Please share your **highlight**, **success**, **small win**, or **something positive** that happened.

What’s a **challenge** you experienced or something you can use more **support** with?

What’s something you are **looking forward** to knowing more about or experiencing?

Image Credit
Nathan, S. (2020). *Rose, Thorn, Bud Activity for Group Presentations.* Life Design Log. Retrieved from: https://lifedesignlog.com/rose-thorn-bud-activity-for-group-presentations/

Data Transformation: Part I

Data Manipulation, Reshaping, & Wrangling in R

Cultivate Learning Innovation Lab Workshop
July 20, 2020 | Monday | 10:00 - 11:30 a.m.

[황보 민] Min Hwangbo

PhD Candidate in Learning Sciences & Human Development
Graduate Certificate Recipient in Demographic Methods

Learning Agenda

- What is *data transformation*?
  - tidyverse: dplyr package
- Creating object in R: Data frame
- Manipulating data frame:
  - Extract data based on conditions:
    - Certain variable(s)
    - Condition based on value
    - Time stamp

[Intro] Inspiration

- Land
    - *"The University of Washington & Cultivate Learning acknowledges that it sits on Indigenous Land, which touches the shared waters of all tribes and bands within the Duwamish, Suquamish, Tulalip, and Muckleshoot Tribes."*
- People
    - Aimée Dechter | Affiliate Assistant Professor & Former Research Coordinator at Center for Studies in Demography & Ecology (CSDE)
    - [Chuck] Charles C. Lanfear | PhD Candidate & R Guru | 2020 Distinguished Teaching Award Recipient
    - [Jose] Jose Hernandez | Data Science Fellow & Research Staff @ eScience Institute
    - [Liz] Elizabeth Sanders | Associate Professor & Quantitative Researcher @ College of Education
    - [RStudio] Hadley Wickham & Garrett Grolemund | Authors of R for Data Science | RStudio Chief Scientist, Data Scientist & Statistician | Creators of RStudio
    - [이근열] Keun Yeol, Lee. | Professor in Busan National University & Qualitative/Dialect/Linguistics Researcher
    - [本橋智光] Motohashi, Tomomitsu. (2018). Maeshoritaizen data bunseki no tame no SQL/R/Python jissen technique (데이터 전처리 대전. 2019).
    - [Nicolas] Nicolas Pröllochs | Tenure-track Professor of Data Science in University of Giessen & Social Network Analysis / Text Mining Expert

Learning Agenda

- What is *data transformation*?
  - tidyverse: dplyr & tidyr package
- Creating object in R: Data frame
- Transforming data frame:
  - Extract data based on conditions:
    - Certain variable(s)
    - Condition based on value
    - Time stamp

[Intro] Data Transformation?

- *“Process of **converting** data from one format or structure into another format or structure.... **fundamental** aspect of most **data integration** and **data management.**”*[1]
    - Data framing
    - Data manipulation
    - Data wrangling
    - Data management
- Data Transformation Steps
    - Data discovery: Where’s my data?
    - Data mapping: Explore your data - What’s in it? Using data transformation technique
    - Code generation: Cheatsheet & Stackflow & Pre-Built codes + Ctrl + C / Ctrl + V
    - Code execution: Ctrl + Enter
    - Data review: head(data), tail(data), View(data), missing data, quality of your data, etc...

[1] Source: Wikipedia

[Intro] Package: tidyverse

- Hadley Wickham & Garrett Grolemund
- tidyverse: “The tidyverse is an opinionated collection of R packages designed for data science.”
  - dplyr: “The grammar of data manipulation.”
  - tidyr: “The goal of tidyr is to help you create tidy data.”
  - readr: “The goal of 'readr' is to provide a fast and friendly way to read rectangular data (like 'csv', 'tsv', and 'fwf').”
  - ggplot2: “ggplot2 is a system for declaratively creating graphics”
  - tibble: “A tibble, or tbl_df, is a modern reimagining of the data.frame, keeping what time has proven to be effective, and throwing out what is not.”
  - purrr: “purrr enhances R’s functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors.”

