IR 412: `dplyr`

Suzie Mulesky

October 27, 2017

Introduction

Most of the time spent on data analysis actually relates to data cleaning/data management. In almost every case, we need to reshape our data, clean/prep our data, and aggregate variables to get our data ready for analysis.

We'll be learning how to clean, prep, manipulate, and aggregate datasets using the `dplyr` package.

Generally speaking, there are two steps we usually take to manipulate our data and get it ready for analysis:

1. **"Tidying" or reshaping our data.** This step shapes our data so that each row is a unit of observation and each column measures some characteristic about that observation.

- We won't be focusing on this step during this lecture because most of the datasets you will work with this semester are already "tidy."
- `tidyr` is the package you would learn to reshape your data.

2. **Transforming the data.** This step can include the process of creating new variables that are a function of existing variables, aggregation, and subsetting.

- We will focus today's lecture on this step.
- `dplyr` is the package you will learn to manipulate and transform your data.

We will be working with data from the World Bank's collection of development indicators. The data file is located in the IR 412 Dropbox folder under "data", called "wdi.RDATA". There are a few variables in the table, including information on GDP, population, government expeditures on education, and gender parity in primary schools.

```
# Load the dplyr package
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Load data from the Dropbox folder.
load("C:/Users/Suzie/Google Drive/SPEC Lab/Misc RA tasks/IR 412/wdi.RDATA")
```

Creating a "region" variable

Since most of you will need to aggregate data to the region level, I'll show you how to quickly create a region variable using the World Bank's regional classification. This step requires you to install and load a package called `countrycode`. The `countrycode` package was developed by a political scientist to standardize

country names and assign common unique country identifiers. You can reference the user manual online: https://cran.r-project.org/web/packages/countrycode/countrycode.pdf.

```
# install.package("countrycode")
library(countrycode)
```

```
## Warning: package 'countrycode' was built under R version 3.3.3
```

```
# Create a region variable that groups countries into the World Bank's
# regional classification scheme.
wdi$region = countrycode(sourcevar = wdi$country, # choose which variable you want to convert
                         origin = "country.name", # coding scheme of sourcevar
                         destination = "region") # coding scheme for new variable
```

```
## Warning in countrycode(sourcevar = wdi$country, origin = "country.name", : Some values were not matc
```

By default, `countrycode` warns you when observations do not receive a new value. You'll notice that many independent states were not classified, which probably means that the list of country names used by `countrycode` doesn't include such countries as Czechoslovakia, German Democratic Republic, Kosovo, etc.

You can manually classify these observations using the `ifelse` command. It works very similarly to the if-then notation in Excel.

```
# Tabulate by region to see all of the region classifications.
table(wdi$region)
```

```
##
## Australia and New Zealand                 Caribbean
##                       116                       754
##           Central America              Central Asia
##                       464                       290
##            Eastern Africa              Eastern Asia
##                       986                       406
##            Eastern Europe                 Melanesia
##                       550                       232
##                Micronesia             Middle Africa
##                       290                       464
##           Northern Africa           Northern America
##                       406                       116
##           Northern Europe                 Polynesia
##                       580                       174
##        South-Eastern Asia              South America
##                       580                       696
##           Southern Africa              Southern Asia
##                       290                       522
##           Southern Europe             Western Africa
##                       802                       928
##              Western Asia             Western Europe
##                       928                       522
```

```
# Czechoslovakia = Eastern Europe
wdi$region = ifelse(wdi$country == "Czechoslovakia", "Eastern Europe", wdi$region)

# German Democratic Republic = Eastern Europe
wdi$region = ifelse(wdi$country == "German Democratic Republic", "Eastern Europe",
                    wdi$region)

# Korea, People's Republic of = Eastern Asia
```

```r
wdi$region = ifelse(wdi$country == "Korea, People's Republic of", "Eastern Asia",
                    wdi$region)

# Kosovo = Southern Europe
wdi$region = ifelse(wdi$country == "Kosovo", "Southern Europe", wdi$region)

# Sao Tome and Principe = Middle Africa
wdi$region = ifelse(wdi$ccode == 403, "Middle Africa", wdi$region)

# Vietnam = South-Eastern Asia
wdi$region = ifelse(wdi$country == "Vietnam, Democratic Republic of",
                    "South-Eastern Asia", wdi$region)
wdi$region = ifelse(wdi$country == "Vietnam, Republic of",
                    "South-Eastern Asia", wdi$region)

# Yemen = Western Asia
wdi$region = ifelse(wdi$country == "Yemen (Arab Republic of Yemen)", "Western Asia",
                    wdi$region)
wdi$region = ifelse(wdi$country == "Yemen, People's Republic of", "Western Asia",
                    wdi$region)

# Yugoslovia = Southern Europe
wdi$region = ifelse(wdi$country == "Yugoslavia", "Southern Europe", wdi$region)

# Hong Kong = Eastern Asia
wdi$region = ifelse(wdi$country == "Hong Kong", "Eastern Asia", wdi$region)

# Macao = Eastern Asia
wdi$region = ifelse(wdi$country == "Macao", "Eastern Asia", wdi$region)
```

Five "verbs" of `dplyr`

1. `filter`: subsets observations
2. `select`: subsets variables
3. `arrange`: reorders observations
4. `mutate`: creates new variables
5. `summarise`: aggregates data

