Working Title: the data science canon

Databrew

2021-04-08

Contents

Chapter 1

Welcome

Welcome to *Working Title*, the data science canon by *DataBrew*

Here is some teacher content.

Part I

Core theory

Chapter 2

Principles of data science

2.1 What is data science?

2.2 What is the data life cycle?

2.3 What is a pipeline?

2.4 Data science 'in the wild'

2.5 The reproducibility crisis

Chapter 3

Visualizing data

3.1 Bad examples

3.2 Good exaples

3.3 Edward Tufte

3.4 Grammar of graphics

3.5 Design principles

3.6 Plots & power

The politics of graphics

Chapter 4

Writing about data

Chapter 5

Data ethics

Part II

Getting started

Chapter 6

Setting up RStudio

Chapter 7

Running R code

7.1 Basic math

7.2 Operators

Chapter 8

RStudio workflows

8.1 Tour of RStudio

8.2 Scripts

8.3 Typical workflows

Chapter 9

Objects in R

9.1 Variables

9.2 Vectors

Chapter 10

Calling functions

Chapter 11

Base plots

Chapter 12

Packages

Chapter 13

Basics of ggplot

Part III

Working with data in R

Chapter 14

Importing data

14.1 Working directories

14.2 Reading in data

Chapter 15

Dataframes

15.1 Exploration

15.2 Summarization

Chapter 16

Data wrangling

16.1 Data transformation

16.1.1 Filtering

16.1.2 Grouping

16.1.3 Joining

16.2 The tidyverse and tibbles

16.3 Transformation with dplyr

16.3.1 Filtering

16.3.2 Grouping

16.3.3 Mutating

Part IV

Exploring & analyzing data

Chapter 17

Exploratory Data Analysis

17.1 Exploring distributions

17.2 Variable types & statistics

17.3 Descriptive statistics

Chapter 18

Significance statistics

18.1 Thinking about significance

18.2 Comparison tests

18.3 Correlation tests

Chapter 19

Displaying data

19.1 Tables

19.2 Base plots

Advanced techniques

19.3 ggplot

Advanced techniques

Part V

Creating your own dataset

Chapter 20

Managing project files

Chapter 21

Formatting your own data

Chapter 22

Reading Excel files

Chapter 23

Reading GoogleSheets

Chapter 24

Reading online data

Part VI

Your R tool bag

Chapter 25

Joining datasets

Chapter 26

`for` loops

Learning goals

- What `for` loops are, and how to use them yourself
- How to use `for` loops for multi-pane plotting
- How to use `for` loops to achieve complex plots
- How to use `for` loops to summarize data efficiently

Coming soon

- Instructor notes and answer keys (hidden from students)

Tutorial video

(coming soon!)

Basics

A `for` loop is a super powerful coding tool. In a `for` loop, `R` loops through a chunk of code for a set number of repititions.

A super basic example:

```
x <- 1:5
for(i in x){
  print(i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Here's an example of a pretty useless `for` loop:

```
for(i in 1:5){
  print("I'm just repeating myself.")
}
```

```
[1] "I'm just repeating myself."
[1] "I'm just repeating myself."
[1] "I'm just repeating myself."
[1] "I'm just repeating myself."
[1] "I'm just repeating myself."
```

This code is saying:
- For each iteration of this loop, step to the next value in `x` (first example) or `1:5` (second example).
- Store that value in an object `i`,
- and run the code inside the curly brackets. - Repeat until the end of `x`.

Look at the basic structure:
- In the`for( )` parenthetical, you tell R what values to step through (`x`), and how to refer to the value in each iteration (`i`).
- Within the curly brackets, you place the chunk of code you want to repeat.

Another basic example, demonsrating that you can update a variable repeatedly in a loop.

```
x <- 2
for(i in 1:5){
  x <- x*x
  print(x)
}
```

```
[1] 4
```

```
[1] 16
[1] 256
[1] 65536
[1] 4294967296
```

Another silly example:

```r
professors <- c("Keri","Deb","Ken")
for(x in professors){
  print(paste0(x," is pretty cool!"))
}
```

```
[1] "Keri is pretty cool!"
[1] "Deb is pretty cool!"
[1] "Ken is pretty cool!"
```

Exercise 1

Use this space to practice the basics of `for loop` formatting.

First, create a vector of names (add at least 3)

```r
# Add your names to this vector
famous.names <- c("Lady Gaga","David Haskell","Tom Cruise")
```

Using the examples above as a guide, create a `for loop` that prints the same silly statement about each of these names.