[Intro] tidyverse Workflow

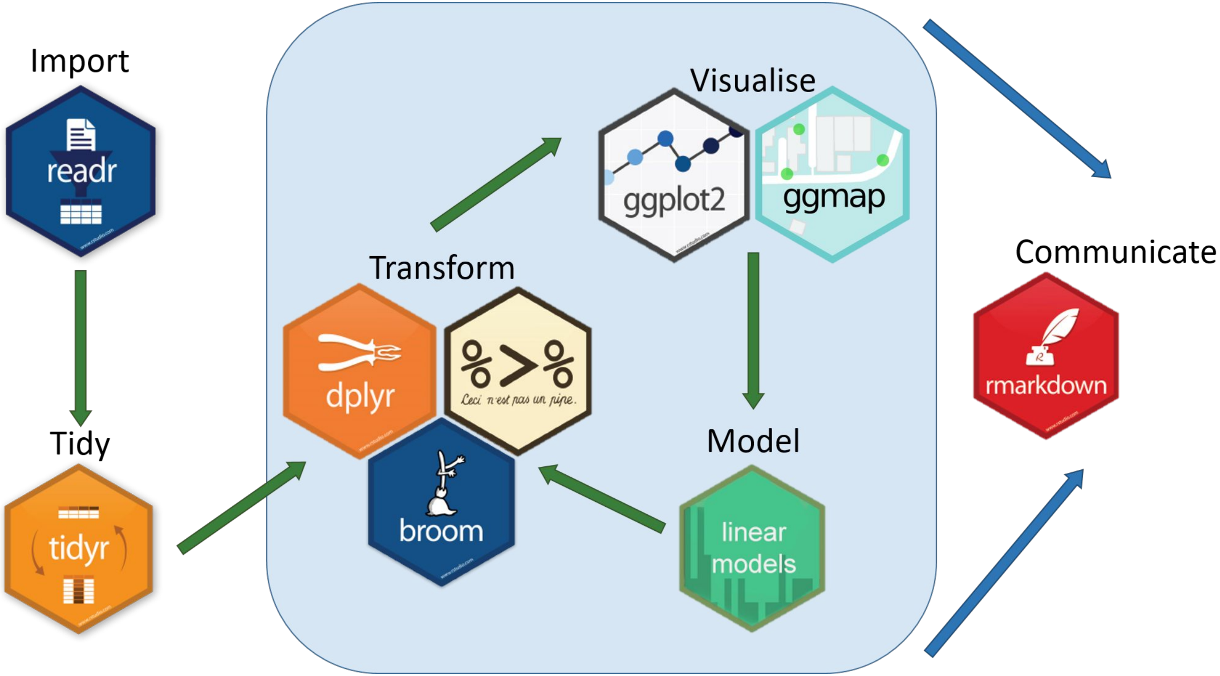


Image retrieved from: Glanz, Hunter. (2019). What is the Tidyverse? *Teach Data Science blog @* https://teachdatascience.com/tidyverse/

[Intro] Package tidyverse: dplyr & tidyr

- dplyr: “The grammar of data manipulation.”
  - **mutate()**: adds new variables that are functions of existing variables
  - **select()**: picks variables based on their names.
  - **filter()**: picks cases based on their values.
  - **summarise()**: reduces multiple values down to a single summary.
  - **arrange()**: changes the ordering of the rows.
- tidyr: “The goal of tidyr is to help you create tidy data.”
  - “Every **column** is **variable**.”
  - “Every **row** is an **observation**.”
  - “Every **cell** is a single **value**.”
- Cheatsheet!: Data Transformation Cheatsheet

[Prep] Preparation

- I’m ready! Okay more things to learn...
  - Wait… before you jump in….
- Packages
  - Installing packages
  - Loading packages
- Data preparation
  - Loading data set
  - Framing data set

Learning Agenda

- What is *data transformation?*
  - tidyverse: dplyr & tidyr package
- Creating object in R: Data frame
- Transforming data frame:
  - Extract data based on conditions:
    - Certain variable(s)
    - Condition based on value
    - Time stamp

[Prep] Packages

- RStudio: Your XBox
- Packages: Software
  - Collection of **R** functions, compiled code and sample data.
  - Installation: **install.packages("packagename")**
  - Stored under "**library("packagename")**"

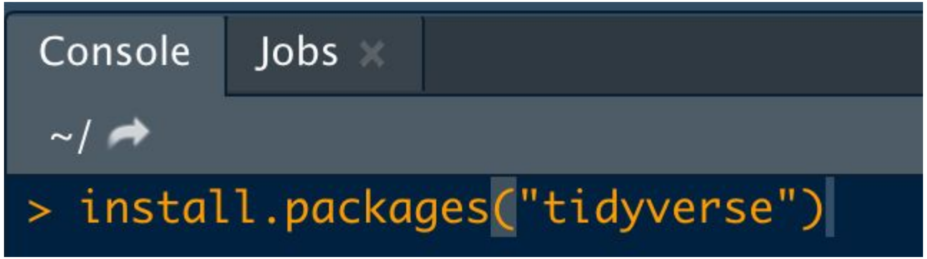
[Prep] Installing Packages

Console

```
install.packages(“tidyverse”)
```

OR

```
install.packages(“readr”) # Loading csv files
install.packages(“readxl”) # Loading excel files - not tidyverse but useful
install.packages(“tidyr”)  # Data cleaning/transformation
install.packages(“dplyr”) # Data wrangling
```



```
> install.packages("tidyverse")
```

