Exploratory Data Analysis
Data Wrangling
Winter Institute in Data Science

Ryan T. Moore

2 January 2020

EDA

"EDA is a state of mind."

– Wickham, *R4DS*, p. 81

"EDA is a state of mind."

– Wickham, *R4DS*, p. 81

- Visualize
- Transform
- Model

…with an eye toward *discovery*

EDA is **not**

- formal hypothesis testing

EDA is **not**

- formal hypothesis testing
  - (if you find $x \to y$ via EDA, "training" data)

EDA is **not**

- formal hypothesis testing
  - (if you find $x \to y$ via EDA, "training" data)
- specification-searching for "effects"

EDA is **not**

- formal hypothesis testing
  - (if you find $x \rightarrow y$ via EDA, “training” data)
- specification-searching for “effects”
- p-hacking

EDA is **not**

- formal hypothesis testing
    - (if you find $x \to y$ via EDA, "training" data)
- specification-searching for "effects"
- p-hacking
- (even in obs designs: pre-specify analyses, standards, decisions, conclusions, . . .)

EDA: Search for Patterns and Models

- What pattern observed?
- What relationship does pattern represent?
- How strong?
- Holds in subgroups?
- Due to another factor? chance?

Visualization for EDA

Unidimensional visualizations

- Discrete variables
  - `geom_bar()`
  - `count()` for tibbular
- Continuous variables
  - `geom_histogram()`
  - `geom_boxplot()`
  - `geom_freqpoly()`
  - `geom_violin()`
  - `geom_density()`

Unidimensional visualizations

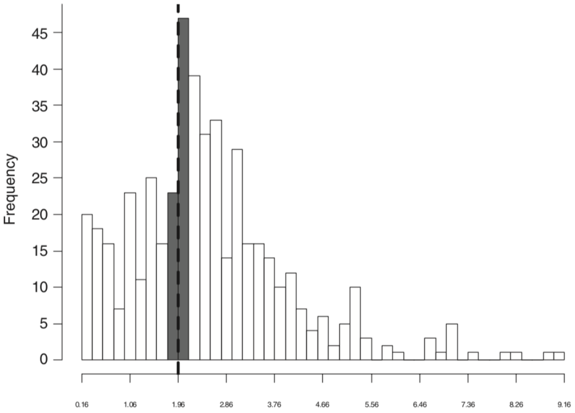
- Discrete variables
  - `geom_bar()`
  - `count()` for tibbular
- Continuous variables
  - `geom_histogram()`
  - `geom_boxplot()`
  - `geom_freqpoly()`
  - `geom_violin()`
  - `geom_density()`

(See the `ggplot2` Cheatsheet for more)

Histograms

Gerber & Malhotra, *SM&R* (2008):

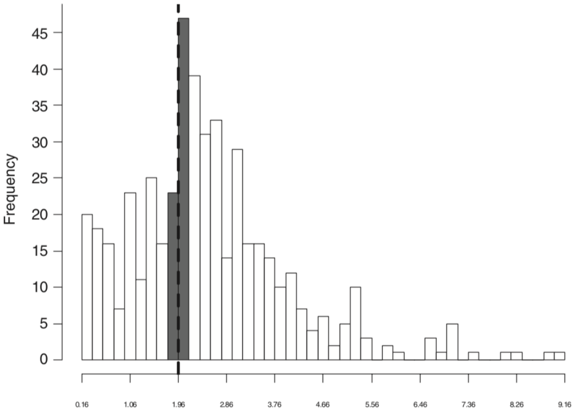
Histogram of *z* Statistics From the *American Sociological Review*, the *American Journal of Sociology*, and *The Sociological Quarterly* (Two-Tailed)



Histograms

Gerber & Malhotra, *SM&R* (2008):

Histogram of *z* Statistics From the *American Sociological Review*, the *American Journal of Sociology*, and *The Sociological Quarterly* (Two-Tailed)



"Publication Bias in Empirical Sociological Research: Do Arbitrary Significance Levels Distort Published Results?"

ANES 2016 Pilot Data

```
anes_16 <- read_csv(here("data", "anes_pilot_2016.csv"))
str(anes_16)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame'
##  $ version                          : chr  "ANES 2016 Pilot
##  $ caseid                           : num  1 2 3 4 5 6 7 8
##  $ weight                           : num  0.951 2.67 1.43
##  $ weight_spss                      : num  0.542 1.522 0.81
##  $ follow                           : num  1 2 1 1 1 1 1 1
##  $ turnout12                        : num  1 2 1 1 1 1 1 1
##  $ turnout12b                       : num  9 9 9 9 9 9 9 9
##  $ vote12                           : num  2 9 1 2 1 3 3 1
##  $ percent16                        : num  100 50 100 100 1
##  $ meet                             : num  1 4 1 5 2 3 1 5
##  $ givefut                          : num  3 5 1 4 1 3 1 4
##  $ info                             : num  4 4 1 5 3 2 1 5
##  $ march                            : num  1 2 1 2 1 2 2 2
##  $ sign                             : num  2 2 1 2 2 1 2 1
```

Some Demographics, Attitudes, Behaviors

```r
demog_names <- c("birthyr", "gender", "race", "educ",
                 "marstat", "speakspanish", "faminc",
                 "state", "pid3", "pid7", "ideo5")

behavior_names <- c("turnout12", "vote12", "meet",
                    "givefut", "info", "march", "sign")

# Feeling thermometers all start with "ft"
```

Recoding with Meaningful Labels

```r
anes_16$pid3_chr <- recode(anes_16$pid3,
                           `1` = "Dem",
                           `2` = "Rep",
                           `3` = "Ind",
                           `4` = "Oth",
                           `5` = "DK")
```

Recoding with Meaningful Labels

```r
anes_16$pid3_chr <- recode(anes_16$pid3,
                           `1` = "Dem",
                           `2` = "Rep",
                           `3` = "Ind",
                           `4` = "Oth",
                           `5` = "DK")
```

(Be careful with `%>%` here . . .)

Some Demographics, Attitudes, Behaviors

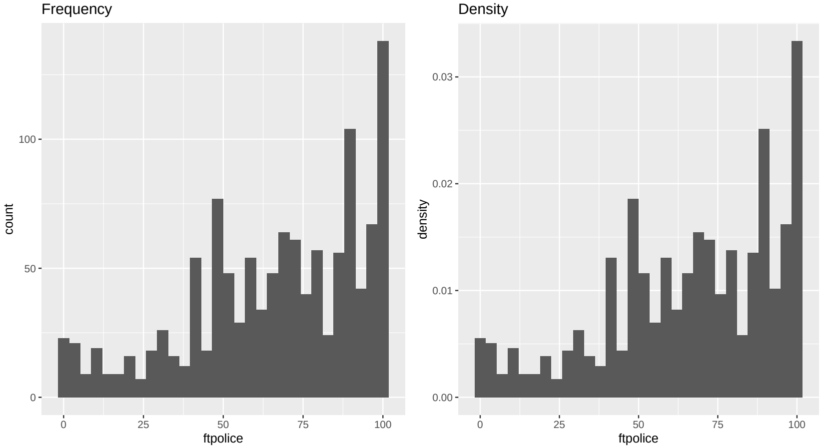
. . . Ratings between 50 degrees and 100 degrees . . . favorable and warm toward the person. Ratings between 0 degrees and 50 degrees . . . don't feel favorable toward the person and that you don't care too much for that person. . . . 50 degree mark if you don't feel particularly warm or cold toward the person.

Histograms

```
p1 <- ggplot(anes_16, aes(ftpolice)) +
  geom_histogram() + ggtitle("Frequency")
p2 <- ggplot(anes_16, aes(ftpolice, ..density..)) +
  geom_histogram() + ggtitle("Density")
```

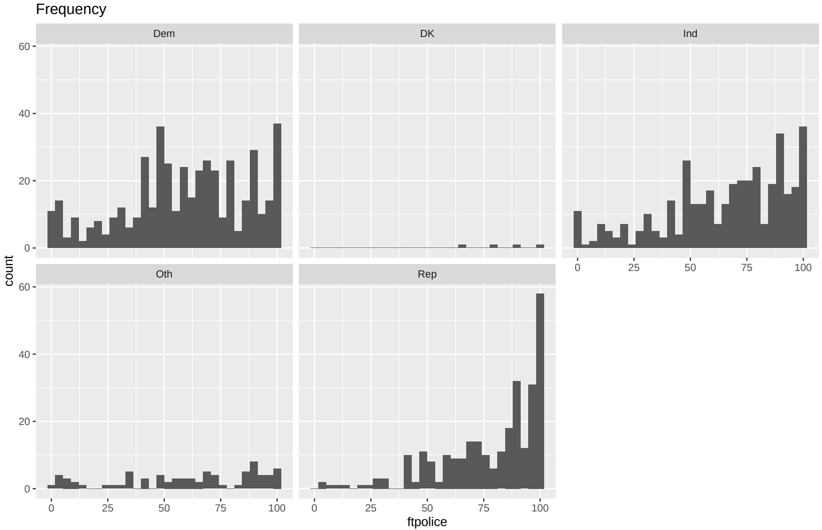
Histograms

```
p1 <- ggplot(anes_16, aes(ftpolice)) +
  geom_histogram() + ggtitle("Frequency")
p2 <- ggplot(anes_16, aes(ftpolice, ..density..)) +
  geom_histogram() + ggtitle("Density")
```



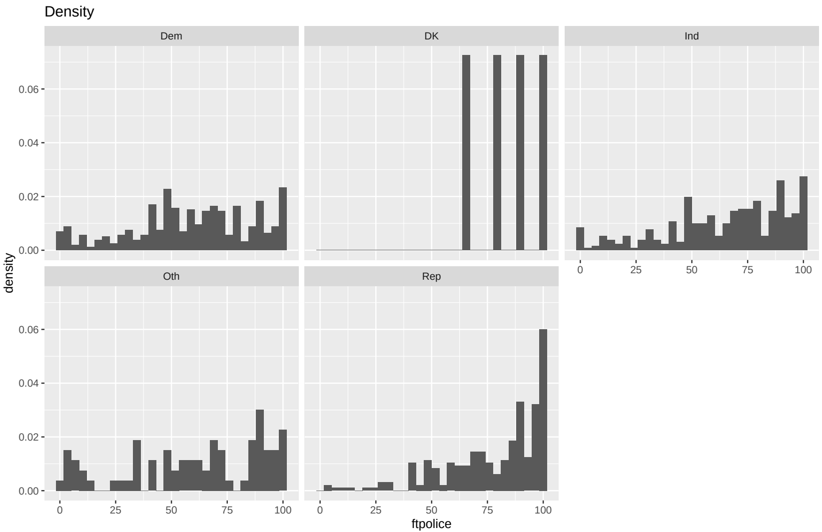
Histograms

```
ggplot(anes_16, aes(ftpolice)) + geom_histogram() +
  ggtitle("Frequency") + facet_wrap(~ pid3_chr)
```