```r
# Do your coding here
for(i in famous.names){
  print(paste0(i," has cooties!"))
}
```

```
[1] "Lady Gaga has cooties!"
[1] "David Haskell has cooties!"
[1] "Tom Cruise has cooties!"
```

Using `for` loops with data

These silly examples above do a poor job of demonstrating how powerful a `for loop` can be.

Multi-panel plots

For example, a `for loop` can be a very efficient way of making multi-panel plots.

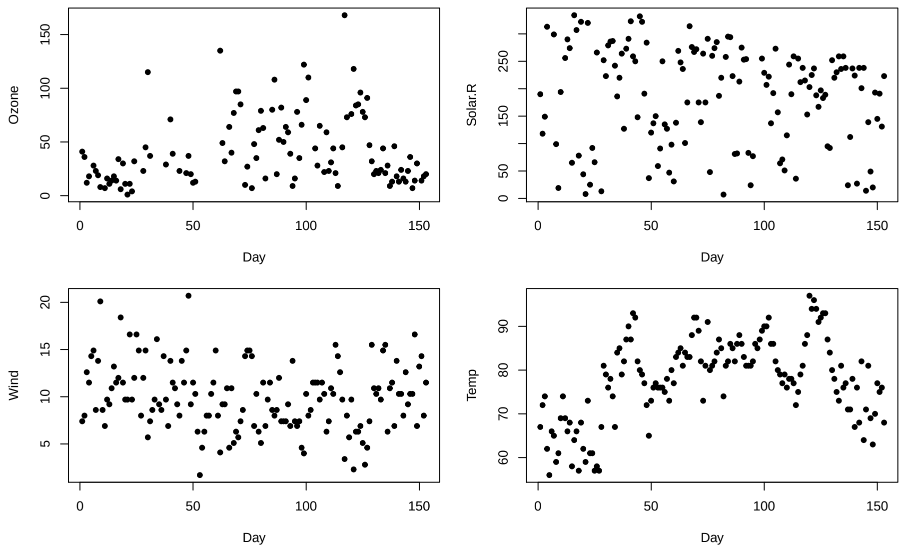
Let's use a `for loop` to get a quick overview of the variables included in the `airquality` dataset built into R.

```
data(airquality)
head(airquality)
```

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

Looks like the first four columns would be interesting to plot.

```
par(mfrow=c(2,2)) # Setup a multi-panel plot # format = c(number of rows, number of co
par(mar=c(4.5,4.5,1,1)) # Set plot margins

for(i in 1:4){
  y <- airquality[,i]
  var.name <- names(airquality)[i]
  plot(y,xlab="Day",ylab=var.name,pch=16)
}
```




```
par(mfrow=c(1,1)) # restore the default single-panel plot
```

Tricky plot solutions

`for loops` are also useful for plotting data in tricky ways. Let's use a different built-in dataset, that shows the performance of various car make/models.

```
data(mtcars)
head(mtcars)
```

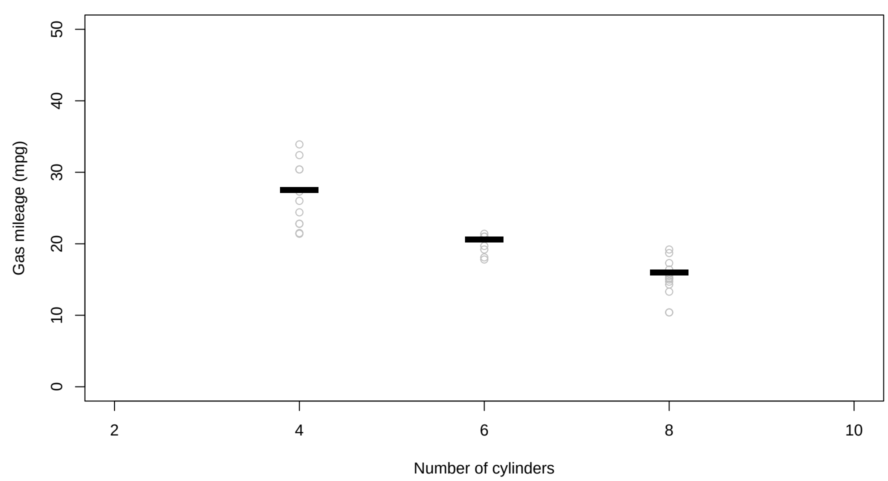
```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Let's say we want to see how gas mileage is affected by the number of cylinders a car has. It would be nice to create a plot that shows the raw data as well as the mean mileage for each cylinder number.

```
# Let's see how many different cylinder types there are in the data
ucyl <- unique(mtcars$cyl) ; ucyl
```

```
[1] 6 4 8
```

```
# Let's make an empty plot
plot(1,type="n", # tell R not to draw anything
     xlim=c(2,10),ylim=c(0,50),
     xlab="Number of cylinders",
     ylab="Gas mileage (mpg)")

# Write your for loop here to add the actual data
i=ucyl[1] # It's always good to use a known value of i as you build up your for loop
for(i in ucyl){

  # Subset the dataframe according to number of cylinders
  cari <- mtcars[mtcars$cyl==i,]

  # Plot the raw data
  points(x=cari$cyl,y=cari$mpg,col="grey")

  # Superimpose the mean on top
  points(x=i,y=mean(cari$mpg),col="black",pch="-",cex=5,)
}
```