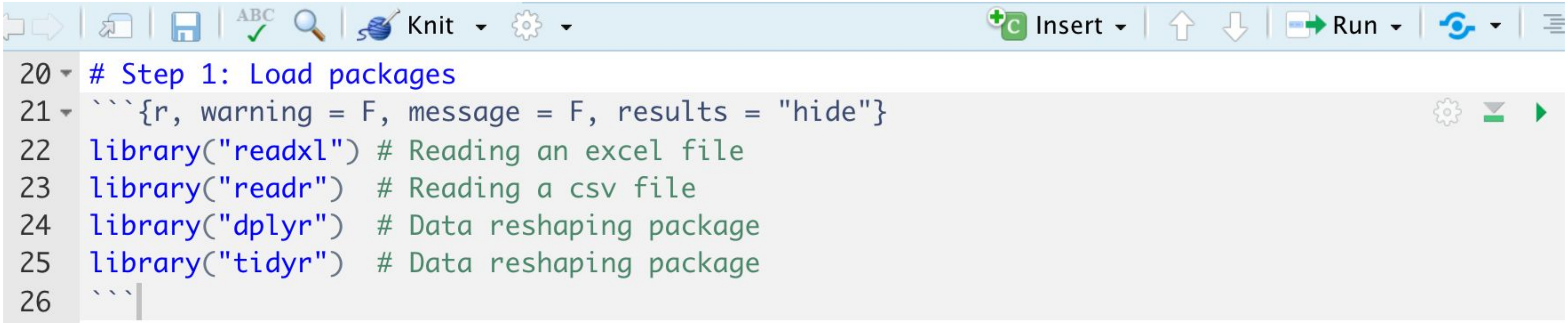
- **DO NOT CODE THESE ON RMarkdown (RMD) !!!**
- Spelling matters!

[Prep] Loading Packages

RMD

~~~
# Step 1: Loading Packages

```{r, warning = F, message = F, results = "hide"}
library("readr") # Reading a csv file
library("readxl") # Reading an excel file
library("dplyr") # Data wrangling package
library("tidyr") # Data cleaning/transformation package
```
~~~

[Prep] Loading Packages



~~~
# Step 1: Load packages
```{r, warning = F, message = F, results = "hide"}
library("readxl") # Reading an excel file
library("readr")  # Reading a csv file
library("dplyr")  # Data reshaping package
library("tidyr")  # Data reshaping package
```
~~~

[Prep] Loading Data Set

- Download data set @ Shared Drive | Data retrieved from data.wa.gov
- Locate your ECEAPSites_DCYF_070120.csv file in your R folder
- Load `` `readr` `` package
- Frame your data set into an object?
    - What do you mean by this?

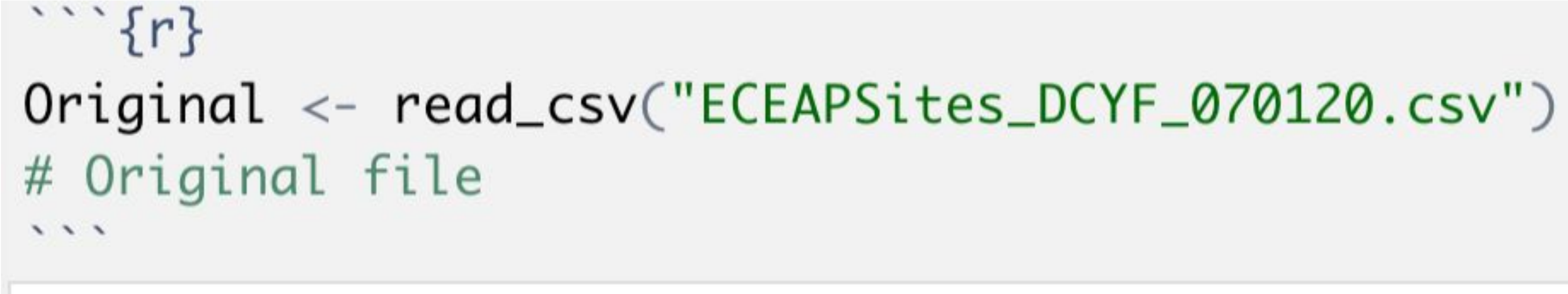
[Prep] Framing an Object

- R Environments
  - **F*rame***, consisting of a set of symbol-value pairs
  - **Enclosure**, a pointer to an enclosing environment.
  - When R looks up the value for a symbol the **frame** is examined and if a matching symbol is found its value will be returned. If not, the **enclosing environment** is then accessed and the process repeated.
    - ??? Too complicated!!!

[Prep] Framing an Object: Simpler Concept

- New Frame/Object **<-** Syntax to create an object
- For example: If you want to load a data from an “Original” / “Raw” file…

Original <- read_csv(file name under your directory) # Windows usually starts with C://



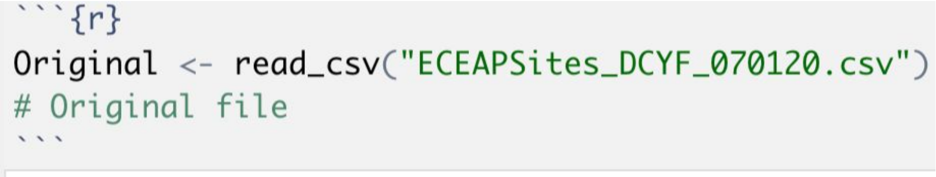
~~~
```{r}
Original <- read_csv("ECEAPSites_DCYF_070120.csv")
# Original file
```
~~~

