# An Introduction to Data Wrangling with R
# (or, an Introduction to R with Data Wrangling)

Rod Alence
International Relations and e-Science
University of the Witwatersrand

Workshop
Introduction to Emerging Methodologies in Social Science Research (IEMeSSR)
10–14 May 2021

# Aims of my sessions

My two sessions aim to introduce R as software for “data science” – understood to include:

1. Data “wrangling” (mainly this session)
   - Importing (possibly “messy”) data into R – from text files, spreadsheets, or other statistical programs (SPSS, Stata, SAS, etc.); and
   - “Tidying” data for analysis – getting the data into a rectangular (one-observation-per-row, one-variable-per-column) format.
2. Data analysis (mainly next session)
   - Transforming data (mathematical calculations and recoding);
   - Visualizing data (graphics); and
   - Modeling data (statistics).

# Approach of my sessions

## “Learning by doing” (as much as possible)

- Start with “traditional” slides/lecture format;
- Shift to “live coding” in R/RStudio as soon as possible (technology permitting);
- Recommend options for offline self-study.

## Programming within R

- Give “non-exclusive emphasis” to Base R over Tidyverse.

  (If you have no idea what that means, don’t worry!)

# Expectations for my sessions (beyond attendance!)

## What I do not expect

- Prior experience programming in R (a bonus if you have it!);
- Mathematical expertise beyond high-school algebra.

## What I do expect

- Some prior experience using some statistical software (e.g., SPSS or Stata);
- Basic familiarity with descriptive statistics and statistical models (not beyond least-squares linear regression).

# Outline

- On learning R
- Some R basics
    - Data in R
    - Functions in R
    - Read data into R
- Data “wrangling” with R: the UNDP Human Development Index
    - Read the data into R
    - “Tidy” the data
- Self-study in R/RStudio with `swirl`
- Wrapping up

# Outline

- On learning R
- Some R basics
    - Data in R
    - Functions in R
    - Read data into R
- Data “wrangling” with R: the UNDP Human Development Index
    - Read the data into R
    - “Tidy” the data
- Self-study in R/RStudio with `swirl`
- Wrapping up

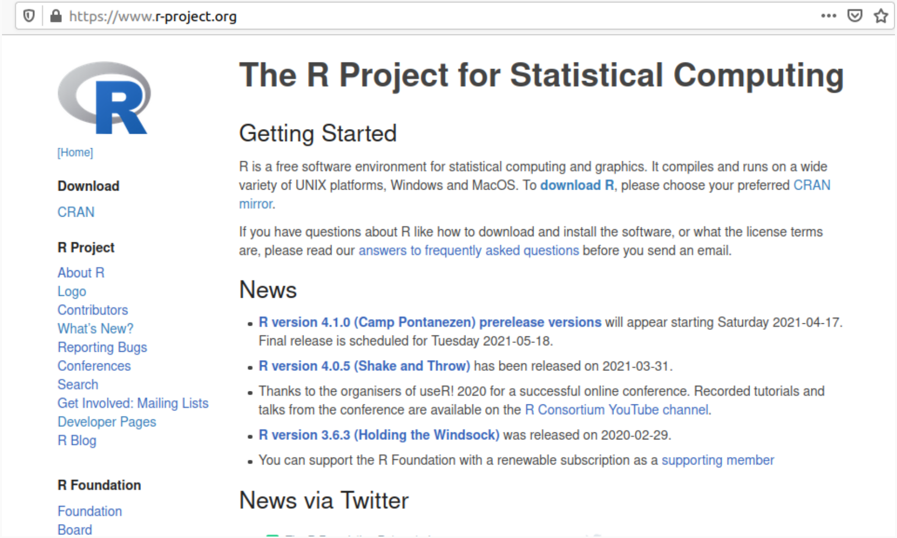
# What is R?

- Free, open-source statistical software that runs on all major operating systems;
- Created in the mid-1990s at the University of Auckland, New Zealand, by Ross Ihaka and Robert Gentleman as an implementation of the S programming language;
- Now maintained by a volunteer Core Development Team, which releases an updated version about twice a year;
- New and updated add-on “packages” appear weekly – more than 17,000 now available;
- For more information: http://www.r-project.org

# Why R?

- Is probably the most powerful software for statistical analysis;
- Has the best graphics capabilities;
- Its package system is “going viral” (in a good way);
- Is “free” – as intellectual property and in price.

