data (Gelman, Fagan, and Kiss 2007) that we used in the last session.

First, we are going to load in the data and make sure that it loaded properly.

```
Data.SAF <- read.csv("data/subset_saf.csv")
head(Data.SAF)
```

```
  year pct ser_num datestop timestop city sex race age
1 2012  44     711 20120112     1505    3   1    1  17
2 2012  32    1753 20120208     1330    1   1    1  53
3 2012  44    3020 20120214     1635    3   1    1  22
4 2012  32    2063 20120217      230    1   1    1  40
```

```
5 2012  44     5339 20120217      2105     3   1     1  57
6 2012  67     2744 20120218      1910     2   1     1  31
```

Everything looks fine, so let's look at the distribution of age ranges.

```r
table(Data.SAF$age)
```

```
   0    1    2    3    5    6    9   11   12   13   14   15   16   17   18 
   3   30    4    1    3    1    1    3   21  104  287  642  926 1047 1094 
  19   20   21   22   23   24   25   26   27   28   29   30   31   32   33 
1061 1016 1047  995  786  766  694  621  555  525  505  495  411  439  332 
  34   35   36   37   38   39   40   41   42   43   44   45   46   47   48 
 289  313  276  229  212  214  225  220  228  174  190  204  184  197  185 
  49   50   51   52   53   54   55   56   57   58   59   60   61   62   63 
 183  179  182  147  134  123  100   80   70   64   52   47   42   29   27 
  64   65   66   67   68   69   70   71   72   73   74   75   76   77   79 
  20   18   14   13   15    7   12    4   11    2    2    7    1    2    1 
  80   82   99  100  117  123  130  150  160  165  169  170  176  180  181 
   1    2   10    2    1    1    1    1    2    2    1    4    1    5    1 
 185  195  215  230  245  347  396  511  520  999 
   2    1    1    1    1    1    1    1    1    4 
```

Now, 999 is commonly used to denote missing data. If 999 were a marker for missing data here, we would expect to see a top age of somewhere in the 90's, and then a big gap between the highest age and 999. That is not the case here. But we do see people recorded at 230, 245, 347, and 396 years old.

Numeric Variables

Let's say that we wanted to look at the distribution of race for different age limits. In that case, we first want to create a variable that is a nominal variable, which R calls a `factor`. To do this, first check the class of the race variable.

```r
class(Data.SAF$race)
```

```
[1] "integer"
```

```r
mean(Data.SAF$race)
```

```
[1] 2.019338
```

You will see that the race variable has a class of `integer`. R interprets variables with the class `numeric` and `integer` as quantitative variables. Therefore, we can perform basic arithmetic operations, such as taking the mean or median on variables of this type.

Factors

R maintains a special class of variable for nominal variables. These variables, called `factors`, are simply interpreted as a set of levels. So, even if R returns levels of 1, 2, 3, etc., these are interpreted as categories, and not numbers. Let's look at an example, where we use the command `as.factor` to create a version of race that is a factor:

```r
Data.SAF$race2 <- as.factor(Data.SAF$race)
head(Data.SAF)
```

```
  year pct ser_num datestop timestop city sex race age race2
1 2012  44     711 20120112     1505    3   1    1  17     1
2 2012  32    1753 20120208     1330    1   1    1  53     1
3 2012  44    3020 20120214     1635    3   1    1  22     1
```

```
4 2012  32     2063 20120217       230     1    1     1  40       1
5 2012  44     5339 20120217      2105     3    1     1  57       1
6 2012  67     2744 20120218      1910     2    1     1  31       1
```

Notice how `race2` was appended to the last column of the data frame `Data.SAF`. To check the class of `race2`, we can do the following:

```r
class(Data.SAF$race2)
```

```
[1] "factor"
```

```r
levels(Data.SAF$race2)
```

```
[1] "1" "2" "3" "4" "5" "6"
```

```r
# mean(Data.SAF$race2) # Run this line -- it returns an NA
```

Notice that the mean of a factor is nonsensical; R returns `NA`. The levels of the variable `race2` are names of each level. We can make these levels more informative using information from the code book, which can be found here.

```r
levels(Data.SAF$race2) <- c("Bl", "Bl-Hs", "Wh-Hs", "Wh", "As", "Am In")
head(Data.SAF)
```

```
  year pct ser_num datestop timestop city sex race age race2
1 2012  44      711 20120112      1505     3    1     1  17     Bl
2 2012  32     1753 20120208      1330     1    1     1  53     Bl
3 2012  44     3020 20120214      1635     3    1     1  22     Bl
4 2012  32     2063 20120217       230     1    1     1  40     Bl
5 2012  44     5339 20120217      2105     3    1     1  57     Bl
6 2012  67     2744 20120218      1910     2    1     1  31     Bl
```

The variable `race2` now displays in a more informative manner. We are using these abbreviations so that the labels fit on the figures and tables we are going to produce; later in the course, we will discuss ways to shrink the text so that it fits on the page better.

Character

Occasionally, you may see a variable that has a class of `character`. In this case, R is interpreting the variable as a string of text. Variables in this form are most commonly used to label axes or figures, not for analysis. To turn the variable `race2` into a character vector, use

```r
char.race <- as.character(Data.SAF$race2)
char.race[1:5] # Look at the first 5 elements
```

```
[1] "Bl" "Bl" "Bl" "Bl" "Bl"
```

The quotation marks indicate that the vector is being interpreted as a character.

Coding Tip

Occasionally, when R reads in data, it will interpret a variable as having class character or factor when you want it to be numeric. We can use `as.numeric` to coerce a variable to be interpreted as numeric.

If you have a variable that is numeric that you want to be a factor, you can use `as.factor` to coerce the variable into a factor.

Subsetting a Vector

In this section, we are going to look at subsets of a vector. In order to produce a subset of a vector, R uses the following syntax:

```
variable[condition]
```

where `variable` is the name of some variable, and `condition` is an expression saying what observations you want to look at. To illustrate, we are first going to look at the distribution of stops-and-frisks (SAFs) by race, for the whole dataset:

