Lecture 15: Better Data Wrangling with `dplyr`

ECON 480 - Econometrics - Fall 2018

Ryan Safner

November 14, 2018

Some Meta Thoughts About Code

- There is a tradeoff between **precision** and **concision** in coding:

HOOD COLLEGE

Some Meta Thoughts About Code

- There is a tradeoff between **precision** and **concision** in coding:
  - `subset(diamonds, x == 0 & y == 0 & z == 0)`

HOOD COLLEGE

Some Meta Thoughts About Code

- There is a tradeoff between **precision** and **concision** in coding:
  - `subset(diamonds, x == 0 & y == 0 & z == 0)`
  - vs.

HOOD COLLEGE

Some Meta Thoughts About Code

- There is a tradeoff between **precision** and **concision** in coding:
  - `subset(diamonds, x == 0 & y == 0 & z == 0)`
  - vs.
  - `diamonds[diamonds$x == 0 & diamonds$y == 0 & diamonds$z == 0, ]`

HOOD COLLEGE

Some Meta Thoughts About Code

- There is a tradeoff between **precision** and **concision** in coding:
  - `subset(diamonds, x == 0 & y == 0 & z == 0)`
  - vs.
  - `diamonds[diamonds$x == 0 & diamonds$y == 0 & diamonds$z == 0, ]`
- It would be ideal for code to be “self-documenting” and easily readable to observers without excess explanation

HOOD COLLEGE

Some Meta Thoughts About Code II

- Compare the following commands, which both subset the `gapminder` data to look only at year and life expectancy for Cambodia

```
gapminder[gapminder$country=="Cambodia", c("year", "lifeExp0")]
```

```
gapminder %>%
  filter(country == "Cambodia") %>%
  select(year, lifeExp)
```

HOOD COLLEGE

Some Meta Thoughts About Code II

- Compare the following commands, which both subset the `gapminder` data to look only at year and life expectancy for Cambodia

```
gapminder[gapminder$country=="Cambodia", c("year", "lifeExp0")]
```

```
gapminder %>%
  filter(country == "Cambodia") %>%
  select(year, lifeExp)
```

- Which is more intuitive to read and understand what we're doing? (without comments!)

HOOD COLLEGE

Some Meta Thoughts About Code II

- Compare the following commands, which both subset the `gapminder` data to look only at year and life expectancy for Cambodia

```
gapminder[gapminder$country=="Cambodia", c("year", "lifeExp0")]
```

```
gapminder %>%
  filter(country == "Cambodia") %>%
  select(year, lifeExp)
```

- Which is more intuitive to read and understand what we're doing? (without comments!)
  - The first is using Base R, the second uses `dplyr`

HOOD COLLEGE

The Pipe Operator, %>%

- The “pipe” operator, **%>%** will change your coding life

HOOD COLLEGE

The Pipe Operator, %>%

- The “pipe” operator, **%>%** will change your coding life
- Keyboard shortcut in R Studio:

HOOD COLLEGE

The Pipe Operator, %>%

- The “pipe” operator, **%>%** will change your coding life
- Keyboard shortcut in R Studio:
    - `CTRL+Shift+M` (Windows)

HOOD COLLEGE

The Pipe Operator, %>%

- The “pipe” operator, **%>%** will change your coding life
- Keyboard shortcut in R Studio:
  - `CTRL+Shift+M` (Windows)
  - `Cmd+Shift+M` (Mac)

HOOD COLLEGE

The Pipe Operator, %>%

- The “pipe” operator, **%>%** will change your coding life
- Keyboard shortcut in R Studio:
  - `CTRL+Shift+M` (Windows)
  - `Cmd+Shift+M` (Mac)
- **%>%** “pipes” the *output* of everything to the *left* of the pipe into the *input* on right

HOOD COLLEGE

The Pipe Operator, %>%

- The "pipe" operator, `%>%` will change your coding life
- Keyboard shortcut in R Studio:
  - `CTRL+Shift+M` (Windows)
  - `Cmd+Shift+M` (Mac)
- `%>%` "pipes" the *output* of everything to the *left* of the pipe into the *input* on right
- Running some function `f` on object `x` as `f(x)` can be "piped" as `x %>% f`

HOOD COLLEGE

The Pipe Operator, %>%

- The "pipe" operator, `%>%` will change your coding life
- Keyboard shortcut in R Studio:
  - `CTRL+Shift+M` (Windows)
  - `Cmd+Shift+M` (Mac)
- `%>%` "pipes" the *output* of everything to the *left* of the pipe into the *input* on right
- Running some function `f` on object `x` as `f(x)` can be "piped" as `x %>% f`
  - i.e. "take `x` and then perform function `f` on it"