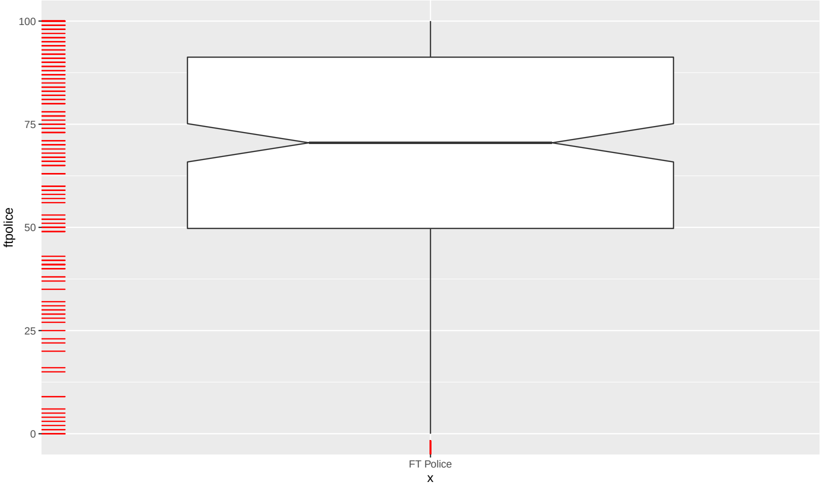
Histograms

```
ggplot(anes_16, aes(ftpolice, ..density..)) + geom_histogra
  ggtitle("Density") + facet_wrap(~ pid3_chr)
```

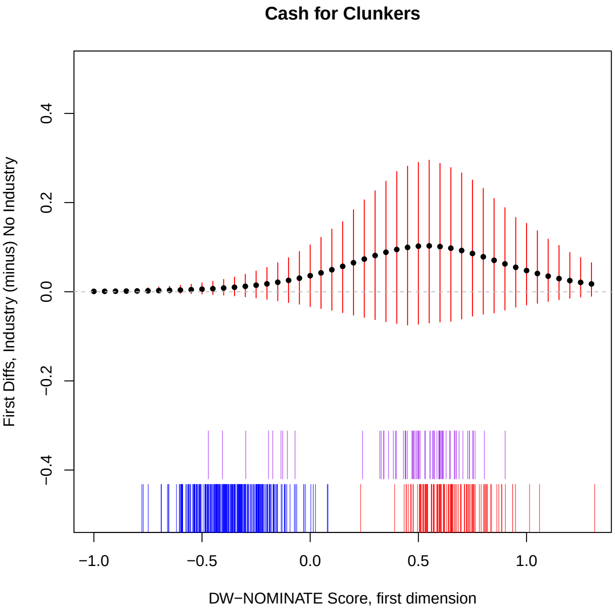


Boxplots

```
ggplot(anes_16 %>% sample_n(200), aes("FT Police", ftpolice
  geom_boxplot(notch = TRUE) + geom_rug(color = "red")
```

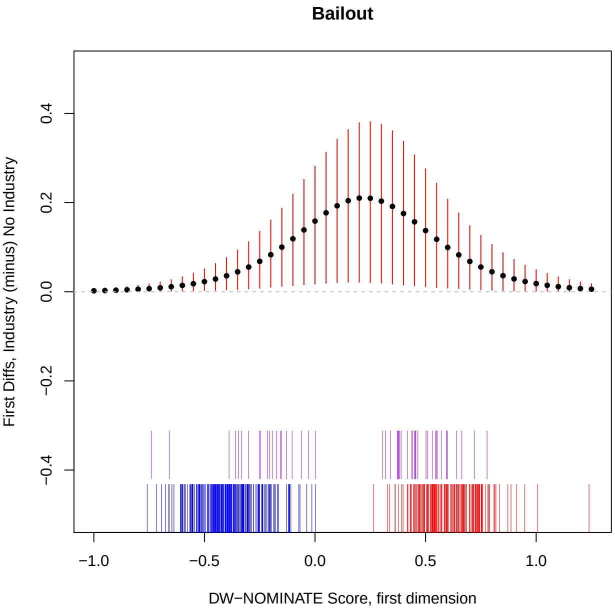


Rugs



Cash for Clunkers
First Diffs, Industry (minus) No Industry
0.4
0.2
0.0
−0.2
−0.4
−1.0
−0.5
0.0
0.5
1.0
DW−NOMINATE Score, first dimension

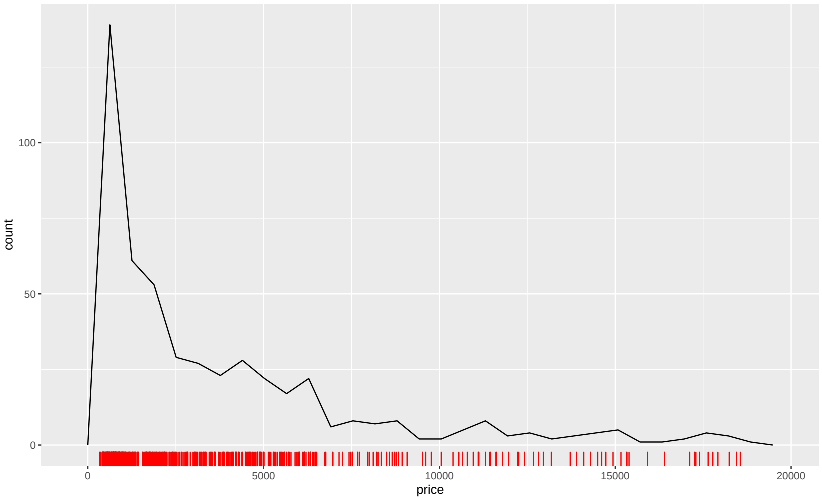
Rugs



Bailout
First Diffs, Industry (minus) No Industry
0.4
0.2
0.0
−0.2
−0.4
−1.0
−0.5
0.0
0.5
1.0
DW−NOMINATE Score, first dimension

Frequency Polygons

```
ggplot(diamonds %>% sample_n(500), aes(price)) +
  geom_freqpoly() + geom_rug(color = "red")
```



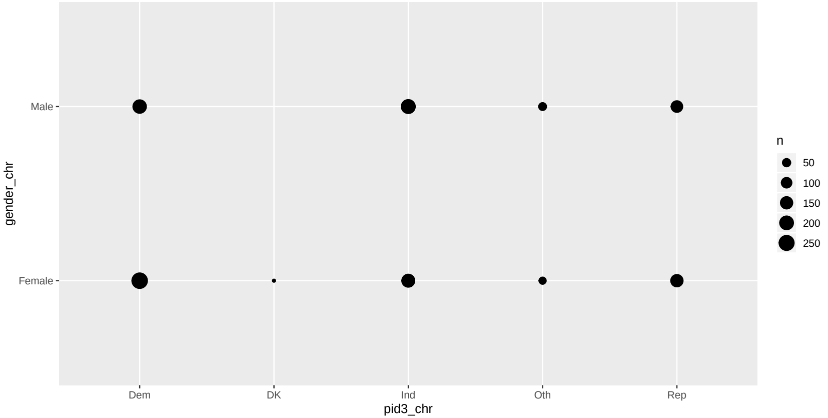
Multidimensional Visualizations

- Discrete × Discrete
  - `geom_count()`
  - `count()` for tibbular
- Continuous × Discrete
  - set of continuous distributions
  - `geom_boxplot(varwidth = TRUE)`
  - dotplot
- Continuous × Continuous
  - scatterplot
  - `geom_bin2d()`
  - `hexbin::geom_hex()`
- Continuous × Continuous × Continuous
  - `geom_contour()`
  - `geom_tile()`
  - heat maps

(See the `ggplot2` Cheatsheet for more)

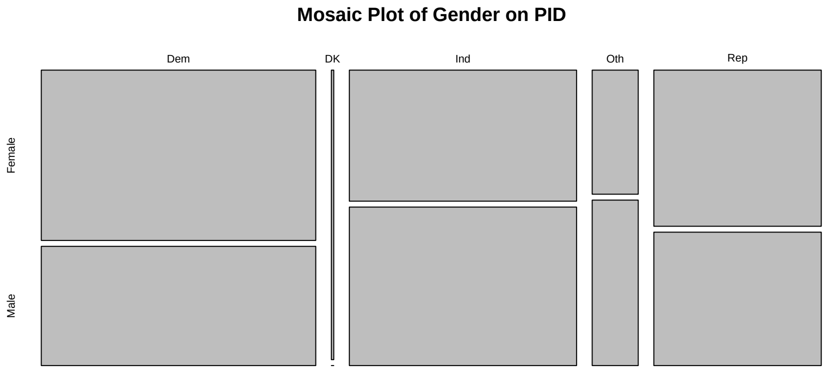
Count Distribution

```r
anes_16$gender_chr <- recode(anes_16$gender,
                             `1` = "Male", `2` = "Female")
ggplot(anes_16, aes(pid3_chr, gender_chr)) + geom_count()
```