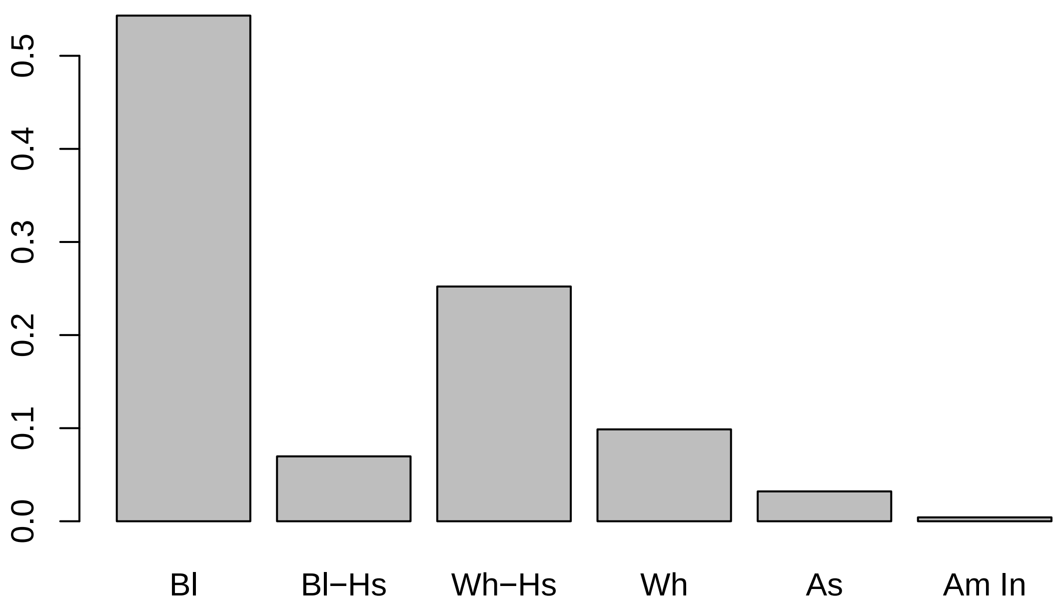
```
table.all <- table(Data.SAF$race2)
table.all
```

```
   Bl Bl-Hs Wh-Hs    Wh    As Am In 
10533  1352  4890  1914   622    81 
```

```
table.all.dens <- table.all / length(Data.SAF$race2)
table.all.dens
```

```
        Bl      Bl-Hs      Wh-Hs         Wh         As      Am In 
0.54316213 0.06971947 0.25216584 0.09870050 0.03207508 0.00417698 
```

```
barplot(table.all.dens)
```



Dividing by the length of the `race2` puts the table on a *density scale*, because it converts the table's frequencies into proportions. This code was a bit long. We can shorten it:

```
table.all <- table(Data.SAF$race2) / length(Data.SAF$race2)
table.all
```

```
        Bl      Bl-Hs      Wh-Hs         Wh         As      Am In 
0.54316213 0.06971947 0.25216584 0.09870050 0.03207508 0.00417698 
```

Either version is fine to use.

Now, let's say we wanted to calculate the distribution of race for people younger than 10 years old. A slightly longer version of the code is:

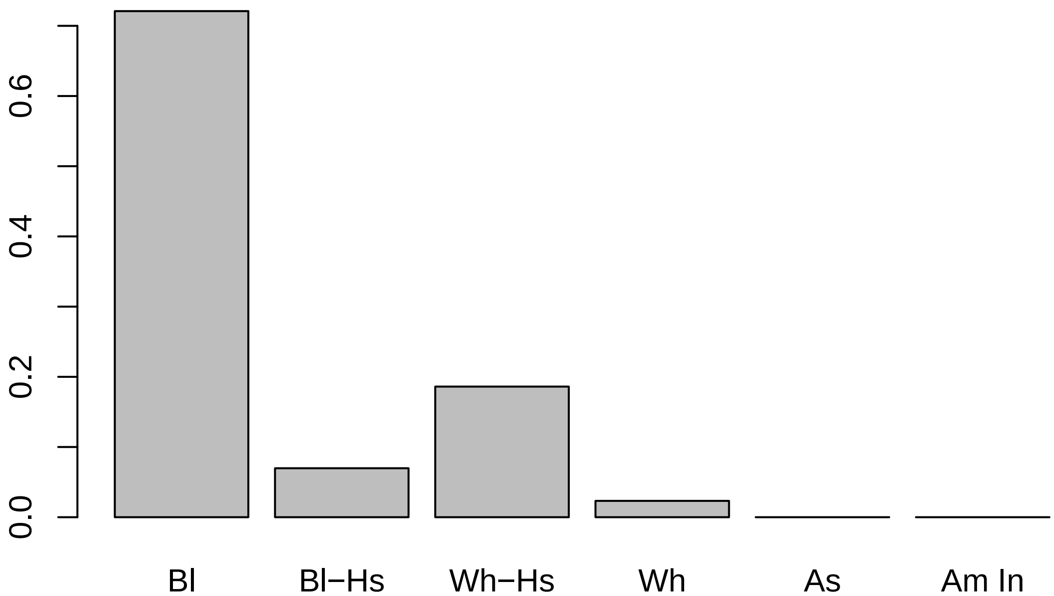
```r
race2.10 <- Data.SAF$race2[Data.SAF$age < 10]

table.10 <- table(race2.10)
table.10
```

```
race2.10
   Bl Bl-Hs Wh-Hs    Wh    As Am In
   31     3     8     1     0     0
```

```r
table.10.dens <- table.10 / length(race2.10)
table.10.dens
```

```
race2.10
        Bl      Bl-Hs      Wh-Hs         Wh         As      Am In
0.72093023 0.06976744 0.18604651 0.02325581 0.00000000 0.00000000
```

```r
barplot(table.10.dens)
```



We could do the same a bit more concisely as:

```r
table.10 <- table(Data.SAF$race2[Data.SAF$age < 10])
table.10
```

```

   Bl Bl-Hs Wh-Hs    Wh    As Am In
   31     3     8     1     0     0
```

```r
table.10.dens <- table.10 / length(Data.SAF$race2[Data.SAF$age < 10])
table.10.dens
```

```

        Bl      Bl-Hs      Wh-Hs         Wh         As      Am In
0.72093023 0.06976744 0.18604651 0.02325581 0.00000000 0.00000000
```

Again, either version is fine. We have found that as people get more familiar with R, their code grows more concise.

Below, we are going to look at density tables of SAFs by race, for a variety of different ages:

People older than 20 (>)

```
race2.g20 <- Data.SAF$race2[Data.SAF$age > 20]
table.g20 <- table(race2.g20)
table.g20
```

```
race2.g20
   Bl Bl-Hs Wh-Hs    Wh    As Am In
 7000   920  3363  1400   407    58
```

```
table.g20.dens <- table.g20 / length(race2.g20)
table.g20.dens
```

```
race2.g20
         Bl       Bl-Hs       Wh-Hs          Wh          As       Am In
0.532400365 0.069972619 0.255780347 0.106480073 0.030955278 0.004411317
```

For people exactly 1 year old (==)

```
race2.1 <- Data.SAF$race2[Data.SAF$age == 1]
table.1 <- table(race2.1)
table.1
```

```
race2.1
   Bl Bl-Hs Wh-Hs    Wh    As Am In
   23     2     4     1     0     0
```

```
table.1.dens <- table.1 / length(race2.1)
table.1.dens
```

```
race2.1
        Bl      Bl-Hs      Wh-Hs         Wh         As      Am In
0.76666667 0.06666667 0.13333333 0.03333333 0.00000000 0.00000000
```