Filter

1. `>` greater than
2. `<` less than
3. `==` equal to
4. `>=` greater than or equal to
5. `<=` less than or equal to
6. `!=` not equal to
7. `x %in% c(a, b, c)`: TRUE if x is in the vector (a,b,c).
8. `is.na()`: missing
9. `!is.na()`: is not missing
10. `&, |, !`, boolean operators

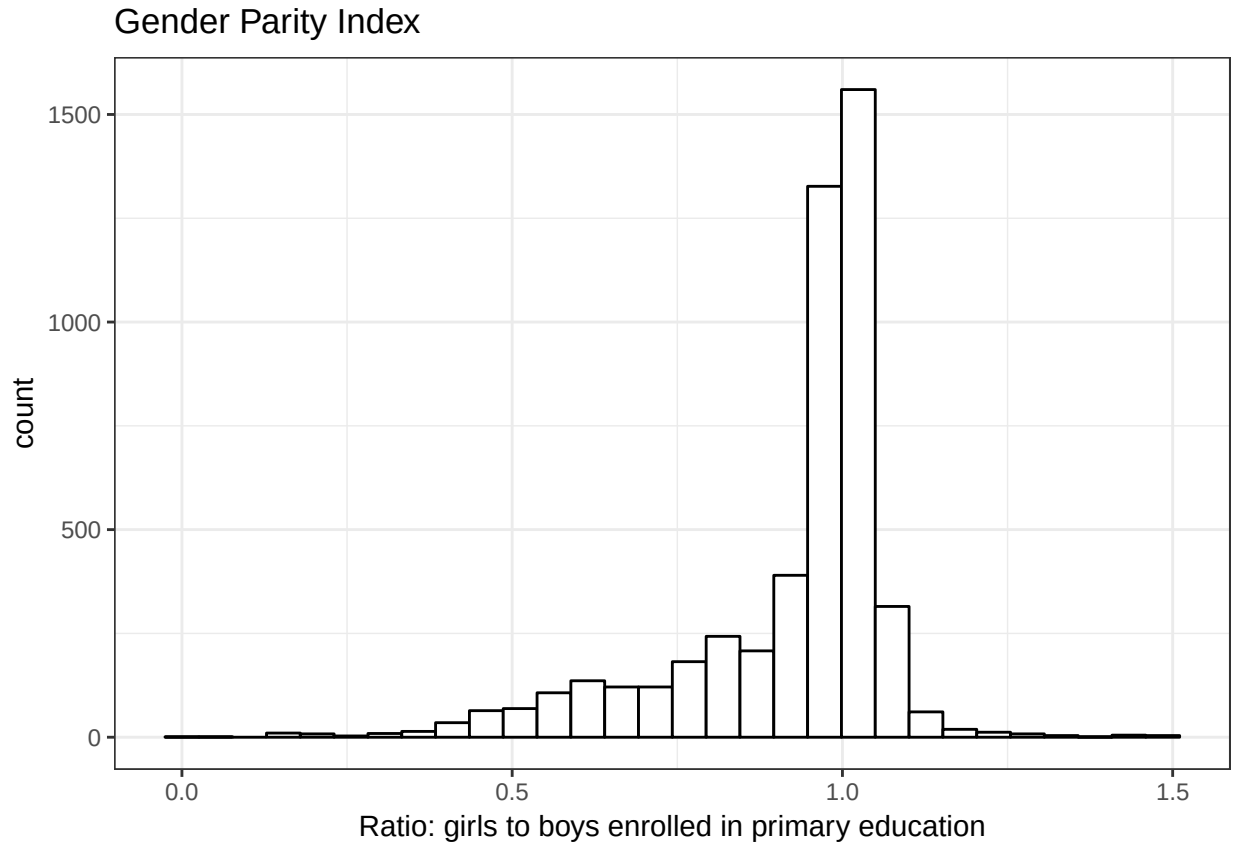
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
# Plot the distribution of gender parity in primary education
ggplot(data = wdi, aes(gender_parity_WDI)) +
  geom_histogram(fill = "white", color = "black") +
  ggtitle("Gender Parity Index") +
  labs(x = "Ratio: girls to boys enrolled in primary education") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 6720 rows containing non-finite values (stat_bin).
```



Compare the distribution of gender parity "then" and "now."