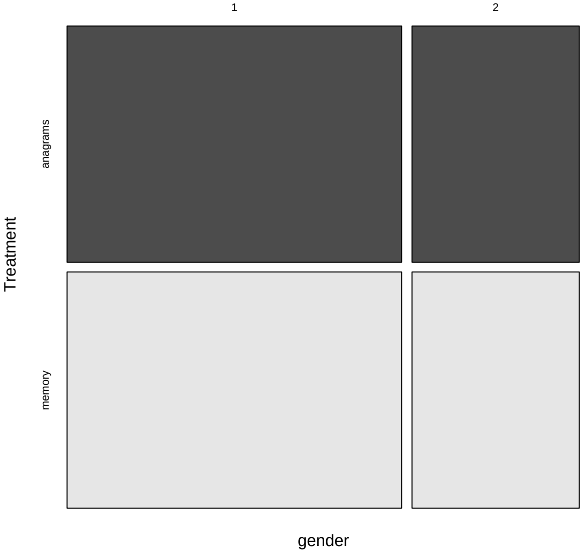


Mosaic Plot

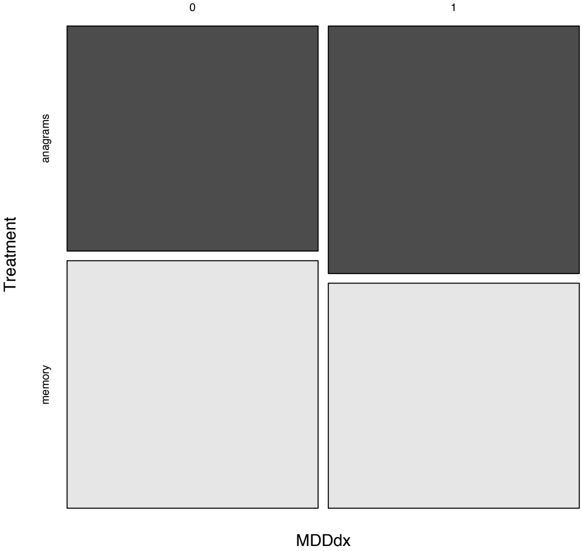
```
mosaicplot(table(anes_16$pid3_chr, anes_16$gender_chr),
           main = "Mosaic Plot of Gender on PID")
```



Mosaic Plot



Mosaic Plot

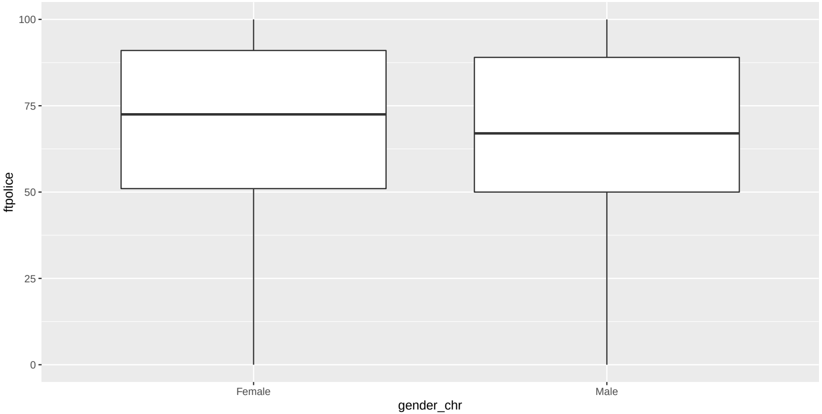


Boxplots

```
ggplot(anes_16, aes(gender_chr, ftpolice)) + geom_boxplot()
```

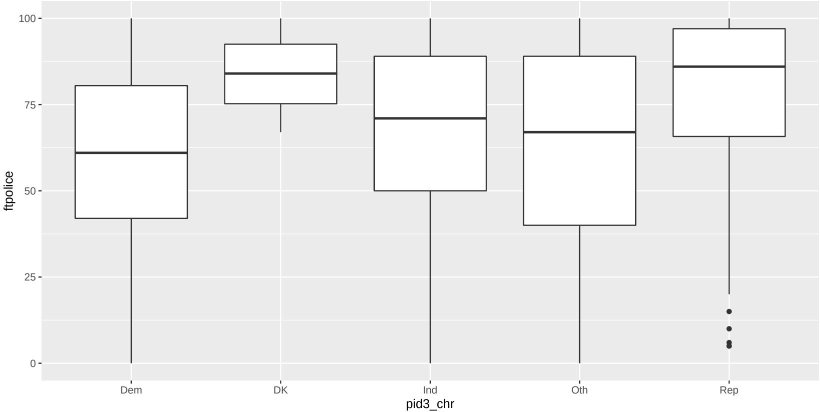
Boxplots

```r
ggplot(anes_16, aes(gender_chr, ftpolice)) + geom_boxplot()
```



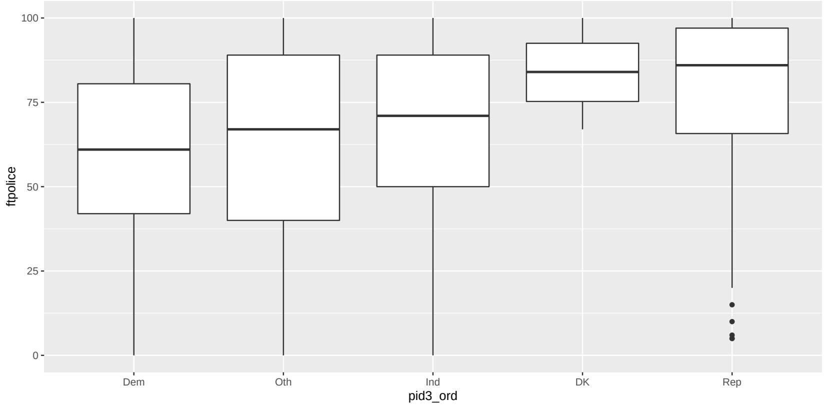
Conditional Boxplots

```
ggplot(anes_16, aes(pid3_chr, ftpolice)) + geom_boxplot()
```



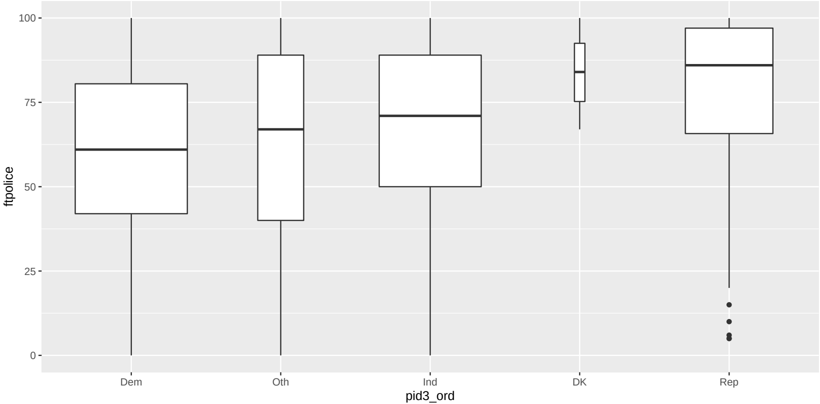
Conditional Boxplots

```r
anes_16$pid3_ord <- reorder(anes_16$pid3_chr,
                            anes_16$ftpolice, median)
#(NB: if NA's in ftpolice, those PIDs are ordered *last*)
ggplot(anes_16, aes(pid3_ord, ftpolice)) + geom_boxplot()
```

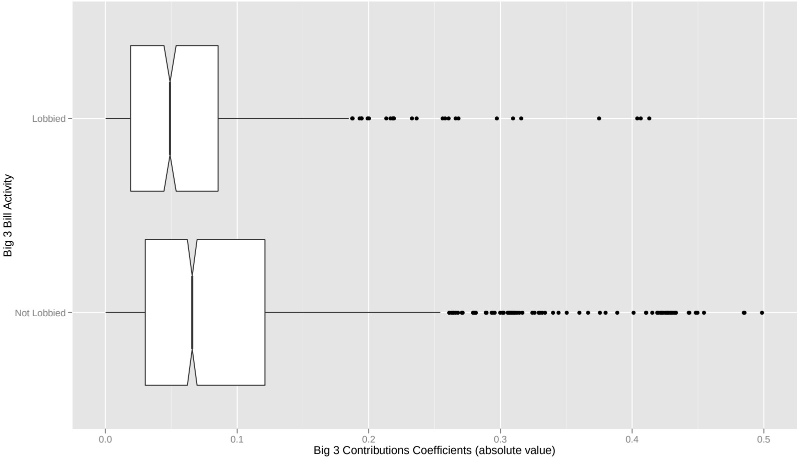


Conditional Boxplots

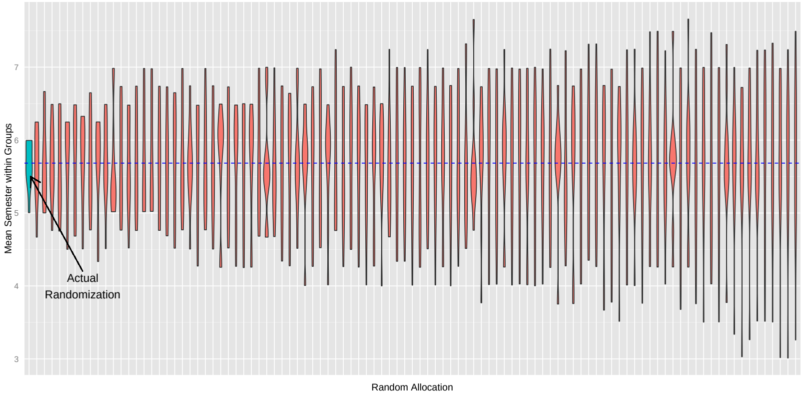
```
ggplot(anes_16, aes(pid3_ord, ftpolice)) +
  geom_boxplot(varwidth = TRUE)
```