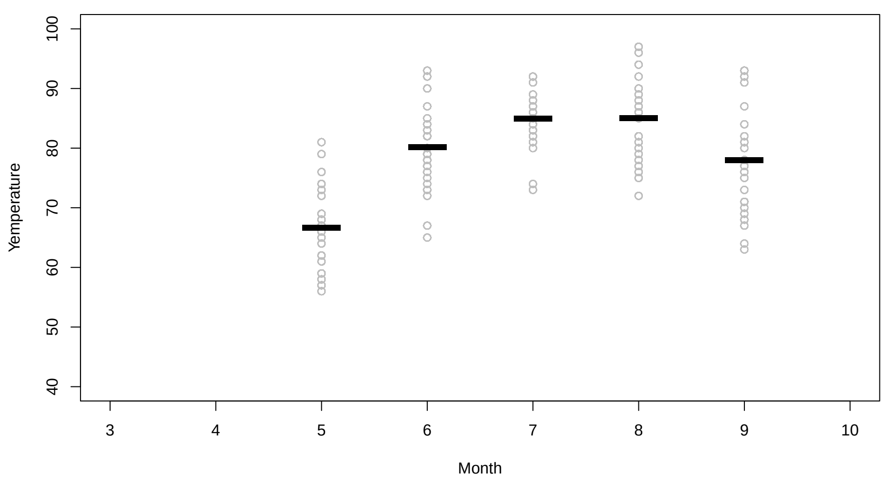


Exercise 2

Now try to do something similar on your own with the `airquality` dataset. Use `for loops` to create a plot with Month on the x axis and Temperature on the y axis. On this plot, depict all the temperatures recorded in each month in the color grey, then superimpose the mean temperature for each month.

We will provide the empty plot, you provide the `for loop`:

```
plot(1,type="n",
     xlim=c(3,10),ylim=c(40,100),
     xlab="Month",
     ylab="Yemperature")

# Write your for loop here to add the actual data
for(i in airquality$Month){
  airi <- airquality[airquality$Month==i,]
  points(x=airi$Month,y=airi$Temp,pch=1,col="grey")
  points(x=i,y=mean(airi$Temp),pch="-",cex=5,col="black")
}
```



Using a `for loop` with more complex data

Here's another good example of the power of a good `for loop`.

First, read in some cool data.

```
kc <- read.csv("./data/keeling-curve.csv") ; head(kc)
```

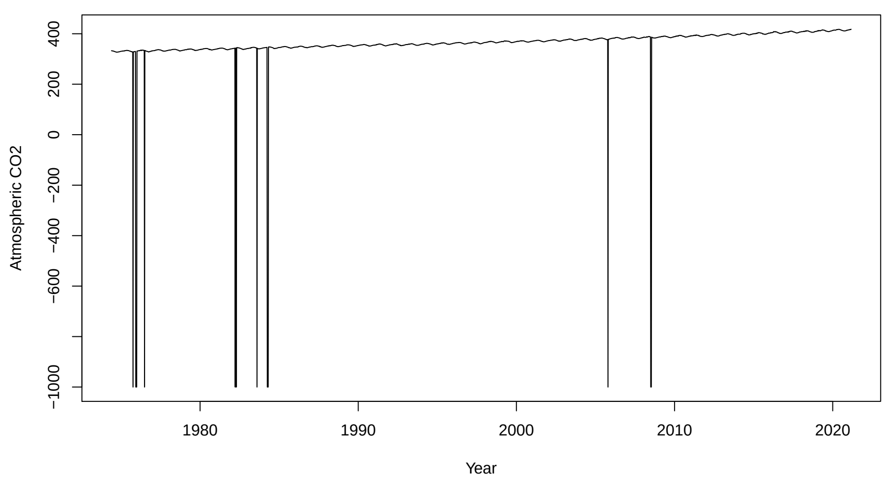
```
  year month day_of_month day_of_year year_dec frac_of_year     CO2
```

```
1 1974     5           26    145.4890 1974.399      0.3986 332.95
2 1974     6            2    152.4970 1974.418      0.4178 332.35
3 1974     6            9    159.5050 1974.437      0.4370 332.20
4 1974     6           16    166.5130 1974.456      0.4562 332.37
5 1974     6           23    173.4845 1974.475      0.4753 331.73
6 1974     6           30    180.4925 1974.495      0.4945 331.68
```

This is the famous Keeling Curve dataset: long-term monitoring of atmospheric CO2 measured at a volcanic observatory in Hawaii.

