# rpapoda Phase 1: Setting the Stage

## Goals

### Goals of this Phase:

- Setup your project
- Find and import data (if necessary)
- Inspect the data
- Ask a question
- Tidy the data, do any other necessary formatting of the data, and document each step along the way
- Create a code book

### Other Goals of this Lesson:

- Intro to R and RStudio
- Intro to Reproducible Research and Literate Programming
- Version Control
- Principles of tidy data
- Intro to working with dates in R

## Setup Your Project

We saw in the intro that the best way to keep your analysis, data, and other documents and materials organized in RStudio is with the “Projects” feature. Each analysis or (fittingly) project that you work on should have its own Project. Projects in RStudio are very simple and straightforward to use, and we will walk through setting one up to use for the remainder of this course.

## Find data (if necessary), and import it

It’s very important to note the interconnectedness between the first four goals listed above. In a formal setting, the analyst may be given a question to answer, then be tasked with collecting the data, or both may be given at the same time. In our informal, educational setting, we will begin with the data in hand, inspect it, then find an appropriate question. Before you do anything in R, you will create your new project.

Its worth mentioning that there are many interesting datasets freely available today. Excellent places to find datasets for your own exploration include the following:
Kaggle (https://www.kaggle.com/datasets)
HealthData.gov (https://www.healthdata.gov/search/type/dataset)
Data.gov (https://catalog.data.gov/dataset)
WorldBank (https://data.worldbank.org/)
The US Census Bureau (https://www.census.gov/data.html)
Amazon Web Services Public Datasets (https://aws.amazon.com/datasets/?_encoding=UTF8&jiveRedirect=1)

These are just a few that are worth browsing, both for datasets and ideas. I’ve also had a lot of luck simply using google if I have something specific I’m looking for.

For this course, I found a very large dataset scraped from Kickstarter.com at https://webrobots.io/kickstarter-datasets/ (https://webrobots.io/kickstarter-datasets/). The data provides information on individual Kickstarter projects. I chose this dataset in hopes that it might provide an interesting look into crowdfunding, a hot trend among small businesses and hopeful startups. Whether or not we actually crack the “code” of Kickstarter is not the point, as our goal is to learn the process of data analysis. The entire dataset contains over 169,000 entries going back to 2009. I took a sample of the dataset to make the size more manageable as we simply didn’t need (or want) that large of a dataset for our purposes. We will learn how to take a sample of a dataset soon.

If you haven’t done so already, you can download the dataset here:

In order to import the data, we will need to load a package to do so. The function we will be using is a part of the `readr` package. We can load it individually, or we can load the `tidyverse` package which contains the `readr` package and many other important packages. You will use the group of packages in `tidyverse` in essentially every R session for the rest of this course.

Remember from the Introduction to R and Rstudio lesson that before you load a package for the first time, you must install it with `install.packages()`. In this case, run `install.packages("tidyverse")` in the console or elsewhere if you never have before.

Load the `tidyverse` package:

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

To import the dataset, we will use the `read_csv()` function from `readr`, and we will store it in an object called `ks`.

The only argument we need here is the name of the csv file, with extension and closed in double quotes.

```
ks <- read_csv("kickstarter_data.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   id = col_integer(),
##   goal = col_double(),
##   pledged = col_double(),
##   currency_trailing_code = col_logical(),
##   deadline = col_datetime(format = ""),
##   state_changed_at = col_datetime(format = ""),
##   created_at = col_datetime(format = ""),
##   launched_at = col_datetime(format = ""),
##   staff_pick = col_logical(),
##   is_starrable = col_logical(),
##   backers_count = col_integer(),
##   static_usd_rate = col_double(),
##   usd_pledged = col_double(),
##   spotlight = col_logical()
## )
```

```
## See spec(...) for full column specifications.
```

# Initial Inspection of the Data

Now that we have imported our dataset, we will inspect it. When we inspect our dataset, we are simply trying to acquaint ourselves with it so that we know what we are working with, if it has any major problems, and whether or not it might be able to help us answer our question. Datasets are oftentimes too big to actually look at the whole thing, so we will focus on looking at the structure of the data, summaries of some of the data, and bite-sized pieces of the data. This is a very surface level look at the data. We are not yet trying to really begin to answer our question or gain insight from our data. I liken it to the “What’s your name? Where are you from?” type of questions we ask when we first meet a stranger. We aren’t leading off with a question about their deepest desires and motivations.

For the first pass of inspection of the dataset, my main goal is simply to get an idea of what questions it will be most suited to answer. At this point in the process, we should not be changing or modifying our original data in any way.