[Prep] Framing an Object: Execution

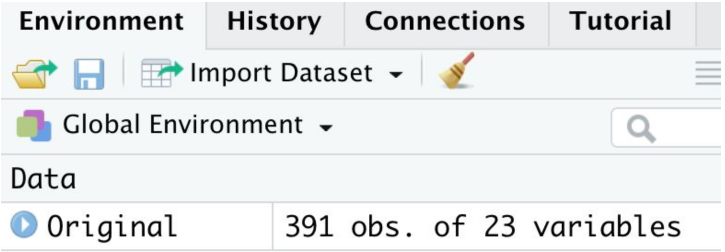
Original <- read_csv(file name under your directory)

Output

Input



~~~
```{r}
Original <- read_csv("ECEAPSites_DCYF_070120.csv")
# Original file
```
~~~



```
Data
Original          391 obs. of 23 variables
```

Questions?

Learning Agenda

- What is *data transformation?*
  - tidyverse: dplyr & tidyr package
- Creating object in R: Data frame
- Manipulating data frame:
  - Extract data based on conditions:
    - Certain variable(s)
    - Condition based on value
    - Time stamp

[Transform] Data Mapping: Variables

- Always check your original file first.

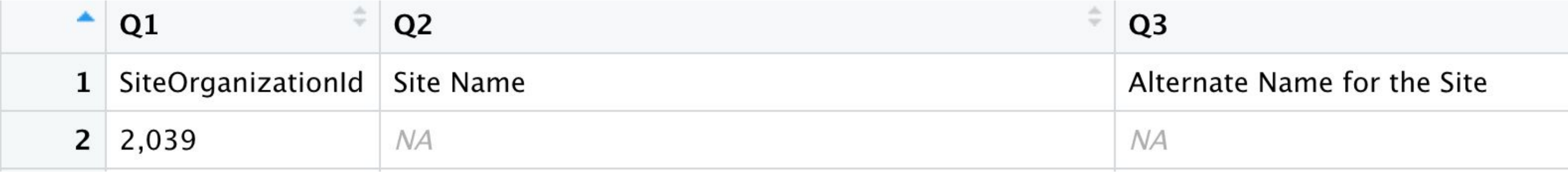
```
summary(Original) # Summary of what this file is about

View(Original)    # Projecting the data set into a familiar spreadsheet format

head(Original, 5) # First five records (default = 6)

tail(Original, 5) # Last five records (default = 6)

ls(Original)      # Vars names
```

[Transform] Data Mapping: Variables

Input

```
View(Original)# Projecting the data set into a familiar spreadsheet format
```

Output



	Q1	Q2	Q3
1	SiteOrganizationId	Site Name	Alternate Name for the Site
2	2,039	NA	NA

Mystery

- Vars names on 2 different rows??? : Which one should I use?

[Transform] Data Mapping: Variables - Mystery

- Mystery always happens :)
  - Similar cases will always occur when you work with any "**Qualtrics**" files
- It's our journey to figure out which variables are the right ones to use.
- Var Names =
  - **Row 1 - Acronyms**: Great if you have a codebook that represents what Q1, Q2, Q3 represents; If not,
  - **Row 2 - Questions**: You'll be able to read the whole thing, but then it will be way too long to process as you conduct your analysis.
- For this example, we'll go with **Row 2** and will rename the data frame as a "**Revision**" file!

[Transform] Revision <- Original

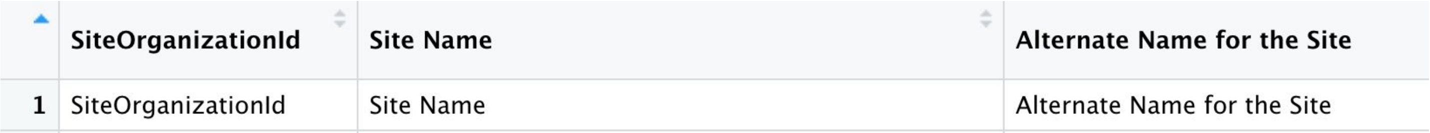
- Frame your “**Revision**” data <- Syntax using data from “**Original**” data

Input

```
names(Original) <- Original[1, ]     # Copy 2nd row as var names
```

Output:

```
View(Original)                       # Execute this code on your Console
```



	SiteOrganizationId	Site Name	Alternate Name for the Site
1	SiteOrganizationId	Site Name	Alternate Name for the Site