Conditional Boxplots



Violins



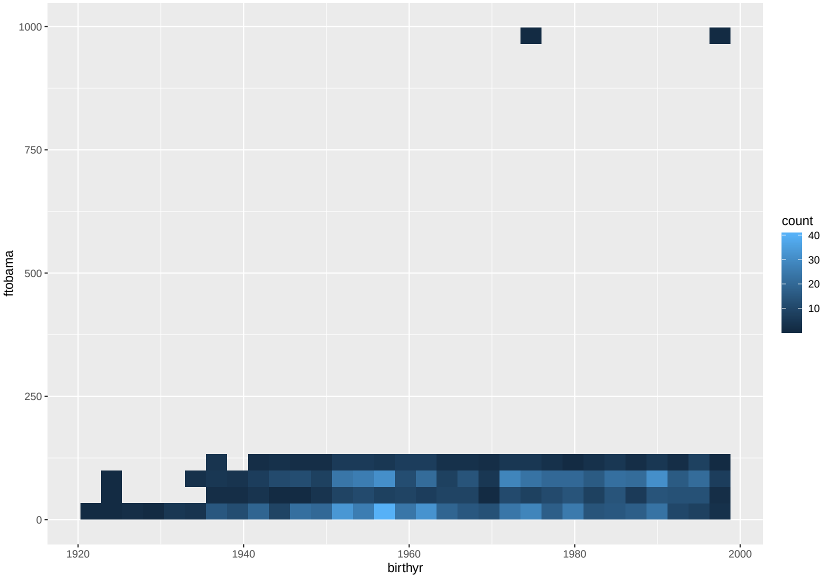
Mean Semester within Groups
7
6
5
4
3
Actual
Randomization
Random Allocation

(Rectangular) Heat Maps

```r
ggplot(anes_16, aes(birthyr, ftobama)) + geom_bin2d()
```

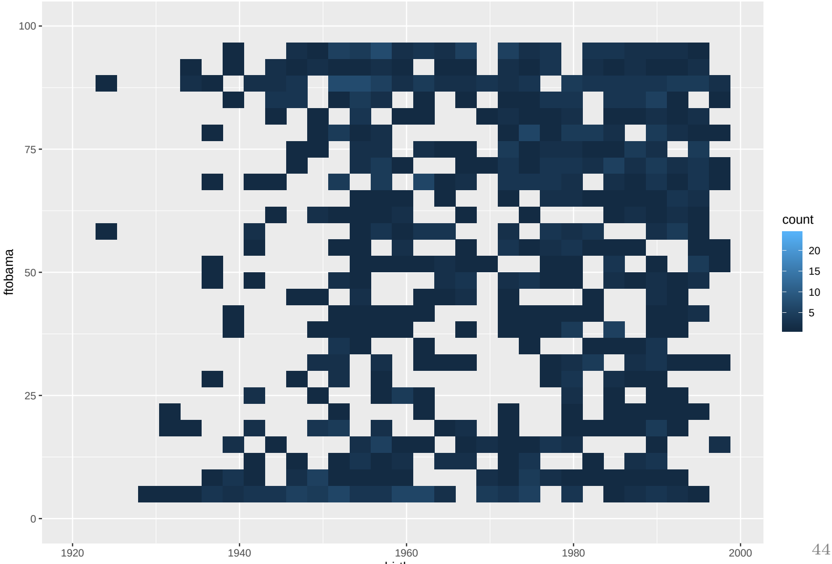
(Rectangular) Heat Maps

```
ggplot(anes_16, aes(birthyr, ftobama)) + geom_bin2d()
```



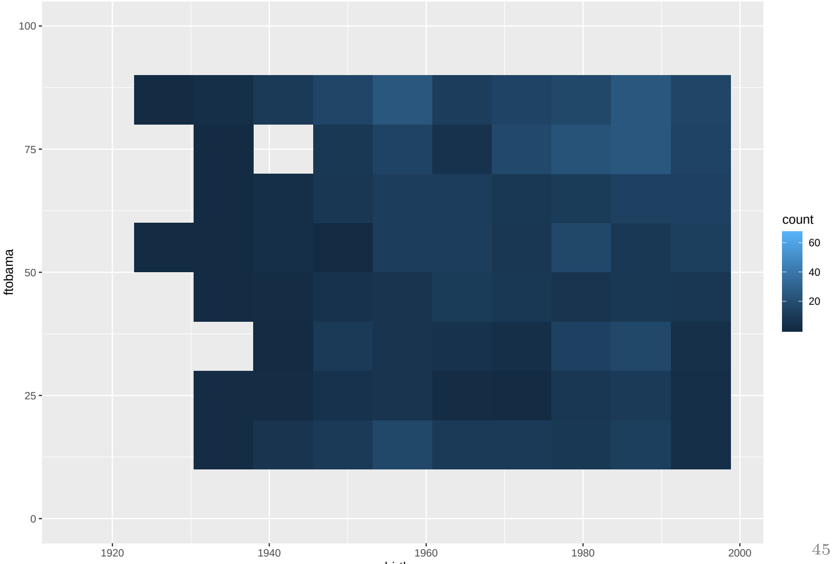
(Rectangular) Heat Maps

```
ggplot(anes_16, aes(birthyr, ftobama)) + geom_bin2d() +
  ylim(0, 100)
```

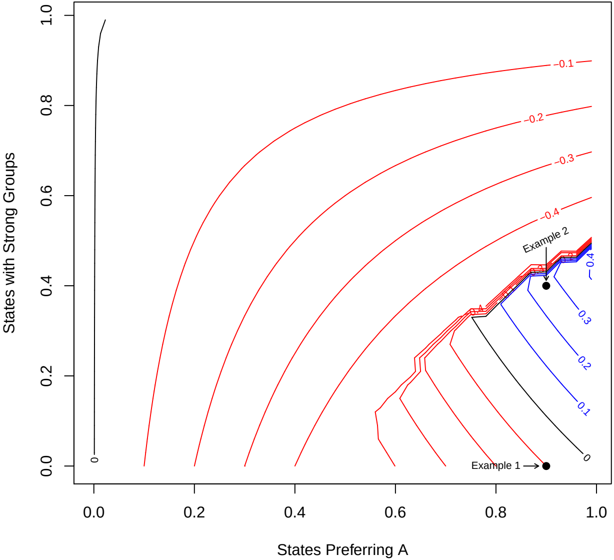


(Rectangular) Heat Maps

```
ggplot(anes_16, aes(birthyr, ftobama)) +
  geom_bin2d(bins = 10) + ylim(0, 100)
```



Contours



States with Strong Groups
States Preferring A
0.0
0.2
0.4
0.6
0.8
1.0
-0.1
-0.2
-0.3
-0.4
Example 2
Example 1
0
0.1
0.2
0.3
0.4

Numeric EDA

Numeric EDA

- *R4DS* focuses on **graphical** EDA for cleaning, discovery
- Do **numeric** EDA, too (esp. for cleaning)

```r
summary(anes_16)
```

```
##    version              caseid          weight          w
##  Length:1200        Min.   :   1.0   Min.   :0.1693   Mi
##  Class :character   1st Qu.: 300.8   1st Qu.:0.3948   1s
##  Mode  :character   Median : 600.5   Median :0.8105   Me
##                     Mean   : 600.5   Mean   :1.0000   Me
##                     3rd Qu.: 900.2   3rd Qu.:1.2210   3r
##                     Max.   :1200.0   Max.   :7.0104   Ma
##
##      follow         turnout12        turnout12b        vot
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:9.000   1st Qu.
##  Median :1.000   Median :1.000   Median :9.000   Median
##  Mean   :1.732   Mean   :1.275   Mean   :8.668   Mean
##  3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:9.000   3rd Qu.
##  Max.   :4.000   Max.   :3.000   Max.   :9.000   Max.
##
##    percent16          meet          givefut          in
##  Min.   :  0.0   Min.   :1.000   Min.   :1.000   Min.
```

```r
str(anes_16)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame'
##  $ version                  : chr  "ANES 2016 Pilot
##  $ caseid                   : num  1 2 3 4 5 6 7 8
##  $ weight                   : num  0.951 2.67 1.43
##  $ weight_spss              : num  0.542 1.522 0.8
##  $ follow                   : num  1 2 1 1 1 1 1 1
##  $ turnout12                : num  1 2 1 1 1 1 1 1
##  $ turnout12b               : num  9 9 9 9 9 9 9 9
##  $ vote12                   : num  2 9 1 2 1 3 3 1
##  $ percent16                : num  100 50 100 100
##  $ meet                     : num  1 4 1 5 2 3 1 5
##  $ givefut                  : num  3 5 1 4 1 3 1 4
##  $ info                     : num  4 4 1 5 3 2 1 5
##  $ march                    : num  1 2 1 2 1 2 2 2
##  $ sign                     : num  2 2 1 2 2 1 2 1
##  $ give12mo                 : num  2 2 1 2 1 1 1 2
##  $ compromise               : num  1 1 2 1 2 2 1 1
##  $ ftobama                  : num  100 39 1 89 1 0
```

Count Distributions

```r
anes_16$gender_chr <- recode(anes_16$gender,
                             `1` = "Male", `2` = "Female")
anes_16 %>% count(pid3_chr, gender_chr) %>%
  arrange(desc(n))
```

```
## # A tibble: 9 x 3
##   pid3_chr gender_chr     n
##   <chr>    <chr>      <int>
## 1 Dem      Female       270
## 2 Ind      Male         208
## 3 Dem      Male         189
## 4 Ind      Female       172
## 5 Rep      Female       151
## 6 Rep      Male         129
## 7 Oth      Male          44
## 8 Oth      Female        33
## 9 DK       Female         4
```

EDA Exercise: ANES data

Find a Friend

One	Two
Cameron	JessicaG
Bryce	Lauren
Xiaofeng	Robin
Olan	Zeinabou
Marc	Tanesia
Hubbert	Ethan
JessicaK	Carine
Milika	Kathleen
Hannah	Mark
AndrewZ	Jocelyn
AndrewE	Edward
Katherine	Kelly
Lucas	Erin

The Exercise

1. Create a new R project: `ex_anes`
2. Download `.csv` from GitHub; store in `ex_anes/data/`
3. Start a new `.R` file; read data with `read_csv()`
4. Create small df with only vars above (incl. feeling therms)
5. Create informative histogram/freqpoly/etc. of feeling therm scores toward Obama (Note *how*.)
6. Write down at least 3 questions/expectations you have about variation or covariation in the data
7. Recode the variables you're interested in
8. Do EDA. Answer your questions by visualizing, transforming, summarizing, iterating over the data

The Exercise

1. Create a new R project: `ex_anes`
2. Download `.csv` from GitHub; store in `ex_anes/data/`
3. Start a new `.R` file; read data with `read_csv()`
4. Create small df with only vars above (incl. feeling therms)
5. Create informative histogram/freqpoly/etc. of feeling therm scores toward Obama (Note *how*.)
6. Write down at least 3 questions/expectations you have about variation or covariation in the data
7. Recode the variables you're interested in
8. Do EDA. Answer your questions by visualizing, transforming, summarizing, iterating over the data