A few common tasks typically included in the preliminary inspection of a dataset:

- The number of observations
- The number of variables
- Number of distinct observations
- The names of the variables
- A summary of the data
- How much data we are missing
- The first few observations
- The last few observations

Let’s dive into inspection of the data. How many observations do we have?

```
nrow(ks)
```

```
## [1] 500
```

How many variables are there?

```
ncol(ks)
```

```
## [1] 31
```

And an easier way to find both the number of observations and the number of variables:

```
dim(ks)
```

```
## [1] 500  31
```

Checking distinct observations (checking for duplicates):

```
n_distinct(ks)
```

```
## [1] 500
```

What variables do we have?

```
names(ks)
```

```
##  [1] "id"                    "photo"
##  [3] "name"                  "blurb"
##  [5] "goal"                  "pledged"
##  [7] "state"                 "slug"
##  [9] "country"               "currency"
## [11] "currency_trailing_code" "deadline"
## [13] "state_changed_at"      "created_at"
## [15] "launched_at"           "staff_pick"
## [17] "is_starrable"          "backers_count"
## [19] "static_usd_rate"       "usd_pledged"
## [21] "creator"               "location"
## [23] "category"              "profile"
## [25] "spotlight"             "urls"
## [27] "source_url"            "friends"
## [29] "is_starred"            "is_backing"
## [31] "permissions"
```

And a summary of the data by variable:

```
summary(ks)
```

```
## Warning in as.POSIXlt.POSIXct(x, tz): unknown timezone 'zone/tz/2019b.1.0/
## zoneinfo/America/Denver'
```

```
##        id              photo              name
##  Min.   :3.703e+06   Length:500         Length:500
##  1st Qu.:5.918e+08   Class :character   Class :character
##  Median :1.162e+09   Mode  :character   Mode  :character
##  Mean   :1.124e+09
##  3rd Qu.:1.649e+09
##  Max.   :2.147e+09
##     blurb                goal           pledged
##  Length:500         Min.   :    10   Min.   :     0.00
##  Class :character   1st Qu.:  2000   1st Qu.:    54.75
##  Mode  :character   Median :  5000   Median :  1076.00
##                     Mean   : 12673   Mean   :  7434.08
##                     3rd Qu.: 12000   3rd Qu.:  5503.58
##                     Max.   :125000   Max.   :172586.00
##     state              slug             country
##  Length:500         Length:500         Length:500
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##   currency         currency_trailing_code    deadline
##  Length:500         Mode :logical          Min.   :2010-03-07 05:00:00
##  Class :character   FALSE:60               1st Qu.:2013-09-16 12:26:19
##  Mode  :character   TRUE :440              Median :2015-03-28 19:49:20
##                     NA's :0                Mean   :2014-12-07 00:39:29
##                                            3rd Qu.:2016-04-16 15:34:53
##                                            Max.   :2017-09-19 22:21:48
##  state_changed_at               created_at
##  Min.   :2010-03-07 05:00:10   Min.   :2010-01-13 17:25:44
##  1st Qu.:2013-09-16 12:26:56   1st Qu.:2013-06-26 11:01:40
##  Median :2015-03-28 19:49:21   Median :2015-01-05 03:29:48
##  Mean   :2014-12-04 11:11:37   Mean   :2014-09-21 12:36:46
##  3rd Qu.:2016-04-15 06:05:55   3rd Qu.:2016-02-07 05:27:22
##  Max.   :2017-08-15 18:01:03   Max.   :2017-08-10 17:16:07
##   launched_at                  staff_pick      is_starrable
##  Min.   :2010-01-13 22:35:38   Mode :logical   Mode :logical
##  1st Qu.:2013-08-11 02:22:38   FALSE:440       FALSE:488
##  Median :2015-02-23 10:40:26   TRUE :60        TRUE :12
##  Mean   :2014-11-03 00:13:57   NA's :0         NA's :0
##  3rd Qu.:2016-03-15 20:33:17
##  Max.   :2017-08-15 18:01:02
##  backers_count     static_usd_rate    usd_pledged       creator
##  Min.   :   0.00   Min.   :0.05474   Min.   :     0   Length:500
##  1st Qu.:   2.00   1st Qu.:1.00000   1st Qu.:    60   Class :character
##  Median :  21.00   Median :1.00000   Median :  1100   Mode  :character
##  Mean   :  87.84   Mean   :1.00932   Mean   :  6632
##  3rd Qu.:  72.25   3rd Qu.:1.00000   3rd Qu.:  5145
##  Max.   :3399.00   Max.   :1.69769   Max.   :152604
##    location           category           profile          spotlight
##  Length:500         Length:500         Length:500         Mode :logical
##  Class :character   Class :character   Class :character   FALSE:264
##  Mode  :character   Mode  :character   Mode  :character   TRUE :236
##                                                           NA's :0
##
##
##     urls            source_url          friends
##  Length:500         Length:500         Length:500
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##   is_starred         is_backing        permissions
##  Length:500         Length:500         Length:500
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

Note that `summary()` provides some information on NA's. However, I find the information regarding NA's in this table to be insconsistent, and I use another method of finding them that I will show in just a minute.

Also, note that the dates don't look much like dates. We'll deal with those when we begin to clean things up.

And a similar way to summarise the data that may be more useful in some situations:

```
str(ks)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    500 obs. of  31 variables:
##  $ id                    : int  832570985 1054135507 1567381435 1329110416 1403665672 568986755 468138435 305131391 10980
19088 700642359 ...
##  $ photo                 : chr  "{\"small\":\"https://ksr-ugc.imgix.net/assets/011/615/756/7bd5961037ef2e1043d39e7e14e8f9
a7_original.jpg?crop=faces&w=160&h=90&f"| __truncated__ "{\"small\":\"https://ksr-ugc.imgix.net/assets/011/713/648/8f2762ddd
84f6076b63144d796720c58_original.jpg?crop=faces&w=160&h=90&f"| __truncated__ "{\"small\":\"https://ksr-ugc.imgix.net/assets/
011/729/113/484d0974b3f352e77ef00f370c4a6740_original.jpg?crop=faces&w=160&h=90&f"| __truncated__ "{\"small\":\"https://ksr-
ugc.imgix.net/assets/011/859/266/cf080ee9fd90215e19a76ad3f4272890_original.jpg?crop=faces&w=160&h=90&f"| __truncated__ ...
##  $ name                  : chr  "Traces 2014" "Dan Phelps' CD-\"Going Home\" promo" "RestorationBid-easiest way to hire P
ROs for home improvement" "Trill League Comic" ...
##  $ blurb                 : chr  "Traces is an immersive and theatrical multi-disciplinary art & design event inspired by
real stories and histories of buildings"| __truncated__ "Dan Phelps is preparing to release a new album, and we're reaching
out to his fans to help get this album out to the world." "Restorationbid.com marketplace is the first of its kind - bridgin
g the gap for consumers to hire local service professionals." "Trill League is hilarious parody combining the world of super
heroes, anime and hip-hop culture." ...
##  $ goal                  : num  2300 1700 75000 7500 5000 109000 250 750 1000 5000 ...
##  $ pledged               : num  2371 205 130 20917 60 ...
##  $ state                 : chr  "successful" "failed" "failed" "successful" ...
##  $ slug                  : chr  "traces-2014" "dan-phelps-cd-going-home-promo" "restorationbid-easiest-way-to-hire-pros-f
or-home-i" "trill-league-comic" ...
##  $ country               : chr  "GB" "US" "US" "US" ...
##  $ currency              : chr  "GBP" "USD" "USD" "USD" ...
##  $ currency_trailing_code: logi  FALSE TRUE TRUE TRUE TRUE TRUE ...
##  $ deadline              : POSIXct, format: "2014-01-27 17:00:00" "2014-07-05 03:19:57" ...
##  $ state_changed_at      : POSIXct, format: "2014-01-27 17:00:17" "2014-07-05 03:19:57" ...
##  $ created_at            : POSIXct, format: "2013-11-30 15:26:36" "2014-05-28 01:22:31" ...
##  $ launched_at           : POSIXct, format: "2014-01-01 15:44:06" "2014-06-05 03:19:57" ...
##  $ staff_pick            : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ is_starrable          : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ backers_count         : int  112 8 13 431 2 1 15 22 26 2 ...
##  $ static_usd_rate       : num  1.65 1 1 1 1 ...
##  $ usd_pledged           : num  3908 205 130 20917 60 ...
##  $ creator               : chr  "Traces" "Dan Phelps CD release" "Restoration Bid Inc." "Anthony Piper" ...
##  $ location              : chr  "London, UK" "Roscoe, IL" "Schaumburg, IL" "Chicago, IL" ...
##  $ category              : chr  "Mixed Media" "Blues" "Web" "Comic Books" ...
##  $ profile               : chr  "{\"background_image_opacity\":0.8,\"should_show_feature_image_section\":true,\"link_text
_color\":null,\"state_changed_at\":1425"| __truncated__ "{\"background_image_opacity\":0.8,\"should_show_feature_image_secti
on\":true,\"link_text_color\":null,\"state_changed_at\":1425"| __truncated__ "{\"background_image_opacity\":0.8,\"should_sho
w_feature_image_section\":true,\"link_text_color\":null,\"state_changed_at\":1425"| __truncated__ "{\"background_image_opaci
ty\":0.8,\"should_show_feature_image_section\":true,\"link_text_color\":\"\",\"state_changed_at\":1439"| __truncated__ ...
##  $ spotlight             : logi  TRUE FALSE FALSE TRUE FALSE FALSE ...
##  $ urls                  : chr  "{\"web\":{\"project\":\"https://www.kickstarter.com/projects/1914203846/traces-2014?ref=
category_newest\",\"rewards\":\"https:/"| __truncated__ "{\"web\":{\"project\":\"https://www.kickstarter.com/projects/104060
2972/dan-phelps-cd-going-home-promo?ref=category_newest\",\""| __truncated__ "{\"web\":{\"project\":\"https://www.kickstarte
r.com/projects/402550103/restorationbid-easiest-way-to-hire-pros-for-home-i?ref=c"| __truncated__ "{\"web\":{\"project\":\"h
ttps://www.kickstarter.com/projects/471907337/trill-league-comic?ref=category_newest\",\"rewards\":\"h"| __truncated__ ...
##  $ source_url            : chr  "https://www.kickstarter.com/discover/categories/art/mixed%20media?ref=category_modal&sor
t=magic" "https://www.kickstarter.com/discover/categories/music/blues?ref=category_modal&sort=magic" "https://www.kickstarte
r.com/discover/categories/technology/web?ref=category_modal&sort=magic" "https://www.kickstarter.com/discover/categories/com
ics/comic%20books?ref=category_modal&sort=magic" ...
##  $ friends               : chr  NA NA NA NA ...
##  $ is_starred            : chr  NA NA NA NA ...
##  $ is_backing            : chr  NA NA NA NA ...
##  $ permissions           : chr  NA NA NA NA ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 31
##   .. ..$ id                    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ photo                 : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ name                  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ blurb                 : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ goal                  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ pledged               : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ state                 : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ slug                  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ country               : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ currency              : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ currency_trailing_code: list()
##   .. .. ..- attr(*, "class")= chr  "collector_logical" "collector"
##   .. ..$ deadline              :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_datetime" "collector"
##   .. ..$ state_changed_at      :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_datetime" "collector"
##   .. ..$ created_at            :List of 1
```