[Transform] Revision <- Original

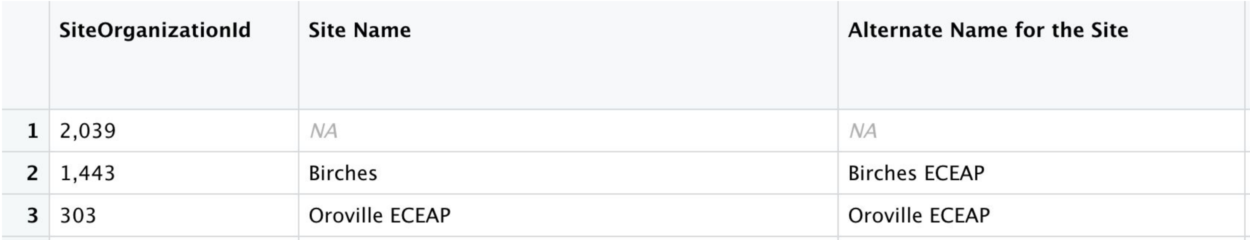
- Frame your “**Revision**” data <- Syntax using data from “**Original**” data

Input

```
Revision <- Original[-1, ]        # Delete duplicate row (1st row)
```

Output:

```
View(Revision)                    # Execute this code on your Console
```



	SiteOrganizationId	Site Name	Alternate Name for the Site
1	2,039	*NA*	*NA*
2	1,443	Birches	Birches ECEAP
3	303	Oroville ECEAP	Oroville ECEAP

[Transform] Quality Check

```
head(Revision, 5)   # First 5 records

tail(Revision, 5)   # Last 5 records

ls(Revision)        # Vars name

# 390 obs & 23 vars

## Important to do this step every time when you create/conduct an analysis
```

10 min Break

Coffee, Tea & Snack Time

Thank you!

감사합니다!

Data Transformation: Part II

Data Manipulation, Reshaping, & Wrangling in R

Cultivate Learning Innovation Lab Workshop
July 23, 2020 | Thursday | 2 - 3:30 p.m.

[황보 민] Min Hwangbo

PhD Candidate in Learning Sciences & Human Development
Graduate Certificate Recipient in Demographic Methods

[Intro] Inspiration

- Land
  - *"The University of Washington & Cultivate Learning acknowledges that it sits on Indigenous Land, which touches the shared waters of all tribes and bands within the Duwamish, Suquamish, Tulalip, and Muckleshoot Tribes."*
- People
  - Aimée Dechter | Affiliate Assistant Professor & Former Research Coordinator at Center for Studies in Demography & Ecology (CSDE)
  - [Chuck] Charles C. Lanfear | PhD Candidate & R Guru | 2020 Distinguished Teaching Award Recipient
  - [Jose] Jose Hernandez | Data Science Fellow & Research Staff @ eScience Institute
  - [Liz] Elizabeth Sanders | Associate Professor & Quantitative Researcher @ College of Education
  - [RStudio] Hadley Wickham & Garrett Grolemund | Authors of R for Data Science | RStudio Chief Scientist, Data Scientist & Statistician | Creators of RStudio
  - [이근열] Keun Yeol, Lee. | Professor in Busan National University & Qualitative/Dialect/Linguistics Researcher
  - [本橋智光] Motohashi, Tomomitsu. (2018). Maeshoritaizen data bunseki no tame no SQL/R/Python jissen technique (데이터 전처리 대전. 2019).
  - [Nicolas] Nicolas Pröllochs | Tenure-track Professor of Data Science in University of Giessen & Social Network Analysis / Text Mining Expert

Learning Agenda

- What is *data transformation?*
  - tidyverse: dplyr & tidyr package
- Creating object in R: Data frame
- Manipulating data frame:
  - Extract data based on conditions:
    - Certain variable(s)
    - Condition based on value
    - Time stamp

[Transform] Extracting Variables

- **What if we're only interested in extracting certain variables such as:**
  - "Emergency Contact Name"
  - "Site Name"
  - "City"
  - "Zip code"
  - "EA Participation"
- Recall the syntax that we used for checking Vars Names???
  - **ls(dataset)**

