

# Intro to R - the Stats

J. Alexander Branham

# Basic statistics in R

# Reading in some data

- ▶ Let's read in some data from the internet:

```
# Data is available on this course's github page:  
# github.com/jabranham/math-camp  
# Data from Herrera et al (forthcoming, AJPS)  
library(foreign)  
my_data <- read.dta("data/herrera-data.dta")  
my_data$fptp <- as.logical(my_data$fptp)
```

# What's in this data?

- ▶ unit of observation: country-year

# What's in this data?

- ▶ unit of observation: country-year
- ▶ DV: turnout

# What's in this data?

- ▶ unit of observation: country-year
- ▶ DV: turnout
- ▶ margin: margin between candidate w/ plurality and runner-up

# What's in this data?

- ▶ unit of observation: country-year
- ▶ DV: turnout
- ▶ margin: margin between candidate w/ plurality and runner-up
- ▶ fptp: dummy for FPTP systems

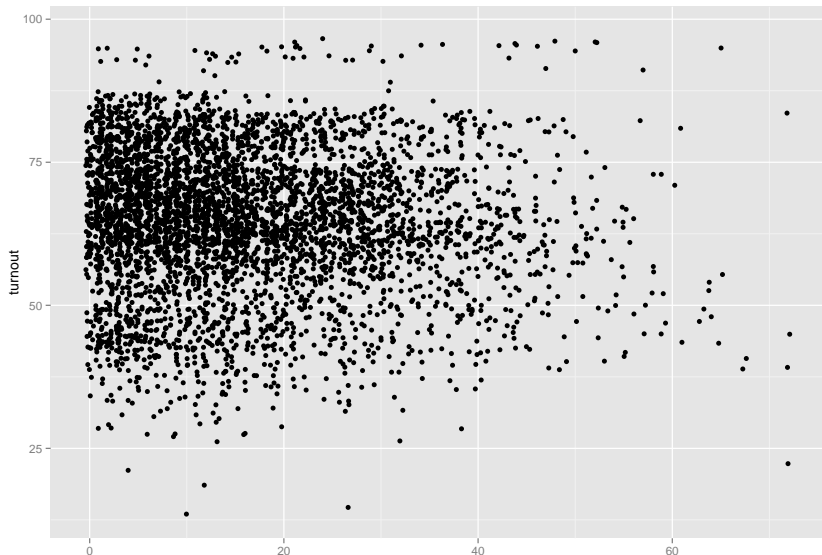
# What's in this data?

- ▶ unit of observation: country-year
- ▶ DV: turnout
- ▶ margin: margin between candidate w/ plurality and runner-up
- ▶ fptp: dummy for FPTP systems
- ▶ ppi: parliamentary power index



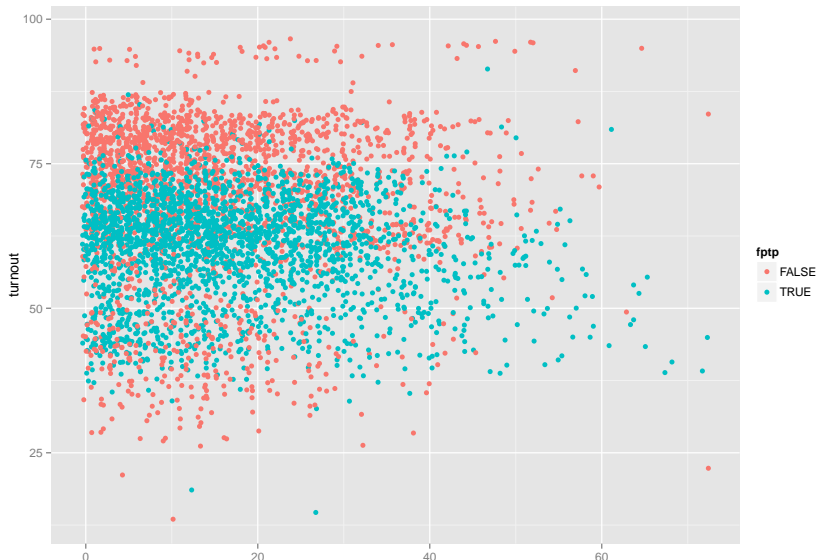
# First step: Plot your data!

```
library(ggplot2)
ggplot(my_data, aes(margin, turnout)) + geom_jitter()
```



