Unleashing the Power of the Tidyverse

with the *pipe operator*

Understanding Political Numbers

Feb 25, 2019

Agenda

The pipe operator: `%>%`

- a revolution in the world of `R`
- Download `lecture_pipes.R` from Canvas ("Lecture Scripts" folder)

Also:

- Research question
- Data sources

Data Sources

Major U.S. Public Opinion Surveys

American National Election Study

- Since 1948(?); standard questions; some panel studies

Current Population Survey ("Voting and Registration Supplement")

- BIG sample; normally economic/family questions
- November supplement: self-reported voter turnout (but no other politics)

Cooperative Congressional Election Study

- since 2006; samples in every district; *validated* turnout

General Social Survey

- More than political

CODEBOOKS ARE SO IMPORTANT

Major International Opinion Sources

Comparative study of electoral systems

- international elections/opinion survey

Eurobarometer

- EU backed regular surveys; thematic, flash, and qualitative surveys

Afrobarometer

- since 1999; attitudes on democracy, governance, economics

LAPOP/AmericasBarometer

- Focus on Latin America, sophisticated survey delivery tech

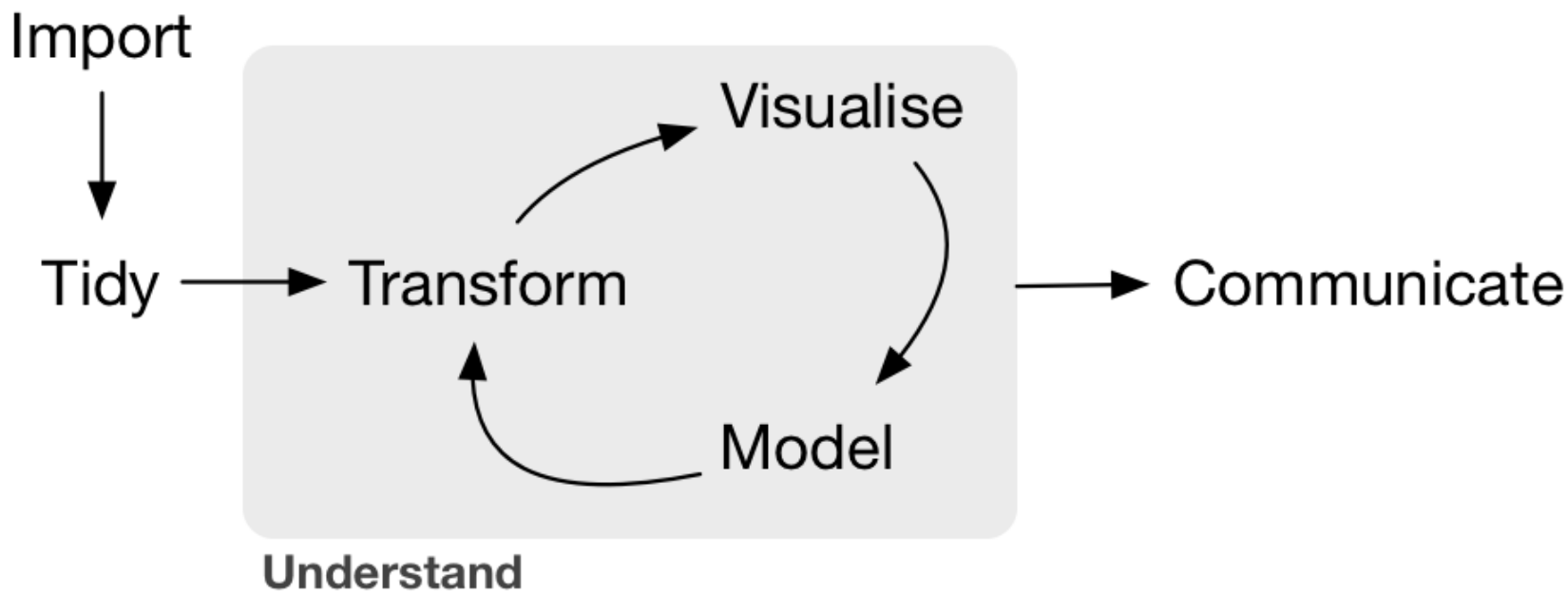
World Values Survey

IR data dump

Paul Hensel Data Site

- Alliances, treaties, organizations
- Conflict, MIDs, political violence
- Trade, aid, development
- Agriculture, food, environment
- Regime type, political systems, demography