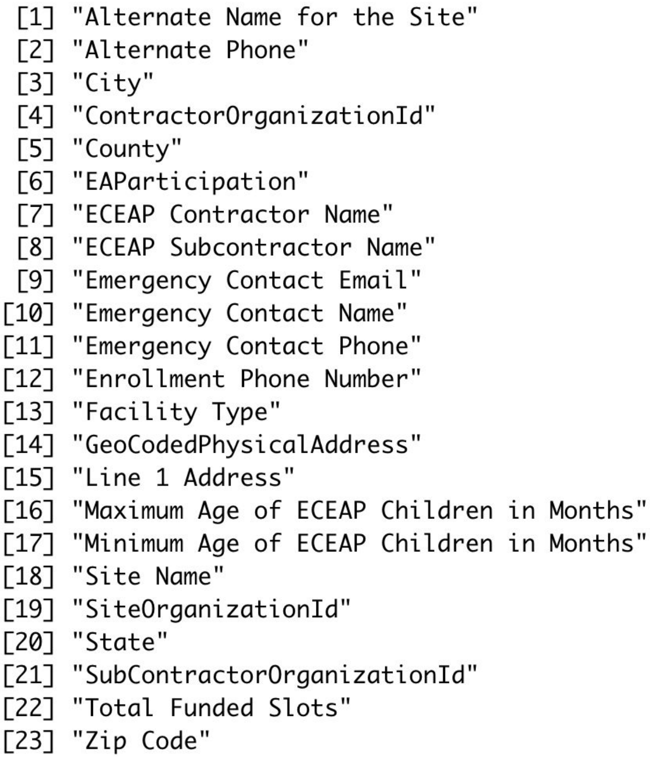
[Transform] Extracting Variables

Input

```
ls(Revision)
```

```
# On your console
```

Output



```
 [1] "Alternate Name for the Site"
 [2] "Alternate Phone"
 [3] "City"
 [4] "ContractorOrganizationId"
 [5] "County"
 [6] "EAParticipation"
 [7] "ECEAP Contractor Name"
 [8] "ECEAP Subcontractor Name"
 [9] "Emergency Contact Email"
[10] "Emergency Contact Name"
[11] "Emergency Contact Phone"
[12] "Enrollment Phone Number"
[13] "Facility Type"
[14] "GeoCodedPhysicalAddress"
[15] "Line 1 Address"
[16] "Maximum Age of ECEAP Children in Months"
[17] "Minimum Age of ECEAP Children in Months"
[18] "Site Name"
[19] "SiteOrganizationId"
[20] "State"
[21] "SubContractorOrganizationId"
[22] "Total Funded Slots"
[23] "Zip Code"
```

[Transform] Extracting Variables: `select`

- `**select**` function: *Extract* variables that you would only need for your analysis

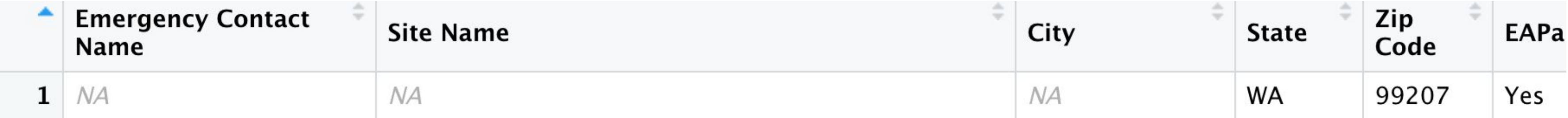
Input

```
ECEAP1 <- Revision %>%
        select("Emergency Contact Name", "Site Name", "City", "State", "Zip Code", "EAParticipation") %>%
as.data.frame()
# 390 obs & 6 vars
```

[Transform] Extracting Variables: `select`

Output

```
View(ECEAP1)                                  # On your console
```

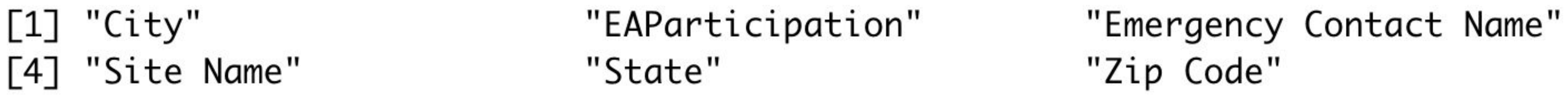


	Emergency Contact Name	Site Name	City	State	Zip Code	EAPa
1	NA	NA	NA	WA	99207	Yes

[Transform] Extracting Variables: `select`

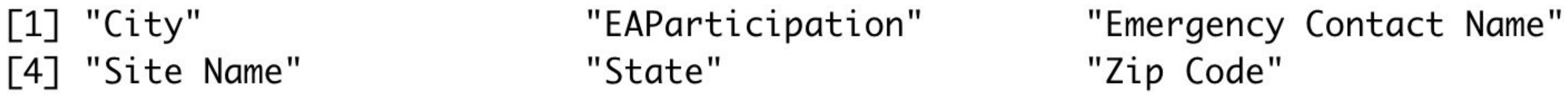
Quality Check

```
head(ECEAP1, 5)          # First 5 records

tail(ECEAP1, 5)          # Last 5 records

ls(ECEAP1)               # Looks good!
```



```
[1] "City"                     "EAParticipation"         "Emergency Contact Name"
[4] "Site Name"                "State"                   "Zip Code"
```

[Transform] Renaming Variables: `rename`



```
[1] "City"                    "EAParticipation"        "Emergency Contact Name"
[4] "Site Name"               "State"                  "Zip Code"
```

- Some Vars Names are too long to type!
- Data scientist = We try to be efficient for any situation if possible.
- **rename("NewName" = `OldName`)**

[Transform] Renaming Variables: `rename`

- Some Vars Names are too long to type!
- Data scientist = We try to be efficient for any situation if possible.

```
rename(“NewName” = `OldName`)
```