```
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_datetime" "collector"
##   .. ..$ launched_at           :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_datetime" "collector"
##   .. ..$ staff_pick            : list()
##   .. .. ..- attr(*, "class")= chr  "collector_logical" "collector"
##   .. ..$ is_starrable          : list()
##   .. .. ..- attr(*, "class")= chr  "collector_logical" "collector"
##   .. ..$ backers_count         : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ static_usd_rate       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ usd_pledged           : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ creator               : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ location              : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ category              : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ profile               : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ spotlight             : list()
##   .. .. ..- attr(*, "class")= chr  "collector_logical" "collector"
##   .. ..$ urls                  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ source_url            : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ friends               : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ is_starred            : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ is_backing            : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ permissions           : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

`str()` shows the dimensions of the data at the top. Then, for each variable, it gives us the class and the first few observations. Because our data is still somewhat raw and messy, it can be a little hard to read.

And here is the better way to find NA's in our data, as promised earlier:

```
colSums(is.na(ks))
```

```
##                     id                  photo                   name 
##                      0                      0                      0 
##                  blurb                   goal                pledged 
##                      0                      0                      0 
##                  state                   slug                country 
##                      0                      0                      0 
##               currency currency_trailing_code               deadline 
##                      0                      0                      0 
##       state_changed_at             created_at            launched_at 
##                      0                      0                      0 
##             staff_pick           is_starrable          backers_count 
##                      0                      0                      0 
##        static_usd_rate            usd_pledged                creator 
##                      0                      0                      0 
##               location               category                profile 
##                      2                      0                      0 
##             spotlight                   urls             source_url 
##                      0                      0                      0 
##                friends             is_starred             is_backing 
##                    500                    500                    500 
##            permissions 
##                    500
```

The four columns towards the end are nearly all NA. We can get rid of them. We will do that in just a little while when we begin cleaning up the dataset. We have two other NAs in location. Having two NAs out of 500 observations likely does not limit the usefulness of the data, but we may have to account for it at some point. For now just keep in mind that they are there.

And to wrap up our initial inspection, we want to look at some of the actual data. Because 500 observations is a little cumbersome, we can look at just the first few observations as well as the last few. We will use `head()` and `tail()`. I will also show a way to open the dataset in its entirety in another window.

```
head(ks)
```

| id <int> |
|---|
| 832570985 |
| 1054135507 |
| 1567381435 |
| 1329110416 |
| 1403665672 |
| 568986755 |

6 rows | 1-1 of 31 columns

```
tail(ks)
```

| id <int> |
|---|
| 1696097401 |
| 1049624554 |
| 1001132597 |
| 1081444630 |
| 116102718 |
| 1315682649 |

6 rows | 1-1 of 31 columns

And to open the entire dataset in another window:

```
View(ks)
```

# Ask a Question

Now that we have a very rough idea of what our dataset contains, we need to decide on a question that the rest of our analysis will focus on answering.