# The R Project website



https://www.r-project.org

[Home]

**Download**

CRAN

**R Project**

About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Developer Pages
R Blog

**R Foundation**

Foundation
Board

## The R Project for Statistical Computing

### Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

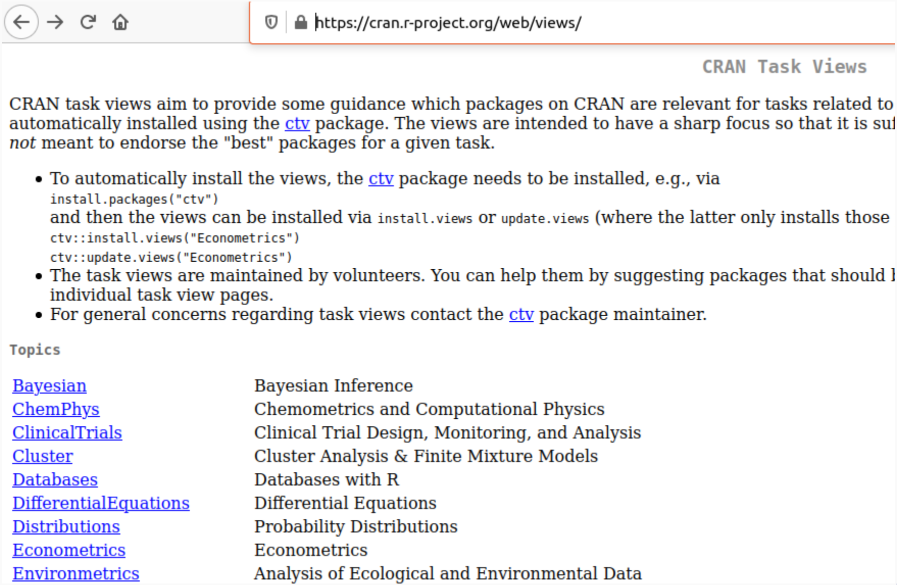
If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

### News

- **R version 4.1.0 (Camp Pontanezen) prerelease versions** will appear starting Saturday 2021-04-17. Final release is scheduled for Tuesday 2021-05-18.
- **R version 4.0.5 (Shake and Throw)** has been released on 2021-03-31.
- Thanks to the organisers of useR! 2020 for a successful online conference. Recorded tutorials and talks from the conference are available on the R Consortium YouTube channel.
- **R version 3.6.3 (Holding the Windsock)** was released on 2020-02-29.
- You can support the R Foundation with a renewable subscription as a supporting member

### News via Twitter

# R package “task views”



CRAN Task Views

CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to automatically installed using the ctv package. The views are intended to have a sharp focus so that it is suf *not* meant to endorse the "best" packages for a given task.

- To automatically install the views, the ctv package needs to be installed, e.g., via

  ```
  install.packages("ctv")
  ```

  and then the views can be installed via `install.views` or `update.views` (where the latter only installs those

  ```
  ctv::install.views("Econometrics")
  ctv::update.views("Econometrics")
  ```

- The task views are maintained by volunteers. You can help them by suggesting packages that should l individual task view pages.
- For general concerns regarding task views contact the ctv package maintainer.

Topics

| | |
|---|---|
| Bayesian | Bayesian Inference |
| ChemPhys | Chemometrics and Computational Physics |
| ClinicalTrials | Clinical Trial Design, Monitoring, and Analysis |
| Cluster | Cluster Analysis & Finite Mixture Models |
| Databases | Databases with R |
| DifferentialEquations | Differential Equations |
| Distributions | Probability Distributions |
| Econometrics | Econometrics |
| Environmetrics | Analysis of Ecological and Environmental Data |

If statistics programs/languages were cars...
Excel
SPSS
SAS
STATA
R

https://twitter.com/statsepi/status/795574223439876100

# Does R have a “steep learning curve”?

## The two most challenging things about R

1. It is entirely command (“expression”) based – you type commands, and R executes them (no “point-and-click” menus).
2. It allows multiple (unlimited) data “objects” in a session simultaneously.