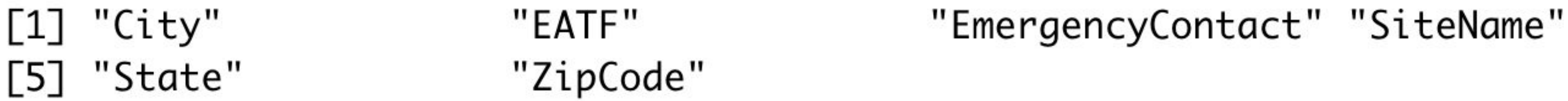
Input

```
ECEAP2 <- ECEAP1 %>% rename("EATF" = `EAParticipation`, "EmergencyContact" =
`Emergency Contact Name`, "SiteName" = `Site Name`, "ZipCode" = `Zip Code`)

# For renaming dataframe column

## Don't forget to conduct your quality check!
```

[Transform] Renaming Variables: `rename`

Output

```
head(ECEAP2, 5)    # First 5 records

tail(ECEAP2, 5)    # Last 5 records

ls(ECEAP2)         # Vars name
```



```
[1] "City"             "EATF"             "EmergencyContact" "SiteName"
[5] "State"            "ZipCode"
```

```
# Clear!

## 390 obs & 6 vars
```

[Transform] Value Condition: `filter`

- What if we're only interested in looking at **ECEAP sites** that are participating in **Early Achievers**?
  - **EATF value = Yes!**
- **`filter`:** Perfect for filtering T/F conditions.

i.e.

```
filter(VarName == Yes)
```

```
filter(VarName == No)
```

```
# "==" instead of "=" as in R lanugage, "=" only works for numerical value.
```

[Transform] Value Condition: `filter`

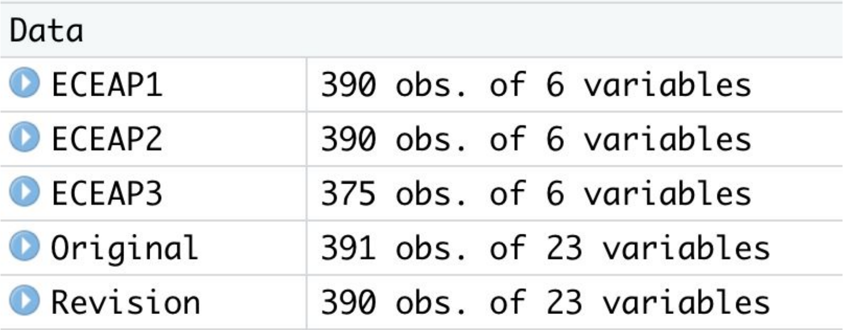
Input

```
ECEAP3 <- ECEAP2 %>%

  filter(EATF == "Yes")

# Time for quality check!

## View data set on your console! View(ECEAP3)
```

[Transform] Value Condition: `filter`

Output

```
head(ECEAP3,5)     # First five records

tail(ECEAP3,5)     # Last five records

ls(ECEAP3)         # Vars name

# Should have equal or smaller obs!

## 390 obs & 6 vars to 375 obs & 6 vars

### Clear?
```

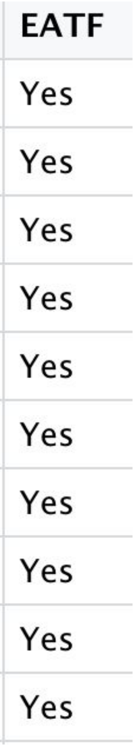


Data	
ECEAP1	390 obs. of 6 variables
ECEAP2	390 obs. of 6 variables
ECEAP3	375 obs. of 6 variables
Original	391 obs. of 23 variables
Revision	390 obs. of 23 variables

[Transform] Value Condition: `filter`

Quality Check: Once more on your Console

```
View(ECEAP3)                # On your Console

# EATF should only indicate “Yes” values
```



EATF
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes

[Transform] Splitting Text: `extract`

- In some cases, a developer of a survey, data set, or data manager prefers to have a cell with “FirstName” & “LastName” combined.
  - This is not considered as a preferred way of collecting data as indicated by the framework of **“Tidy Data”**
  - “Every column is variable.”
  - “Every row is an observation.”
  - **“Every cell is a single value.”**
- Well, there’s a way around to at least “split” this cell value using `**extract**` function
- Let's split names from “**EmergencyContact**” to
  - **“FirstName”**
  - **“LastName”**

[Transform] Splitting Text: `extract`

Input

```
ECEAP4 <- extract(ECEAP3, "EmergencyContact", c("FirstName", "LastName"), "([^ ]+) (.*)")

# Still can't figure out this code - Retrieved from Stackflow

## You'll notice on the Environment that now we have 7 vars!
```

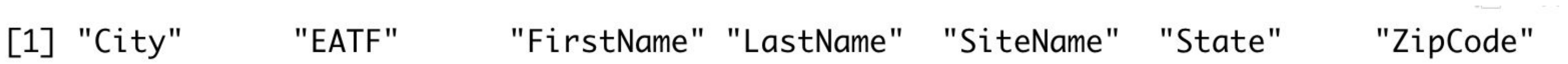
[Transform] Splitting Text: `extract`

Quality Check | Output

head(ECEAP4, 5)

tail(ECEAP4, 5)

ls(ECEAP4)



```
[1] "City"       "EATF"       "FirstName" "LastName"   "SiteName"   "State"      "ZipCode"
```

# Yes! 375 obs & 7 vars!

[Transform] Save Applicable Variables: `select`

- What if we don’t need some of the variables? i.e. “EATF” or “SiteName”?
- We’re going to **`select`** variables that we’d only need for our final data set!