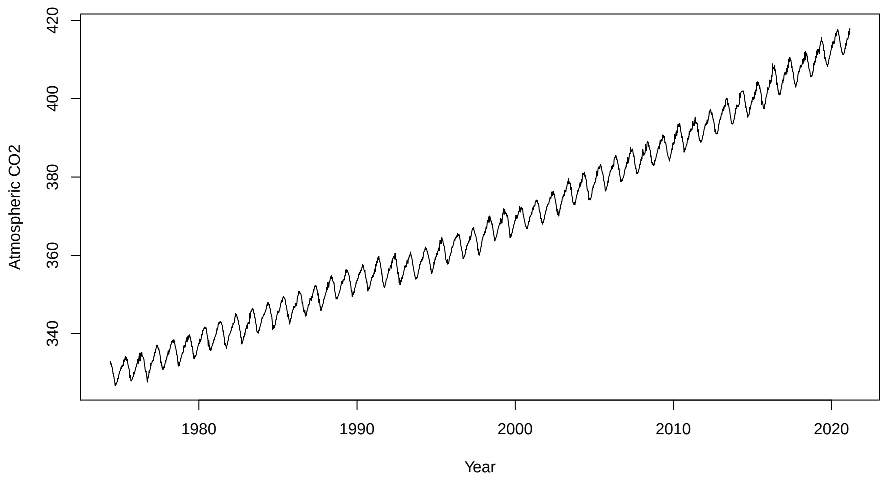
Try plotting the Keeling Curve:

```
plot(kc$CO2 ~ kc$year_dec,type="l",xlab="Year",ylab="Atmospheric CO2")
```



There are some erroneous data points! We clearly can't have negative CO2 values. Let's remove those and try again:

```
kc <- kc[kc$CO2 >0,]
plot(kc$CO2 ~ kc$year_dec,type="l",xlab="Year",ylab="Atmospheric CO2")
```



What's the deal with those squiggles? Let's investigate!

Let's look at the data a different way: *by focusing in on a single year.*

```
# Stage an empty plot for what you are trying to represent
plot(1, # plot a single point
     type="n",
     xlim=c(0,365),xlab="Day of year",
     ylim=c(-5,5),ylab="CO2 anomaly")
abline(h=0,col="grey") # add nifty horizontal line

# Reduce the dataset to a single year (any year)
kcy <- kc[kc$year=="1990",] ; head(kcy)
```

```
    year month day_of_month day_of_year year_dec frac_of_year    CO2
816 1990     1            7      6.4970 1990.018       0.0178 353.58
817 1990     1           14     13.5050 1990.037       0.0370 353.99
818 1990     1           21     20.5130 1990.056       0.0562 353.92
819 1990     1           28     27.4845 1990.075       0.0753 354.39
820 1990     2            4     34.4925 1990.094       0.0945 355.04
821 1990     2           11     41.5005 1990.114       0.1137 355.09
```

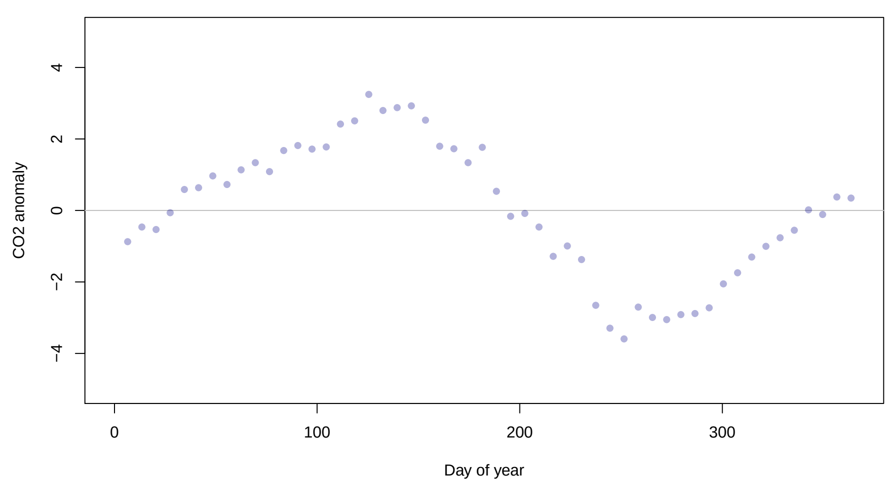
```
# Let's convert each CO2 reading to an 'anomaly' compared to the year's average.
CO2.mean <- mean(kcy$CO2,na.rm=TRUE) ; CO2.mean  # Take note of how useful that 'na.rm=TRUE' inpu
```

```
[1] 354.4538
```

```
y <- kcy$CO2 - CO2.mean ; y # Translate each data point to an anomaly
```

```
 [1] -0.87384615 -0.46384615 -0.53384615 -0.06384615  0.58615385  0.63615385
 [7]  0.96615385  0.72615385  1.13615385  1.33615385  1.08615385  1.67615385
[13]  1.81615385  1.71615385  1.77615385  2.41615385  2.50615385  3.24615385
[19]  2.79615385  2.87615385  2.92615385  2.52615385  1.79615385  1.72615385
[25]  1.33615385  1.76615385  0.53615385 -0.16384615 -0.08384615 -0.46384615
[31] -1.28384615 -0.99384615 -1.37384615 -2.65384615 -3.29384615 -3.59384615
[37] -2.70384615 -2.99384615 -3.05384615 -2.91384615 -2.88384615 -2.72384615
[43] -2.05384615 -1.74384615 -1.30384615 -1.00384615 -0.76384615 -0.55384615
[49]  0.01615385 -0.11384615  0.37615385  0.34615385          NA
```

```
# Add points to your plot
points(y~kcy$day_of_year,pch=16,col=adjustcolor("darkblue",alpha.f=.3))
```