- Codebook and survey at http://j.mp/2E3RzR4
- ``recode(x, `1` = "Male", `2` = "Female")``

Wrangling

Tibbles v. Data Frames

Creating a `df` with `tibble()`:

- Preserves input types
- Preserves variable names
- Allows sequential variable creation
- Avoids row names ($\rightsquigarrow$ make them a variable!)
- Only recycles if `length(variable) == 1`

Tibbles v. Data Frames

Tibbles

- Print more reasonably
- Subset more strictly
  - `[` always → `tbl`
  - `df$x` won't get `df$xyz`

Data frames

- Print less reasonably
- Subset more liberally

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

What does R return?

```
df$x
tbl$x
```

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

What does R return?

```
df$x
tbl$x
```

```
df$x # scalar factor, length 1
tbl$x # NULL
```

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

What does R return?

```
df$x
tbl$x
```

```
df$x # scalar factor, length 1
tbl$x # NULL
```

```
tbl$xyz # vector, length 1
```

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

What does R return?

```
df[, "xyz"]
tbl[, "xyz"]
```

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

What does R return?

```
df[, "xyz"]
tbl[, "xyz"]
```

```
df[, "xyz"] # scalar factor, length 1
tbl[, "xyz"] # tibble, 1x1
```

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

What does R return?

```
df[, c("abc", "xyz")]
tbl[, c("abc", "xyz")]
```

Tibbles v. Data Frames

Consider

```
df <- data.frame(abc = 1, xyz = "a")
tbl <- tibble(abc = 1, xyz = "a")
```

What does R return?

```
df[, c("abc", "xyz")]
tbl[, c("abc", "xyz")]
```

```
df[, c("abc", "xyz")] # df, 1x2
tbl[, c("abc", "xyz")] # tibble, 1x2
```

Type depends on how many cols you select!

Reading Data

`read_csv()` is the tidyverse workhorse. Creates a `tbl`:

```
# 'vignettes' data from Imai, Ch 3:
vign <- read_csv("https://raw.githubusercontent.com/kosukeimai/q
vign
```

```
## # A tibble: 781 x 6
##     self alison  jane moses china   age
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1      5     5     2     0    31
##  2     1      1     5     5     0    54
##  3     2      3     1     1     0    50
##  4     2      4     2     1     0    22
##  5     2      3     3     3     0    52
##  6     1      3     1     5     0    50
##  7     1      1     1     1     0    35
##  8     4      4     4     5     0    56
##  9     3      2     1     2     0    53
## 10     1      3     1     1     0    22
## # ... with 771 more rows
```

Reading Data

`read.csv()` is the base R workhorse. Creates a `data.frame`:

```
vign2 <- read.csv("https://raw.githubusercontent.com/kosukeimai/
vign2
```

```
##    self alison jane moses china age
## 1     1      5    5     2     0  31
## 2     1      1    5     5     0  54
## 3     2      3    1     1     0  50
## 4     2      4    2     1     0  22
## 5     2      3    3     3     0  52
## 6     1      3    1     5     0  50
## 7     1      1    1     1     0  35
## 8     4      4    4     5     0  56
## 9     3      2    1     2     0  53
## 10    1      3    1     1     0  22
## 11    1      1    1     1     0  32
## 12    3      3    5     1     0  27
## 13    1      1    1     1     0  18
## 14    2      3    3     3     0  82
## 15    2      3    4     4     0  22
```

Arguments to `read_csv()`, etc.

- `col_names = TRUE`
- `locale = ...`
- `na = c("", "NA")`
- `quote = "\""`
- `comment = ""`
- `trim_ws = TRUE`
- `skip = 0`
- `n_max = Inf`

Create data with `tibble()`

```r
self <- c(1, 1, 3)
alison <- c(5, 1, 3)
jane <- c(5, 5, 1)
tibble(self, alison, jane)
```

Create data with tibble()

```r
self <- c(1, 1, 3)
alison <- c(5, 1, 3)
jane <- c(5, 5, 1)
tibble(self, alison, jane)
```

```
## # A tibble: 3 x 3
##    self alison  jane
##   <dbl>  <dbl> <dbl>
## 1     1      5     5
## 2     1      1     5
## 3     3      3     1
```

Create data with `tibble()`

```r
tibble(self = c(1, 1, 3), alison = c(5, 1, 3),
       jane = c(5, 5, 1))
```

```
## # A tibble: 3 x 3
##    self alison  jane
##   <dbl>  <dbl> <dbl>
## 1     1      5     5
## 2     1      1     5
## 3     3      3     1
```

Create data with `tibble()`

```
tibble(self = c(1, 1, 3), alison = c(5, 1, 3),
       jane = c(5, 5, 1))
```

```
## # A tibble: 3 x 3
##    self alison  jane
##   <dbl>  <dbl> <dbl>
## 1     1      5     5
## 2     1      1     5
## 3     3      3     1
```

Fine, unless thinking across rows. Create example where a low sees both high, a low sees polar opposites, a med sees peer-low.

Using `tribble()`

A low sees both high, a low sees polar opposites, a med sees peer-low:

```r
tribble(
  ~ self, ~ alison, ~ jane,
  #
  1, 5, 5,
  1, 1, 5,
  3, 3, 1
  )
```

```
## # A tibble: 3 x 3
##    self alison  jane
##   <dbl>  <dbl> <dbl>
## 1     1      5     5
## 2     1      1     5
## 3     3      3     1
```

Viewing, Extracting options

Viewing a `tbl`:

- `print(tbl, n = 5, width = Inf)` (temporary)

Viewing, Extracting options

Viewing a `tbl`:

- `print(tbl, n = 5, width = Inf)` (temporary)
- `options()` ("permanent" for session)

Viewing, Extracting options

Viewing a `tbl`:

- `print(tbl, n = 5, width = Inf)` (temporary)
- `options()` ("permanent" for session)
  - `options(tibble.print_max = Inf)`

Viewing, Extracting options

Viewing a `tbl`:

- `print(tbl, n = 5, width = Inf)` (temporary)
- `options()` ("permanent" for session)
  - `options(tibble.print_max = Inf)`
  - `options(tibble.width = Inf)`

Viewing, Extracting options

Viewing a `tbl`:

- `print(tbl, n = 5, width = Inf)` (temporary)
- `options()` ("permanent" for session)
  - `options(tibble.print_max = Inf)`
  - `options(tibble.width = Inf)`
- `as.data.frame(tbl)`

Viewing, Extracting options

Viewing a `tbl`:

- `print(tbl, n = 5, width = Inf)` (temporary)
- `options()` ("permanent" for session)
  - `options(tibble.print_max = Inf)`
  - `options(tibble.width = Inf)`
- `as.data.frame(tbl)`
- `View(tbl)`

Quick review of Viewing, Extracting

Extracting from df `df`:

```
##   var1  x
## 1    a 29
## 2    b 30
```

```r
df$x
```

```
## [1] 29 30
```

```r
df[["x"]]
```

```
## [1] 29 30
```

```r
df[[2]]
```

```
## [1] 29 30
```

Quick review of Viewing, Extracting

Extracting from tibble `tbl`:

```
##   var1  x
## 1    a 29
## 2    b 30
```

```r
tbl$x
```

```
## [1] 29 30
```

```r
tbl[["x"]]
```

```
## [1] 29 30
```

```r
tbl[[2]]
```

```
## [1] 29 30
```

Reading

A *locale* is a set of language, region, etc. parameters.

Locales

A *locale* is a set of language, region, etc. parameters.

The locale defines the parsing defaults.

Locales

You can define a locale locally:

```
parse_double("1,23",
             locale = locale(decimal_mark = ","))
```

```
## [1] 1.23
```

Locales

Or use a pre-defined locale:

```
parse_date("15 enero 2000", format = "%d %B %Y")
```

```
## [1] NA
```

Locales

Or use a pre-defined locale:

```r
parse_date("15 enero 2000", format = "%d %B %Y")
```

```
## [1] NA
```

```r
parse_date("15 enero 2000", locale = locale("es"),
           format = "%d %B %Y")
```

```
## [1] "2000-01-15"
```

Encodings

An *encoding* is part of a locale.

Encodings map from raw hexadecimals to characters.

E.g.,

```
parse_character("\x52\x54\x4d",
                locale = locale(encoding = "Latin1"))
```

```
## [1] "RTM"
```

Encodings

An *encoding* is part of a locale.

Encodings map from raw hexadecimals to characters.

E.g.,

```
parse_character("\x52\x54\x4d",
                locale = locale(encoding = "Latin1"))
```

```
## [1] "RTM"
```

```
parse_character("\x52\x54\x4d",
                locale = locale(encoding = "Latin2"))
```

```
## [1] "RTM"
```

Encodings

But,

```
parse_character("\xa1\x45\x73\x70\x61\xf1\x61\x21",
                locale = locale(encoding = "Latin2"))
```

```
## [1] "ĄEspańa!"
```

Encodings

But,

```
parse_character("\xa1\x45\x73\x70\x61\xf1\x61\x21",
                locale = locale(encoding = "Latin2"))
```

```
## [1] "ĄEspańa!"
```

```
parse_character("\xa1\x45\x73\x70\x61\xf1\x61\x21",
                locale = locale(encoding = "Latin1"))
```

```
## [1] "¡España!"
```

Encodings

When you get bad characters copy-pasting from Excel . . .

Encodings

When you get bad characters copy-pasting from Excel …

or even opening/closing file in Excel …

Encodings

When you get bad characters copy-pasting from Excel …

or even opening/closing file in Excel …

Excel doesn't use UTF-8.

Encodings

When you get bad characters copy-pasting from Excel ...

or even opening/closing file in Excel ...

Excel doesn't use UTF-8.

For writing a `.csv` that Excel will read:

```
write_excel_csv()
```

Encodings

When you get bad characters copy-pasting from Excel …

or even opening/closing file in Excel …

Excel doesn't use UTF-8.

For writing a `.csv` that Excel will read:

```
write_excel_csv()
```

(But, why?)