Input

```
Final <- ECEAP4 %>%

  select("FirstName", "LastName", "City", "State", "ZipCode") %>%

as.data.frame()

# Should have 5 vars
```

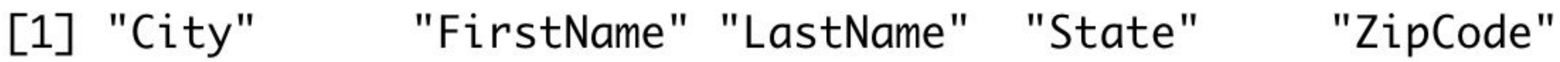
[Transform] Save Applicable Variables: `select`

Quality Check | Output

```
head(Final, 5)
```

```
tail(Final, 5)
```

```
ls(Final)
```



```
 [1] "City"        "FirstName" "LastName"  "State"     "ZipCode"
```

```
# 375 obs & 5 variables!
```

[Transform] Save as a CSV file: `write.csv`

- Final step! How do I save it as a new csv file?

```
write.csv(FinalDataFrame, "FileName_ProjectName_MMDDYY”)

# Make sure to check your final csv data.

## R will likely generate id on the first column.
```

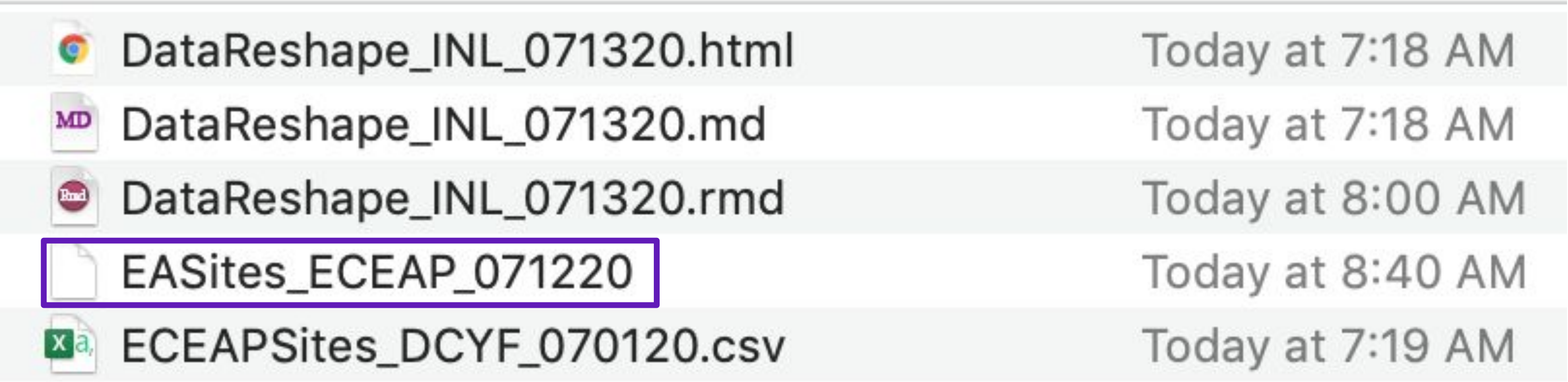
[Transform] Save as a CSV file: `write.csv`

Input

```
write.csv(Final, "EASites_ECEAP_071220")

# Make sure to check your final csv data.

## R will likely generate id on the first column.
```

[Transform] Save as a CSV file: `write.csv`

Output (In your folder that you saved your RMD file)



DataReshape_INL_071320.html	Today at 7:18 AM
DataReshape_INL_071320.md	Today at 7:18 AM
DataReshape_INL_071320.rmd	Today at 8:00 AM
EASites_ECEAP_071220	Today at 8:40 AM
ECEAPSites_DCYF_070120.csv	Today at 7:19 AM

# Make sure to check your final csv data & add .csv after file name if this happens.

## R will likely generate id on the first column.

Questions?

Summary

- Understanding the workflow of “tidyverse” packages & RMD is considered strong suit for becoming a data scientist.
- Transformation order matters.
- Applying to your own data set will enhance your learning!

Next Steps

- Scenario!

MERIT Scenario

- DCYF MERIT team notified Cultivate Learning that they have a capacity to upload our Circle Time Magazine data in a mass upload format!
- The format requires you to extract variables i.e. “FirstName”, “LastName”, “CompletedDate”, “StarsID”, and “Amount.”
- CTM Qualtrics file is available as a raw file but it’s a mess.
- On the other hand, you were able to find a RMD file that the senior data scientist created long time ago!
- **You have 72 hours to accomplish this task to clean a raw data set into a format that DCYF wants!**

Thank you!

감사합니다!