

# R Introduction

Econ 5/Poli 5D Lecture 12

## Announcements

- Fourth quiz due this Friday
- Any questions?

Today's Question: How do we detect discrimination in the labor market?

- We will explore one way: resume audit studies
- First of two-part class on the topic

Today's Software/Data Skills:

- Introduce R

## Today's Application – Discrimination in the labor market

Pervasive evidence of inequality in the labor market

- Black workers are about twice as likely as white worker to be unemployed (Council of Economic Advisors, 1998)

Two stories:

- Employers are prejudiced against black workers and less likely to hire them
- Faced with two identical workers, one white and one black, employers are not prejudiced, but researchers do not have access to all variables employers are interested in when making hiring decisions

## Bertrand and Mullainathan (2004)

To test the two theories, the authors send fictitious resumes to help-wanted ads in Boston and Chicago

They manipulate the perception of race by altering the name of the fictitious applicants

- Examples of white-sounding names from the paper: Emily Walsh and Greg Baker
- Examples of black-sounding names from the paper: Lakisha Washington and Jamal Jones

They randomize all other characteristics of the applicant: experience, education, job history

They then compare callback rates (for interviews) for white and Black applicants

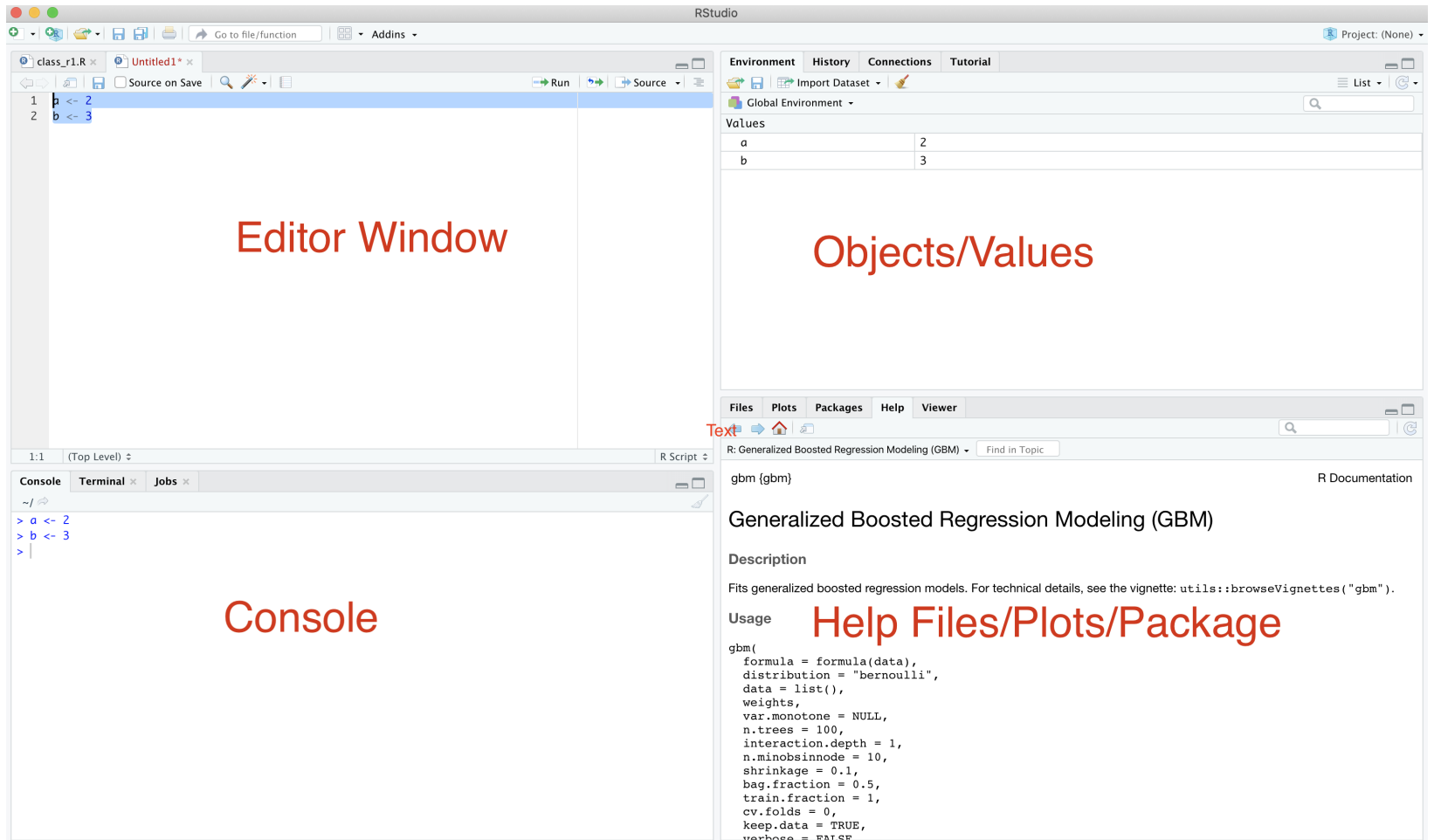
## Resume Audit Studies

Bertrand and Mullainathan (2004) launched a large literature exploring discrimination in the labor market using resume audit studies

Key strength of the design: all other factors relevant for a job are held fixed

Therefore, cannot explain away any disparities as stemming from omitted variables

Before looking at the data, though, we need to learn how to use R!



**Editor** Where you type your scripts (R scripts are the R version of .do files)

**Console Window** Where results of scripts appear

**Objects/Values** Objects and dataframes appear here

**Help/Packages/Plots/Files** Useful window depends on what you are currently trying to accomplish

## Arithmetic Operations

R can be used as a calculator to perform arithmetic operations