```
# Filter observations before 1980
gender70 = filter(wdi, year < 1980)
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.3
```

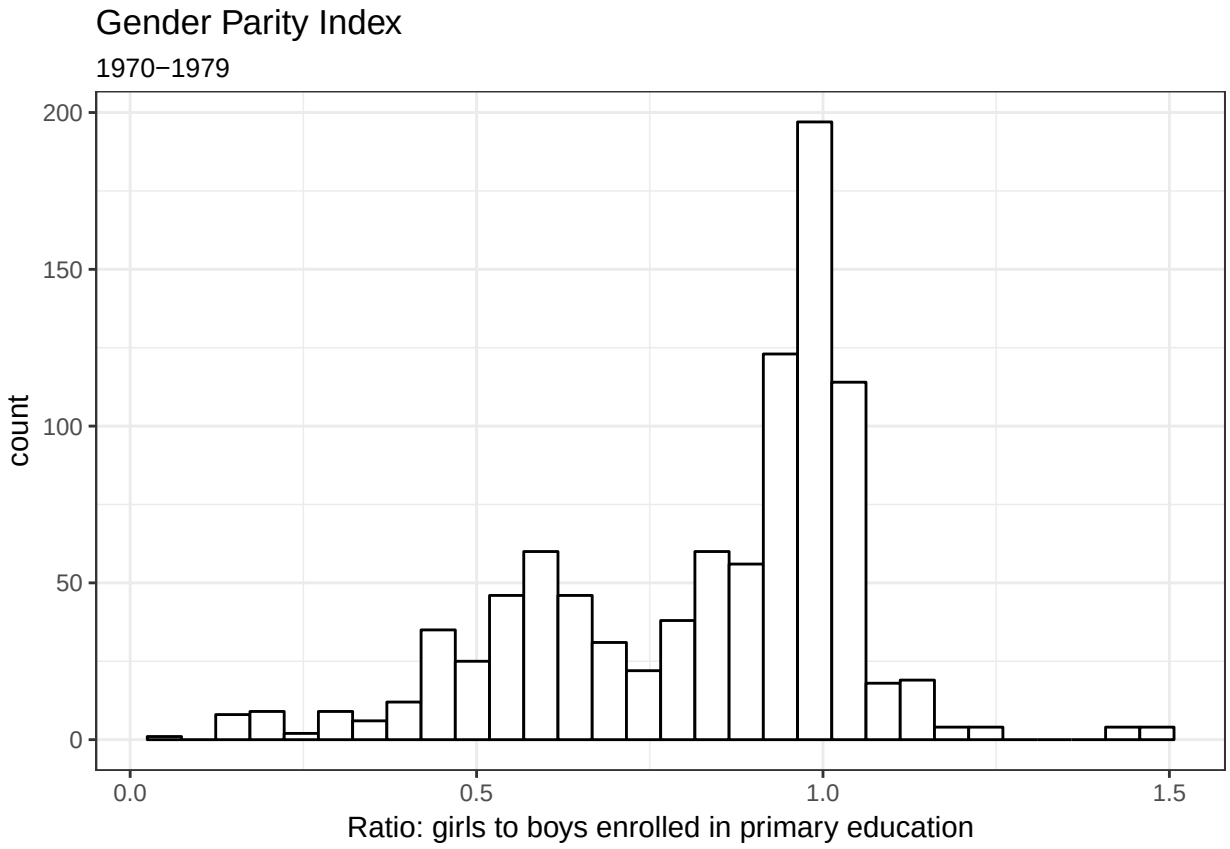
```
# Filter observations after 2005
gender05 = filter(wdi, year >= 2005)

# Histogram using data from the 1970s
ggplot(data = gender70, aes(gender_parity_WDI)) +
  geom_histogram(fill = "white", color = "black") +
  ggtitle("Gender Parity Index", subtitle = "1970-1979") +
```

```
  labs(x = "Ratio: girls to boys enrolled in primary education") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

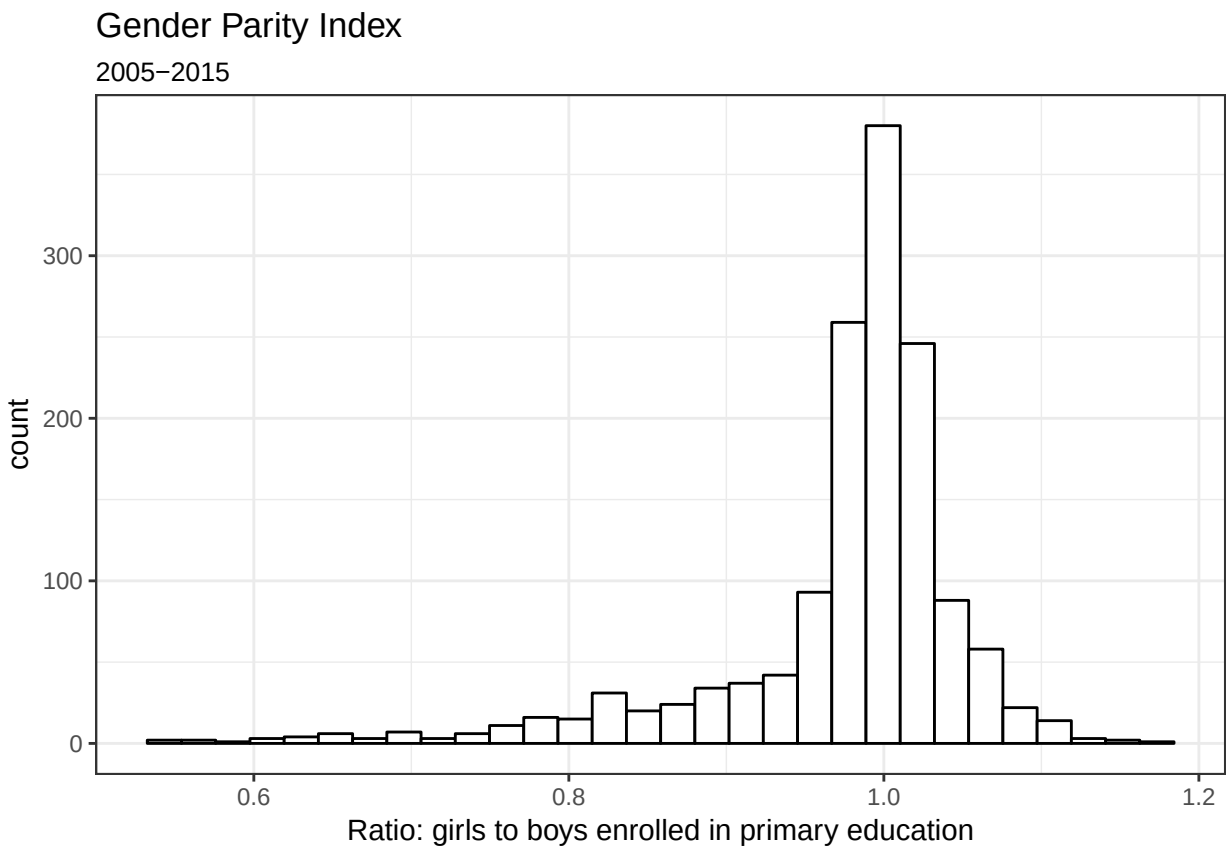
```
## Warning: Removed 3097 rows containing non-finite values (stat_bin).
```




```
# Histogram using data from 2005-2015
ggplot(data = gender05, aes(gender_parity_WDI)) +
  geom_histogram(fill = "white", color = "black") +
  ggtitle("Gender Parity Index", subtitle = "2005-2015") +
  labs(x = "Ratio: girls to boys enrolled in primary education") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1197 rows containing non-finite values (stat_bin).
```