Some reading functions

- `read_csv()`: the workhorse
- `read_csv2()`: the `,`-not-`.` workhorse
- `read_fwf()`: conquer old-school survey files

Some reading functions

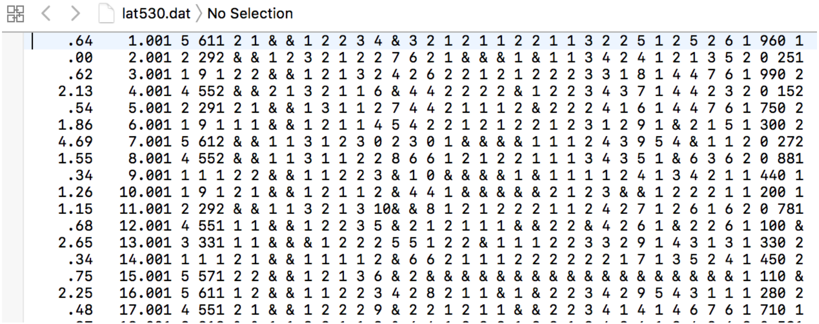
- `read_csv()`: the workhorse
- `read_csv2()`: the ,-not-. workhorse
- `read_fwf()`: conquer old-school survey files

For `.xlsx`/`.xls`:

- `library(readxl)`
- `read_excel()`

Fixed-width Files

The bad old days:



lat530.dat 〉 No Selection

```
 .64    1.001 5 611 2 1 & & 1 2 2 3 4 & 3 2 1 2 1 1 2 2 1 1 3 2 2 5 1 2 5 2 6 1 960 1
 .00    2.001 2 292 & & 1 2 3 2 1 2 2 7 6 2 1 & & & 1 & 1 1 3 4 2 4 1 2 1 3 5 2 0 251
 .62    3.001 1 9 1 2 2 & & 1 2 1 3 2 4 2 6 2 2 1 2 1 2 2 2 3 3 1 8 1 4 4 7 6 1 990 2
2.13    4.001 4 552 & & 2 1 3 2 1 1 6 & 4 4 2 2 2 2 & 1 2 2 3 4 3 7 1 4 4 2 3 2 0 152
 .54    5.001 2 291 2 1 & & 1 3 1 1 2 7 4 4 2 1 1 1 2 & 2 2 2 4 1 6 1 4 4 7 6 1 750 2
1.86    6.001 1 9 1 1 1 & & 1 2 1 1 4 5 4 2 2 1 2 1 2 2 1 2 3 1 2 9 1 & 2 1 5 1 300 2
4.69    7.001 5 612 & & 1 1 3 1 2 3 0 2 3 0 1 & & & & 1 1 1 2 4 3 9 5 4 & 1 1 2 0 272
1.55    8.001 4 552 & & 1 1 3 1 1 2 2 8 6 6 1 2 1 2 2 1 1 1 3 4 3 5 1 & 6 3 6 2 0 881
 .34    9.001 1 1 1 2 2 & & 1 1 2 2 3 & 1 0 & & & & 1 & 1 1 1 1 2 4 1 3 4 2 1 1 440 1
1.26   10.001 1 9 1 2 1 & & 1 2 1 1 2 & 4 4 1 & & & & & 2 1 2 3 & & 1 2 2 2 1 1 200 1
1.15   11.001 2 292 & & 1 1 3 2 1 3 10& & 8 1 2 1 2 2 2 1 1 2 4 2 7 1 2 6 1 6 2 0 781
 .68   12.001 4 551 1 1 & & 1 2 2 3 5 & 2 1 2 1 1 1 & & 2 2 & 4 2 6 1 & 2 2 6 1 100 &
2.65   13.001 3 331 1 1 & & & 1 2 2 2 5 5 1 2 2 & 1 1 1 2 2 3 3 2 9 1 4 3 1 3 1 330 2
 .34   14.001 1 1 1 2 1 & & 1 1 1 1 2 & 6 6 2 1 1 1 2 2 2 2 2 2 1 7 1 3 5 2 4 1 450 2
 .75   15.001 5 571 2 2 & & 1 2 1 3 6 & 2 & & & & & & & & & & & & & & & & & 1 110 &
2.25   16.001 5 611 1 2 & & 1 1 2 2 3 4 2 8 2 1 1 & 1 & 2 2 3 4 2 9 5 4 3 1 1 1 280 2
 .48   17.001 4 551 2 1 & & 1 2 2 2 9 & 2 2 1 2 1 1 & & 2 2 3 4 1 4 1 4 6 7 6 1 710 1
```

(Free half-day preschool, 2006)

Fixed-width Files

```r
wid530 <- c(7,8,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
            2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2)
lat530 <- read.fwf("../data/lat530.dat", widths = wid530)
```

Fixed-width Files

```r
wid530 <- c(7,8,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
            2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2)
lat530 <- read.fwf("../data/lat530.dat", widths = wid530)
```

```r
dim(lat530)
```

```
## [1] 2838   39
```

```r
head(lat530)
```

```
##      V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V
## 1 0.64  1  1  5 61 1  2  1  &   &   1   2   2   3   4
## 2 0.00  2  1  2 29 2  &  &  1   2   3   2   1   2   2
## 3 0.62  3  1  1  9 1  2  2  &   &   1   2   1   3   2
## 4 2.13  4  1  4 55 2  &  &  2   1   3   2   1   1   6
## 5 0.54  5  1  2 29 1  2  1  &   &   1   3   1   1   2
## 6 1.86  6  1  1  9 1  1  1  &   &   1   2   1   1   4
##   V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V3
## 1  1   1   2   2   1   1   3   2   2   5   1   2   5   2
## 2  &   &   1   &   1   1   3   4   2   4   1   2   1   3
```

Fixed-width Files

```
n530 <- c("region","party","prop82","gender","ideology","ag
which530 <- c("V4","V11","V12","V13","V29","V30","V33","V35
lat530s <- lat530[, which530]
names(lat530s) <- n530
head(lat530s)
```

```
##   region party prop82 gender ideology age educ income
## 1      5     1      2      2        2   5    5      6
## 2      2     3      2      1        2   4    1      5
## 3      1     1      2      1        1   8    4      6
## 4      4     3      2      1        3   7    4      3
## 5      2     1      3      1        1   6    4      6
## 6      1     1      2      1        2   9    2      5
```

Fixed-width Files

```r
lat530_guess <- read_fwf("../data/lat530.dat",
                         fwf_empty("../data/lat530.dat"))
lat530_guess
```

```
## # A tibble: 2,838 x 33
##       X1 X2        X3 X4    X5    X6    X7    X8    X9
##    <dbl> <chr>  <dbl> <chr> <chr> <chr> <chr> <chr> <chr
##  1  0.64 1.001      5 611   2     1     &     &     1
##  2  0    2.001      2 292   &     &     1     2     3
##  3  0.62 3.001      1 9 1   2     2     &     &     1
##  4  2.13 4.001      4 552   &     &     2     1     3
##  5  0.54 5.001      2 291   2     1     &     &     1
##  6  1.86 6.001      1 9 1   1     1     &     &     1
##  7  4.69 7.001      5 612   &     &     1     1     3
##  8  1.55 8.001      4 552   &     &     1     1     3
##  9  0.34 9.001      1 1 1   2     2     &     &     1
## 10  1.26 10.001     1 9 1   2     1     &     &     1
## # ... with 2,828 more rows, and 21 more variables: X13 <
```

Tidyverse v. Base R

Translation:
http://www.science.smith.edu/~amcnamara/Syntax-cheatsheet.pdf

Why read data with `tidyverse`?

- Speed
- Better defaults
    - `stringsAsFactors = FALSE`
    - ~~row names~~
    - `col names` preserved
- Reproducible
    - Indep of OS, environment

Parsing

Reading data is a composition of *parsings*.

`parse_<type>()`:

- `logical`
- `integer`
- `double`
- `number`

Parsing

```
tmp <- "Hi $1.000,00"
parse_number(tmp)
```

```
## [1] 1
```

Parsing

```r
tmp <- "Hi $1.000,00"
parse_number(tmp)
```

```
## [1] 1
```

```r
parse_number(tmp, locale = locale(decimal_mark = ","))
```

```
## [1] 1000
```

Parsing

```r
tmp <- "Hi $1.000,00"
parse_number(tmp)
```

```
## [1] 1
```

```r
parse_number(tmp, locale = locale(decimal_mark = ","))
```

```
## [1] 1000
```

```r
parse_number(tmp, locale = locale(decimal_mark = ",",
                                  grouping_mark = "."))
```

```
## [1] 1000
```

You will be tempted …

But use the parsers.

Parsing

Reading data is a composition of *parsings*.

`parse_<type>()`:

- `time`
- `date`
- `datetime`
- `factor`
- `character`

Parsing Dates and Times

Page 137, 7

```r
d1 <- "January 1, 2010"
parse_date(d1, "%B %d, %Y")
```

```
## [1] "2010-01-01"
```

Parsing Dates and Times

Page 137, 7

```
d1 <- "January 1, 2010"
parse_date(d1, "%B %d, %Y")
```

```
## [1] "2010-01-01"
```

```
parse_date(d1, "%B%.%d,%.%Y")
```

```
## [1] "2010-01-01"
```

Parsing Dates and Times

Page 137, 7

```
d1 <- "January 1, 2010"
parse_date(d1, "%B %d, %Y")
```

```
## [1] "2010-01-01"
```

```
parse_date(d1, "%B%.%d,%.%Y")
```

```
## [1] "2010-01-01"
```

```
parse_date(d1, "%B%.%d%.%.%Y")
```

```
## [1] "2010-01-01"
```

Parsing Dates and Times

Page 137, 7

```r
d2 <- "2015-Mar-07"
parse_date(d2, "%Y-%b-%d")
```

```
## [1] "2015-03-07"
```

```r
d3 <- "06-Jun-2017"
parse_date(d3, "%d-%b-%Y")
```

```
## [1] "2017-06-06"
```

```r
d4 <- c("August 19 (2015)", "July 1 (2015)")
parse_date(d4, "%B %d (%Y)")
```

```
## [1] "2015-08-19" "2015-07-01"
```

Parsing Dates and Times

Page 137, 7

```
d5 <- "12/30/14" # Dec 30, 2014
parse_date(d5, "%m/%d/%y")

t1 <- "1705"
parse_time(t1, "%H%M")

t2 <- "11:15:10.12 PM"
parse_time(t2, "%H:%M:%OS %p")
```

Writing

Write to more-universal standards

- Write strings with UTF-8
- Write dates/times in ISO-8601
- Write rectangular files to `.csv`

Write to more-universal standards

- Write strings with UTF-8
- Write dates/times in ISO-8601
- Write rectangular files to `.csv`

Be nice.