## Arithmetic Operations

R can be used as a calculator to perform arithmetic operations

In [1]:

```
5+3
```

```
5/3
```

```
5^3
```

8

1.6666666666666667

125

## R is an "object-oriented" programming language

Objects can be any piece of information stored by R:

- A dataset (referred to as data frame in R)
- A subset of a dataset (unlike Stata, easy to hold multiple datasets in memory)
- A number (e.g.  $2\pi + 1$ )
- A text string (e.g. "UCSD is awesome")
- A function (e.g. a function that takes in  $x$  and gives you  $x^2 + 8$ )

## Creating Objects

Use `<-` as an assignment operator for objects

## Creating Objects

Use `<-` as an assignment operator for objects

In [2]:

```
object_1 <- 5 + 3  
object_1
```

8

## Creating Objects

Use `<-` as an assignment operator for objects

In [2]:

```
object_1 <- 5 + 3  
object_1
```

8

If we assign a new value to the same object, we overwrite the object

## Creating Objects

Use `<-` as an assignment operator for objects

```
In [2]: object_1 <- 5 + 3  
object_1
```

8

If we assign a new value to the same object, we overwrite the object

```
In [3]: object_1 <- 5-3  
object_1
```

2

## More Objects

R can also represent other values, such as string as objects

## More Objects

R can also represent other values, such as string as objects

In [4]:

```
MySchool <- "UCSD"  
MySchool
```

'UCSD'

## Vectors

A vector represents a collection of information stored in a specific order

We use the function `c()` (concatenate) to enter a data vector

## Vectors

A vector represents a collection of information stored in a specific order

We use the function `c()` (concatenate) to enter a data vector

In [5]:

```
vector.1 <- c(93, 92, 83, 99, 96, 97)
vector.1
```

```
93 · 92 · 83 · 99 · 96 · 97
```

## Vectors (continued)

To access a specific **element** of a vector, we use square brackets `[]`. This is called **indexing**

## Vectors (continued)

To access a specific **element** of a vector, we use square brackets `[]`. This is called **indexing**

In [6]:

```
vector.1[2]
```

92

## Vectors (continued)

To access a specific **element** of a vector, we use square brackets `[]`. This is called **indexing**

```
In [6]: vector.1[2]
```