Select

Select the variables related to education and assign them to a new dataframe object.

```
edu = select(wdi, # specify the dataframe first
                  country, # specify the variable names
                  year,
                  genderpar = gender_parity_WDI, # you can rename variables
                  govexp = govexp_edu_WDI)
```

Question 1:

Figure out how to use one of the helper functions for `select` to choose only the country identifier variables (gwno:gwabbrev, region). Assign this to a new dataframe.

Hint: `dplyr` cheat sheet: https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Mutate

`mutate` helps you create new variables.

We have a variable measuring a country's GDP and a variable measuring a country's population. We can use `mutate` to create a GDP per capita variable.

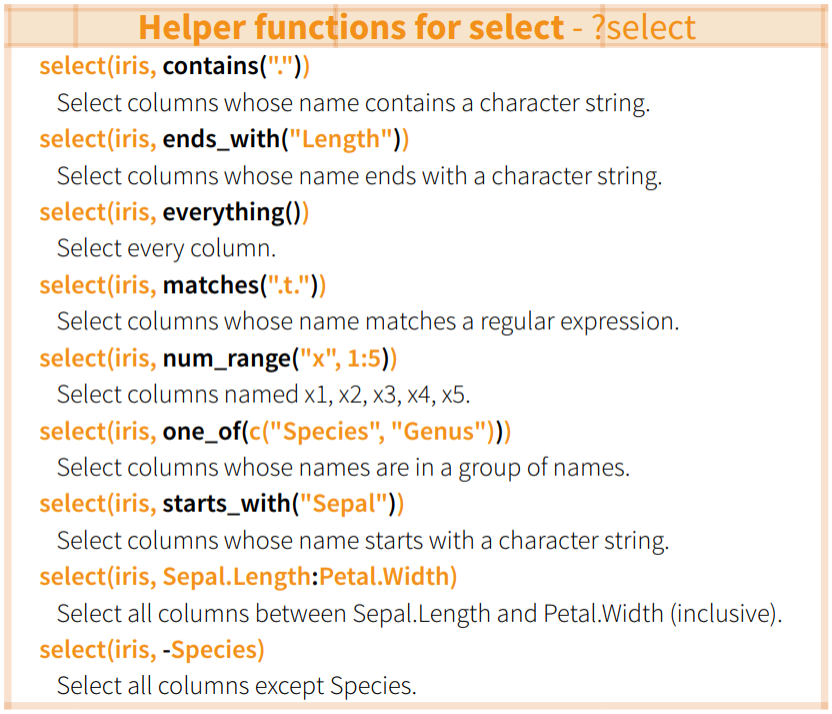


Figure 1: Helper functions for `dplyr`'s `select` function.

```
wdi = mutate(wdi, gdppc = gdp_WDI/pop_WDI)
```

We can transform the government education expenditures variable from a percentage to a proportion by dividing by 100.

```
wdi = mutate(wdi, govexp_edu_WDI = govexp_edu_WDI/100)
```

Question 2:

Create a dummy variable measuring if the value for gender parity (`gender_parity_WDI`) lies between 0.9 and 1.1. (1 = yes; 0 = no). Call this new variable `genderpar_dummy`.

Data Pipeline

The pipe operator in `dplyr` makes writing and reading code so easy and intuitive. This operator allows you to "pipe" the output from one function to the input of another function. With the pipe operator, you can forget about having to nest functions. Instead, you can write out your functions step-by-step.

Here's a basic example of a nested function. If you wanted to view the first 6 rows for country and gdp, you would need to nest `select()` into `head()`.

```
head(select(wdi, country, year, gdp_WDI))
```

```
##                    country year      gdp_WDI
## 1 United States of America 1960 3.078071e+12
## 2 United States of America 1961 3.148867e+12
## 3 United States of America 1962 3.340948e+12
```

```
## 4 United States of America 1963 3.487949e+12
## 5 United States of America 1964 3.690250e+12
## 6 United States of America 1965 3.926426e+12
```

But with the pipe operator, we can write this code differently. We can pipe the `wdi` dataframe to the function that will select 3 variables (`country`, `year`, `gdp_WDI`) to the function that will print the first 6 rows.

```
wdi %>%
  select(country, year, gdp_WDI) %>%
  head
```

```
##                    country year      gdp_WDI
## 1 United States of America 1960 3.078071e+12
## 2 United States of America 1961 3.148867e+12
## 3 United States of America 1962 3.340948e+12
## 4 United States of America 1963 3.487949e+12
## 5 United States of America 1964 3.690250e+12
## 6 United States of America 1965 3.926426e+12
```