There are two basic types of question that this type of analysis is particularly useful for answering: inferential and predictive. Predictive is the most useful for business applications, and that is what we will focus on in this course. We will spend enough time discussing ways to deal with the question if it were inferential that you will be prepared to tackle an inferential question as well. # Could move this to discussion of specific criteria 4 below

# General criteria of a good question:

1. Must be answerable with the available data or data than can be obtained
2. Must be useful or of interest
3. Must be grounded in logic
4. Must be sufficiently sharp to provide useful insight

# Specific criteria for this analysis:

1. The answer (if found) may provide insight into factors that lead to successful funding on Kickstarter
2. Can be answered with the current dataset
3. Allows for exploration of the similarities and differences between Inferential and Predictive questions. The question should fall into one of the categories, but could be varied slightly to fall into the other for learning purposes
4. Provides the opportunity to apply the Process of Data Science

The question I propose is this:

```
How can we predict success on Kickstarter?
```

## Applying the general criteria to our question:

**1. Must be answerable with the available data or data than can be obtained**
The dataset we have is as relvant as it gets and gives us a decent chance at being able to answer the question.
**2. Must be useful or of interest**
As worded, the answer to our question could be useful either for people considering putting their project on Kickstarter and also for people considering funding projects.
**3. Must be grounded in logic**
The logic behind the question is that there may be some link between some of those variables and success. I think that is sound.
**4. Must be sufficiently sharp to provide useful insight**
My definition of sharp is that you will know when you have accomplished your task. If we are able to put together a model using the available data that predicts success on Kickstarter that meets some level of statistical significance, then we will have accomplished our task.

## Applying the specific criteria for this analysis to our question:

**1. The answer (if found) may provide insight into factors that lead to successful funding on Kickstarter**
Yes, we will have the opportunity to build a model that will hopefully predict success.
**2. Can be answered with the current dataset**
Yes, see above.
**3. Allows for exploration of the similarities and differences between Inferential and Predictive questions. The question should fall into one of the categories, but could be varied slightly to fall into the other for learning purposes**
**4. Provides the opportunity to apply the Process of Data Science**

# Tidy the data, and any other necessary formatting

Data rarely come to us in an ideal format. There are almost always some formatting issues that we will have to deal with.

Working with tidy data makes every subsequent step of the analysis much simpler and more straightforward. Every bit of time spent at this stage is an investment that will pay dividend throughout the rest of the analysis.

However, this can be one of the most challenging parts of any data anaylsis. We know what we want the data to look like at the end, but the ways it can differ from what we want are endless. In other words, tidy, clean data fits a nice, tight script, but there are no limits to how dirty a dataset may be.

Fortunately, we have a set of principles to define what a clean or tidy dataset looks like, and we have many excellent tools to help us get there. In this section of the course, we will learn the principles of tidy data and how to use the primary tools for tidying data. We will also use several other tools to address the specific issues that need to be addressed in our dataset.

The caveat here is that no course could possibly teach how to overcome every possible obstacle a dataset may face. However, with the underlying principles of tidy data guiding us, we can rely on our accumulated and always growing experience, our creativity, and a little research to get through those problems.

# Topics covered in this section

- Principles of tidy data
- Converting and parsing dates
- Encoding factors and an intro to important data types

# Tidy Data, and meet Hadley Wickham

As with so many things in modern data analysis using R, we have Dr. Hadley Wickham to thank for providing both structure and tools for dealing with messy data. Hadley holds many titles, including Adjunct Professor of Statistics at the University of Auckland, Stanford University, and Rice University, Chief Data Scientist for RStudio, and is the author of many books about R programming. He has a genius for reducing common problems and difficulties to their basic principles and developing tools to help us implement those principles in our analyses.

His contributions include the following (these are just a few of the most popular and important; there are many more):

- ggplot2 for graphing and visualizing data
- dplyr for transforming data
- ggvis for interactive visualizations of data
- readr for importing and exporting data
- and the subject of this section: tidyr for tidying data

We will go over many of these and more throughout the rest of this course.

I could go on about the impact that Hadley has had on the world of R programming and data analysis in general, but instead I will refer you this article, which does a great job of deomonstrating his impact: Hadley Wickham, the Man Who Revolutionized R (https://priceonomics.com/hadley-wickham-the-man-who-revolutionized-r/)

I also highly recommend any of his books, two of which are available online for free:

- R for Data Science: Import, Tidy, Transform, Visualize, and Model Data (http://r4ds.had.co.nz/)
- Advanced R (http://adv-r.had.co.nz/)

All of his books can be purchased from Amazon and make great additions to any data analyst's library:

- ggplot2: Elegant Graphics for Data Analysis (https://www.amazon.com/ggplot2-Elegant-Graphics-Data-Analysis/dp/331924275X/ref=as_li_ss_tl?ie=UTF8&qid=1518551437&sr=8-1&keywords=ggplot&linkCode=ll1&tag=jaycreighton-20&linkId=ddf3a29b7f2a08e66cb2ed8cd9d102a5)
- Hands-On Programming with R: Write Your Own Functions and Simulations (https://www.amazon.com/Hands-Programming-Write-Functions-Simulations/dp/1449359019/ref=as_li_ss_tl?_encoding=UTF8&pd_rd_i=1449359019&pd_rd_r=FB5MENM8TTVMCTGQQ3HN&pd_rd_w=c3K0H&pd_rd_wg=xys2E&psc=1&refRID=FB5MENM8TTVMCTGQQ320&linkId=711b99d45fcb277d8fdcea044f61dfa4)
- R Packages: Organize, Test, Document, and Share Your Code (https://www.amazon.com/Packages-Organize-Test-Document-Share/dp/1491910593/ref=as_li_ss_tl?ie=UTF8&qid=1518551528&sr=8-1&keywords=r+packages+wickham&linkCode=ll1&tag=jaycreighton-20&linkId=cdd21eb92453652a033f047445fe2369)
- R for Data Science: Import, Tidy, Transform, Visualize, and Model Data (https://www.amazon.com/Data-Science-Transform-Visualize-Model/dp/1491910399/ref=as_li_ss_tl?s=books&ie=UTF8&qid=1518551559&sr=1-2&keywords=r+for+data+science+hadley+wickham&linkCode=ll1&tag=jaycreighton-20&linkId=133441a126c4a9346c47c7ccc328ec61)
- Advanced R (https://www.amazon.com/Advanced-Chapman-Hall-Hadley-Wickham/dp/1466586966/ref=as_li_ss_tl?_encoding=UTF8&pd_rd_i=1466586966&pd_rd_r=C1C394EFNS710CVDFPZ6&pd_rd_w=6QyJR&pd_rd_wg=8VtUp&psc=1&refRID=C1C394EFNS710CVDFPZ6&li20&linkId=3549d5d55049b3704b47ac6d4feb56c7)

Now onto his principles of **tidy data**.

As mentioned earlier, data rarely comes to us in the exact format that we need. We could call such data raw, messy, or even un-tidy. And we must take measures to organize, clean, or tidy it. But what exactly do we want the data to look like when we're done with it? Dr. Wickham put together a set of guidelines or principles of tidy data. I think it's worth noting that the definition given here refers to tidy data in a somewhat strict sense. Data

can meet all of these criteria and still need some formatting. We will first discuss this strict definition of tidy data, then move on to other formatting tasks, all of which can fall under a more general definition of tidying data.

From Hadley's book, R for Data Science:

> There are three interrelated rules which make a dataset tidy (Wickham & Grolemund, 2017, ch. 12): 1. Each variable must have its own column. 2. Each observation must have its own row. 3. Each value must have its own cell.

Let's compare our dataset to these criteria to see if we have any work to do on it. We can look at the top of our data by using the `head()` function or we can open the whole thing in a separate window with the `View()` function like we did earlier.

Let's start with the head().

```
head(ks)
```

| id <int> |
|---|
| 832570985 |
| 1054135507 |
| 1567381435 |
| 1329110416 |
| 1403665672 |
| 568986755 |

6 rows | 1-1 of 31 columns

You can get quite a bit of information from the head table here. However, when getting a feel for the whole dataset, I find it useful to open the whole document. If the dataset is especially large, using the `View()` can be a little cumbersome. In our case, however, it's just fine.

```
# View(ks)
```

Remember those three rules we are looking for. We can walk through each of those, asking ourselves if our data meet the criteria.

**First, does each variable have its own column?** Looking at the data, none of the variables are combinations of data that would need to be separated. Additionally, none of the columns are single variables spread across multiple columns. Check that one off the list.

**Second, does each observation have its own row?** Since each row of data refers to one case, yes, each observation has its own row.

**Third, does each value have it's own cell?** If we have addressed the first two principles concerning rows and columns, then each cell must be its own value. Looking at our dataset confirms this.

So our dataset is tidy, but not all data that you come across will be. While all tidy datasets fit a common format, untidy data can be untidy in literally an infinite number of ways. Because there is an endless number of problems you may have to fix in tidying a dataset, it is necessarily out of the scope of this class. Fortunately, there are many great tools for tidying data (with the tidyr package at the top of the list). These tools are simple and intuitive, especially once you have a little experience working with them. Tidying data requires a lot of free form thinking and creative problem solving, and it only gets easier with time and experience. Early on, I encourage you to find datasets that are pretty close to tidy, and work your way up to more challenging datasets.

```
set.seed(97)

(untidy_df2 <- data.frame(make = LETTERS[1:5], v6 = sample(100:200, 5), v8 = sample(200:350, 5)))
```

| make <fctr> | v6 <int> | v8 <int> |
|---|---|---|
| A | 104 | 330 |
| B | 156 | 344 |
| C | 197 | 321 |
| D | 117 | 243 |
| E | 181 | 221 |