The Pipe


Import
Tidy
Transform
Visualise
Model
Communicate
Understand

```
library("tidyverse")

# COUNTY-level data from Midwest states
midwest
```

```
## # A tibble: 437 x 28
##      PID county state  area poptotal popdensity popwhite popblack
##    <int> <chr>  <chr> <dbl>    <int>      <dbl>    <int>    <int>
##  1   561 ADAMS  IL    0.052    66090      1271.    63917     1702
##  2   562 ALEXA… IL    0.014    10626       759      7054     3496
##  3   563 BOND   IL    0.022    14991       681.    14477      429
##  4   564 BOONE  IL    0.017    30806      1812.    29344      127
##  5   565 BROWN  IL    0.018     5836       324.     5264      547
##  6   566 BUREAU IL    0.05     35688       714.    35157       50
##  7   567 CALHO… IL    0.017     5322       313.     5298        1
##  8   568 CARRO… IL    0.027    16805       622.    16519      111
##  9   569 CASS   IL    0.024    13437       560.    13384       16
## 10   570 CHAMP… IL    0.058   173025      2983.   146506    16559
## # … with 427 more rows, and 20 more variables: popamerindian <int>,
## #   popasian <int>, popother <int>, percwhite <dbl>, percblack <dbl>,
## #   percamerindan <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>
```

We want to:

- Calculate the proportion of the population that is *children* (for every county)
- Then, calculate the mean proportion of children in every state

We want to:

- Calculate the proportion of the population that is *children* (for every county): mutate()
- Then, calculate the mean proportion of children in every state: group_by() and summarize()

We want to:

- Calculate the proportion of the population that is *children* (for every county): mutate()
- Then, calculate the mean proportion of children in every state: group_by() and summarize()

Overwrite original data at each step?

```
# all tidyverse functions: first argument is the DATASET

# MUTATE
# first, find the number of children (total minus adults)
# then, find the proportion (children / total)
midwest <- mutate(midwest,
                  popchildren = poptotal - popadults,
                  pr_children = popchildren / poptotal)

# group by state
midwest <- group_by(midwest, state)

# mean pr_children by state
summarize(midwest, mean_pr_children = mean(pr_children))
```

```
## # A tibble: 5 x 2
##   state mean_pr_children
##   <chr>            <dbl>
## 1 IL               0.350
## 2 IN               0.366
## 3 MI               0.359
## 4 OH               0.368
## 5 WI               0.359
```

We want to:

- Calculate the proportion of the population that is *children* (for every county): mutate()
- Then, calculate the mean proportion of children in every state: group_by() and summarize()

Create intermediate objects at each step?

```
# create a new object,
#   which gets results from mutate()
midwest_mut <- mutate(midwest,
                      popchildren = poptotal - popadults,
                      pr_children = popchildren / poptotal)

# group the NEW OBJECT by state
midwest_grp <- group_by(midwest_mut, state)

# Summarize the GROUPED OBJECT
summarize(midwest_grp, mean_pr_children = mean(pr_children))
```

```
## # A tibble: 5 x 2
##   state mean_pr_children
##   <chr>            <dbl>
## 1 IL               0.350
## 2 IN               0.366
## 3 MI               0.359
## 4 OH               0.368
## 5 WI               0.359
```

We want to:

- Calculate the proportion of the population that is *children* (for every county): mutate()
- Then, calculate the mean proportion of children in every state: group_by() and summarize()

Use the order of operations?

```
# every f() creates a new data frame
# so pass results to next function: f(g(h(x)))

# FROM THE INSIDE OUT: mutate() the midwest data
# Then, mutate() result is the data for group_by()
# Then, group_by() result is the data for summarize()

summarize(
  group_by(
    mutate(midwest,
           popchildren = poptotal - popadults,
           pr_children = popchildren / poptotal),
    state),
  mean_pr_children = mean(pr_children)
)
```

```
## # A tibble: 5 x 2
##   state mean_pr_children
##   <chr>            <dbl>
## 1 IL               0.350
## 2 IN               0.366
## 3 MI               0.359
## 4 OH               0.368
## 5 WI               0.359
```

There's a better way

Linear data processing plan:

- Start with the data, *then*...
- Create new variables, *then*...
- Group the data by state, *then*...
- Calculate the mean in each state

There's a better way

Linear data processing plan:

- Start with the data, *then*...
- Create new variables, *then*...
- Group the data by state, *then*...
- Calculate the mean in each state

Linear code:

```
# enter... the pipe operator %>%
# read it as "and then..."
midwest %>%
  mutate(popchildren = poptotal - popadults,
         pr_children = popchildren / poptotal) %>%
  group_by(state) %>%
  summarize(mean_pr_children = mean(pr_children))
```

```
## # A tibble: 5 x 2
##   state mean_pr_children
##   <chr>            <dbl>
## 1 IL               0.350
## 2 IN               0.366
## 3 MI               0.359
## 4 OH               0.368
## 5 WI               0.359
```