But this only shows one year of data! How can we include the seasonal squiggle from other years?

Let's use a `for loop`!

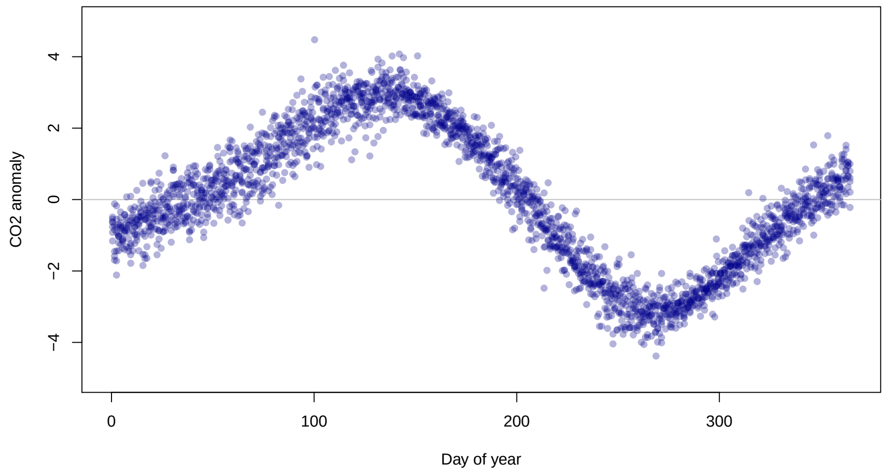
OK – let's redo that graph and add a `for loop` into the mix:

```
# First, stage your empty plot:
plot(1,type="n",
     xlim=c(0,365),xlab="Day of year",
     ylim=c(-5,5),ylab="CO2 anomaly")
```

```
abline(h=0,col="grey")

# Now we will loop through each year of data. First, get a vector of the years included in the da
years <- unique(kc$year) ; years
```

```
 [1] "1974" "1975" "1976" "1977" "1978" "1979" "1980" "1981" "1982" "1983"
[11] "1984" "1985" "1986" "1987" "1988" "1989" "1990" "1991" "1992" "1993"
[21] "1994" "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002" "2003"
[31] "2004" "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012" "2013"
[41] "2014" "2015" "2016" "2017" "2018" "2019" "2020" "2021" NA
```

```
# Now build your for loop.
# Notice that the contents of the `for loop` are exactly the same
# as the single plot above --  with one exception.
# Notice the use of the symbol i

for(i in years){

  # Reduce the dataset to a single year
  kcy <- kc[kc$year==i,] ; head(kcy)

  # Let's convert each CO2 reading to an 'anomaly' compared to the year's average.
  CO2.mean <- mean(kcy$CO2,na.rm=TRUE) ; CO2.mean # Get average CO2 for year

  y <- kcy$CO2 - CO2.mean ; y # Translate each data point to an anomaly

  # Add points to your plot
  points(y~kcy$day_of_year,pch=16,col=adjustcolor("darkblue",alpha.f=.3))

}
```



Beautiful! So how do you interpret this graph? Why does the squiggle happen every year?

Review assignment

First, read in and format some other cool data. The code for doing so is provided for you here:

```
df <- read.csv("./data/renewable-energy.csv")
```

This dataset, freely available from World Bank, shows the renewable electricity output for various countries, presented as a percentage of the nation's total electricity output. They provide this data as a time series.