92

```
In [7]: vector.1[c(2,4)]
```

92 · 99

## Vectors (continued)

To access a specific **element** of a vector, we use square brackets `[]`. This is called **indexing**

```
In [6]: vector.1[2]
```

92

```
In [7]: vector.1[c(2,4)]
```

92 · 99

```
In [8]: vector.1[-4]
```

93 · 92 · 83 · 96 · 97

## Functions

A function takes input object(s) and returns an output object. In R, a functions generally runs as `funcname(input)`. Some basic functions useful for summarizing data are:

- `length()` : length of a vector (number of elements)
- `min()` : minimum value
- `max()` : maximum value
- `range()` : range of data
- `mean()` : mean value
- `sum()` : sum

## Practicing Functions

In [9]: `length(vector.1)`

6

```
In [9]: length(vector.1)
```

6

```
In [10]: min(vector.1)
```

83

```
In [9]: length(vector.1)
```

6

```
In [10]: min(vector.1)
```

83

```
In [11]: max(vector.1)
```

99

## Practicing Functions

## Practicing Functions

In [12]:

```
range(vector.1)
```

83 · 99

## Practicing Functions

```
In [12]: range(vector.1)
```

```
83.99
```

```
In [13]: mean(vector.1)
```

```
93.33333333333333
```

## Practicing Functions

```
In [12]: range(vector.1)
```

83.99

```
In [13]: mean(vector.1)
```

93.33333333333333

```
In [14]: sum(vector.1)
```

560

## On the the Data

Now let's open the data

Reminder: this data comes from the actual experiment run by Bertrand and Mullainaita (2004)

The outcome we are interested is whether a given application receives a callback for an interview. The applications (on average) are the same for white and black applicants

## Set the working directory

Step 1: Need to remember to set the working directory in R. To see your working directory type

## Set the working directory

Step 1: Need to remember to set the working directory in R. To see your working directory type

In [15]:

```
getwd()
```

```
'/Users/Brian/Dropbox/Grad School/Sixth Year/Econ:Poli 5/Lectures/Week 7'
```

I'm going to change mine to the "data" subfolder that holds the data for this lecture

In [16]:

```
setwd('/Users/Brian/Dropbox/Grad School/Sixth Year/Econ:Poli 5/Lectures/Week 7/data')
```

## Loading data

R has different functions to load the data depending on the "type" of data (remember all of our Stata datasets had a ".dta")

The most common ones you will use are either `read.csv` which reads in comma separated files or `load` which loads in R data files

## Loading data

R has different functions to load the data depending on the "type" of data (remember all of our Stata datasets had a ".dta")

The most common ones you will use are either `read.csv` which reads in comma separated files or `load` which loads in R data files

```
In [17]: resume <- read.csv("resume.csv")
```

## Loading data

R has different functions to load the data depending on the "type" of data (remember all of our Stata datasets had a ".dta")

The most common ones you will use are either `read.csv` which reads in comma separated files or `load` which loads in R data files

```
In [17]: resume <- read.csv("resume.csv")
```

Note: In R you can load multiple datasets at once! Datasets in R are referred to as data frames.

## Exploring the data frame

There are a number of useful function that will allow you to explore the dataset more

One particular useful function is `head()`. This will show you the first 5 observations of the dataset

## Exploring the data frame

There are a number of useful function that will allow you to explore the dataset more

One particular useful function is `head()`. This will show you the first 5 observations of the dataset

In [18]:

```
head(resume)
```

A data.frame: 6 × 5

	<b>X</b>	<b>firstname</b>	<b>sex</b>	<b>race</b>	<b>call</b>
	<b>&lt;int&gt;</b>	<b>&lt;fct&gt;</b>	<b>&lt;fct&gt;</b>	<b>&lt;fct&gt;</b>	<b>&lt;int&gt;</b>
<b>1</b>	1	Allison	female	white	0
<b>2</b>	2	Kristen	female	white	0
<b>3</b>	3	Lakisha	female	black	0
<b>4</b>	4	Latonya	female	black	0
<b>5</b>	5	Carrie	female	white	0
<b>6</b>	6	Jay	male	white	0