For people aged 10-19 (>, <=, and &)

```
race2.teen <- Data.SAF$race2[(Data.SAF$age > 9) & (Data.SAF$age <= 19)]
table.teen <- table(race2.teen)
table.teen
```

```
race2.teen
   Bl Bl-Hs Wh-Hs    Wh    As Am In
 2952   356  1264   408   188    17
```

```
table.teen.dens <- table.teen / length(race2.teen)
table.teen.dens
```

```
race2.teen
         Bl       Bl-Hs       Wh-Hs          Wh          As       Am In
0.569334619 0.068659595 0.243780135 0.078688525 0.036258438 0.003278689
```

For people ages 20-29 but only in the morning (>, <=, and &)

```
race2.20s.morn <- Data.SAF$race2[(Data.SAF$age > 19) &
                                 (Data.SAF$age <= 29) &
                                 (Data.SAF$timestop < 1200)]
```

```
table.20s.morn <- table(race2.20s.morn)
table.20s.morn
```

```
race2.20s.morn
   Bl Bl-Hs Wh-Hs    Wh    As Am In 
 1188   161   679   280    99    10 
```

```
table.20s.morn.dens <- table.20s.morn / length(race2.20s.morn)
table.20s.morn.dens
```

```
race2.20s.morn
        Bl      Bl-Hs      Wh-Hs         Wh         As      Am In 
0.49151841 0.06661150 0.28092677 0.11584609 0.04095987 0.00413736 
```

For people in the early morning *or* late evening, between 11pm and 3am (>, <=, and |)

```
race2.night <- Data.SAF$race2[(Data.SAF$timestop <= 300) |
                              (Data.SAF$timestop >= 2300)]
table.night <- table(race2.night)
table.night
```

```
race2.night
   Bl Bl-Hs Wh-Hs    Wh    As Am In 
 2429   268  1099   429   120    17 
```

```
table.night.dens <- table.night / length(race2.night)
table.night.dens
```

```
race2.night
         Bl       Bl-Hs       Wh-Hs          Wh          As       Am In 
0.556854654 0.061439707 0.251948647 0.098349381 0.027510316 0.003897295 
```

Subsetting a Data Frame

Now that we know how to subset a vector, we may want to subset a data frame. This allows us to subset all of the columns in a data frame simultaneously. The syntax for subsetting a data frame is

```
data[rows, columns]
```

where the first argument in the brackets tells you what rows to consider and the second tells what columns. At its simplest, if we wanted the third row of `Data.SAF`, we would use

```
Data.SAF[3, ]
```

```
  year pct ser_num datestop timestop city sex race age race2
3 2012  44    3020 20120214     1635    3   1    1  22    Bl
```

while if we wanted the value of the 12,234rd row and fifth column, we would use

```
Data.SAF[12234, 5]
```

```
[1] 1520
```

Often, we will want to subset an entire data frame. As an example, let's say we wanted to consider the subset of `Data.SAF` for people with a reported age of 1. In this case, we would subset the data as

```
Data.SAF.1 <- Data.SAF[Data.SAF$age == 1, ]
head(Data.SAF.1)
```

```
     year pct ser_num datestop timestop city sex race age race2
381  2012  73     176 20120106     1910    2   0    3   1 Wh-Hs
1251 2012  52     886 20120116     2340    3   1    3   1 Wh-Hs
1788 2012  14     639 20120123     1611    1   0    4   1    Wh
2378 2012  14    1165 20120130     1450    1   1    1   1    Bl
4082 2012  18     691 20120219     1826    1   1    1   1    Bl
4574 2012  73    4072 20120225     1720    2   1    1   1    Bl
```

which is telling R to take all of the rows of `Data.SAF` for which `age == 1` (the first argument) and then return all columns (the second argument).

Here's the first three columns:

```
Data.SAF.1 <- Data.SAF[Data.SAF$age == 1, 1:3]
head(Data.SAF.1)
```

```
     year pct ser_num
381  2012  73     176
1251 2012  52     886
1788 2012  14     639
2378 2012  14    1165
4082 2012  18     691
4574 2012  73    4072
```

Density Plots

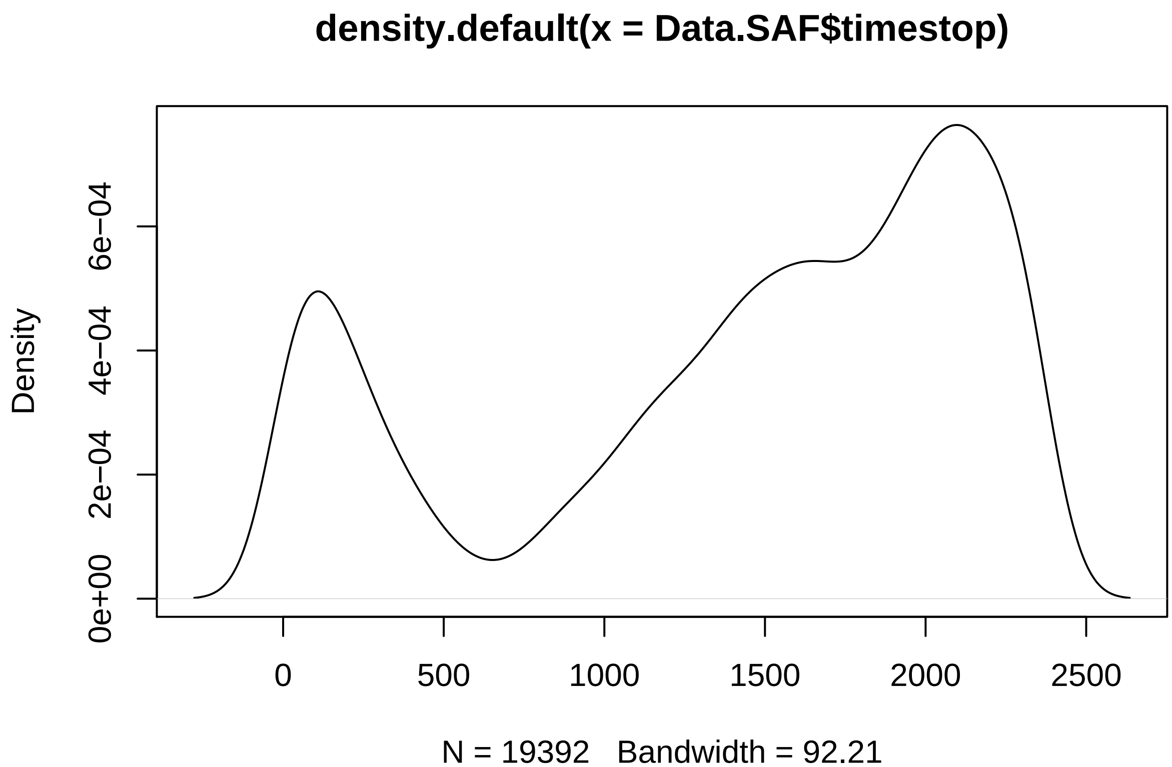
Familiarity with subsetting is a crucial part of data analysis. Yet, subsetting data is never a goal in and of itself. We subset data so that we can analyze different subsets, and see how the relationship between our treatment variable and outcome changes from subset to subset.