HOOD COLLEGE

The Pipe Operator, %>% II

- With ordinary math functions, we read operations from outside⟵(inside):

$$g(f(x))$$

  i.e. take `x` and then perform function `f` on `x`, then perform function `g` on that result

HOOD COLLEGE

The Pipe Operator, %>% II

- With ordinary math functions, we read operations from outside⟵(inside):

$$g(f(x))$$

  i.e. take `x` and then perform function `f` on `x`, then perform function `g` on that result

- With pipes, we read operations from left⟶ right:

HOOD COLLEGE

The Pipe Operator, %>% II

- With ordinary math functions, we read operations from outside⟵(inside):

$$g(f(x))$$

  i.e. take `x` and then perform function `f` on `x`, then perform function `g` on that result

- With pipes, we read operations from left⟶ right:

```
x %>% f %>% g
```

take `x` and then perform function `f` on it, then perform function `g` on that result

HOOD COLLEGE

The Pipe Operator, %>% II

- With ordinary math functions, we read operations from outside⟵(inside):

$$g(f(x))$$

  i.e. take `x` and then perform function `f` on `x`, then perform function `g` on that result

- With pipes, we read operations from left⟶ right:

```
x %>% f %>% g
```

take `x` and then perform function `f` on it, then perform function `g` on that result

- So read `%>%` mentally as “and then”

HOOD COLLEGE

The Pipe Operator, %>% Example

```
library("gapminder") # load gapminder package for gapminder dataset

head(gapminder) # look at top 6 rows
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

HOOD COLLEGE

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

- Core packages include ones we've discussed before: `ggplot2`, `dplyr`, `magrittr` among several others (`tidy readr`, `purr`, `forcats`, `stringr`)

HOOD COLLEGE

The tidyverse

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

- Core packages include ones we've discussed before: `ggplot2`, `dplyr`, `magrittr` among several others (`tidy readr`, `purr`, `forcats`, `stringr`)
- Loading any `tidyverse` package loads `magrittr` (so you can use `%>%`)

The tidyverse

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

- Core packages include ones we've discussed before: `ggplot2`, `dplyr`, `magrittr` among several others (`tidy readr`, `purr`, `forcats`, `stringr`)
- Loading any `tidyverse` package loads `magrittr` (so you can use `%>%`)
- Learn more at tidyverse.org

The tidyverse II

- Easiest to just load the core tidyverse all at once

```
library("tidyverse")
```

The tidyverse II

- Easiest to just load the core tidyverse all at once
    - Note loading the tidyverse is “noisy”, it will spew a lot of messages

```
library("tidyverse")
```

HOOD COLLEGE

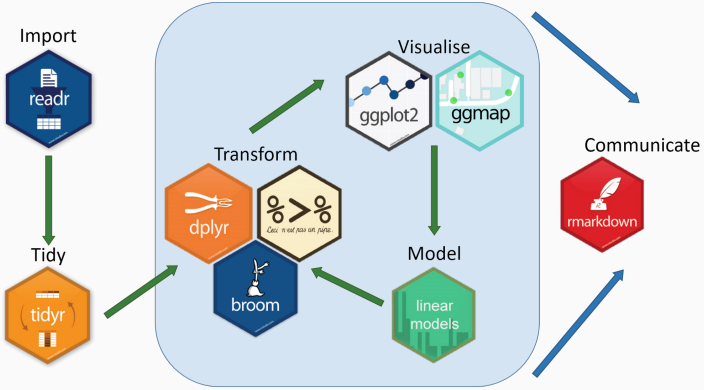
The tidyverse II

- Easiest to just load the core tidyverse all at once
  - Note loading the tidyverse is "noisy", it will spew a lot of messages
  - Remember you can hide these by setting `R` chunks (if using `R Markdown`) with `message=FALSE` and `warning=FALSE`