5 rows

```
(tidy_df2 <- untidy_df2 %>%
    gather(v6, v8, key = "cylinders", value = "hp" ))
```

| make <fctr> | cylinders <chr> | hp <int> |
|---|---|---|
| A | v6 | 104 |
| B | v6 | 156 |
| C | v6 | 197 |
| D | v6 | 117 |

| make<br><fctr> | cylinders<br><chr> | hp<br><int> |
|---|---|---|
| E | v6 | 181 |
| A | v8 | 330 |
| B | v8 | 344 |
| C | v8 | 321 |
| D | v8 | 243 |
| E | v8 | 221 |

1-10 of 10 rows

```
# This might work just to explain the concept and the function
```

```
# Do this one first (before gather)

(untidy_df1 <- data.frame(id = c(101, 101, 102, 102, 103, 103, 104, 104, 105, 105), measure = c("hp", "mpg"), number = c(11
1, 27, 149, 30, 114, 19, 223, 10, 136, 12)))
```

| id<br><dbl> | measure<br><fctr> | number<br><dbl> |
|---|---|---|
| 101 | hp | 111 |
| 101 | mpg | 27 |
| 102 | hp | 149 |
| 102 | mpg | 30 |
| 103 | hp | 114 |
| 103 | mpg | 19 |
| 104 | hp | 223 |
| 104 | mpg | 10 |
| 105 | hp | 136 |
| 105 | mpg | 12 |

1-10 of 10 rows

```
(tidy_df1 <- untidy_df1 %>%
    spread(key = measure, value = number))
```

| | id<br><dbl> | hp<br><dbl> | mpg<br><dbl> |
|---|---|---|---|
| 1 | 101 | 111 | 27 |
| 2 | 102 | 149 | 30 |
| 3 | 103 | 114 | 19 |
| 4 | 104 | 223 | 10 |
| 5 | 105 | 136 | 12 |

5 rows

```
# I THINK THIS ONE IS DONE!!!!
```

```
# a toy df
(dates_df <- data.frame(id = 101:110, date = as.Date(sample(13000:15000, 10), origin = '1970-01-01', tz = 'GMT')))
```

| id<br><int> | date<br><date> |
|---|---|
| 101 | 2008-04-13 |
| 102 | 2007-06-05 |
| 103 | 2008-07-22 |
| 104 | 2007-05-05 |
| 105 | 2006-05-28 |
| 106 | 2007-05-27 |
| 107 | 2008-09-30 |
| 108 | 2007-09-23 |

| id <int> | date <date> |
|---|---|
| 109 | 2007-02-01 |
| 110 | 2005-11-12 |

1-10 of 10 rows

```
(dates_df_sep <- dates_df %>%
    separate(date, into = c("year", "month", "day"), sep = "-"))
```

| | id <int> | year <chr> | month <chr> | day <chr> |
|---|---|---|---|---|
| 1 | 101 | 2008 | 04 | 13 |
| 2 | 102 | 2007 | 06 | 05 |
| 3 | 103 | 2008 | 07 | 22 |
| 4 | 104 | 2007 | 05 | 05 |
| 5 | 105 | 2006 | 05 | 28 |
| 6 | 106 | 2007 | 05 | 27 |
| 7 | 107 | 2008 | 09 | 30 |
| 8 | 108 | 2007 | 09 | 23 |
| 9 | 109 | 2007 | 02 | 01 |
| 10 | 110 | 2005 | 11 | 12 |

1-10 of 10 rows

```
# sep argument is optional, as the default is to separate on any non-alphanumeric character

# Now let's put it back together!
(together_again <- dates_df_sep %>%
    unite(date, year, month, day))
```

| | id <int> | date <chr> |
|---|---|---|
| 1 | 101 | 2008_04_13 |
| 2 | 102 | 2007_06_05 |
| 3 | 103 | 2008_07_22 |
| 4 | 104 | 2007_05_05 |
| 5 | 105 | 2006_05_28 |
| 6 | 106 | 2007_05_27 |
| 7 | 107 | 2008_09_30 |
| 8 | 108 | 2007_09_23 |
| 9 | 109 | 2007_02_01 |
| 10 | 110 | 2005_11_12 |