## But – these features are essential to R’s strengths

- No menu system could ever keep up with software as powerful and dynamic as R.
- Allowing multiple objects is essential to a programming language in which the output of nearly any command can be the input of another.

And... RStudio makes learning and executing R command syntax

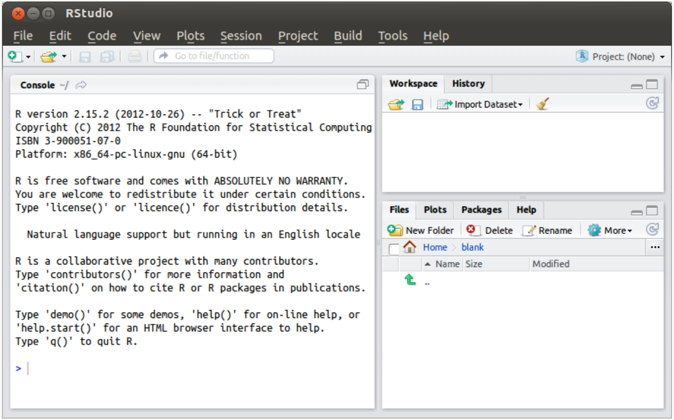
# What is RStudio?

- An “integrated development environment” (IDE) for R (but not a “point-and-click” interface to R commands);
- Launched in 2011;
- Free and open source;
- Available for all major operating systems (Windows, MacOS, and Linux);
- For more information: http://www.rstudio.org

# RStudio at first start-up

## Three windows

### R console occupies full left side



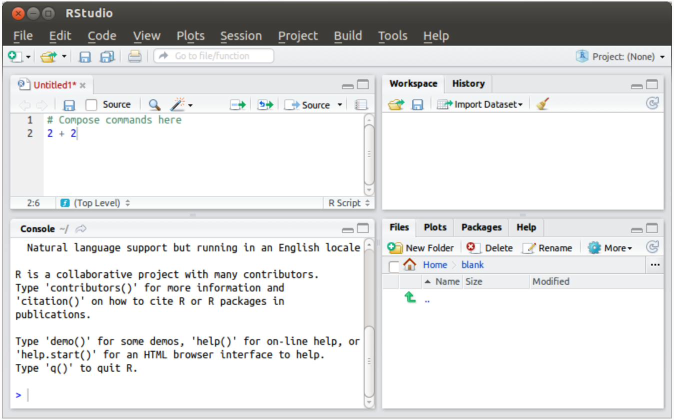
RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
Console ~/
Workspace History
Import Dataset
Files Plots Packages Help
New Folder Delete Rename More
Home blank
Name Size Modified
..

```
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

# RStudio with editor window open (the usual way)

## Four windows

Left side split between editor (top) and R console (bottom)



```
# Compose commands here
2 + 2
```

```
 Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in
publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

# Outline

On learning R

Some R basics

- Data in R
- Functions in R
- Read data into R

Data “wrangling” with R: the UNDP Human Development Index

- Read the data into R
- “Tidy” the data

Self-study in R/RStudio with `swirl`

Wrapping up

# Outline

- On learning R
- Some R basics
    - Data in R
    - Functions in R
    - Read data into R
- Data “wrangling” with R: the UNDP Human Development Index
    - Read the data into R
    - “Tidy” the data
- Self-study in R/RStudio with `swirl`
- Wrapping up

## Data structures: vectors

The most basic data structure in R is the vector – which is just a fancy word for “a list of things in a particular order.”

- Even a single number is a vector to R – it is a vector that happens to contain only one thing.
- When a vector holds data about units, like a column in a spreadsheet, a vector is synonymous with what is often called a “variable.”
- Each vector has two intrinsic structural attributes:

  Length

  How many things (elements) does it contain?

  Mode

  What kinds of things does it contain – e.g., numeric values, typographic characters?

# Data structures: data frames

A data frame is a rectangular, spreadsheet-like data structure – typically organized with “observations” in rows and “variables” in columns.

Data frames

- May contain column vectors of any class; but
- Must contain column vectors of the same length.

The collection of packages known as the “Tidyverse” often use a special type of data frame called the tibble, which

- Has the same basic structure as the “traditional” data frame, with a few distinct features; and
- Can easily be converted to the “traditional” data frame.