```
library("tidyverse")
```

HOOD COLLEGE

The tidyverse III



dplyr Verbs

- Base R is about running functions on nouns, e.g. `function(object)`

HOOD COLLEGE

dplyr Verbs

- Base R is about running functions on nouns, e.g. `function(object)`
- `dplyr` is all about using active English-language verbs to accomplish tasks

HOOD COLLEGE

dplyr Verbs

- Base R is about running functions on nouns, e.g. `function(object)`
- `dplyr` is all about using active English-language verbs to accomplish tasks

Function	Does
`filter`	Keep only selected *observations*
`select`	Keep only selected *variables*
`arrange`	Reorder rows (e.g. in numerical order)
`mutate`	Create new variables
`recode`	Change a variable's values or categories/`factor` levels
`summarize`	Collapse data into summary statistics
`group_by`	Perform any of the above functions by groups/categories

HOOD COLLEGE

dplyr Verbs II

- Syntax of any `dplyr` function is the same: `dyplrfunction(dataframe, condition)`, which returns a `data.frame`

HOOD COLLEGE

dplyr Verbs II

- Syntax of any `dplyr` function is the same: `dyplrfunction(dataframe, condition)`, which returns a `data.frame`
    - Or if you prefer to try out the pipe `%>%`:

```
dataframe %>%
  dplyrfunction(condition)
```

HOOD COLLEGE

filter

- `filter` keeps only selected **observations**

HOOD COLLEGE

filter

- `filter` keeps only selected **observations**

```
# look only at African observations
gapminder %>%
  filter(continent==”Africa”)
```

```
## # A tibble: 624 x 6
##    country continent  year lifeExp      pop gdpPercap
##    <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Algeria Africa     1952    43.1  9279525     2449.
##  2 Algeria Africa     1957    45.7 10270856     3014.
##  3 Algeria Africa     1962    48.3 11000948     2551.
##  4 Algeria Africa     1967    51.4 12760499     3247.
##  5 Algeria Africa     1972    54.5 14760787     4183.
##  6 Algeria Africa     1977    58.0 17152804     4910.
##  7 Algeria Africa     1982    61.4 20033753     5745.
##  8 Algeria Africa     1987    65.8 23254956     5681.
##  9 Algeria Africa     1992    67.7 26298373     5023.
## 10 Algeria Africa     1997    69.2 29072015     4797.
## # ... with 614 more rows
```

HOOD COLLEGE

`dplyr` Allows You to Test Without Overwriting

- Great thing about `dplyr` is we don't necessarily need to store our results as objects until we're ready

HOOD COLLEGE

`dplyr` Allows You to Test Without Overwriting

- Great thing about `dplyr` is we don't necessarily need to store our results as objects until we're ready
    - Won't overwrite an object inccorectly, e.g.:

HOOD COLLEGE

`dplyr` Allows You to Test Without Overwriting

- Great thing about `dplyr` is we don't necessarily need to store our results as objects until we're ready
  - Won't overwrite an object inccorectly, e.g.:
  - `gapminder %>% select(country == "United States")` does not overwrite `gapminder`.

HOOD COLLEGE

`dplyr` Allows You to Test Without Overwriting

- Great thing about `dplyr` is we don't necessarily need to store our results as objects until we're ready
  - Won't overwrite an object inccorectly, e.g.:
  - `gapminder %>% select(country == "United States")` does not overwrite `gapminder`.
- You can still deliberately save (and overwrite) objects with the assignment operator:

HOOD COLLEGE

`dplyr` Allows You to Test Without Overwriting

- Great thing about `dplyr` is we don't necessarily need to store our results as objects until we're ready
  - Won't overwrite an object inccorectly, e.g.:
  - `gapminder %>% select(country == "United States")` does not overwrite `gapminder`.
- You can still deliberately save (and overwrite) objects with the assignment operator:
  - `gapminder <- gapminder %>% select(country == "United States")` *would* overwrite gapminder with *just* the U.S. observations

HOOD COLLEGE

`filter` Multiple AND Conditions

- `filter` multiple conditions with commas (implicitly, having multiple “AND” conditions)

HOOD COLLEGE

`filter` Multiple AND Conditions

- `filter` multiple conditions with commas (implicitly, having multiple "AND" conditions)

```
# look only at observations that are in Europe AND in 1997
gapminder %>%
  filter(continent=="Europe", year==1997)
```

```
## # A tibble: 30 x 6
##    country                continent  year lifeExp      pop gdpPercap
##    <fct>                  <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Albania                Europe     1997    73.0  3428038     3193.
##  2 Austria                Europe     1997    77.5  8069876    29096.
##  3 Belgium                Europe     1997    77.5 10199787    27561.
##  4 Bosnia and Herzegovina Europe     1997    73.2  3607000     4766.
##  5 Bulgaria               Europe     1997    70.3  8066057     5970.
##  6 Croatia                Europe     1997    73.7  4444595     9876.
##  7 Czech Republic         Europe     1997    74.0 10300707    16049.
##  8 Denmark                Europe     1997    76.1  5283663    29804.
##  9 Finland                Europe     1997    77.1  5134406    23724.
## 10 France                 Europe     1997    78.6 58623428    25890.
## # ... with 20 more rows
```

HOOD COLLEGE

`filter` OR Conditions

- `filter` multiple alternative conditions with | (“OR”)

HOOD COLLEGE

`filter` OR Conditions

- `filter` multiple alternative conditions with | ("OR")