Which observations have the highest boy-to-girl ratio?

```
wdi %>%
  arrange(gender_parity_WDI) %>% # puts the data in ascending order
  select(country, year, gender_parity_WDI) %>%
  slice(1:20) # you could also use head()
```

```
## # A tibble: 20 x 3
##         country  year gender_parity_WDI
##           <chr> <dbl>             <dbl>
##  1 Afghanistan  2001           0.00000
##  2      Bhutan  1970           0.05231
##  3        Oman  1972           0.15031
##  4 Afghanistan  1972           0.16127
##  5 Afghanistan  1971           0.16162
##  6        Oman  1971           0.16204
##  7 Afghanistan  1974           0.16741
##  8 Afghanistan  1970           0.16762
##  9 Afghanistan  1973           0.16919
## 10       Nepal  1973           0.16993
## 11 Afghanistan  1975           0.17439
## 12       Nepal  1974           0.17846
## 13 Afghanistan  1976           0.18116
## 14       Nepal  1975           0.18625
## 15 Afghanistan  1977           0.19284
## 16 Afghanistan  1978           0.19940
## 17      Bhutan  1972           0.20575
## 18        Oman  1973           0.20625
## 19       Nepal  1972           0.21300
## 20      Bhutan  1971           0.22775
```

Question 3:

Which observations have the highest girl-to-boy ratio?

Summarise

We can combine `group_by()`, `summarise()`, and the pipe operator to find the average value of gender parity each year.

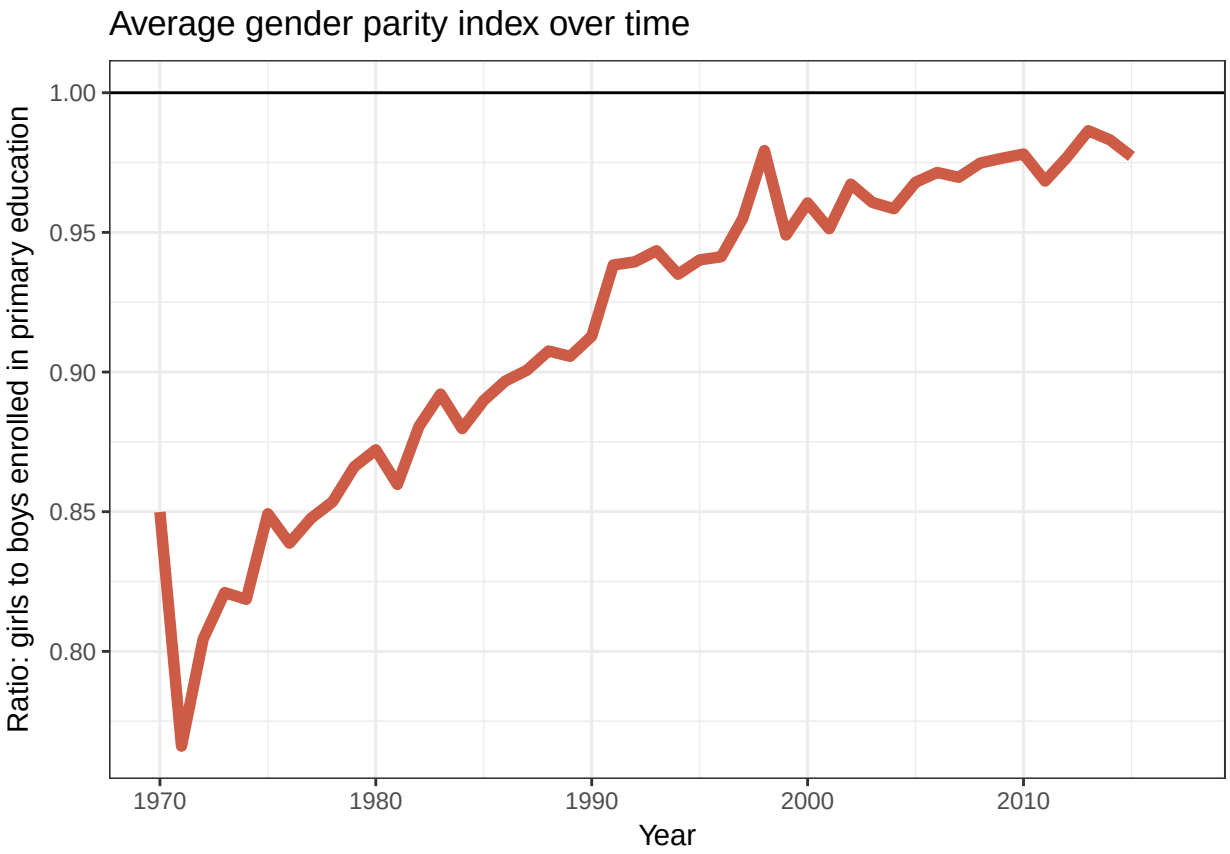
```
wdi %>%
  group_by(year) %>%
  summarise(genderpar_avg = mean(gender_parity_WDI, na.rm = TRUE)) %>%
  filter(year >= 1970)
```

```
## # A tibble: 48 x 2
##     year genderpar_avg
##    <dbl>         <dbl>
##  1  1970     0.8498986
##  2  1971     0.7660553
##  3  1972     0.8043357
##  4  1973     0.8210581
##  5  1974     0.8186184
##  6  1975     0.8492747
##  7  1976     0.8387251
##  8  1977     0.8476392
##  9  1978     0.8535775
## 10  1979     0.8660880
## # ... with 38 more rows
```