# Assigning names to data

Any non-trivial “data” is assigned a name for further use, using the “backward-arrow” operator.

For example, we can “stick together” some numbers as a vector using `c` (for combine) and assign it a name, like `some_numbers`:

```r
some_numbers <- c(1, 2, 3, 4, 5)
```

And we can do the same with some letters (in quotation marks):

```r
some_letters <- c("e", "d", "c", "b", "a")
```

Assignment is “silent” – but we can check the objects’ contents by typing its name and pressing return.

```r
some_numbers
# [1] 1 2 3 4 5

some_letters
# [1] "e" "d" "c" "b" "a"
```

# Combining vectors in a data frame

Because the two vectors assigned in the previous slide are the same length, we can stick them together side-by-side in a data frame using `data.frame`.

```
boring_df <- data.frame(some_numbers, some_letters)
```

And to view the data frame, enter its name.

```
boring_df
#   some_numbers some_letters
# 1            1            e
# 2            2            d
# 3            3            c
# 4            4            b
# 5            5            a
```

# Notes on naming

## R’s rules about names

- May contain lower-case and upper-case letters, numbers, dots (`.`), and underscores (`_`).
- May not start with numbers – and they should almost always start with letters.
- Are case-sensitive – lower-case and upper-case versions of the same letter are treated as entirely different characters.
- Overwrite any existing object with the same name.

## Common sense about names

- Should be concise (to avoid too much typing).
- Should be Informative (to clarify content).

# Numeric indexing

Elements of data structures can be accessed by position using numeric square-bracket indexes.

## Vectors

```
some_letters
# [1] "e" "d" "c" "b" "a"

some_letters[4]  # get the fourth element
# [1] "b"
```

## Data frames

```
boring_df
#   some_numbers some_letters
# 1            1            e
# 2            2            d
# 3            3            c
# 4            4            b
# 5            5            a

boring_df[3, 2]  # row-by-col (third row, second col)
# [1] "c"
```

# Indexing columns (variables) in data frames

Three common ways to select a (column) variable in a data frame:

1. By numeric position

```
boring_df[ , 2]  # get the second col (all rows)
# [1] "e" "d" "c" "b" "a"
```

2. By column name

```
boring_df[ , "some_letters"]  # get the col called "some_letters"
# [1] "e" "d" "c" "b" "a"
```

3. Dollar-sign (list) notation

```
boring_df$some_letters
# [1] "e" "d" "c" "b" "a"
```

# Indexing rows (observations) in data frames

Two common ways to select rows in a data frame:

1. By numeric position

```
boring_df[c(1, 3), ]  # get the first and third rows (all cols)
#   some_numbers some_letters
# 1            1            e
# 3            3            c
```

2. By logical expression

```
## Get rows in which the logical expression holds
boring_df[boring_df$some_numbers > 3, ]
#   some_numbers some_letters
# 4            4            b
# 5            5            a
```

# Outline

On learning R

Some R basics

- Data in R
- Functions in R
- Read data into R

Data “wrangling” with R: the UNDP Human Development Index

- Read the data into R
- “Tidy” the data

Self-study in R/RStudio with `swirl`

Wrapping up

# Functions in R

Functions are what “do things” in R – if data objects are like nouns, functions are the verbs.

## Function syntax

To use a function:

1. Type its name (exactly, remember case-sensitivity),
2. Followed immediately by parentheses (curved brackets),
3. Insert any inputs (“arguments”) inside the parentheses, separated by commas.

Often the reason for using a function is to generate output which is immediately assigned to an object.

# An example using functions

Which two functions are used here?

```r
marks <- c(78, 56, 91, 88, NA, 62, 67)  # one student was absent
class_ave <- mean(marks, na.rm=TRUE)
class_ave
# [1] 73.66667
```

Each function has a help page, which explains what the function does and what inputs it takes.

Typing a question mark followed by a function name calls up the help page. To find out what the `na.rm=TRUE` is about, try entering `?mean` in the R console.