```
# look only at observations that are in Europe OR in 1997
gapminder %>%
  filter(continent=="Europe" | year==1997)
```

```
## # A tibble: 472 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1997    41.8 22227415      635.
##  2 Albania     Europe     1952    55.2  1282697     1601.
##  3 Albania     Europe     1957    59.3  1476505     1942.
##  4 Albania     Europe     1962    64.8  1728137     2313.
##  5 Albania     Europe     1967    66.2  1984060     2760.
##  6 Albania     Europe     1972    67.7  2263554     3313.
##  7 Albania     Europe     1977    68.9  2509048     3533.
##  8 Albania     Europe     1982    70.4  2780097     3631.
##  9 Albania     Europe     1987    72    3075321     3739.
## 10 Albania     Europe     1992    71.6  3326498     2497.
## # ... with 462 more rows
```

HOOD COLLEGE

`filter` Other Useful Operators

- We can `filter` by membership `%in%` a particular set (represented by a vector)

HOOD COLLEGE

filter Other Useful Operators

- We can `filter` by membership `%in%` a particular set (represented by a vector)

```
# look only at observations that are in the "set" of (Europe, Africa, Asia)
gapminder %>%
  filter(continent %in% c("Europe", "Africa", "Asia"))
```

```
## # A tibble: 1,380 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## #     with 1,370 more rows
```

HOOD COLLEGE

select

- `select` keeps only selected **variables**

select

- `select` keeps only selected **variables**

```
# Only keep country, year, and population variables
gapminder %>%
  select(country, year, pop)
```

```
## # A tibble: 1,704 x 3
##    country      year      pop
##    <fct>       <int>    <int>
##  1 Afghanistan  1952  8425333
##  2 Afghanistan  1957  9240934
##  3 Afghanistan  1962 10267083
##  4 Afghanistan  1967 11537966
##  5 Afghanistan  1972 13079460
##  6 Afghanistan  1977 14880372
##  7 Afghanistan  1982 12881816
##  8 Afghanistan  1987 13867957
##  9 Afghanistan  1992 16317921
## 10 Afghanistan  1997 22227415
## # ... with 1,694 more rows
```

HOOD COLLEGE

`select` Helper Functions

- `select` has a lot of nice helper functions

HOOD COLLEGE

`select` Helper Functions

- `select` has a lot of nice helper functions
- Type `?select` to get more information

HOOD COLLEGE

`select` Helper Functions

- `select` has a lot of nice helper functions
- Type `?select` to get more information
- Some examples (where `string` is some text that you are searching for):

Function	Description
`starts_with("string")`	Variable name begins with `string`
`ends_with("string")`	Variable name ends with `string`
`contains("string")`	Variable name contains `string` in it

HOOD COLLEGE

select Example

select Example

```
# Only keep gdpPercap and othervariables that start with "c"
gapminder %>%
  select(gdpPercap,starts_with("c"))
```

```
## # A tibble: 1,704 x 3
##    gdpPercap country     continent
##        <dbl> <fct>       <fct>
##  1      779. Afghanistan Asia
##  2      821. Afghanistan Asia
##  3      853. Afghanistan Asia
##  4      836. Afghanistan Asia
##  5      740. Afghanistan Asia
##  6      786. Afghanistan Asia
##  7      978. Afghanistan Asia
##  8      852. Afghanistan Asia
##  9      649. Afghanistan Asia
## 10      635. Afghanistan Asia
## # ... with 1,694 more rows
```

HOOD COLLEGE

`select` NEG

- `select` allows you to “negate” columns with a negative sign (-)

HOOD COLLEGE

`select` NEG

- `select` allows you to “negate” columns with a negative sign (-)
  - A way of keeping “everything but” certain variables

HOOD COLLEGE

`select` NEG

- `select` allows you to “negate” columns with a negative sign (-)
  - A way of keeping “everything but” certain variables

HOOD COLLEGE

select NEG

- `select` allows you to "negate" columns with a negative sign (-)
  - A way of keeping "everything but" certain variables