What makes the pipe operator so powerful is that you can pipe into ggplot. So we can use our aggregated data to plot gender parity over time.

```
wdi %>%
  group_by(year) %>%
  summarise(genderpar_avg = mean(gender_parity_WDI, na.rm = TRUE)) %>%
  filter(year >= 1970) %>%
  ggplot(aes(x = year, y = genderpar_avg)) +
  geom_line(size = 2, color = "coral3") +
  geom_hline(yintercept = 1) +
  theme_bw() +
  ggtitle("Average gender parity index over time") +
  labs(x = "Year", y = "Ratio: girls to boys enrolled in primary education")
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```

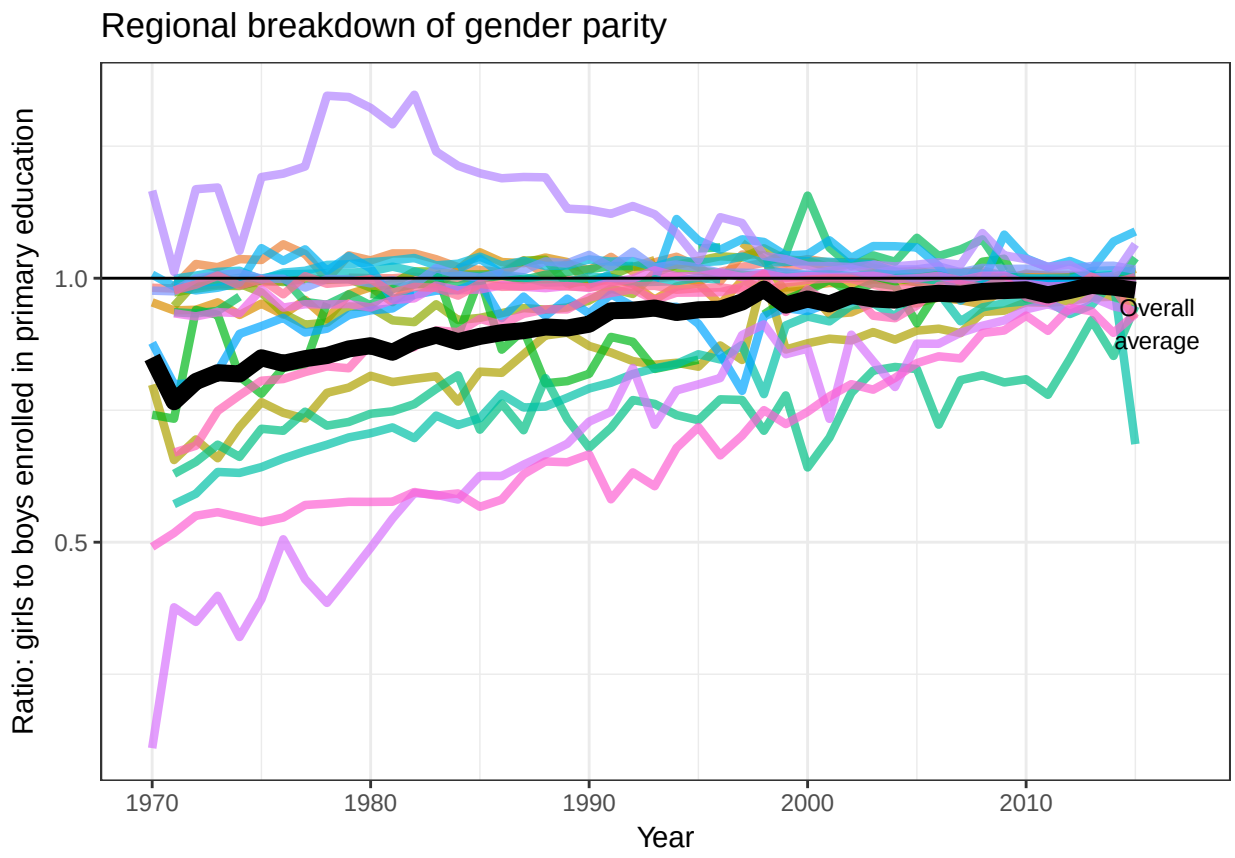


Question 4:

How might you plot the overall average gender parity along with regional averages over time? Something like this:

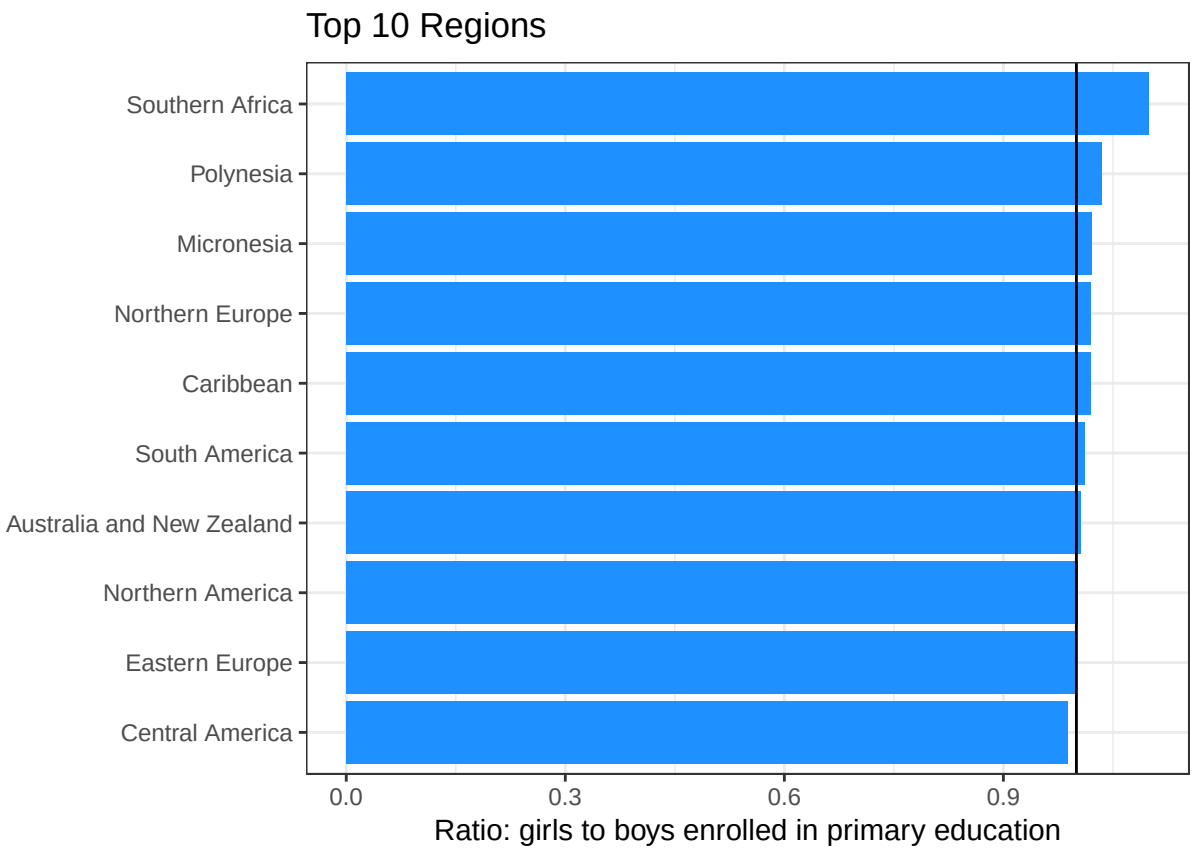
```
## Warning: Removed 71 rows containing missing values (geom_path).
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```



