R libraries for Data Mining

Introduction

- There are 15340 CRAN R packages available

https://cran.r-project.org/web/packages/

- For a quick list of libraries by category

https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages

Categories of R libraries

- Import data
- Read data
- Manipulate data
- Visualize data
- Report results
- Model building
- High performance

Categories of R libraries

- Import data
- Read data (readr, data.table)
- Manipulate data (dplyr, tidyr, tibble, stringr, …)
- Visualize data (ggplot2, ggmap)
- Report results (rmarkdown, shiny, xtable)
- Model building (car,randomForest,glmnet,…)
- High performance (parallel, h2o, …)

Data Wrangling

Organizing the data in a useful way

for visualization, modeling and reporting

Import → Clean → Organize → Transform →

→ Visualize → Model → Report

tidyverse library

- A collection of R libraries for Data Wrangling
  - core libraries
  - non-core libraries

www.tidyverse.org

tidyverse ecosystem

- readr
- dplyr
- tidyr
- ggplot2
- stringr
- lubridate
- forcats

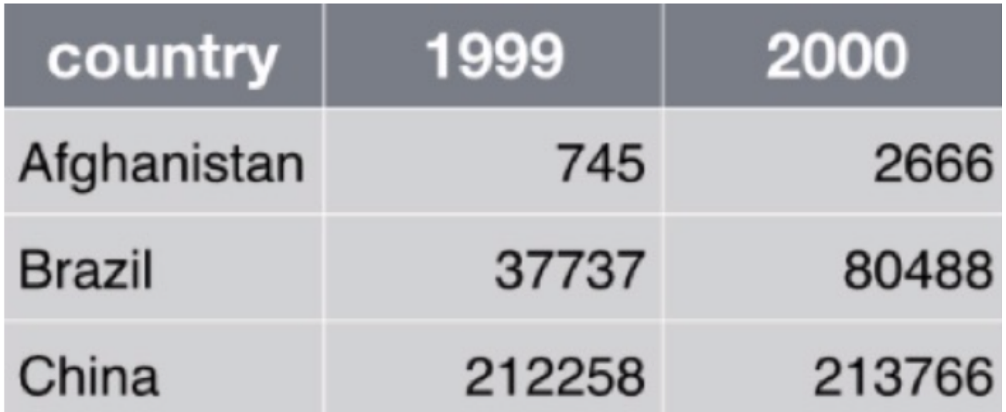
www.tidyverse.org

tidyverse ecosystem

- readr
- dplyr
- tidyr
- ggplot2
- stringr
- lubridate
- forcats

tidy and non-tidy data

- non-tidy data

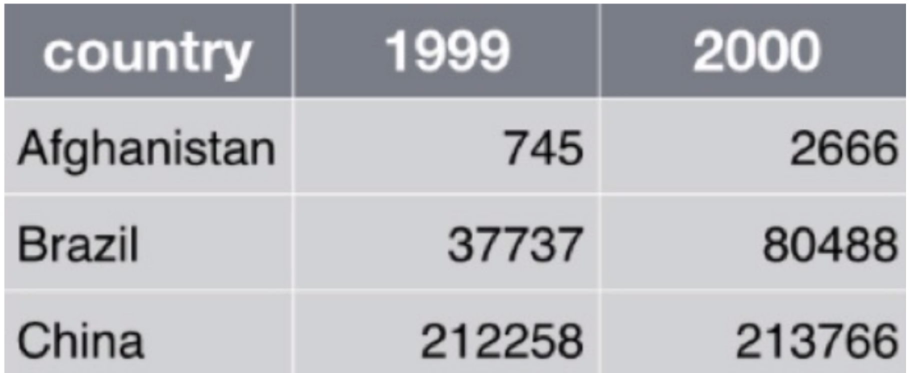


country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

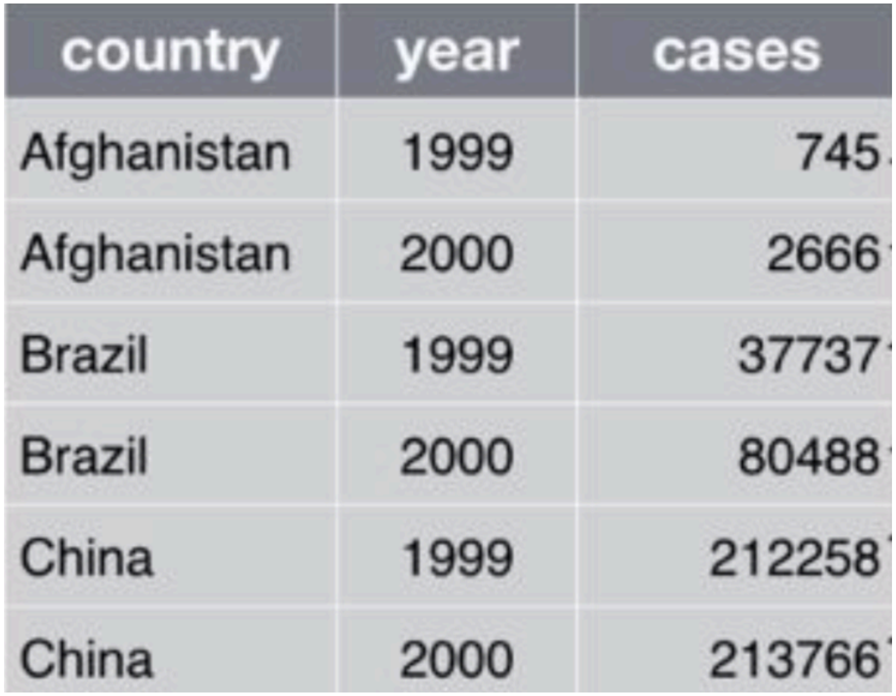
rows and columns are not variables

they are levels of 2 different categorical variables

tidy and non-tidy data



country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766



country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

tidy data

- each variable in a different column
- each observation in a different row
- each value in a different cell

non-tidy to tidy data

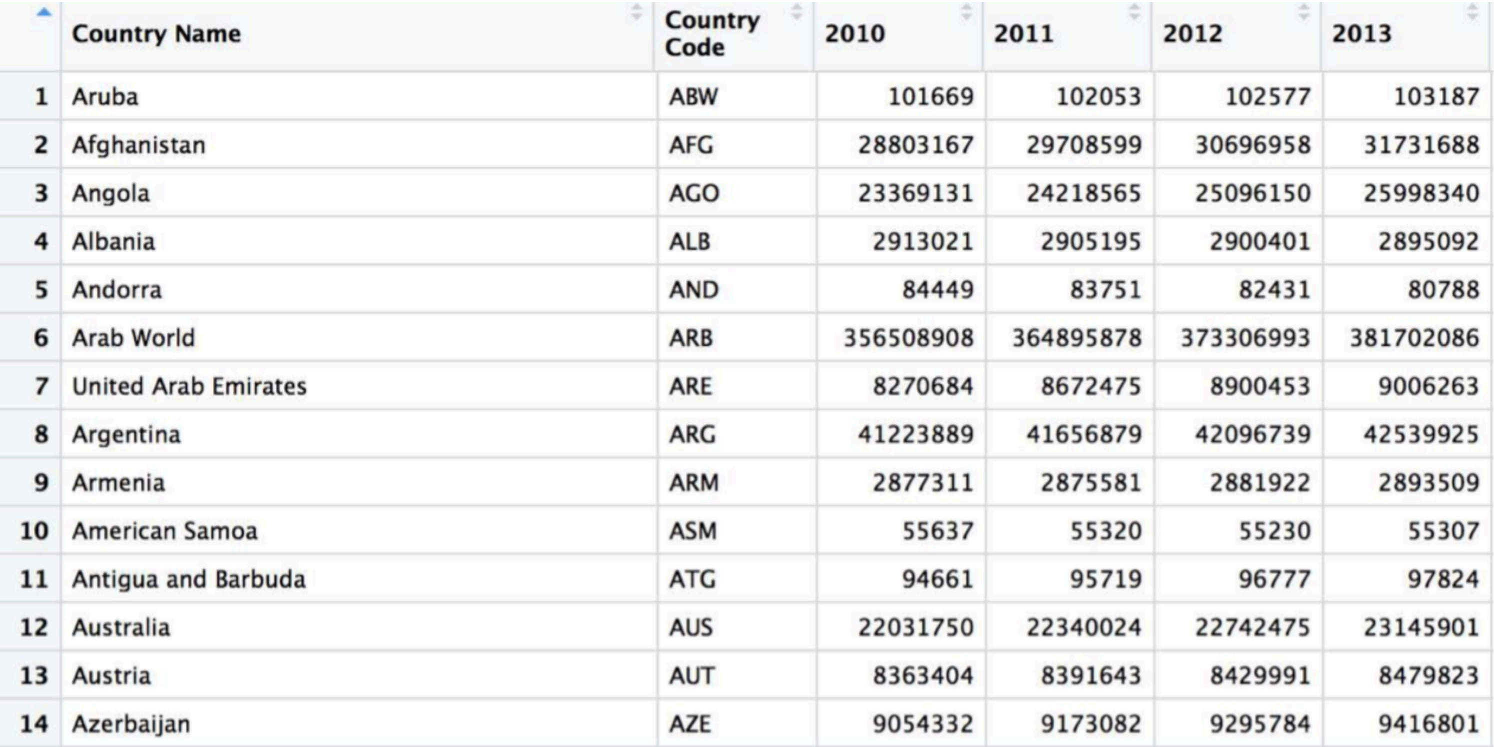
- Use library tidyr

function gather( )