```
# keep all variables EXCEPT pop
gapminder %>%
  select(-pop)
```

```
## # A tibble: 1,704 x 5
##    country     continent  year lifeExp gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <dbl>
##  1 Afghanistan Asia       1952    28.8      779.
##  2 Afghanistan Asia       1957    30.3      821.
##  3 Afghanistan Asia       1962    32.0      853.
##  4 Afghanistan Asia       1967    34.0      836.
##  5 Afghanistan Asia       1972    36.1      740.
##  6 Afghanistan Asia       1977    38.4      786.
##  7 Afghanistan Asia       1982    39.9      978.
##  8 Afghanistan Asia       1987    40.8      852.
##  9 Afghanistan Asia       1992    41.7      649.
```

HOOD COLLEGE

rename

- `rename` changes the name of a variable in the following format: `newname=oldname`

HOOD COLLEGE

rename

- `rename` changes the name of a variable in the following format: `newname=oldname`

```
# Rename gdpPercap to just GDP
gapminder %>%
  rename(GDP=gdpPercap)
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop   GDP
##    <fct>       <fct>     <int>   <dbl>    <int> <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333  779.
##  2 Afghanistan Asia       1957    30.3  9240934  821.
##  3 Afghanistan Asia       1962    32.0 10267083  853.
##  4 Afghanistan Asia       1967    34.0 11537966  836.
##  5 Afghanistan Asia       1972    36.1 13079460  740.
##  6 Afghanistan Asia       1977    38.4 14880372  786.
##  7 Afghanistan Asia       1982    39.9 12881816  978.
##  8 Afghanistan Asia       1987    40.8 13867957  852.
##  9 Afghanistan Asia       1992    41.7 16317921  649.
## 10 Afghanistan Asia       1997    41.8 22227415  635.
## # ... with 1,694 more rows
```

HOOD COLLEGE

arrange

- `arrange` orders the observations (rows) in some logical order

HOOD COLLEGE

arrange

- `arrange` orders the observations (rows) in some logical order
  - e.g. (reverse) alphabetical, (reverse) numerical, largest to smallest (smallest to largest)

HOOD COLLEGE

arrange

- `arrange` orders the observations (rows) in some logical order
  - e.g. (reverse) alphabetical, (reverse) numerical, largest to smallest (smallest to largest)

```
# Sort by lifeExp
gapminder %>%
  arrange(lifeExp)
```

```
## # A tibble: 1,704 x 6
##    country      continent  year lifeExp     pop gdpPercap
##    <fct>        <fct>     <int>   <dbl>   <int>     <dbl>
##  1 Rwanda       Africa     1992    23.6 7290203      737.
##  2 Afghanistan  Asia       1952    28.8 8425333      779.
##  3 Gambia       Africa     1952    30    284320      485.
##  4 Angola       Africa     1952    30.0 4232095     3521.
##  5 Sierra Leone Africa     1952    30.3 2143249      880.
##  6 Afghanistan  Asia       1957    30.3 9240934      821.
##  7 Cambodia     Asia       1977    31.2 6978607      525.
##  8 Mozambique   Africa     1952    31.3 6446316      469.
##  9 Sierra Leone Africa     1957    31.6 2295678     1004.
```

HOOD COLLEGE

arrange

- use `desc()` for descending order

HOOD COLLEGE

arrange

- use `desc()` for descending order

```
# Sort by country name (reverse alphabetically)
gapminder %>%
  arrange(desc(country))
```

```
## # A tibble: 1,704 x 6
##    country  continent  year lifeExp      pop gdpPercap
##    <fct>    <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Zimbabwe Africa     1952    48.5  3080907      407.
##  2 Zimbabwe Africa     1957    50.5  3646340      519.
##  3 Zimbabwe Africa     1962    52.4  4277736      527.
##  4 Zimbabwe Africa     1967    54.0  4995432      570.
##  5 Zimbabwe Africa     1972    55.6  5861135      799.
##  6 Zimbabwe Africa     1977    57.7  6642107      686.
##  7 Zimbabwe Africa     1982    60.4  7636524      789.
##  8 Zimbabwe Africa     1987    62.4  9216418      706.
##  9 Zimbabwe Africa     1992    60.4 10704340      693.
## 10 Zimbabwe Africa     1997    46.8 11404948      792.
## #     with 1,694 more rows
```

HOOD COLLEGE

mutate

- `mutate` creates a new variable according to some operation on other variables

HOOD COLLEGE

mutate

- `mutate` creates a new variable according to some operation on other variables
  - syntax: `new.variable.name=operation`

HOOD COLLEGE

mutate

- `mutate` creates a new variable according to some operation on other variables
  - syntax: `new.variable.name=operation`