Plot the top 10 ranked regions in terms of gender parity.

```
wdi %>%
  group_by(region) %>%
  summarise(genderpar_avg = mean(gender_parity_WDI, na.rm = TRUE)) %>%
  filter(!is.na(region)) %>%
  arrange(-genderpar_avg) %>%
  slice(1:10) %>%
  ggplot(aes(x = reorder(region, genderpar_avg), y = genderpar_avg)) +
  geom_bar(stat = "identity", fill = "dodgerblue") +
  coord_flip() +
  geom_hline(yintercept = 1) +
  labs(x = "", y = "Ratio: girls to boys enrolled in primary education") +
  ggtitle("Top 10 Regions") +
  theme_bw()
```



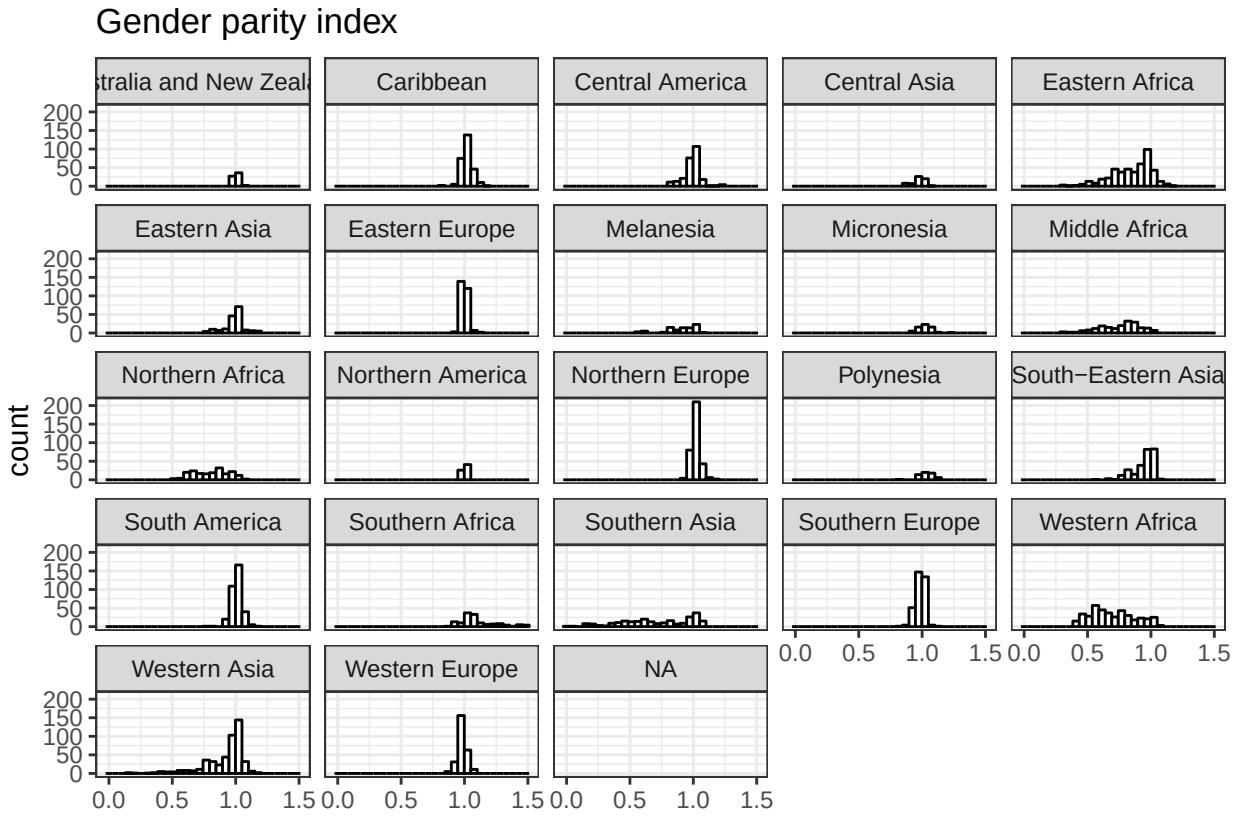
What's going on in Africa?