In this section, we are going to learn how to produce density plots with multiple densities. These figures will be similar to the NJ-PA minimum wage example from lecture.

Producing a Single Density Plot

The basic syntax for producing a density plot in R is

```
plot(density(Data.SAF$timestop))
```



Now, the way I teach plotting involves four steps:

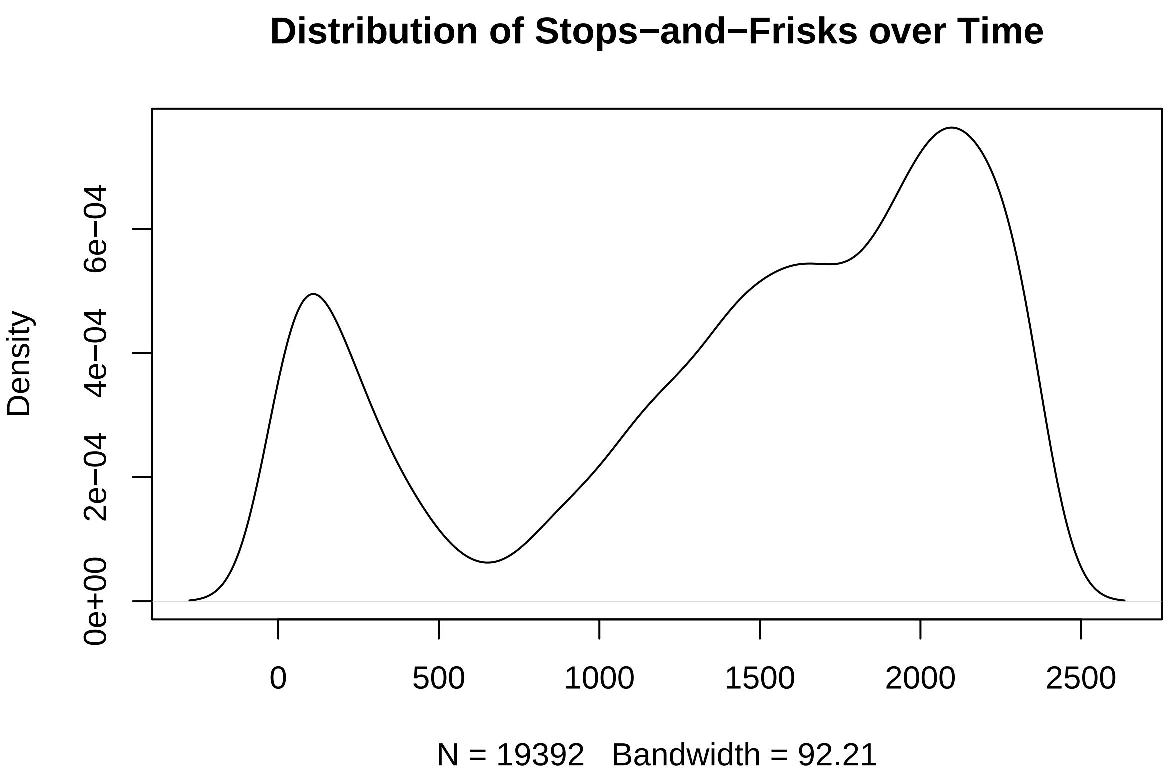
1. Produce the simplest possible figure
2. Look at the figure, and identify the first thing that grabs your eye as wrong
3. Fix it
4. Go to step 2

We are now going to clean up the density plot above.

Fixing the Title

The first thing that grabbed my eye was the main title. The R function `plot` takes a parameter `main` that allows you to set the title. Like this:

```
plot(density(Data.SAF$timestop),
     main = "Distribution of Stops-and-Frisks over Time")
```

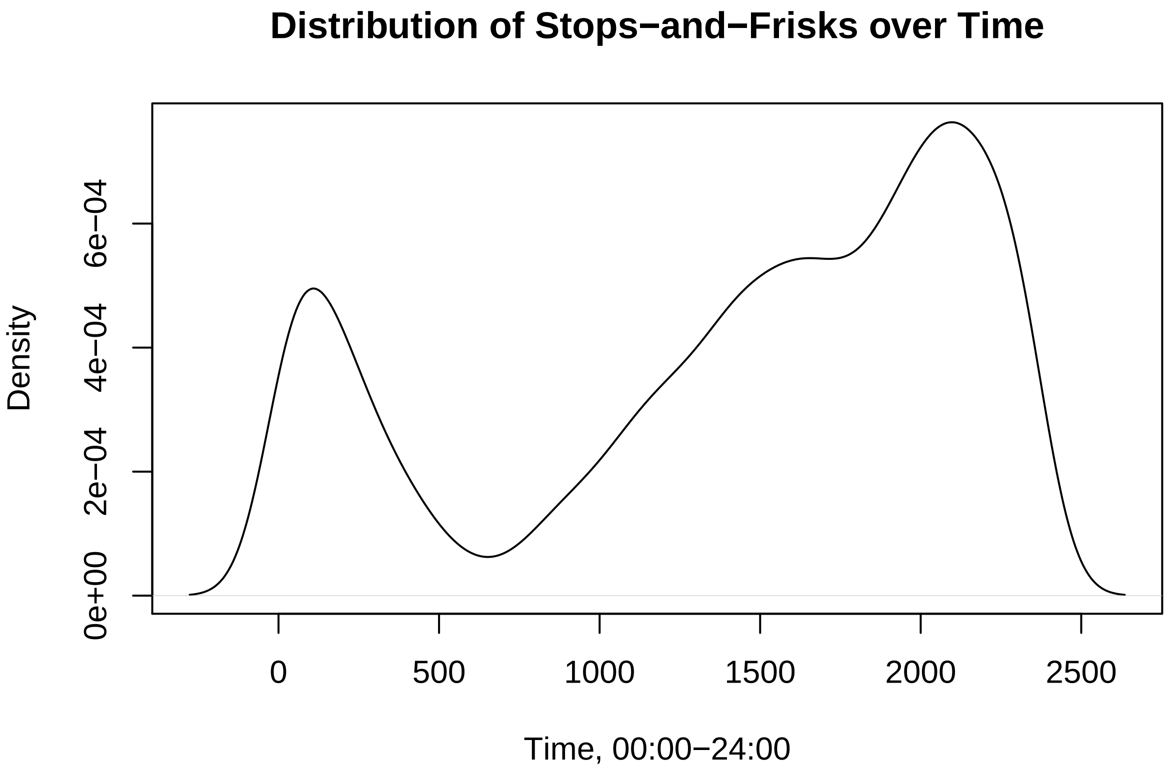


Note that `main` goes *outside* the parentheses for the `density` but *inside* the parentheses for `plot`, because it changes how `plot` operates, but not how `density` operates.