26.0.1 Summarize columns with a `for loop`

Task 1: Use a `for loop` to find the change in renewable energy output for each nation in the dataset between 1990 and 2015. Print the difference for each nation in the console.

```
# Write your code here
names(df)
```

```
 [1] "year"            "World"            "Australia"        "Canada"
```

```
 [5] "China"          "Denmark"        "India"          "Japan"
 [9] "New_Zealand"    "Sweden"         "Switzerland"    "United_Kingdom"
[13] "United_States"
```

```
i=2
for(i in 2:ncol(df)){
  dfi <- df[,i] ; dfi
  diffi <- dfi[length(dfi)] - dfi[1] ; diffi
  print(paste0(names(df)[i]," : ",round(diffi),"% change."))
}
```

```
[1] "World : 3% change."
[1] "Australia : 4% change."
[1] "Canada : 1% change."
[1] "China : 4% change."
[1] "Denmark : 62% change."
[1] "India : -9% change."
[1] "Japan : 5% change."
[1] "New_Zealand : 0% change."
[1] "Sweden : 12% change."
[1] "Switzerland : 7% change."
[1] "United_Kingdom : 23% change."
[1] "United_States : 2% change."
```

Task 2: Re-do this loop, but instead of printing the differences to the console, save them in a vector.

```
# Write your code here
diffs <- c()
i=2
for(i in 2:ncol(df)){
  dfi <- df[,i] ; dfi
  diffi <- dfi[length(dfi)] - dfi[1] ; diffi
  print(paste0(names(df)[i]," : ",round(diffi),"% change."))
  diffs <- c(diffs,diffi)
}
```

```
[1] "World : 3% change."
[1] "Australia : 4% change."
[1] "Canada : 1% change."
[1] "China : 4% change."
[1] "Denmark : 62% change."
[1] "India : -9% change."
[1] "Japan : 5% change."
```

```
[1] "New_Zealand : 0% change."
[1] "Sweden : 12% change."
[1] "Switzerland : 7% change."
[1] "United_Kingdom : 23% change."
[1] "United_States : 2% change."
```

```
diffs
```

```
 [1]  3.49241703  3.98181045  0.63273122  3.51887728 62.33064943 -9.14624362
 [7]  4.73004321  0.07524008 12.26263811  7.21543884 23.01128298  1.69994636
```

Multi-pane plots with `for loops`

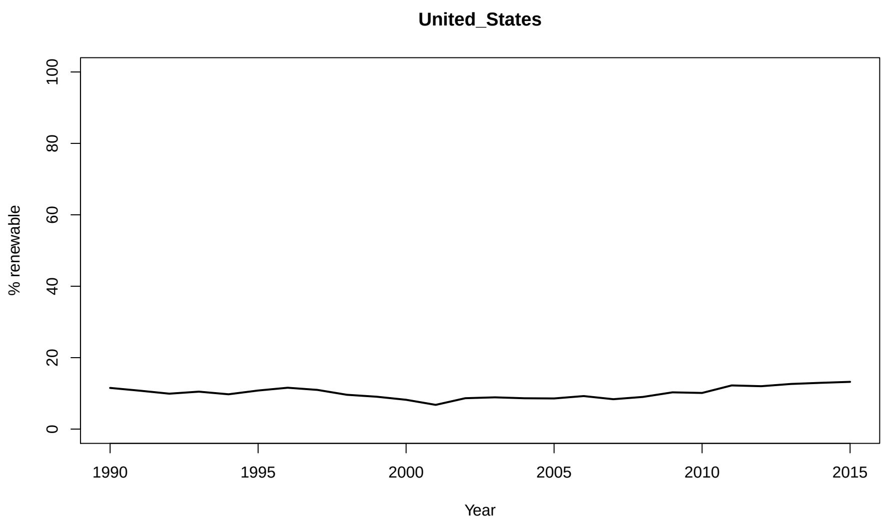
Practice with a single plot

Task 3: First, get your bearings by figuring out how to use the `df` dataset to plot the time series for the United States, for the years 1990 - 2015. Label the x axis "Year" and the y axis "% Renewable". Include the full name of the county as the `main` title for the plot.

```
# Write code here
head(df)
```

```
  year    World Australia   Canada    China  Denmark    India     Japan
1 1990 19.36204  9.656031 62.37872 20.40794 3.175275 24.48929 11.254738
2 1991 19.23357 10.598201 61.41041 18.47113 2.892325 22.80740 11.856735
3 1992 19.15840 10.066865 61.67921 17.58468 4.398464 20.75265 10.162888
4 1993 19.78795 10.549144 61.72233 18.12526 4.730088 19.55881 11.454528
5 1994 19.53812 10.194474 60.40045 18.08844 4.295431 21.21910  7.993026
6 1995 19.83536  9.624143 61.00410 19.21414 5.035639 17.26054  9.416323
  New_Zealand   Sweden Switzerland United_Kingdom United_States
1    80.00620 51.00011    54.98254       1.828767     11.528647
2    77.18945 44.30088    57.16370       1.656439     10.757414
3    72.58771 52.33321    56.90938       2.005662      9.916110
4    77.02407 52.92433    59.57279       1.777626     10.484326
5    82.05216 43.02873    60.57322       2.139842      9.747236
6    83.85281 47.57878    57.42996       2.066535     10.801085
```