We can take a closer look at the distribution of gender parity region-by-region by plotting histograms for each region. We can do this using a `facet_wrap()` layer in `ggplot`.

```
ggplot(wdi, aes(gender_parity_WDI)) +
  geom_histogram(fill = "white", color = "black") +
  facet_wrap(~ region) +
  theme_bw() +
  labs(x = "") +
  ggtitle("Gender parity index")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 6720 rows containing non-finite values (stat_bin).
```

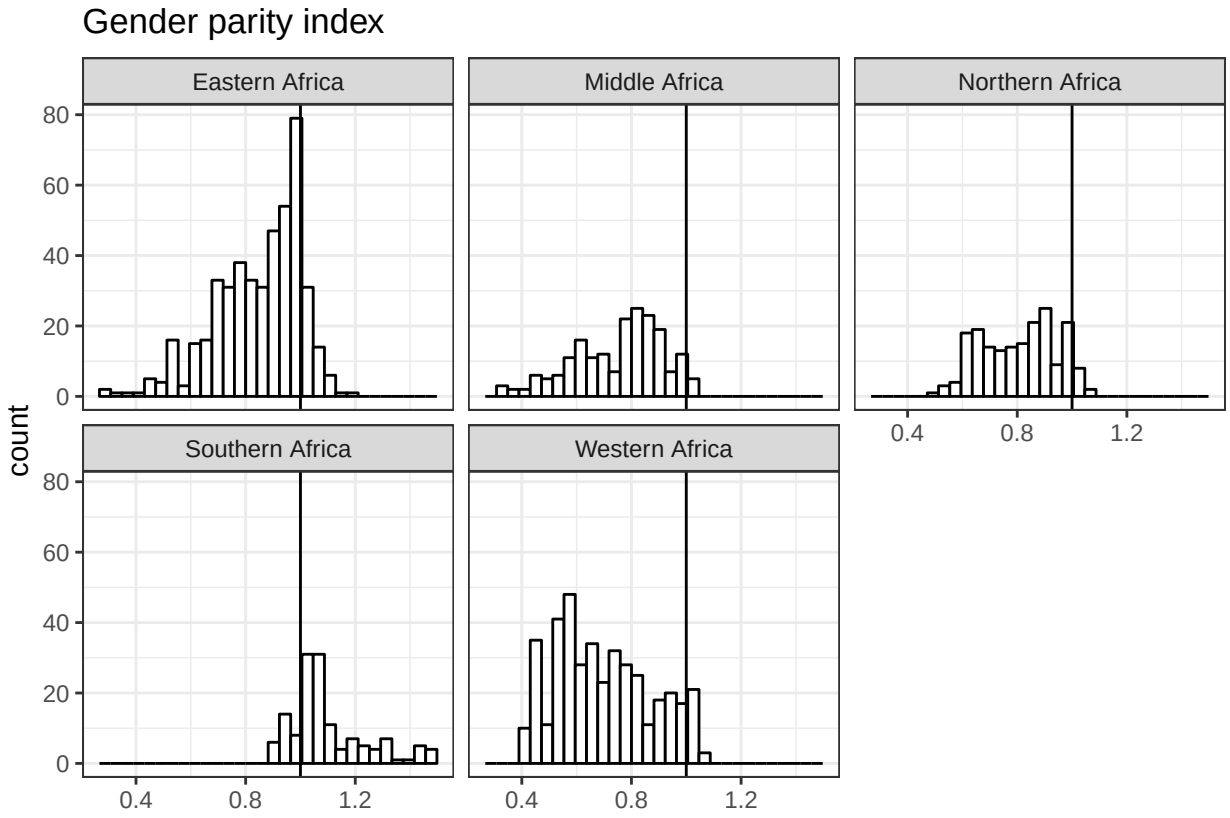


As you can see, there's too much going on, and it's difficult to read.

Question 5:

How would we plot histograms for African regions only?

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 1744 rows containing non-finite values (stat_bin).
```


Gender parity index
Eastern Africa
Middle Africa
Northern Africa
Southern Africa
Western Africa
count
0
20
40
60
80
0.4
0.8
1.2