Fixing the X-axis label

`plot` also takes an option `xlab` which allow us to change the x-axis label. If we wanted to change the y-axis label, we could use `ylab`, but the default setting "Density" will work fine here. So, now our code looks like

```
plot(density(Data.SAF$timestop),
     main = "Distribution of Stops-and-Frisks over Time",
     xlab = "Time, 00:00-24:00")
```

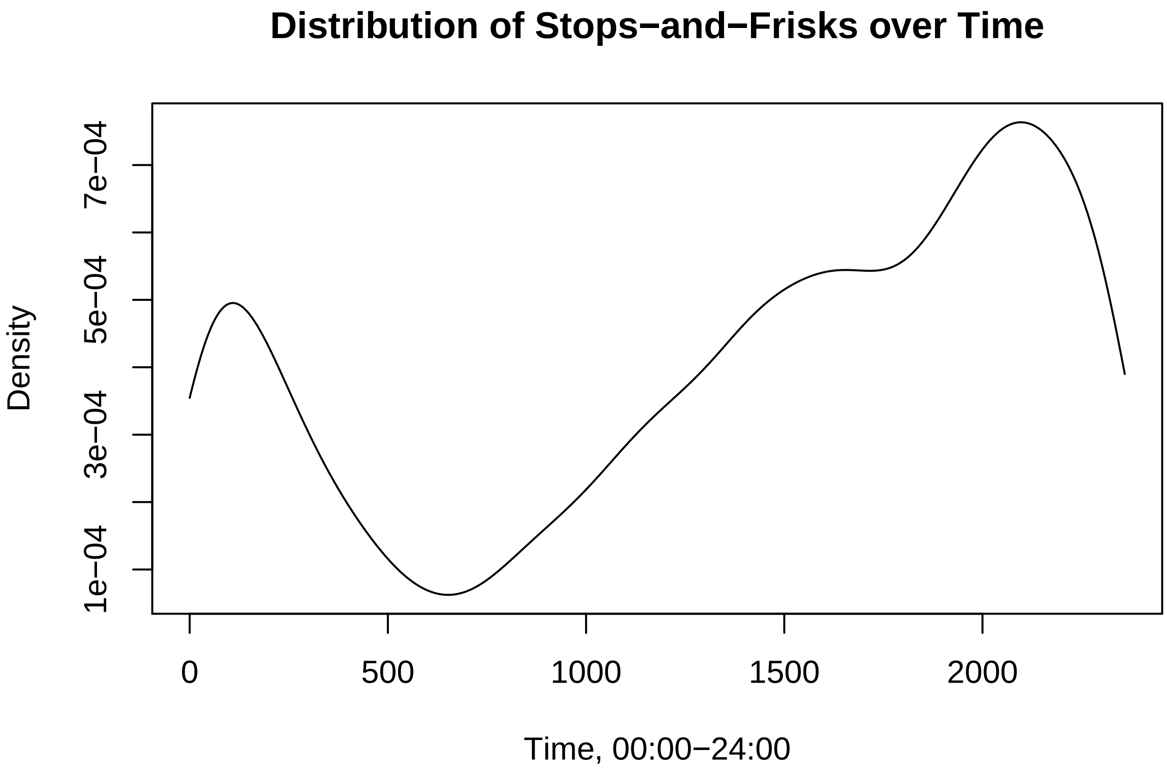


where I include the time range so the reader knows we are using military time.

Truncating the Density Plot at 0000 and 2400

At this point, if you look at the figure, you will notice that the density plot goes over 2400 and below 0. This does not make any sense. `density` takes an option `cut = 0` that cuts the density plot at the minimal and maximal values of `timestop`. Let's go ahead and do that.

```
plot(density(Data.SAF$timestop, cut = 0),
     main = "Distribution of Stops-and-Frisks over Time",
     xlab = "Time, 00:00-24:00")
```



Notice how the `cut = 0` option is *inside* the `density`.

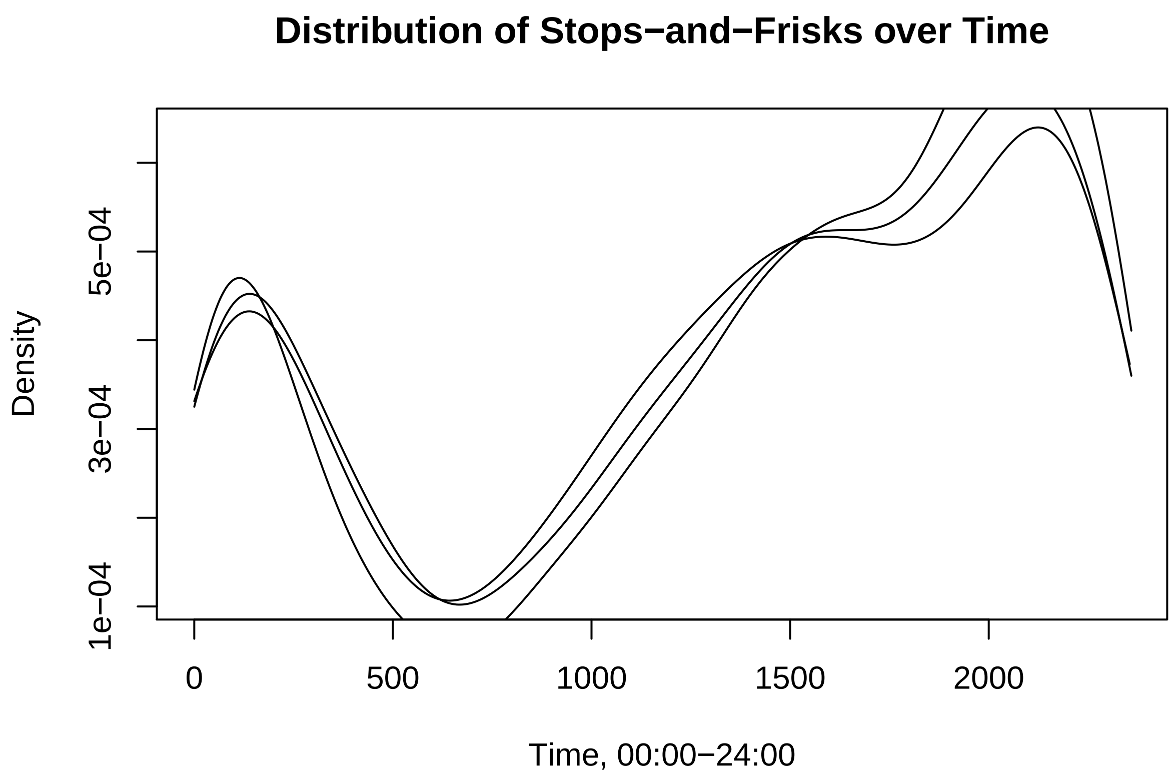
Next, we move on to placing several densities in a single figure.