gather(df, colnames, new categorical column,
new numeric column)

non-tidy to tidy data

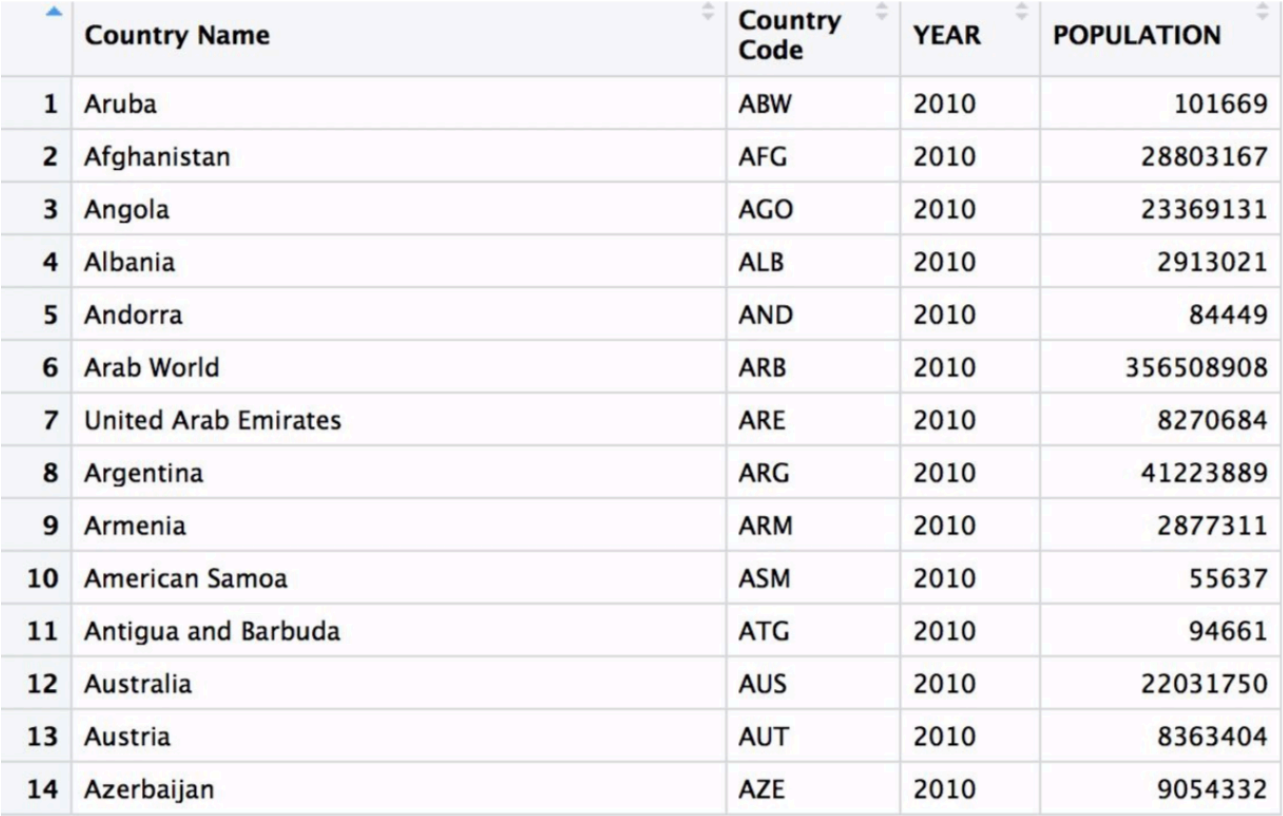
```
years = colnames(df)
years = years[3:6]
```