```
# make a GDP variable by multiplying gdpPercap and pop
gapminder %>%
  mutate(gdp= gdpPercap * pop)
```

```
## # A tibble: 1,704 x 7
##    country     continent  year lifeExp      pop gdpPercap          gdp
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>        <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.  6567086330.
##  2 Afghanistan Asia       1957    30.3  9240934      821.  7585448670.
##  3 Afghanistan Asia       1962    32.0 10267083      853.  8758855797.
##  4 Afghanistan Asia       1967    34.0 11537966      836.  9648014150.
##  5 Afghanistan Asia       1972    36.1 13079460      740.  9678553274.
##  6 Afghanistan Asia       1977    38.4 14880372      786. 11697659231.
##  7 Afghanistan Asia       1982    39.9 12881816      978. 12598563401.
##  8 Afghanistan Asia       1987    40.8 13867957      852. 11820990309.
##  9 Afghanistan Asia       1992    41.7 16317921      649. 10595901589.
```

HOOD COLLEGE

`mutate` Multiple Variables at Once

- Can create multiple new variables with the same command using commas

HOOD COLLEGE

mutate Multiple Variables at Once

- Can create multiple new variables with the same command using commas

```
gapminder %>%
  mutate(gdp= gdpPercap * pop,
         gdp.billions=gdp/1000000000)
```

```
## # A tibble: 1,704 x 8
##    country  continent  year lifeExp     pop gdpPercap     gdp gdp.billions
##    <fct>    <fct>     <int>   <dbl>   <int>     <dbl>   <dbl>        <dbl>
##  1 Afghani~ Asia       1952    28.8  8.43e6      779. 6.57e 9         6.57
##  2 Afghani~ Asia       1957    30.3  9.24e6      821. 7.59e 9         7.59
##  3 Afghani~ Asia       1962    32.0  1.03e7      853. 8.76e 9         8.76
##  4 Afghani~ Asia       1967    34.0  1.15e7      836. 9.65e 9         9.65
##  5 Afghani~ Asia       1972    36.1  1.31e7      740. 9.68e 9         9.68
##  6 Afghani~ Asia       1977    38.4  1.49e7      786. 1.17e10        11.7
##  7 Afghani~ Asia       1982    39.9  1.29e7      978. 1.26e10        12.6
##  8 Afghani~ Asia       1987    40.8  1.39e7      852. 1.18e10        11.8
##  9 Afghani~ Asia       1992    41.7  1.63e7      649. 1.06e10        10.6
## 10 Afghani~ Asia       1997    41.8  2.22e7      635. 1.41e10        14.1
## # ... with 1,694 more rows
```

HOOD COLLEGE

summarize

- `summarize`[1] calculates desired summary statistics of a variable

[1] Also the non-U.S. English spelling ‘summarise‘ works. ‘dplyr‘ was written by a Kiwi after all!

HOOD COLLEGE

summarize

- `summarize`[1] calculates desired summary statistics of a variable
  - common summary statistics: `n()`, `mean()`, `sd()`, `min()`, `max()`

[1] Also the non-U.S. English spelling 'summarise' works. 'dplyr' was written by a Kiwi after all!

summarize

- `summarize`[1] calculates desired summary statistics of a variable
  - common summary statistics: `n()`, `mean()`, `sd()`, `min()`, `max()`
  - `summarize` outputs a new `data.frame`

[1] Also the non-U.S. English spelling 'summarise' works. 'dplyr' was written by a Kiwi after all!

HOOD COLLEGE

summarize

- `summarize`[1] calculates desired summary statistics of a variable
  - common summary statistics: `n()`, `mean()`, `sd()`, `min()`, `max()`
  - `summarize` outputs a new `data.frame`
  - Can give a name to the summary variable as if you are `mutate`-ing a new variable

[1] Also the non-U.S. English spelling 'summarise' works. 'dplyr' was written by a Kiwi after all!

HOOD COLLEGE

summarize

- `summarize`[1] calculates desired summary statistics of a variable
  - common summary statistics: `n()`, `mean()`, `sd()`, `min()`, `max()`
  - `summarize` outputs a new `data.frame`
  - Can give a name to the summary variable as if you are `mutate`-ing a new variable

```
# get average life expectancy
gapminder %>%
  summarize(avg.LE=mean(lifeExp))
```