# Outline

- On learning R
- Some R basics
  - Data in R
  - Functions in R
  - Read data into R
- Data “wrangling” with R: the UNDP Human Development Index
  - Read the data into R
  - “Tidy” the data
- Self-study in R/RStudio with `swirl`
- Wrapping up

# Reading data into R

R has functions for reading in data in various formats, for example:

- Comma- or tab-delimited text: `read.csv` and `read.delim` in Base R, or `read_csv` and `read_tsv` in the readr package
- Spreadsheets (.xls, .xlsx): `read_excel` in the readxl package;
- Stata (.dta): `read.dta` (Stata 5-12) in the foreign package, `read.dta13` (Stata 13 onwards) in the readstata13 package, or `read_dta` (all versions) in the haven package;
- SPSS (.sav): `read.spss` in the foreign package, or `read_spss` in the haven package.

# Outline

On learning R

Some R basics

- Data in R
- Functions in R
- Read data into R

Data “wrangling” with R: the UNDP Human Development Index

- Read the data into R
- “Tidy” the data

Self-study in R/RStudio with `swirl`

Wrapping up

# Setting up

## Install two packages

The installation only needs to be run once – I use the console.

```
install.packages(c("readxl", "tidyverse"))
```

## Download the data spreadsheet

The download only needs to be run once (if the data set is static).

```
## Break up the url for convenience, because it is long
site_url <- "http://hdr.undp.org/"
path_url <- "sites/default/files/"
fn_url   <- "2020_statistical_annex_table_1.xlsx"
## Paste the parts together
link_url <- paste0(site_url, path_url, fn_url)

## Download using the full url
download.file(url=link_url,
              destfile="hdi2020.xlsx")
```

# Outline

On learning R

Some R basics

- Data in R
- Functions in R
- Read data into R

Data “wrangling” with R: the UNDP Human Development Index

- Read the data into R
- “Tidy” the data

Self-study in R/RStudio with `swirl`

Wrapping up

# Read in the spreadsheet (and have a quick look)

```
library(readxl)
HDI <- read_excel("hdi2020.xlsx",
                  range = "B8:K200",  # cells to read
                  col_names = FALSE,  # column names not read
                  na = c("", ".."))   # missing value strings
dim(HDI)
# [1] 193  10

head(HDI)
# # A tibble: 6 x 10
#   ...1        ...2 ...3   ...4 ...5   ...6 ...7   ...8 ...9
#   <chr>      <dbl> <lgl> <dbl> <chr> <dbl> <chr> <dbl> <chr>
# 1 VERY HIG~ NA     NA     NA   <NA>   NA   <NA>   NA   <NA>
# 2 Norway     0.957 NA     82.4 <NA>   18.1 b      12.9 <NA>
# 3 Ireland    0.955 NA     82.3 <NA>   18.7 b      12.7 <NA>
# 4 Switzerl~  0.955 NA     83.8 <NA>   16.3 <NA>   13.4 <NA>
# 5 Hong Kon~  0.949 NA     84.9 <NA>   16.9 <NA>   12.3 <NA>
# 6 Iceland    0.949 NA     83.0 <NA>   19.1 b      12.8 c
# # ... with 1 more variable: ...10 <dbl>
```

# The structure of the data frame

The data frame is still a bit “messy.”

```
str(HDI)
# tibble[,10] [193 x 10] (S3: tbl_df/tbl/data.frame)
#  $ ...1 : chr [1:193] "VERY HIGH HUMAN DEVELOPMENT" "Norway" "Ireland
#  $ ...2 : num [1:193] NA 0.957 0.955 0.955 0.949 0.949 0.947 0.945 0.
#  $ ...3 : logi [1:193] NA NA NA NA NA NA ...
#  $ ...4 : num [1:193] NA 82.4 82.3 83.8 84.9 ...
#  $ ...5 : chr [1:193] NA NA NA NA ...
#  $ ...6 : num [1:193] NA 18.1 18.7 16.3 16.9 ...
#  $ ...7 : chr [1:193] NA "b" "b" NA ...
#  $ ...8 : num [1:193] NA 12.9 12.7 13.4 12.3 ...
#  $ ...9 : chr [1:193] NA NA NA NA ...
#  $ ...10: num [1:193] NA 66494 68371 69394 62985 ...
```