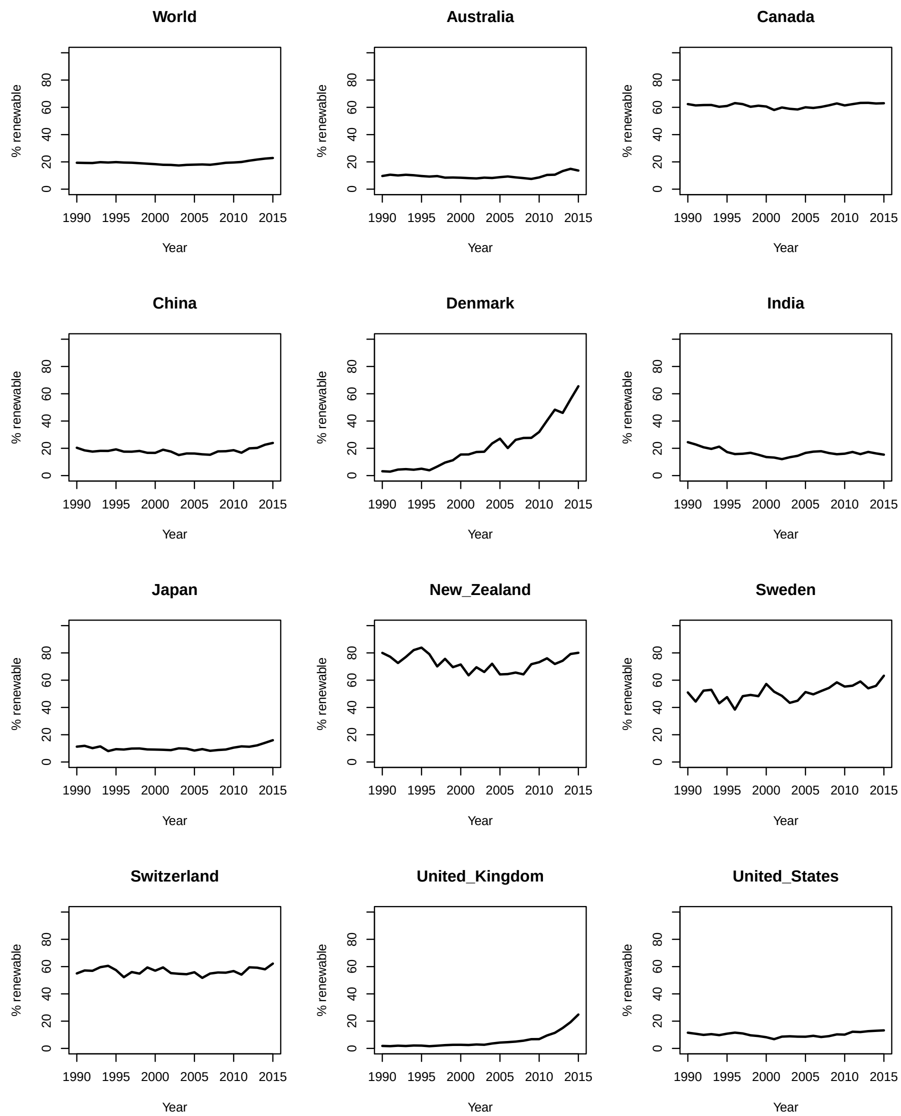
```
dfi <- df[,c(1,13)]
plot(x=dfi[,1],
     y=dfi[,2],
     type="l",lwd=2,
     xlim=c(1990,2015),ylim=c(0,100),
     xlab="Year",ylab="% renewable",
     main=names(dfi)[2])
```



Now loop it!