```
## # A tibble: 1 x 1
##   avg.LE
##    <dbl>
## 1   59.5
```

[1] Also the non-U.S. English spelling 'summarise' works. 'dplyr' was written by a Kiwi after all!

HOOD COLLEGE

summarize II

- Can `summarize` multiple variables at once

HOOD COLLEGE

summarize II

- Can `summarize` multiple variables at once

```
# get average life expectancy, gdp per capita, and population
gapminder %>%
  summarize(avg.LE=mean(lifeExp),
            avg.GDP=mean(gdpPercap),
            abg.pop=mean(pop))
```

```
## # A tibble: 1 x 3
##   avg.LE avg.GDP   abg.pop
##    <dbl>   <dbl>     <dbl>
## 1   59.5   7215. 29601212.
```

HOOD COLLEGE

summarize Other Statistics

```
# get count, mean, sd, min, and max life expectancy
gapminder %>%
  summarize(count.LE=n(),
            avg.LE=mean(lifeExp),
            sd.LE=sd(lifeExp),
            min.LE=min(lifeExp),
            max.LE=max(lifeExp))
```

```
## # A tibble: 1 x 5
##   count.LE avg.LE sd.LE min.LE max.LE
##      <int>  <dbl> <dbl>  <dbl>  <dbl>
## 1     1704   59.5  12.9   23.6   82.6
```

HOOD COLLEGE

group_by

- If we have `factor` variables (such as `continent`), we can run all of our `dplyr` verb commands by group

HOOD COLLEGE

group_by

- If we have `factor` variables (such as `continent`), we can run all of our `dplyr` verb commands by group
- First we define the groups as the `continent`

HOOD COLLEGE

group_by

- If we have `factor` variables (such as `continent`), we can run all of our `dplyr` verb commands by group
- First we define the groups as the `continent`

```
gapminder %>%
  group_by(continent) %>% # first group by continent
  summarize(mean_life=mean(lifeExp), # get averages for life exp
            mean_gdp=mean(gdpPercap), # and gdp per capita
            mean_pop=mean(pop)) # and population by continent
```

```
## # A tibble: 5 x 4
##   continent mean_life mean_gdp  mean_pop
##   <fct>         <dbl>    <dbl>     <dbl>
## 1 Africa         48.9    2194.  9916003.
## 2 Americas       64.7    7136. 24504795.
## 3 Asia           60.1    7902. 77038722.
## 4 Europe         71.9   14469. 17169765.
## 5 Oceania        74.3   18622.  8874672.
```

HOOD COLLEGE

group_by Without the Pipe

- Since there are several steps going on here, let's think about what this would look like without the %>% operator:

group_by Without the Pipe

- Since there are several steps going on here, let's think about what this would look like without the %>% operator:

```
# first generate a variable for the groups
by_continent<-group_by(gapminder, continent)

# then summarize the variable
summarize(by_continent, mean_life=mean(lifeExp),
          mean_gdp=mean(gdpPercap),
          mean_pop=mean(pop))
```

```
## # A tibble: 5 x 4
##   continent mean_life mean_gdp  mean_pop
##   <fct>         <dbl>    <dbl>     <dbl>
## 1 Africa         48.9    2194.  9916003.
## 2 Americas       64.7    7136. 24504795.
## 3 Asia           60.1    7902. 77038722.
## 4 Europe         71.9   14469. 17169765.
## 5 Oceania        74.3   18622.  8874672.
```

HOOD COLLEGE

group_by Example II

group_by Example II

```
# track change in average Life Expectancy over time
gapminder %>%
  group_by(year) %>%
  summarize(mean.LE=mean(lifeExp),
            meean.GPD=mean(gdpPercap))
```

```
## # A tibble: 12 x 3
##     year mean.LE meean.GPD
##    <int>   <dbl>     <dbl>
##  1  1952    49.1     3725.
##  2  1957    51.5     4299.
##  3  1962    53.6     4726.
##  4  1967    55.7     5484.
##  5  1972    57.6     6770.
##  6  1977    59.6     7313.
##  7  1982    61.5     7519.
##  8  1987    63.2     7901.
##  9  1992    64.2     8159.
## 10  1997    65.0     9090.
## 11  2002    65.7     9918.
```

HOOD COLLEGE

group_by and summarize_at to Combine Statistics

- Use `summarize_at` to summarize multiple variables with multiple summary statistics

HOOD COLLEGE

group_by and summarize_at to Combine Statistics

- Use `summarize_at` to summarize multiple variables with multiple summary statistics
- Syntax: `summarize_at(vars(var1, var2), funs(stat1, stat2))` where `var1` and `var2` are your variables and `stat1` and `stat2` are the summary statistics you'd like (e.g. mean, median, etc)

HOOD COLLEGE

group_by and summarize_at to Combine Statistics

- Use `summarize_at` to summarize multiple variables with multiple summary statistics
- Syntax: `summarize_at(vars(var1, var2), funs(stat1, stat2))` where `var1` and `var2` are your variables and `stat1` and `stat2` are the summary statistics you'd like (e.g. mean, median, etc)