## Outline

On learning R

Some R basics

- Data in R
- Functions in R
- Read data into R

Data “wrangling” with R: the UNDP Human Development Index

- Read the data into R
- “Tidy” the data

Self-study in R/RStudio with `swirl`

Wrapping up

# Remove unneeded columns

## Which columns are not needed?

```
head(HDI)
# # A tibble: 6 x 10
#   ...1        ...2 ...3   ...4 ...5   ...6 ...7   ...8 ...9
#   <chr>      <dbl> <lgl> <dbl> <chr> <dbl> <chr> <dbl> <chr>
# 1 VERY HIG~ NA     NA     NA   <NA>   NA   <NA>   NA   <NA>
# 2 Norway      0.957 NA     82.4 <NA>   18.1 b      12.9 <NA>
# 3 Ireland     0.955 NA     82.3 <NA>   18.7 b      12.7 <NA>
# 4 Switzerl~   0.955 NA     83.8 <NA>   16.3 <NA>   13.4 <NA>
# 5 Hong Kon~   0.949 NA     84.9 <NA>   16.9 <NA>   12.3 <NA>
# 6 Iceland     0.949 NA     83.0 <NA>   19.1 b      12.8 c
# # ... with 1 more variable: ...10 <dbl>
```

## Remove the “skinny” footnote columns.

```
## Use negative column indexes to remove columns
HDI <- HDI[ , -c(3, 5, 7, 9)]  # negative column index to remove

## Tidyverse alternative (dplyr package) (NOT RUN)
## HDI <- dplyr::select(HDI, -c(3, 5, 7, 9))
```

# Add meaningful column names

Add meaningful (informative and concise) names by “assigning into” the data frame’s `colnames`:

```
## Old column names
colnames(HDI)
# [1] "...1"  "...2"  "...4"  "...6"  "...8"  "...10"

## Assign new column names
colnames(HDI) <- c("country", "hdi", "life_exp",
                   "school_exp", "school_mean", "gni_pc")
head(HDI)
# # A tibble: 6 x 6
#   country         hdi life_exp school_exp school_mean gni_pc
#   <chr>         <dbl>    <dbl>      <dbl>       <dbl>  <dbl>
# 1 VERY HIGH H~ NA         NA         NA          NA      NA
# 2 Norway        0.957     82.4       18.1        12.9 66494.
# 3 Ireland       0.955     82.3       18.7        12.7 68371.
# 4 Switzerland   0.955     83.8       16.3        13.4 69394.
# 5 Hong Kong, ~  0.949     84.9       16.9        12.3 62985.
# 6 Iceland       0.949     83.0       19.1        12.8 54682.
```

# Remove unneeded rows

## Which rows are not needed?

```
head(HDI, n=2)
# # A tibble: 2 x 6
#   country          hdi life_exp school_exp school_mean gni_pc
#   <chr>          <dbl>    <dbl>      <dbl>       <dbl>  <dbl>
# 1 VERY HIGH H~ NA          NA         NA          NA       NA
# 2 Norway         0.957     82.4       18.1        12.9 66494.

HDI[HDI$country == "MEDIUM HUMAN DEVELOPMENT", ]
# # A tibble: 1 x 6
#   country          hdi life_exp school_exp school_mean gni_pc
#   <chr>          <dbl>    <dbl>      <dbl>       <dbl>  <dbl>
# 1 MEDIUM HUMAN~     NA       NA         NA          NA     NA
```

Remove rows with no numeric data (e.g., in the `hdi` column).

```
HDI <- HDI[! is.na(HDI$hdi), ]

## Tidyverse alternative (NOT RUN)
## HDI <- dplyr::filter(HDI, ! is.na(HDI$hdi))
```

# Check the data structure

Things to check:

- Dimensions (rows by columns);
- Column names;
- Object “classes” (e.g., character vs. numeric).

```
str(HDI)
# tibble[,6] [189 x 6] (S3: tbl_df/tbl/data.frame)
#  $ country    : chr [1:189] "Norway" "Ireland" "Switzerland" "Hong Ko
#  $ hdi        : num [1:189] 0.957 0.955 0.955 0.949 0.949 0.947 0.945
#  $ life_exp   : num [1:189] 82.4 82.3 83.8 84.9 83  ...
#  $ school_exp : num [1:189] 18.1 18.7 16.3 16.9 19.1  ...
#  $ school_mean: num [1:189] 12.9 12.7 13.4 12.3 12.8  ...
#  $ gni_pc     : num [1:189] 66494 68371 69394 62985 54682  ...
```