Tidy Data, Anonymization, Exercise

Structuring Data: `tidy` Definitions

- Variable: measured quantity
- Value: state of variable as measured
- Observation/Unit/Case: set of values under similar conditions
- Tidy data
  - each value is its own cell
  - each variable is its own column

Tidy Data

1. Each variable is a column.

Tidy Data

1. Each variable is a column.
2. Each observation is a row.

Tidy Data

1. Each variable is a column.
2. Each observation is a row.
3. Each table is a type of observational unit.

The Most Common Messes

1. Column headers are values, not variable names.

The Most Common Messes

1. Column headers are values, not variable names.
2. Multiple variables are stored in one column.

The Most Common Messes

1. Column headers are values, not variable names.
2. Multiple variables are stored in one column.
3. Variables are stored in both rows and columns.

The Most Common Messes

1. Column headers are values, not variable names.
2. Multiple variables are stored in one column.
3. Variables are stored in both rows and columns.
4. Multiple types of obs units are stored in the same table.

The Most Common Messes

1. Column headers are values, not variable names.
2. Multiple variables are stored in one column.
3. Variables are stored in both rows and columns.
4. Multiple types of obs units are stored in the same table.
5. A single observational unit is stored in multiple tables.

Mess 1: Column headers are values, not variable names

```
table4a
```

```
## # A tibble: 3 x 3
##   country      `1999` `2000`
## * <chr>         <int>  <int>
## 1 Afghanistan     745   2666
## 2 Brazil        37737  80488
## 3 China        212258 213766
```

Mess 2: Multiple variables stored in one column

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>       <int> <chr>
## 1 Afghanistan  1999 745/19987071
## 2 Afghanistan  2000 2666/20595360
## 3 Brazil       1999 37737/172006362
## 4 Brazil       2000 80488/174504898
## 5 China        1999 212258/1272915272
## 6 China        2000 213766/1280428583
```

Mess 3: Variables stored in both rows and columns

```
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <int>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

Mess 4. Multiple types of obs units stored in same table

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

Mess 4. Multiple types of obs units stored in same table

id	artist	track	time
1	2 Pac	Baby Don't Cry	4:22
2	2Ge+her	The Hardest Part Of ...	3:15
3	3 Doors Down	Kryptonite	3:53
4	3 Doors Down	Loser	4:24
5	504 Boyz	Wobble Wobble	3:35
6	98^0	Give Me Just One Nig...	3:24
7	A*Teens	Dancing Queen	3:44
8	Aaliyah	I Don't Wanna	4:15
9	Aaliyah	Try Again	4:03
10	Adams, Yolanda	Open My Heart	5:30
11	Adkins, Trace	More	3:05
12	Aguilera, Christina	Come On Over Baby	3:38
13	Aguilera, Christina	I Turn To You	4:00
14	Aguilera, Christina	What A Girl Wants	3:18
15	Alice Deejay	Better Off Alone	6:50

id	date	rank
1	2000-02-26	87
1	2000-03-04	82
1	2000-03-11	72
1	2000-03-18	77
1	2000-03-25	87
1	2000-04-01	94
1	2000-04-08	99
2	2000-09-02	91
2	2000-09-09	87
2	2000-09-16	92
3	2000-04-08	81
3	2000-04-15	70
3	2000-04-22	68
3	2000-04-29	67
3	2000-05-06	66

Mess 5: A single obs unit stored in multiple tables

What is the observational unit?

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
## * <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

The Vignettes data

What's the unit of observation?

```
vign
```

```
## # A tibble: 781 x 6
##     self alison  jane moses china   age
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1      5     5     2     0    31
##  2     1      1     5     5     0    54
##  3     2      3     1     1     0    50
##  4     2      4     2     1     0    22
##  5     2      3     3     3     0    52
##  6     1      3     1     5     0    50
##  7     1      1     1     1     0    35
##  8     4      4     4     5     0    56
##  9     3      2     1     2     0    53
## 10     1      3     1     1     0    22
## # ... with 771 more rows
```

The Vignettes data

What change? What's lost? What's gained?

```
vign %>% gather(alison, jane, moses, key = "person",
                value = "score") %>% arrange(age, person, s
```

The Vignettes data

What change? What's lost? What's gained?

```
vign %>% gather(alison, jane, moses, key = "person",
                value = "score") %>% arrange(age, person, s
```

```
## # A tibble: 2,343 x 5
##      self china   age person score
##     <dbl> <dbl> <dbl> <chr>  <dbl>
##  1      1     0    18 alison     1
##  2      1     0    18 alison     3
##  3      1     0    18 alison     3
##  4      1     0    18 alison     2
##  5      1     0    18 alison     3
##  6      1     0    18 alison     1
##  7      1     0    18 alison     1
##  8      2     0    18 alison     3
##  9      2     0    18 alison     3
## 10      2     1    18 alison     3
## # ... with 2,333 more rows
```

The Vignettes data

```
vign %>% gather(self, alison, jane, moses, key = "person",
                value = "score") %>% arrange(age, person)
```

The Vignettes data

```
vign %>% gather(self, alison, jane, moses, key = "person",
                value = "score") %>% arrange(age, person)
```

```
## # A tibble: 3,124 x 4
##    china   age person score
##    <dbl> <dbl> <chr>  <dbl>
##  1     0    18 alison     1
##  2     0    18 alison     3
##  3     0    18 alison     3
##  4     0    18 alison     3
##  5     0    18 alison     3
##  6     0    18 alison     3
##  7     0    18 alison     4
##  8     0    18 alison     2
##  9     0    18 alison     3
## 10     0    18 alison     3
## # ... with 3,114 more rows
```

The Vignettes data

```
# A tbl of respondent data:
df_respondents <- vign %>% transmute(id = 1:nrow(vign),
                                     self = self,
                                     china = china,
                                     age = age)
```

The Vignettes data

```
# A tbl of respondent data:
df_respondents <- vign %>% transmute(id = 1:nrow(vign),
                                     self = self,
                                     china = china,
                                     age = age)
```

```
# A tbl of scores:
df_scores <- vign %>% add_column(id = 1:nrow(vign)) %>%
  select(-age, -china) %>%
  gather(self, alison, jane, moses,
         key = "person", value = "score")
```

The Vignettes data

```
df_respondents
```

```
## # A tibble: 781 x 4
##        id  self china   age
##     <int> <dbl> <dbl> <dbl>
##  1      1     1     0    31
##  2      2     1     0    54
##  3      3     2     0    50
##  4      4     2     0    22
##  5      5     2     0    52
##  6      6     1     0    50
##  7      7     1     0    35
##  8      8     4     0    56
##  9      9     3     0    53
## 10     10     1     0    22
## # ... with 771 more rows
```

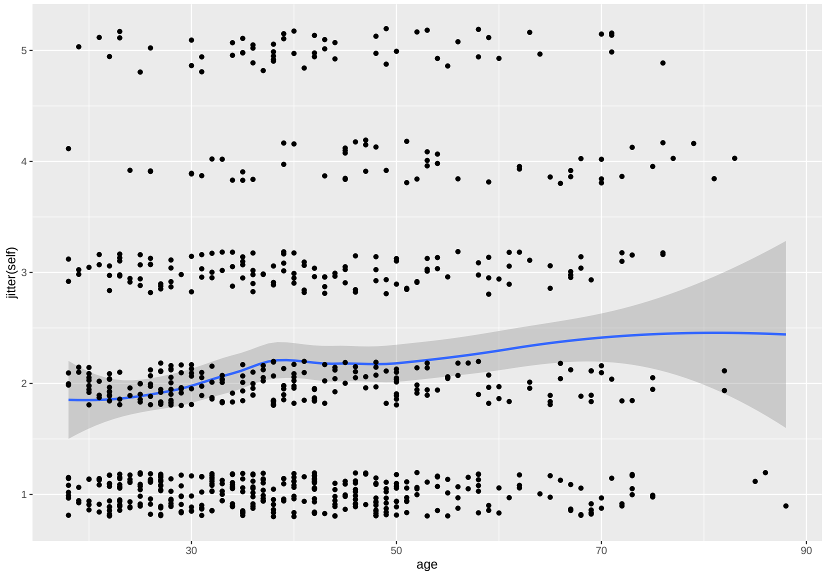
The Vignettes data

```
df_scores
```

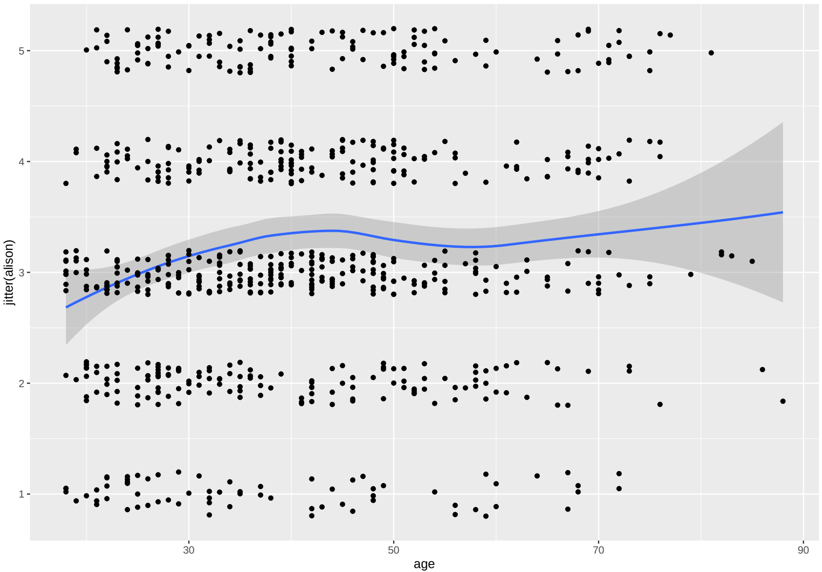
```
## # A tibble: 3,124 x 3
##        id person score
##     <int> <chr>  <dbl>
##  1      1 self       1
##  2      2 self       1
##  3      3 self       2
##  4      4 self       2
##  5      5 self       2
##  6      6 self       1
##  7      7 self       1
##  8      8 self       4
##  9      9 self       3
## 10     10 self       1
## # ... with 3,114 more rows
```

```
ggplot(vign, aes(age, jitter(self))) + geom_smooth() +
  geom_point()
```