1-10 of 10 rows

```
# We can choose the separator
(together_again2 <- dates_df_sep %>%
    unite(date, year, month, day, sep = "-"))
```

| | id <int> | date <chr> |
|---|---|---|
| 1 | 101 | 2008-04-13 |
| 2 | 102 | 2007-06-05 |
| 3 | 103 | 2008-07-22 |
| 4 | 104 | 2007-05-05 |
| 5 | 105 | 2006-05-28 |
| 6 | 106 | 2007-05-27 |
| 7 | 107 | 2008-09-30 |
| 8 | 108 | 2007-09-23 |
| 9 | 109 | 2007-02-01 |

| | id <int> | date <chr> |
|---|---|---|
| 10 | 110 | 2005-11-12 |

1-10 of 10 rows

As you come accross issues with datasets that you encounter, here are the best resources to use:

tidyr vignette (https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html)
Package documentation for 'tidyr' (https://cran.r-project.org/web/packages/tidyr/tidyr.pdf)
Hadley Wickham's paper on tidy data in the Journal of Statistical Software (http://vita.had.co.nz/papers/tidy-data.pdf)
R for Data Science, Chapter 12: Tidy Data (http://r4ds.had.co.nz/tidy-data.html)

# Other formatting and tidying of the data

So we have ensured that our data is tidy in the strict sense, but still have plenty of other formatting to do.

Although it may come earlier in other analyses, this will be the first time that we will change our modify our data. To be sure that we never lose any data or unintentionally change our data, be sure that you only modify the copy data that we have imported into an R object (ours is called `ks`). We will leave our raw data exactly as it is and do all of our work on a copy of it. Furthermore, we will document and re-run each step of the process at the beginning of every session, so that we know our analysis is reproducible. This is accomplished by simply working down the page, adding each step to the bottom of this document in the order that we do it. If we are diligent in doing so, each time we open and run this document in a new session, we should get the same result.

# Dates

We noticed earlier that our date variables didnt look much like dates.

Let's quickly take another look at the date variables to refresh our memory as to what we are dealing with.

```
head(ks)
```

| id <int> |
|---|
| 832570985 |
| 1054135507 |
| 1567381435 |
| 1329110416 |
| 1403665672 |
| 568986755 |

6 rows | 1-1 of 31 columns

Dates can come to you in many formats. They make look like 5/27/2005, 27/5/2005, 4-30-17, 1.16.87, or even 537813061. The first four are simply stylistic differences, but the last one is a format known as Unix time.

The "Date" or `as.Date` class in R is simply the number of days since January 1, 1970. There are also two Date-Time classes in R (which are similar to the "Date" class, but also include time information), "POSIXlt" and "POSIXct". "POSIXct" is the number of seconds since the beginning of …

The date variables include `deadline`, `state_changed_at`, `created_at`, and `launched_at`. It appears that value of these variables is a 10 digit number rather than a date. That very large number represents the number of seconds since January 1, 1970. This format is known as Unix time. In order to convert them to a reasonable format, we will use the r function `as.POSIXct()`. We can use the `$` operator to call a specific variable from the `ks` dataset: `ks$created_at` in the first example. We will also need to use the `as.numeric()` function since the original function requires a numeric object. Be sure the parentheses for the `as.numeric()` function enclose only the variable we are calling. We will replace the old values with the new ones using the assignment operator, `<-`, assigning the new values in place of the old ones.

```
ks$created_at <- as.POSIXct(as.numeric(ks$created_at), origin = '1970-01-01', tz = 'GMT')
```

There's a lot going on here, so let's go back through each part to be sure its all clear as can be…

Let's check our work

```
head(ks)
```

| id <int> |
|---|
| 832570985 |
| 1054135507 |
| 1567381435 |
| 1329110416 |
| 1403665672 |
| 568986755 |

6 rows | 1-1 of 31 columns

Note the difference both in how the dates look, but also the class listed right below variable name.

Let's do the rest of them:

```
ks$deadline <- as.POSIXct(as.numeric(ks$deadline), origin = '1970-01-01', tz = 'GMT')

ks$state_changed_at <- as.POSIXct(as.numeric(ks$state_changed_at), origin = '1970-01-01', tz = 'GMT')

ks$launched_at <- as.POSIXct(as.numeric(ks$launched_at), origin = '1970-01-01', tz = 'GMT')
```

```
head(ks)
```

| id <int> |
|---|
| 832570985 |
| 1054135507 |
| 1567381435 |
| 1329110416 |
| 1403665672 |
| 568986755 |

6 rows | 1-1 of 31 columns

They're all good!

Another useful tool for working with dates is the lubridate package. It's most useful when dates already look like dates (but may not yet act like dates). There are a very simple set of functions for parsing (putting into a usable format). All you have to do is choose the function that corresponds to the correct order of month, day and year. It does not matter how they are separated.

Here's an example:

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
mdy("05/01/2008")
```

```
## [1] "2008-05-01"
```

```
dmy("21.01.2013")
```

```
## [1] "2013-01-21"
```

```
ymd("1989-06-23")
```

```
## [1] "1989-06-23"
```

You can find more information on the lubridate package at Dates and Times Made Easy with lubridate (https://www.jstatsoft.org/article/view/v040i03).

# References

Wickham, H., & Grolemund, G. (2017). *R for data science* (First Edition). O'Reilly Media.