## Check the data summary

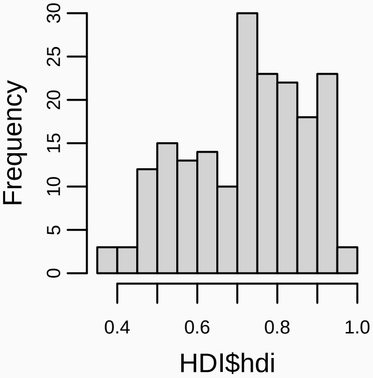
Things to check:

- Descriptive statistics;
- Missing values (NA frequencies are reported for each variable).

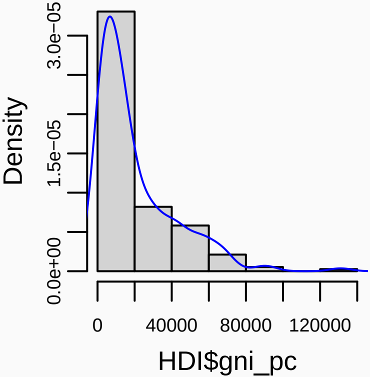
```
summary(HDI)
#    country               hdi            life_exp    
#  Length:189         Min.   :0.3940   Min.   :53.28  
#  Class  :character   1st Qu.:0.6020   1st Qu.:67.44  
#  Mode   :character   Median :0.7400   Median :74.05  
#                      Mean   :0.7224   Mean   :72.71  
#                      3rd Qu.:0.8290   3rd Qu.:77.91  
#                      Max.   :0.9570   Max.   :84.86  
#    school_exp      school_mean         gni_pc       
#  Min.   : 5.005   Min.   : 1.644   Min.   :   753.9  
#  1st Qu.:11.431   1st Qu.: 6.437   1st Qu.:  4910.2  
#  Median :13.188   Median : 9.032   Median : 12707.4  
#  Mean   :13.325   Mean   : 8.728   Mean   : 20219.7  
#  3rd Qu.:15.227   3rd Qu.:11.326   3rd Qu.: 29497.2  
#  Max.   :21.954   Max.   :14.152   Max.   :131031.6  
```

# Run a few histograms?

```r
hist(HDI$hdi,
     main="")
```



```r
hist(HDI$gni_pc,
     freq=FALSE,
     main="")
lines(density(HDI$gni_pc),
      col="blue")
```



# Outline

On learning R

Some R basics

- Data in R
- Functions in R
- Read data into R

Data “wrangling” with R: the UNDP Human Development Index

- Read the data into R
- “Tidy” the data

Self-study in R/RStudio with `swirl`

Wrapping up

# What is swirl

swirl

An R package that provides the infrastructure to run interactive self-study lessons, right in the R console.

(For more information: https://swirlstats.com/.)

“Introduction to R Programming”

A foundational swirl course consisting of 15 short lessons (work out your own pace, but most take about 15–20 minutes each).

# Install and run `swirl` courses

## Install (once-off)

```r
## Install the swirl package -- the "infrastructure"
install.packages("swirl")

## Install the R programming course -- the content
swirl::install_course("R Programming")
```

## Run the course

```r
## "Load" (attach) the swirl package
library(swirl)

## Run swirl
swirl()  # follow the prompts to choose a course, lessons
```

(Mostly do the courses in order, except you may want to skip the second one on “Workspace and Files” and come back to it later.)

# Outline

- On learning R
- Some R basics
    - Data in R
    - Functions in R
    - Read data into R
- Data “wrangling” with R: the UNDP Human Development Index
    - Read the data into R
    - “Tidy” the data
- Self-study in R/RStudio with `swirl`
- Wrapping up

# Wrapping up

## Learning R

- Push ahead with the `swirl` lessons on “R Programming”;
- For further self-study, with a Tidyverse focus, try:

  Hadley Wickham and Garrett Grolemund, *R for Data Science* (O’Reilly Media, 2017) – available for free online at https://r4ds.had.co.nz/.

## For tomorrow

- Data *analysis* in R.