Adding Density Plots to an Existing Figure

Placing several densities in a single figure can be a particularly useful means for communicating information. In this section, we are going to look at whether there is variation in what times of the day or evening people are stopped and frisked, by race.

Specifically, we are going to place three densities on the same figure: one for Blacks, one for Whites, and one for White-Hispanics. We are going to start with the following code:

```
plot(density(Data.SAF$timestop[Data.SAF$race2 == "Wh"], cut = 0),
             main = "Distribution of Stops-and-Frisks over Time",
             xlab = "Time, 00:00-24:00")
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Bl"], cut = 0))
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Wh-Hs"], cut = 0))
```

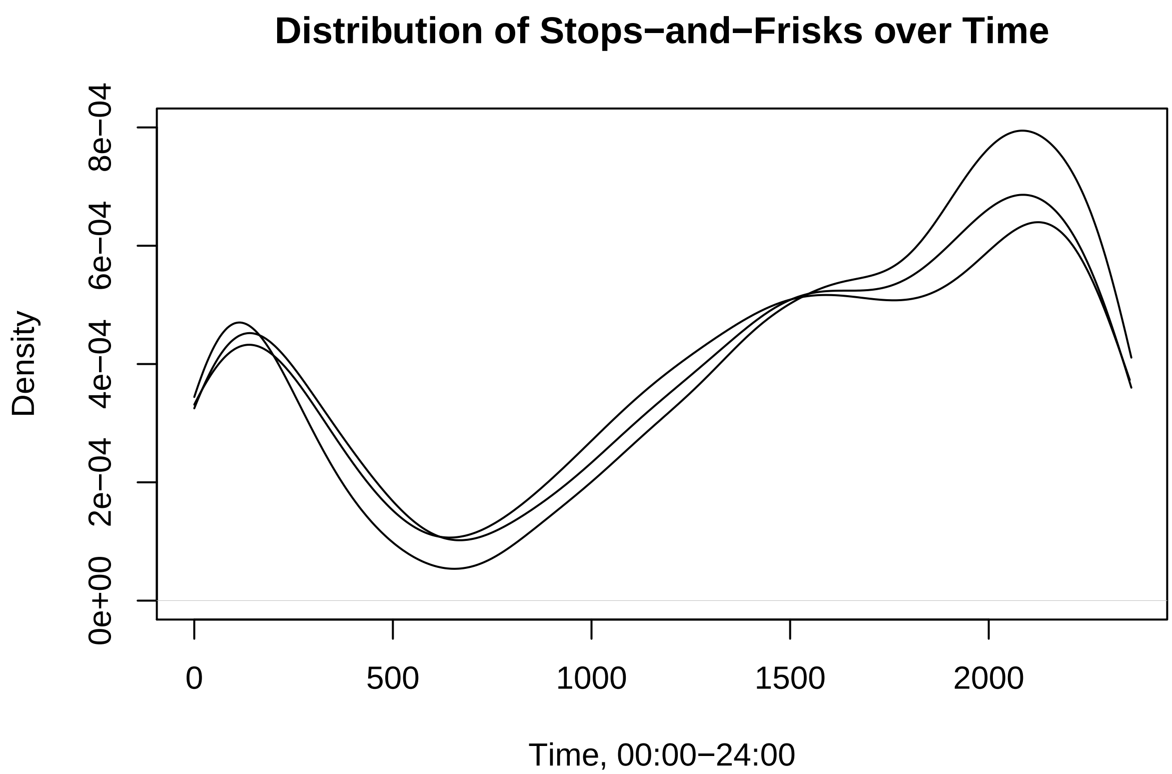


`plot` opens and produces a single figure and `lines` places the density on an already existing figure. So, you should notice three densities in this figure. We will now begin fixing this figure.

Adjusting the y-axis limits

The first thing I noticed in this figure is how the densities are truncated. This happens because the dimensions for the plot are set by the range on the first call to `plot`, that for White people. So, we need to adjust the y-axis. We do this using `ylim = c(upper, lower)` where `upper` and `lower` give the desired upper and lower limits on the y-axis. `ylim` is only in the call to `plot`.

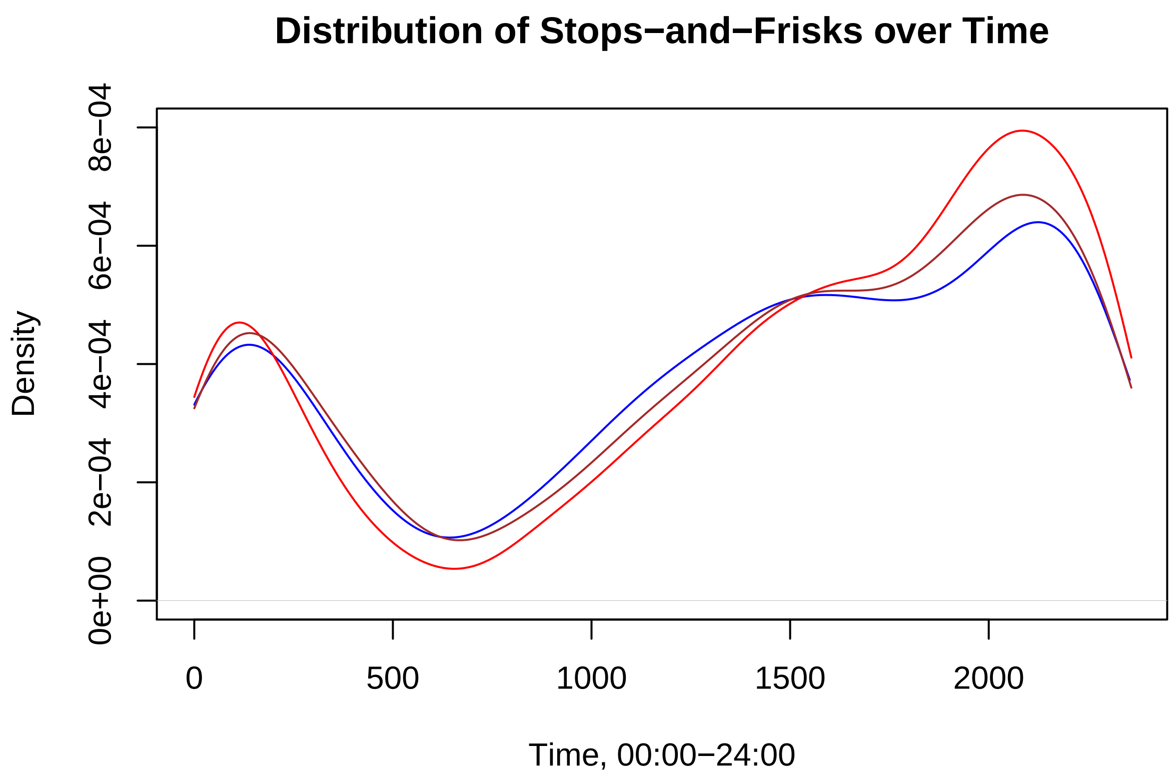
```
plot(density(Data.SAF$timestop[Data.SAF$race2 == "Wh"], cut = 0),
     main = "Distribution of Stops-and-Frisks over Time",
     xlab = "Time, 00:00-24:00",
     ylim = c(0, .0008))
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Bl"], cut = 0))
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Wh-Hs"], cut = 0))
```