Task 4: Use that code as the foundation for building up a `for loop` that displays the same time series for every country in the dataset on a multi-pane graph that with 4 rows and 3 columns.

```
par(mfrow=c(4,3))
i=3
for(i in 2:ncol(df)){
  dfi <- df[,c(1,i)] ; dfi
  plot(x=dfi[,1],
       y=dfi[,2],
       type="l",lwd=2,
       xlim=c(1990,2015),ylim=c(0,100),
       xlab="Year",ylab="% renewable",
       main=names(dfi)[2])
}
```

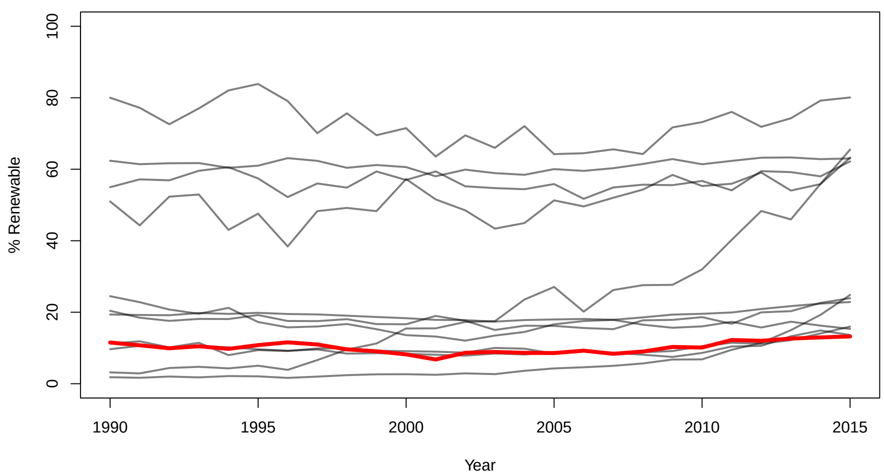


Now loop it differently!

Task 5: Now try a different presentation. Instead of producing 12 different plots, superimpose the time series for each country on the *same single plot.*

To add some flare, highlight the USA curve by coloring it red and making it thicker.

```r
par(mfrow=c(1,1))
plot(1,type="n",lwd=2,
     xlim=c(1990,2015),ylim=c(0,100),
     xlab="Year",ylab="% Renewable")

for(i in 2:ncol(df)){
  dfi <- df[,c(1,i)] ; dfi
  lines(dfi[,2]~dfi[,1],lwd=2,col=adjustcolor("black",alpha.f=.5))
}

lines(df$United_States~df$year,lwd=4,col="red")
```



Chapter 27

Writing functions

Chapter 28

Working with text

Chapter 29

Working with dates & times

Chapter 30

Working with factors

Chapter 31

Cleaning messy data

Chapter 32

Matrices & lists

Chapter 33

Pipes

Chapter 34

Exporting data & plots

Part VII

Interactive dashboards

Chapter 35

Intro to Shiny apps

Chapter 36

Shiny dashboards

Chapter 37

Data entry apps

Part VIII

Databases

Chapter 38

Introduction

38.1 What

38.2 Why

38.3 When

38.4 When not

Chapter 39

Platforms

39.1 PostgreSQL

39.2 mySQL

39.3 SQLite

Chapter 40

Alternatives

40.1 NoSQL

Chapter 41

Practices

Spinning up a local DB

Part IX

Documenting your work

Chapter 42

R Markdown

Chapter 43

Reproducible research

Chapter 44

Automated reporting

Chapter 45

Formatting standards

45.1 Tables

45.2 Figures

45.3 Captions

Part X

Version control and teamwork

Chapter 46

What is version control?

Chapter 47

What is Git?

47.1 Repositories

47.2 Github

Chapter 48

Standard git operations

Chapter 49

A git workflow

Chapter 50

Other git platforms

Part XI

Writing about data

Chapter 51

Types of writing

51.1 Grant proposals

51.2 Reports and publications

51.3 Fundraising

51.4 Press releases

Chapter 52

Elements of style

Chapter 53

Sections of a report

53.1 Abstract

53.2 Introduction

53.3 Methods

53.4 Results

53.5 Discussion

53.6 Other elements

53.6.1 Acknowledgments

53.6.2 Literature Cited

53.6.3 Tables

53.6.4 Figures

53.6.5 Supplementary Materials

Part XII

Creating websites

Part XIII

Advanced skills

Chapter 54

Mapping

Chapter 55

Geographic computing & GIS

Chapter 56

Statistical modeling

Chapter 57

Apply family

Chapter 58

Iterative statistics

Chapter 59

Iterative simulations

Chapter 60

Image analysis

Chapter 61

Machine learning

Chapter 62

Template

Learning goals

- Item 1
- Item 2
- Item 3

Here is some teacher content.

Tutorial video

Bangarang - Crew Briefing from Luke Padgett on Vimeo.

Basics

Exercise 1

Review assignment

Introduce data

Introduce task(s)

62.1 Other Resources

https://desiree.rbind.io/post/2020/learnr-iframes/

https://rstudio.github.io/learnr/