```
# get summary statistics (mean, median, sd) for lifeExp and gdpPercap over time
gapminder %>%
  group_by(year) %>%
  summarize_at(vars(lifeExp, gdpPercap), funs(mean, median, sd))
```

```
## # A tibble: 12 x 7
##     year lifeExp_mean gdpPercap_mean lifeExp_median gdpPercap_median
##    <int>        <dbl>          <dbl>          <dbl>            <dbl>
##  1  1952         49.1          3725.           45.1            1969.
##  2  1957         51.5          4299.           48.4            2173.
##  3  1962         53.6          4726.           50.9            2335.
##  4  1967         55.7          5484.           53.8            2678.
##  5  1972         57.6          6770.           56.5            3339.
```

HOOD COLLEGE

tally

- `tally` is shorthand for just getting the counts of observations by group (instead of `summarize` and `n()`)

HOOD COLLEGE

tally

- `tally` is shorthand for just getting the counts of observations by group (instead of `summarize` and `n()`)

```
gapminder %>%
  group_by(continent) %>%
  tally
```

```
## # A tibble: 5 x 2
##   continent     n
##   <fct>     <int>
## 1 Africa      624
## 2 Americas    300
## 3 Asia        396
## 4 Europe      360
## 5 Oceania      24
```

HOOD COLLEGE

Piping Across Packages

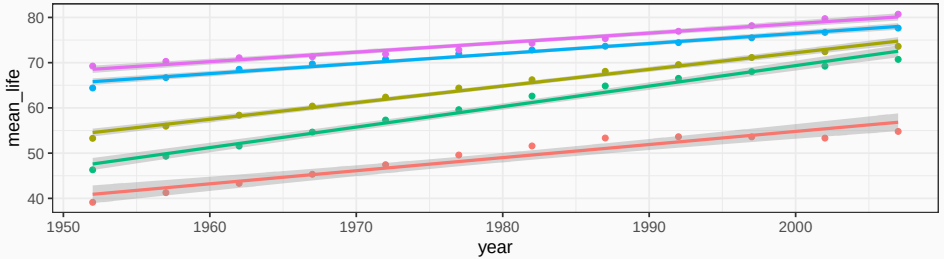
- The `tidyverse` uses the same grammar and design philosophy, so you can (almost always) pipe things across packages and functions

HOOD COLLEGE

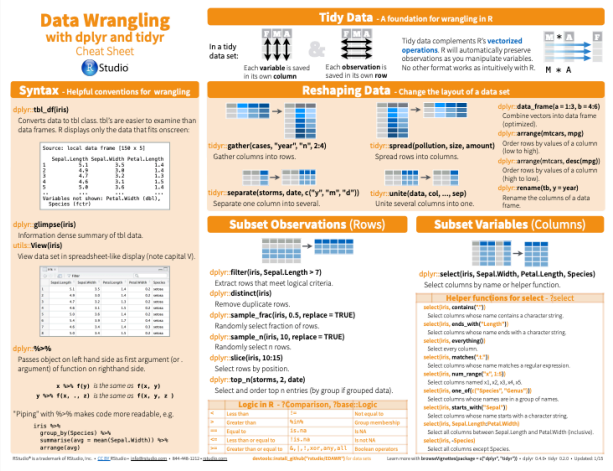
Piping Across Packages

- The `tidyverse` uses the same grammar and design philosophy, so you can (almost always) pipe things across packages and functions
- Example: graph the change in average life expectancy by continent over time

```
gapminder %>% # start with gapminder data
  group_by(year, continent) %>% # create groups of years and of continents
  summarize(mean_life = mean(lifeExp)) %>% # get average life expectancy for each group
  ggplot(aes(year, mean_life, color = continent))+ # plot this over time
  geom_point() + geom_smooth(method="lm")
```



As usual, there is a fantastic cheatsheet for `dplyr` via RStudio



RStudio's dplyr Cheatsheet