	Country Name	Country Code	2010	2011	2012	2013
1	Aruba	ABW	101669	102053	102577	103187
2	Afghanistan	AFG	28803167	29708599	30696958	31731688
3	Angola	AGO	23369131	24218565	25096150	25998340
4	Albania	ALB	2913021	2905195	2900401	2895092
5	Andorra	AND	84449	83751	82431	80788
6	Arab World	ARB	356508908	364895878	373306993	381702086
7	United Arab Emirates	ARE	8270684	8672475	8900453	9006263
8	Argentina	ARG	41223889	41656879	42096739	42539925
9	Armenia	ARM	2877311	2875581	2881922	2893509
10	American Samoa	ASM	55637	55320	55230	55307
11	Antigua and Barbuda	ATG	94661	95719	96777	97824
12	Australia	AUS	22031750	22340024	22742475	23145901
13	Austria	AUT	8363404	8391643	8429991	8479823
14	Azerbaijan	AZE	9054332	9173082	9295784	9416801

non-tidy to tidy data

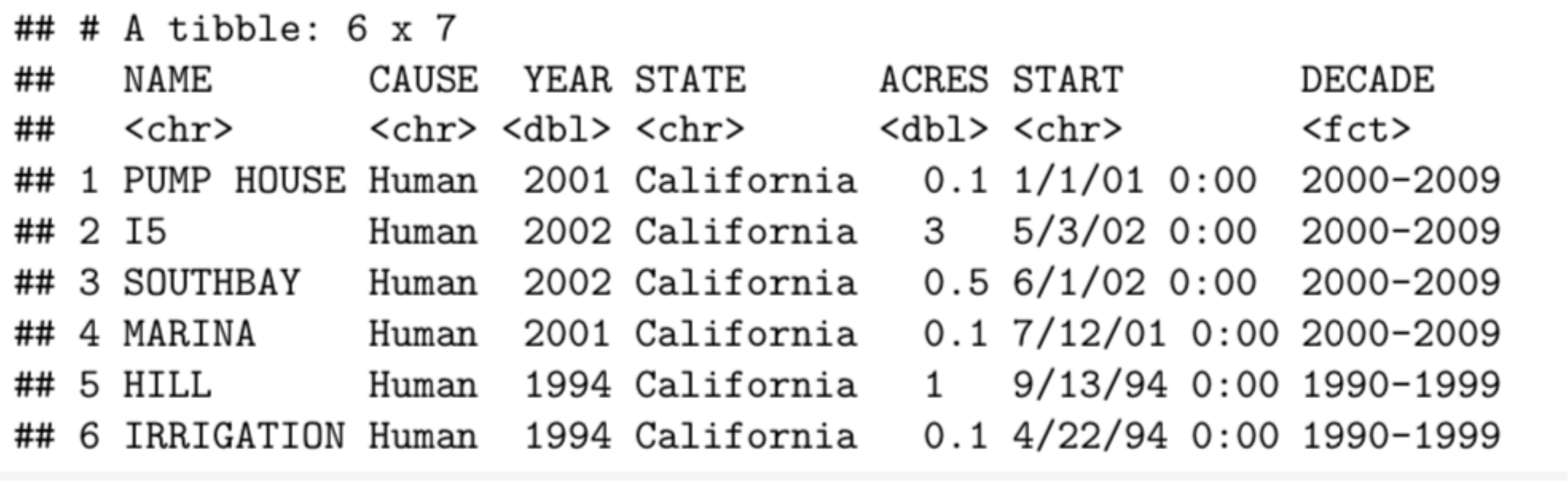
```
years = colnames(df)
years = years[3:6]
df2 = gather(df, years, YEAR, POPULATION)
```



	Country Name	Country Code	YEAR	POPULATION
1	Aruba	ABW	2010	101669
2	Afghanistan	AFG	2010	28803167
3	Angola	AGO	2010	23369131
4	Albania	ALB	2010	2913021
5	Andorra	AND	2010	84449
6	Arab World	ARB	2010	356508908
7	United Arab Emirates	ARE	2010	8270684
8	Argentina	ARG	2010	41223889
9	Armenia	ARM	2010	2877311
10	American Samoa	ASM	2010	55637
11	Antigua and Barbuda	ATG	2010	94661
12	Australia	AUS	2010	22031750
13	Austria	AUT	2010	8363404
14	Azerbaijan	AZE	2010	9054332

tidyverse libraries

- These libraries create dataframes called tibbles
- Tibbles are slight modification of dataframes



```
## # A tibble: 6 x 7
##   NAME       CAUSE  YEAR STATE      ACRES START         DECADE
##   <chr>      <chr> <dbl> <chr>      <dbl> <chr>         <fct>
## 1 PUMP HOUSE Human  2001 California   0.1 1/1/01 0:00   2000-2009
## 2 I5         Human  2002 California   3   5/3/02 0:00   2000-2009
## 3 SOUTHBAY   Human  2002 California   0.5 6/1/02 0:00   2000-2009
## 4 MARINA     Human  2001 California   0.1 7/12/01 0:00 2000-2009
## 5 HILL       Human  1994 California   1   9/13/94 0:00 1990-1999
## 6 IRRIGATION Human  1994 California   0.1 4/22/94 0:00 1990-1999
```