Adding Color to Differentiate Lines

Next, we need to be able to tell the lines apart. To do this, let's give each line its own color. In R, the parameter `col` can be used either in `plot` or `lines`. We can set color as follows:

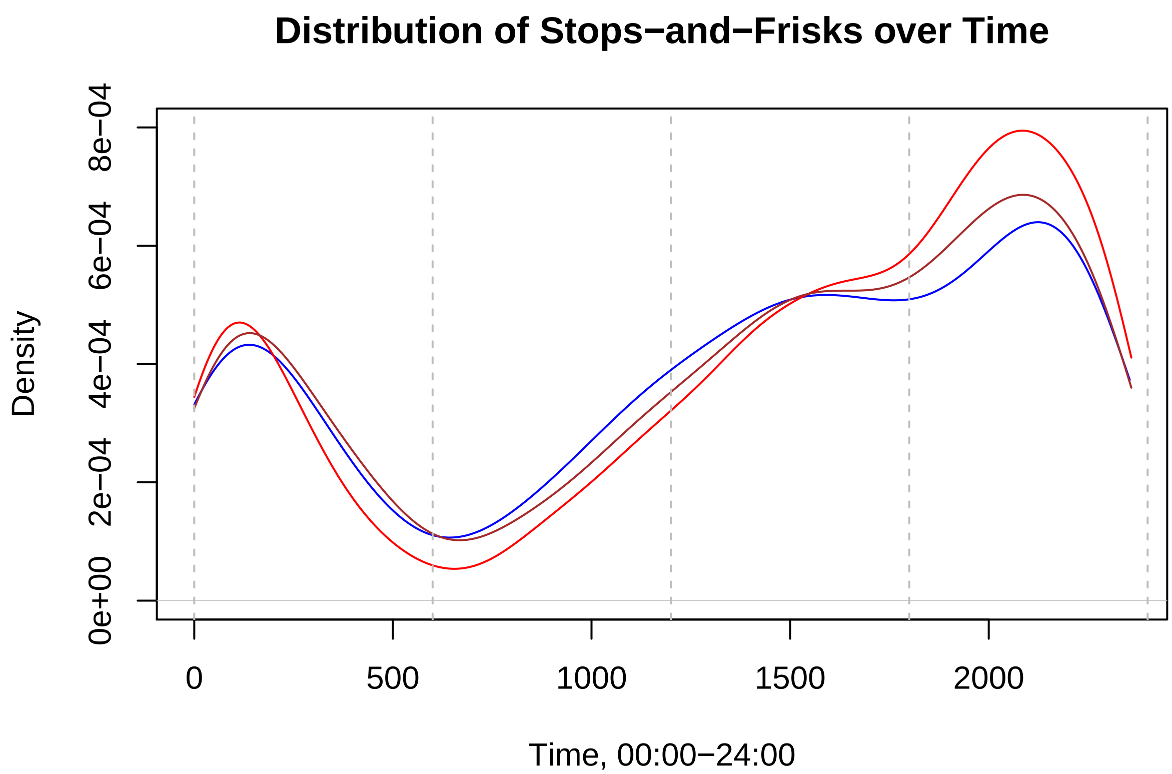
```
plot(density(Data.SAF$timestop[Data.SAF$race2 == "Wh"], cut = 0),
             main = "Distribution of Stops-and-Frisks over Time",
             xlab = "Time, 00:00-24:00",
             ylim = c(0, .0008), col = "blue")
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Bl"], cut = 0),
      col = "red" )
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Wh-Hs"], cut = 0),
      col = "brown" )
```



Adding Vertical Lines

At this point, we are going to use the `abline` to add vertical lines at midnight, 6am, noon, and 6pm, just to help guide the reader's eye.

```
plot(density(Data.SAF$timestop[Data.SAF$race2 == "Wh"], cut = 0),
     main = "Distribution of Stops-and-Frisks over Time",
     xlab = "Time, 00:00-24:00",
     ylim = c(0, .0008),
     col = "blue" )
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Bl"], cut = 0),
      col = "red")
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Wh-Hs"], cut = 0),
      col = "brown" )
abline(v = c(0,600, 1200, 1800, 2400),
       lty = 2, col = "gray" )
```