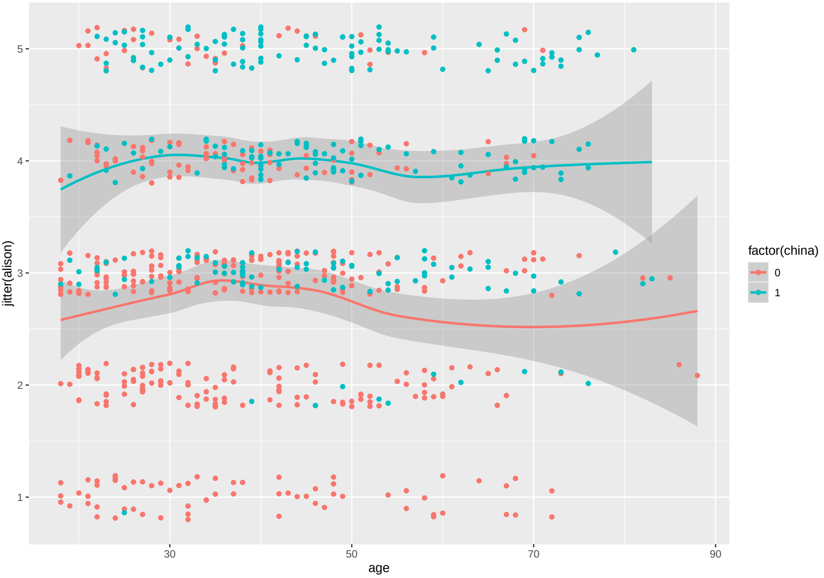
```r
ggplot(vign, aes(age, jitter(self))) + geom_smooth() +
  geom_point()
```



```r
ggplot(vign, aes(age, jitter(alison))) + geom_smooth() +
  geom_point()
```



```r
ggplot(vign, aes(age, jitter(alison), group = china, color
  geom_smooth() + geom_point()
```



The Vignettes data

Better: decide nature of `self`, how to relate it to others, transform, . . .

Tidying functions

- `spread()` and `gather()`
  - $\rightsquigarrow$ wider and narrower as f(data combinations)
- `pivot_wider()` and `pivot_longer()`
- `separate()` and `unite()`
  - $\rightsquigarrow$ wider and narrower as f(each cell characteristics)

Tidying functions: spread()

```
df_scores
```

```
## # A tibble: 3,124 x 3
##        id person score
##     <int> <chr>  <dbl>
##  1      1 self       1
##  2      2 self       1
##  3      3 self       2
##  4      4 self       2
##  5      5 self       2
##  6      6 self       1
##  7      7 self       1
##  8      8 self       4
##  9      9 self       3
## 10     10 self       1
## # ... with 3,114 more rows
```

Tidying functions: spread()

```
df_scores %>% spread(key = person, value = score)
```

```
## # A tibble: 781 x 5
##       id alison  jane moses  self
##    <int>  <dbl> <dbl> <dbl> <dbl>
##  1     1      5     5     2     1
##  2     2      1     5     5     1
##  3     3      3     1     1     2
##  4     4      4     2     1     2
##  5     5      3     3     3     2
##  6     6      3     1     5     1
##  7     7      1     1     1     1
##  8     8      4     4     5     4
##  9     9      2     1     2     3
## 10    10      3     1     1     1
## # ... with 771 more rows
```

Anonymizing sensitive data

```
sens
```

```
## # A tibble: 3 x 4
##   Donor Address         Phone Score
##   <chr> <chr>           <dbl> <dbl>
## 1 Ryan  10 Downing St     123    90
## 2 Esme  667 Dark Ave      456    50
## 3 Simon 10 Downing St     789    70
```

Anonymizing sensitive data

```r
sens
```

```
## # A tibble: 3 x 4
##   Donor Address         Phone Score
##   <chr> <chr>           <dbl> <dbl>
## 1 Ryan  10 Downing St     123    90
## 2 Esme  667 Dark Ave      456    50
## 3 Simon 10 Downing St     789    70
```

```r
library(digest)
```

Anonymizing sensitive data

```r
cols_to_mask <- c("Address", "Phone")
for(i in cols_to_mask){
  anon <- sapply(unlist(sens[, i]), digest, algo = "sha1")
  short_anon <- substr(anon, 1, 10)
  sens[, i] <- short_anon
}
```

Anonymizing sensitive data

```
cols_to_mask <- c("Address", "Phone")
for(i in cols_to_mask){
  anon <- sapply(unlist(sens[, i]), digest, algo = "sha1")
  short_anon <- substr(anon, 1, 10)
  sens[, i] <- short_anon
}
```

```
sens
```

```
## # A tibble: 3 x 4
##   Donor Address    Phone      Score
##   <chr> <chr>      <chr>      <dbl>
## 1 Ryan  c209e019b6 dac97294d4    90
## 2 Esme  7ccc94e690 0f2c3261a0    50
## 3 Simon c209e019b6 b13ceddf81    70
```

Anonymizing sensitive data

```r
cols_to_mask <- c("Address", "Phone")
for(i in cols_to_mask){
  anon <- sapply(unlist(sens[, i]), digest, algo = "sha1")
  short_anon <- substr(anon, 1, 10)
  sens[, i] <- short_anon
}
```

```r
sens
```

```
## # A tibble: 3 x 4
##   Donor Address    Phone      Score
##   <chr> <chr>      <chr>      <dbl>
## 1 Ryan  c209e019b6 dac97294d4    90
## 2 Esme  7ccc94e690 0f2c3261a0    50
## 3 Simon c209e019b6 b13ceddf81    70
```

(Warning: don't shorten!)

Wrangling Exercise

Find a Friend

One	Two
Tanesia	Olan
Edward	JessieG
Lauren	AndrewZ
LucasA	Erin
Carine	Jocelyn
Zeinabou	Ethan
Marc	Kathleen
Cameron	JessicaK
Kelly	Milika
LucasG	Mark
AndrewE	Hannah
Hubbert	Kate
Sophie	Bryce

Exercise: Laws Passed data

3-row sample:

```
laws
```

```
## # A tibble: 3 x 6
##   section statyear statmonth statday authorA authorB
##   <chr>      <dbl> <chr>       <dbl> <chr>   <chr>
## 1 1.2         2016 Jan             1 yes     no
## 2 3.8         2017 Feb             2 yes     no
## 3 4.1         2018 Mar             3 no      yes
```

Exercise: Laws Passed data

1. Combine year, month, day into one variable, `stat_date` (use `-` between components).
2. Split section into `section`, `subsection` (numeric).
3. Create a single variable for `author`.
4. Parse the date variable.

Exercise: Laws Passed data

```
laws2 <- laws %>% unite(date, starts_with("stat"),
                        sep = "-")
laws2
```

```
## # A tibble: 3 x 4
##   section date        authorA authorB
##   <chr>   <chr>       <chr>   <chr>
## 1 1.2     2016-Jan-1 yes     no
## 2 3.8     2017-Feb-2 yes     no
## 3 4.1     2018-Mar-3 no      yes
```

Exercise: Laws Passed data

```r
laws3 <- laws2 %>% separate(section,
                            into = c("section", "subsection
                            convert = TRUE)
laws3
```

```
## # A tibble: 3 x 5
##   section subsection date       authorA authorB
##     <int>      <int> <chr>      <chr>   <chr>
## 1       1          2 2016-Jan-1 yes     no
## 2       3          8 2017-Feb-2 yes     no
## 3       4          1 2018-Mar-3 no      yes
```

Exercise: Laws Passed data

```r
laws4 <- laws3 %>% gather(starts_with("author"),
                          key = author,
                          value = wrote_bill)
laws4
```

```
## # A tibble: 6 x 5
##   section subsection date        author  wrote_bill
##     <int>      <int> <chr>       <chr>   <chr>
## 1       1          2 2016-Jan-1 authorA yes
## 2       3          8 2017-Feb-2 authorA yes
## 3       4          1 2018-Mar-3 authorA no
## 4       1          2 2016-Jan-1 authorB no
## 5       3          8 2017-Feb-2 authorB no
## 6       4          1 2018-Mar-3 authorB yes
```

Exercise: Laws Passed data

```
laws5 <- laws4 %>% filter(wrote_bill == "yes")
laws5
```

```
## # A tibble: 3 x 5
##   section subsection date        author  wrote_bill
##     <int>      <int> <chr>       <chr>   <chr>
## 1       1          2 2016-Jan-1 authorA yes
## 2       3          8 2017-Feb-2 authorA yes
## 3       4          1 2018-Mar-3 authorB yes
```

Exercise: Laws Passed data

```
laws5 <- laws4 %>% filter(wrote_bill == "yes")
laws5
```

```
## # A tibble: 3 x 5
##   section subsection date        author  wrote_bill
##     <int>      <int> <chr>       <chr>   <chr>
## 1       1          2 2016-Jan-1 authorA yes
## 2       3          8 2017-Feb-2 authorA yes
## 3       4          1 2018-Mar-3 authorB yes
```

```
(laws5 <- laws5 %>% select(- wrote_bill))
```

```
## # A tibble: 3 x 4
##   section subsection date        author
##     <int>      <int> <chr>       <chr>
## 1       1          2 2016-Jan-1 authorA
## 2       3          8 2017-Feb-2 authorA
## 3       4          1 2018-Mar-3 authorB
```

Exercise: Laws Passed data

```
laws5$date <- parse_date(laws5$date, "%Y-%b-%d")
laws5
```

```
## # A tibble: 3 x 4
##   section subsection date       author
##     <int>      <int> <date>     <chr>
## 1       1          2 2016-01-01 authorA
## 2       3          8 2017-02-02 authorA
## 3       4          1 2018-03-03 authorB
```