How it works

```
# Typical R code: f(data)
dim(midwest)
```

```
## [1] 437  30
```

How it works

```
# Typical R code: f(data)
dim(midwest)
```

```
## [1] 437  30
```

```
# with the pipe: data %>% f()
#   data serves as first arg in f()
#   (implicitly)
midwest %>% dim()
```

```
## [1] 437  30
```

How it works

```
# Typical R code: f(data)
dim(midwest)
```

```
## [1] 437  30
```

```
# with the pipe: data %>% f()
#   data serves as first arg in f()
#   (implicitly)
midwest %>% dim()
```

```
## [1] 437  30
```

Summon the pipe

Macs: `Cmd + Shift + m`

Windows: `Ctrl + Shift + m`

Do lots of things easily

```
# Typical R code: f(g(x))
#   g() is first, then f()
length(names(midwest))
```

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

Do lots of things easily

```
# Typical R code: f(g(x))
#   g() is first, then f()
length(names(midwest))
```

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

```
# with the pipe: data %>% g() %>% f()
# just as it should be
midwest %>% names() %>% length()
```

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

```
# or, break across lines for readability
midwest %>%
  names() %>%
  length()
```

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

Why we like the pipe operator

Series of operations, without piping

$$f(g(h(x)))$$

Works inside-out

Counter-intuitive

Difficult to write

Difficult to read

With piping

$$x \rightarrow h() \rightarrow g() \rightarrow f()$$

Works linearly

Intuitive

Easy to write

Easy to read

Tidyverse flow

Let's write our own chain

1. Start with `midwest`
2. For each state, count metro/non-metro area counties (`inmetro`)

Let's do another

1. Start with `gapminder`
2. For each year, keep the country with the highest GDP per capita
3. sort by year

Printing the results of a chain

```
# results don't print when you make new object
just_WI <- midwest %>%
  filter(state == "WI")

# But they do with print()
just_WI <- midwest %>%
  filter(state == "WI") %>%
  print()
```

```
## # A tibble: 72 x 30
## # Groups:   state [1]
##      PID county state  area poptotal popdensity popwhite popblack
##    <int> <chr>  <chr> <dbl>    <int>      <dbl>    <int>    <int>
##  1  2981 ADAMS  WI    0.041    15682       382.    15001      375
##  2  2982 ASHLA… WI    0.054    16307       302.    14749       17
##  3  2983 BARRON WI    0.053    40750       769.    40346       40
##  4  2984 BAYFI… WI    0.089    14008       157.    12707       29
##  5  2985 BROWN  WI    0.032   194594      6081.   186621     1012
##  6  2986 BUFFA… WI    0.04     13584       340.    13521        5
##  7  2987 BURNE… WI    0.053    13084       247.    12497       22
##  8  2988 CALUM… WI    0.023    34291      1491.    33910       29
##  9  2989 CHIPP… WI    0.063    52360       831.    51854       31
## 10  2990 CLARK  WI    0.072    31647       440.    31437       29
```

When data aren't the first argument?

```
# linear relationship between poverty (y) and education (x)
lm(percelderlypoverty ~ percollege, data = midwest)
```

```
##
## Call:
## lm(formula = percelderlypoverty ~ percollege, data = midwest)
##
## Coefficients:
## (Intercept)   percollege
##     17.0055      -0.3074
```

When data aren't the first argument?

```
# linear relationship between poverty (y) and education (x)
lm(percelderlypoverty ~ percollege, data = midwest)
```

```
##
## Call:
## lm(formula = percelderlypoverty ~ percollege, data = midwest)
##
## Coefficients:
## (Intercept)   percollege
##     17.0055      -0.3074
```

```
# use a period to stand in for the piped data
midwest %>%
  lm(percelderlypoverty ~ percollege, data = .)
```

```
##
## Call:
## lm(formula = percelderlypoverty ~ percollege, data = .)
##
## Coefficients:
## (Intercept)   percollege
##     17.0055      -0.3074
```

Tips for pipes

- Not necessary if just one function
- Break multiple functions across lines
- Indent functions (two spaces)
- Write full chain *before* creating new object
- Test chain line-by-line while writing
- Tidyverse flipbook by Gina Reynolds
- Need variables out of a data frame? Check out `pull()` or `%$%` from the `magrittr` package

Enjoy your new powers

Practice on your first exercise!

Exercises due *before class* Wednesday. Late work NOT ACCEPTED!