# Maybe it looks different in FPTP systems?

```
ggplot(my_data, aes(margin, turnout, color=fptp)) +  
  geom_jitter()
```



# Means by group

- ▶ To do this, we need to *subset* by ftp or not

# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,

# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,
- ▶ and another mean for non-ftp systems

# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,
- ▶ and another mean for non-ftp systems
- ▶ Subsetting in R can be done several ways

# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,
- ▶ and another mean for non-ftp systems
- ▶ Subsetting in R can be done several ways
- ▶ One way uses square brackets `[]` to subset row by column

# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,
- ▶ and another mean for non-ftp systems
- ▶ Subsetting in R can be done several ways
- ▶ One way uses square brackets `[]` to subset row by column
- ▶ Another way uses `$` to subset by variable name



# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,
- ▶ and another mean for non-ftp systems
- ▶ Subsetting in R can be done several ways
- ▶ One way uses square brackets `[]` to subset row by column
- ▶ Another way uses `$` to subset by variable name
- ▶ We can combine these two types of subsetting too

# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,
- ▶ and another mean for non-ftp systems
- ▶ Subsetting in R can be done several ways
- ▶ One way uses square brackets `[]` to subset row by column
- ▶ Another way uses `$` to subset by variable name
- ▶ We can combine these two types of subsetting too
- ▶ This is how base R thinks about it: there are other (better?) ways using the `dplyr` or `data.table` packages

# Means by group

- ▶ To do this, we need to *subset* by ftp or not
- ▶ So we need one mean for ftp systems,
- ▶ and another mean for non-ftp systems
- ▶ Subsetting in R can be done several ways
- ▶ One way uses square brackets `[]` to subset row by column
- ▶ Another way uses `$` to subset by variable name
- ▶ We can combine these two types of subsetting too
- ▶ This is how base R thinks about it: there are other (better?) ways using the `dplyr` or `data.table` packages
  - ▶ Personally, I prefer `dplyr`

## The code

```
mean_fptp <- mean(my_data$turnout[my_data$fptp==TRUE])  
mean_notfptp <- mean(my_data$turnout[my_data$fptp==FALSE])  
c(mean_fptp, mean_notfptp)
```

```
## [1] 60.25896 69.64283
```

Maybe we want uncertainty too...

```
sd_fptp <- sd(my_data$turnout[my_data$fptp==TRUE])  
sd_notfptp <- sd(my_data$turnout[my_data$fptp==FALSE])  
c(sd_fptp, sd_notfptp)
```

```
## [1] 9.293468 13.688340
```

# There's a formal test

- Tests whether the mean is statistically different from each other

```
t.test(my_data$turnout[my_data$fptp==TRUE],  
       my_data$turnout[my_data$fptp==FALSE])
```

```
##
```

```
##  Welch Two Sample t-test
```

```
##
```

```
## data:  my_data$turnout[my_data$fptp == TRUE] and my_data$turnout[my_data$fptp == FALSE]
```

```
## t = -27.097, df = 3933.7, p-value < 2.2e-16
```

```
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
```

```
##  -10.062820  -8.704922
```

```
## sample estimates:
```

```
## mean of x mean of y
```

```
##  60.25896  69.64283
```

# There's a formal test

- ▶ Tests whether the mean is statistically different from each other
- ▶ LOTS more of this in Stats I

```
t.test(my_data$turnout[my_data$fptp==TRUE],  
       my_data$turnout[my_data$fptp==FALSE])
```

```
##
```

```
##  Welch Two Sample t-test
```

```
##
```

```
## data:  my_data$turnout[my_data$fptp == TRUE] and my_data$turnout[my_data$fptp == FALSE]
```

```
## t = -27.097, df = 3933.7, p-value < 2.2e-16
```

```
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
```

```
##  -10.062820  -8.704922
```

```
## sample estimates:
```

```
## mean of x mean of y
```

```
##  60.25896  69.64283
```