`summary()` provides statistics about each variable in your dataset

`summary()` provides statistics about each variable in your dataset

In [19]:

```
summary(resume)
```

	X	firstname	sex	race	call
Min.	: 1	Tamika : 256	female:3746	black:2435	Min. :0.00000
1st Qu.	:1218	Anne : 242	male :1124	white:2435	1st Qu.:0.00000
Median	:2436	Allison: 232			Median :0.00000
Mean	:2436	Latonya: 230			Mean :0.08049
3rd Qu.	:3653	Emily : 227			3rd Qu.:0.00000
Max.	:4870	Latoya : 226			Max. :1.00000
		(Other):3457			

---

To reference a particular variable within a data frame, we can use the syntax `data_frame$variable_name`

In [20]:

```
head(resume$firstname)
```

Allison · Kristen · Lakisha · Latonya · Carrie · Jay

► **Levels:**

## Taking the mean of a variable

Our outcome of interest is `call` which indicates whether a person received a callback for an interview

Let's see what the average callback rate is in this experiment

## Taking the mean of a variable

Our outcome of interest is `call` which indicates whether a person received a callback for an interview

Let's see what the average callback rate is in this experiment

```
In [21]: mean(resume$call)
```

```
0.0804928131416838
```

So only about 8 percent of applicants receive a callback for an interview. Callback rates are commonly quite low in these types of studies

## Racial Disparities

Our main question of interest is whether there is a racial disparity in callback rates. We need to retrieve the average callback rate for white and Black defendants separately

One way is to create a new data frame that is a subset of our data

Let's create a dataframe composed of Black applicants

To create our dataset, we need to understand how brackets `[]` work in R

Syntax:

```
resume[i,j]
```

Returns the row and column of the resume data frame

To create our dataset, we need to understand how brackets `[]` work in R

Syntax:

```
resume[i,j]
```

Returns the  $i^{th}$  row and  $j^{th}$  column of the resume data frame

```
In [22]: # ID of the first observation
         resume[1,1]

         # name of the first observation
         resume[1,2]
```

1

Allison

► **Levels:**

You aren't constrained by just putting numbers into brackets. You can also put lists of numbers. For example `a:b` works just as it did in excel.

Imagine I want the first five names in the data frame

You aren't constrained by just putting numbers into brackets. You can also put lists of numbers. For example `a:b` works just as it did in excel.

Imagine I want the first five names in the data frame

In [23]:

```
resume[1:5,2]
```

Allison · Kristen · Lakisha · Latonya · Carrie

► **Levels:**

You aren't constrained by just putting numbers into brackets. You can also put lists of numbers. For example `a:b` works just as it did in excel.

Imagine I want the first five names in the data frame

In [23]:

```
resume[1:5,2]
```

Allison · Kristen · Lakisha · Latonya · Carrie

► **Levels:**

If you want all the variables to be included, you can leave the second entry blank

You aren't constrained by just putting numbers into brackets. You can also put lists of numbers. For example `a:b` works just as it did in excel.

Imagine I want the first five names in the data frame

```
In [23]: resume[1:5,2]
```

Allison · Kristen · Lakisha · Latonya · Carrie

### ► Levels:

If you want all the variables to be included, you can leave the second entry blank

```
In [24]: resume[1:5,]
```

A data.frame: 5 × 5

	X	firstname	sex	race	call
	<int>	<fct>	<fct>	<fct>	<int>
1	1	Allison	female	white	0
2	2	Kristen	female	white	0
3	3	Lakisha	female	black	0
4	4	Latonya	female	black	0
5	5	Carrie	female	white	0

Next class we will directly compare callback rates for black applicants and white applicants

To do so we will learn how to right logic statements and subset data in R

Today's Application: Resume Audit studies provide clever way to study discrimination

- Application to be continued next class

Today's Software/Data Skills:

- How to set directory/load data
- Basic functions in R