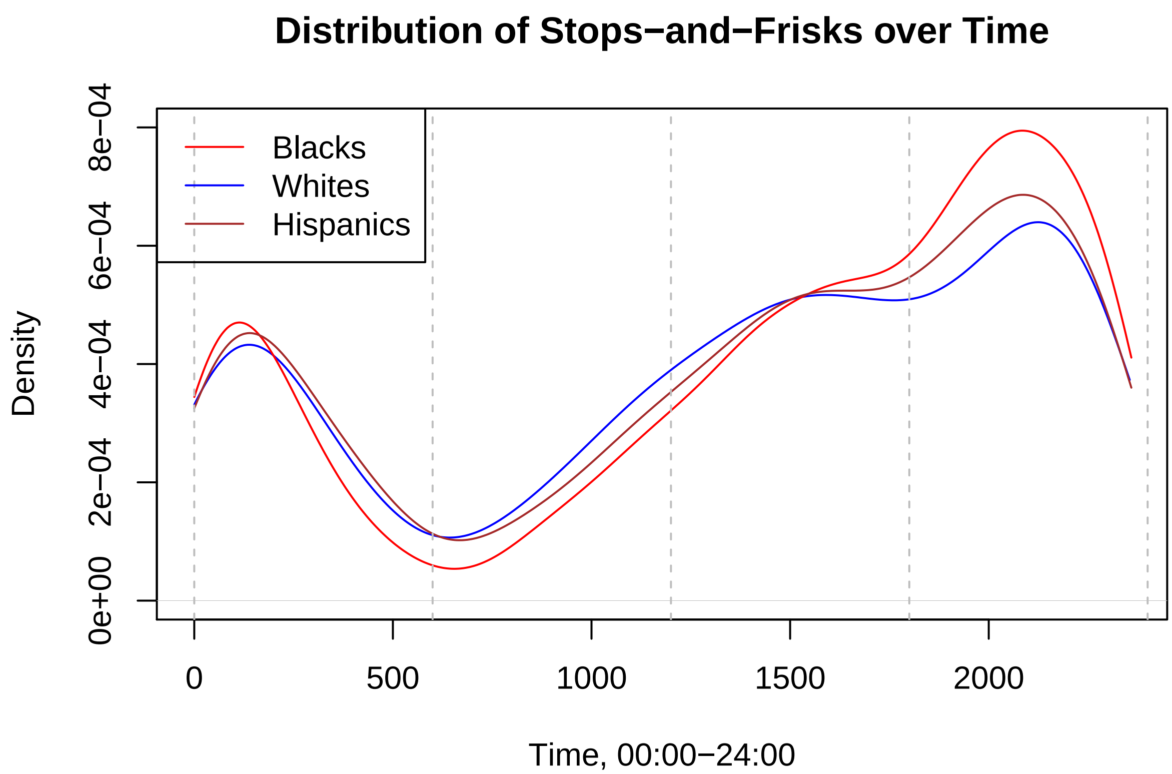
The `abline` function takes several arguments:

- `v` (or `h`). Whether and where to make a vertical (or horizontal) line
- `lty`. Line type: 1 is a solid line (and the default), 2 gives a dashed line. 3-5 give different types of dashed line. *You can also specify the line type in either* `plot` *or* `lines`
- `col`: Color, as described above

Adding a Legend

At this point, we can tell the difference between the densities, but the reader will have no idea as to which color denotes which race. Therefore, we need to add a legend:

```
plot(density( Data.SAF$timestop[Data.SAF$race2 == "Wh"], cut = 0),
     main = "Distribution of Stops-and-Frisks over Time",
     xlab = "Time, 00:00-24:00",
     ylim = c(0, .0008),
     col = "blue" )
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Bl"], cut = 0),
      col = "red")
lines(density(Data.SAF$timestop[Data.SAF$race2 == "Wh-Hs"], cut = 0),
      col = "brown" )
abline(v = c(0,600, 1200, 1800, 2400),
       lty = 2, col = "gray" )
legend("topleft", legend  = c("Blacks", "Whites", "Hispanics"),
       col = c("red", "blue", "brown"),
       lty = c(1,1,1))
```



Note: Saving a figure in RStudio can be accomplished through clicking on "Export" on the top bar of the figure, which should be visible in the lower right hand corner. Select "Export > Save Plot as PDF" and then you can adjust the size and name the file.

Precept Questions

For these questions, we will analyze the relationship between oil, Islam, and female participation in the workforce. Female participation in the workforce, and in the government, is lower in high-oil countries relative to the rest of the world. Researchers have posited that Islam is at the basis of this discrepancy. Note the underlying causal claim—higher levels of a Muslim population *causes* a decrease in workforce participation.

Ross (2008) argued that oil, rather than Islam, is the underlying cause of this low female participation. You will not need to do so for this assignment, but I recommend reading the paper. An overview of the paper and debate over its findings appeared on the political science blog, the Monkey Cage. Please read the blog post for a discussion of the underlying causal mechanism that Ross posits.

We will be using Ross's data in this problem. The dataset `RossOilWomenIslamData.csv` is in the `data` folder. Reminder: if your working directory is the directory this document lives in then the *relative path* to the data is `data/RossOilWomenIslamData.csv`.

This data set contains the following variables:

Name	Description
`country`	The name of each country
`loggdp`	Logged Gross Domestic Product (GDP) per capita, averaged over 1993-2002
`islam`	Proportion of the country that is Muslim
`me`	1 if the country is in the Middle East and 0 otherwise
`femlabor`	A measure of female participation in the workforce

Name	Description
`oil`	Logged oil rents per capita

Note that the variables `loggdp`, `islam`, `femlabor`, and `oil` have been converted to a z-scale. When a variable is converted to a z-scale, the variables have their mean subtracted and have been divided by their standard deviation. After a z-transformation, observations with a positive value are above the mean for that variable, and observations with a negative value are below the mean of that variable. We will discuss this more in lecture, but researchers will occasionally standardize the data this way to put variables 'on the same scale'.

We will start by looking at mean differences in `femlabor` in different subsets of the data.

Question 1

First calculate the difference-in-means for `femlabor` between countries with high and low levels of Islam (i.e., between countries with positive and negative or zero values of `islam`).

Question 2

Now calculate the difference-in-means for `femlabor` between countries with high and low levels of oil (i.e., between countries with positive and negative or zero values of `oil`).

Question 3

Just from looking at the means, does it appear that higher female labor force participation is associated with higher or lower levels of Islam? Oil? Is this result causal? Why or why not?

Question 4

Now calculate the difference-in-means for `femlabor` between countries with high and low levels of oil *only for countries with a **high** level of Islam*.

Question 5

Now for countries with high and low levels of oil but *only for countries with a **low** level of Islam*.

Question 6

How does the relationship between oil and female labor participation vary between countries with a high and low level of Islam?

Question 7

Next, we are going to create two figures and analyze the results.

First, we are going to create the figures. The first figure should consider only countries with a **low** level of oil rents. This figure should:

1. Contain the density for female labor participation for countries with a high level of Islam (in red) and a low level of Islam (in blue).
2. The figure should have a red or blue dashed vertical line at the mean level of female labor participation, corresponding to each density
3. Each plot should have informative axis labels and titles.
4. Each plot should have a legend. If necessary, you should lengthen the y-axis so that the legend does not overlap with the density plot.

The second figure should be the same as the first figure, except it should consider countries with a **high** level of oil rents.

Question 8

In which subset of the data do we see a stronger relationship between Islam and female labor participation? Explain why we are treating oil as a confounder, and how we are attempting to control for it.

References

Gelman, Andrew, Jeffrey Fagan, and Alex Kiss. 2007. "An Analysis of the New York City Police Department's "Stop-and-Frisk" Policy in the Context of Claims of Racial Bias." *Journal of the American Statistical Association* 102 (479): 813–23. https://doi.org/10.1198/016214506000001040.

Ross, Michael L. 2008. "Oil, Islam, and Women." *American Political Science Review* 102 (01): 107–23. https://doi.org/10.1017/S0003055408080040.