tidyverse libraries

- These libraries create dataframes called tibbles
- Tibbles are slight modification of dataframes
- In general, no need to convert, but if needed use

```
df2 = as_tibble(df1)

df1 = as.dataframe(df2)
```

df1: dataframe, df2: tibble

dplyr functions

- select( )
- filter( )
- arrange( )
- mutate( )
- summarize( )
- pipe

dplyr functions for dataframes

select( )	choose columns
filter( )	choose rows
arrange( )	sorting
mutate( )	create/add columns
summarize( )	summarizing by categories
%>%	create pipeline

select( )

- df2 = select(df1, column name)
- df2 = select(df1, new name = column name)
- df2 = select(df1, columns names)
- df2 = select(df1, helper functions)
- helper functions
  - starts_with(“string”)
  - ends_with(“string”)
  - contains(“string”)
  - matches(“string”)

select( )

- df2 = select(df1,SEX, GPA, GRE)
- df2 = select(df1, -c(GPA, GRE))
- df2 = select(df1, -c(3,5))
- df2 = select(df1,SEX, starts_with(“G”))

filter( )

- df2 = filter(df1, conditions)
- conditions
    - &	and
    - |	or
    - %in%	in a set

filter( )

- df2 = filter(df1, SEX == “male” & GPA>3.0)
- df2 = filter(df1, GPA > 3.5 | GRE > 165)
- df2 = filter(df1,SEX == “male”, GPA>3.0)
- categs = c(“single”,”married”)
- df2 = filter(df1,STATUS %in% categs)
- df2 = filter(df1,SEX == “male”, STATUS %in% categs)

arrange( )

- df2 = arrange(df1, column names)

- Examples
- df2 = arrange(df1, GPA, GRE)
- df2 = arrange(df1, desc(GPA), AGE)

mutate( )

- df1 = mutate(df1, new col_name = expression)
- Example: how much each student GPA is

  above or below the class GPA average
- df2 = mutate(df1, GPA_diff = GPA – mean(GPA, na.rm = TRUE))

mutate( )

- df1 = mutate(df1, new col_name = expression)
- Example: how much each student GPA is

  above or below the class GPA average
- df2 = mutate(df1, GPA_diff = GPA – mean(GPA, na.rm = TRUE))

with R base

- GPA_diff = GPA – mean(GPA, na.rm = TRUE)
- df2 = data.frame(df1,GPA_diff)

summarize( )

- summary statistics by categories
    - average, median
    - standard deviation, mad
    - min, max
    - first, last row by category
    - n() number of rows per category

summarize( )

- summary statistics by categories
  - average GPA by STATUS
  - max GPA by MAJOR
- two-steps

```
df_temp = group_by(df1, categorical variable)

summarize(df_temp, columns)
```

summarize( )

Find average and highest GPA by major

- df_temp = group_by(df1, major)
- summarize(df_temp,GPA_avg =mean(GPA,na.rm=TRUE),
  GPA_max =max(GPA,na.rm=TRUE))

summarize( )

Find average and highest GPA by major

- df_temp = group_by(df1, major)
- summarize(df_temp,GPA_avg =mean(GPA,na.rm=TRUE),

  GPA_max =max(GPA,na.rm=TRUE))
- df1 = tapply(df1$GPA,df1$MAJOR,mean)
- df2 = tapply(df1$GPA,df1$MAJOR,max)
- df = merge(df1,df2,by="MAJOR")

pipe

- A sequence of multiple operations
- Use %>% to separate each operation
- %>% comes from library magrittr
- All packages from tidyverse load %>% function
- Use library(magrittr) if tidyverse is not in use

pipe

- Sintax

```
df = read_csv(“StudyArea.csv”,col_names=TRUE) %>%
                    select(columns) %>%
                    filter(conditions) %>%
                    mutate(new column = expression)
```

%>% must go at the end of line (EOL)

pipe

Find major programs with average GPA > 3.0

```
df2 = df1 %>%
        group_by(major) %>%
        summarize(
            GPA_avg  = mean(GPA,na.rm=TRUE),
            GPA_max = max(GPA,na.rm=TRUE)
        ) %>%
        filter(GPA_avg > 3.0)
```