



2. Introduction to the Tidyverse

Jonathan Hersh, PhD (Chapman Argyros School of Business)

12/2/20

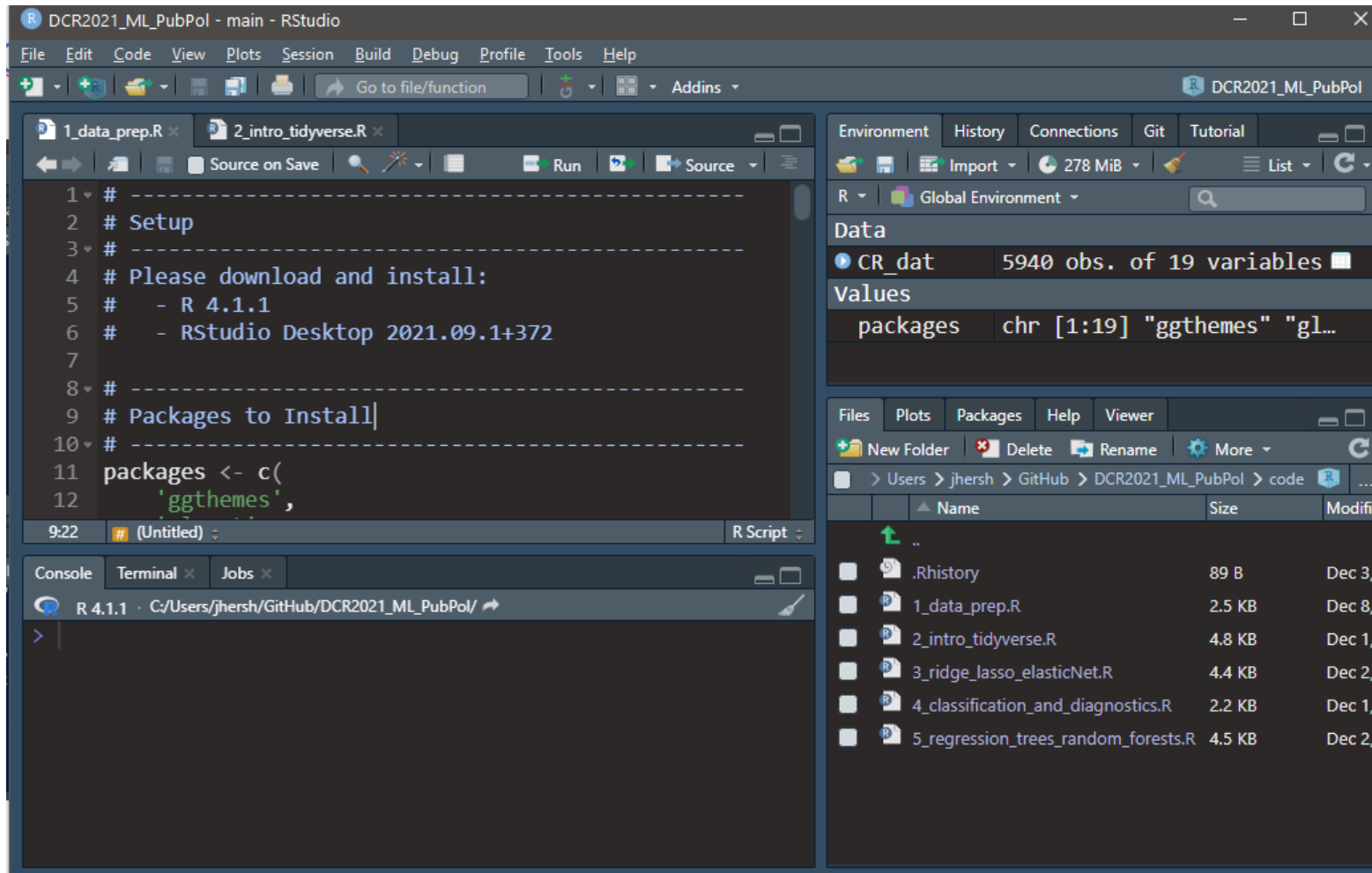
Section 2: Outline

1. R Projects

2. Data Analysis

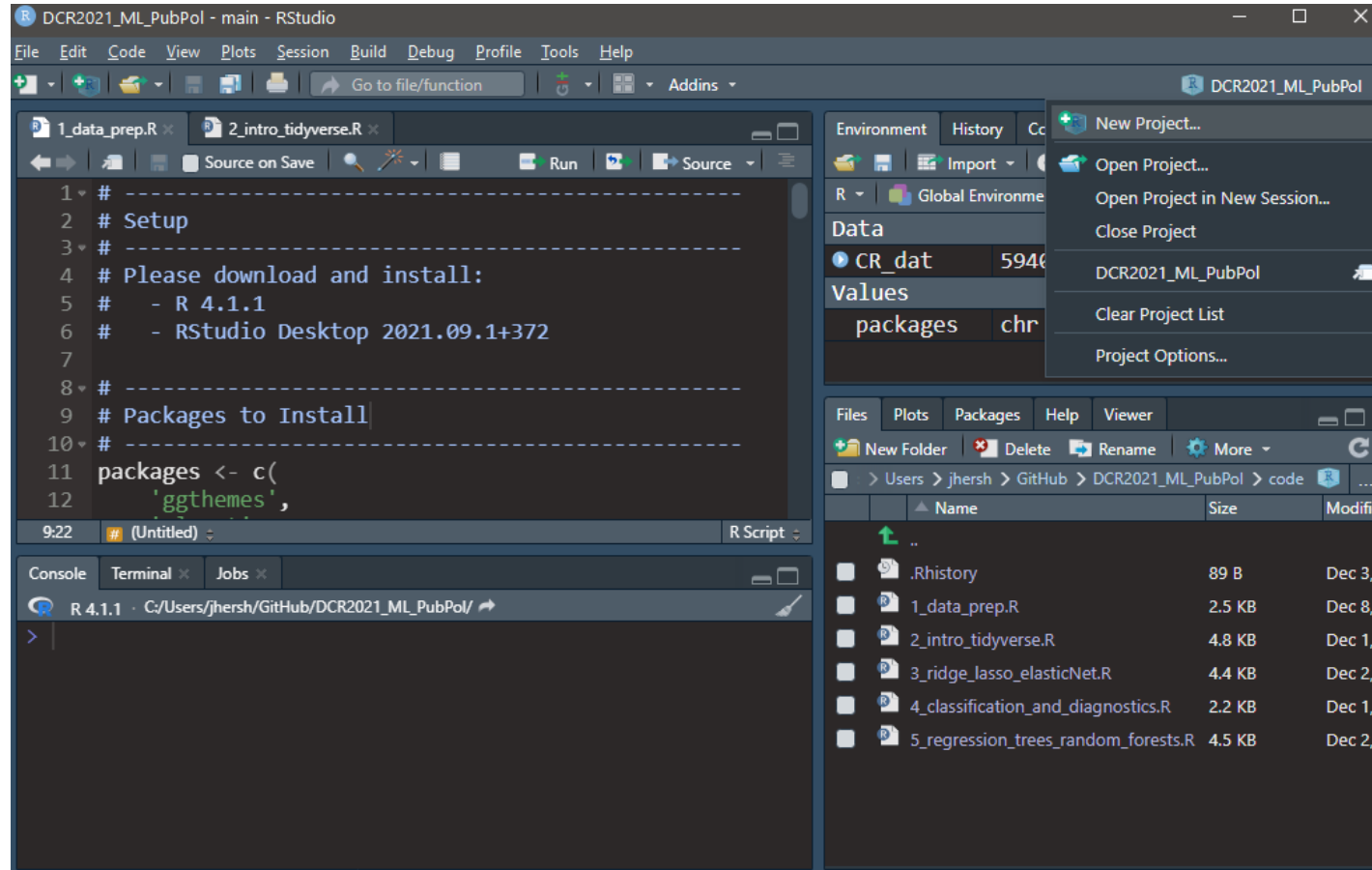
- Loading data
- Glimpse to view
- Pipe operator
- slice() to select rows
- arrange() to order data frame
- select() to choose variables
- rename() to rename variables
- filter() to select rows matching characteristics
- mutate() to create new variables
- group_by() and summarize() to create group

Rstudio Projects



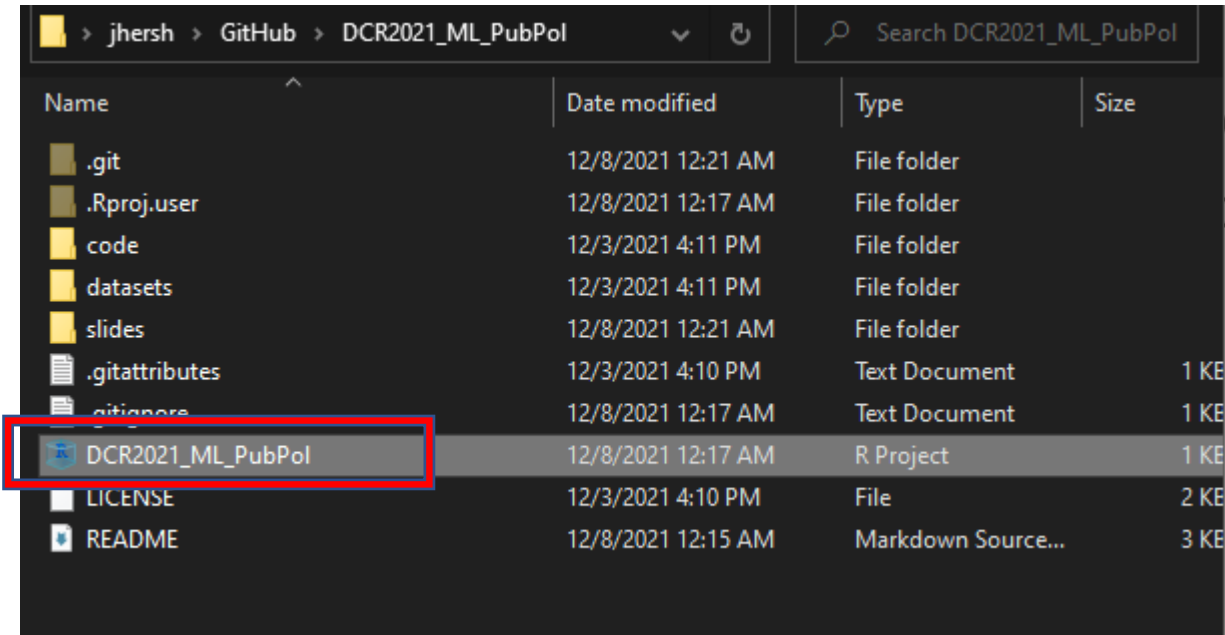
- Rstudio Projects are environments for your code
- They make your life easier, especially across teams and platforms
- Always use Rstudio Projects!

Rstudio Projects



- I recommend you create a new project in your github repo folder
- Select file -> new project

Rstudio Projects



- This creates an .Rproj file you should click and open whenever you want to run code for this class

The screenshot displays the RStudio Desktop environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar below the menu bar contains icons for file operations and a 'Go to file/function' search bar. The source editor on the left shows an R script with the following code:

```

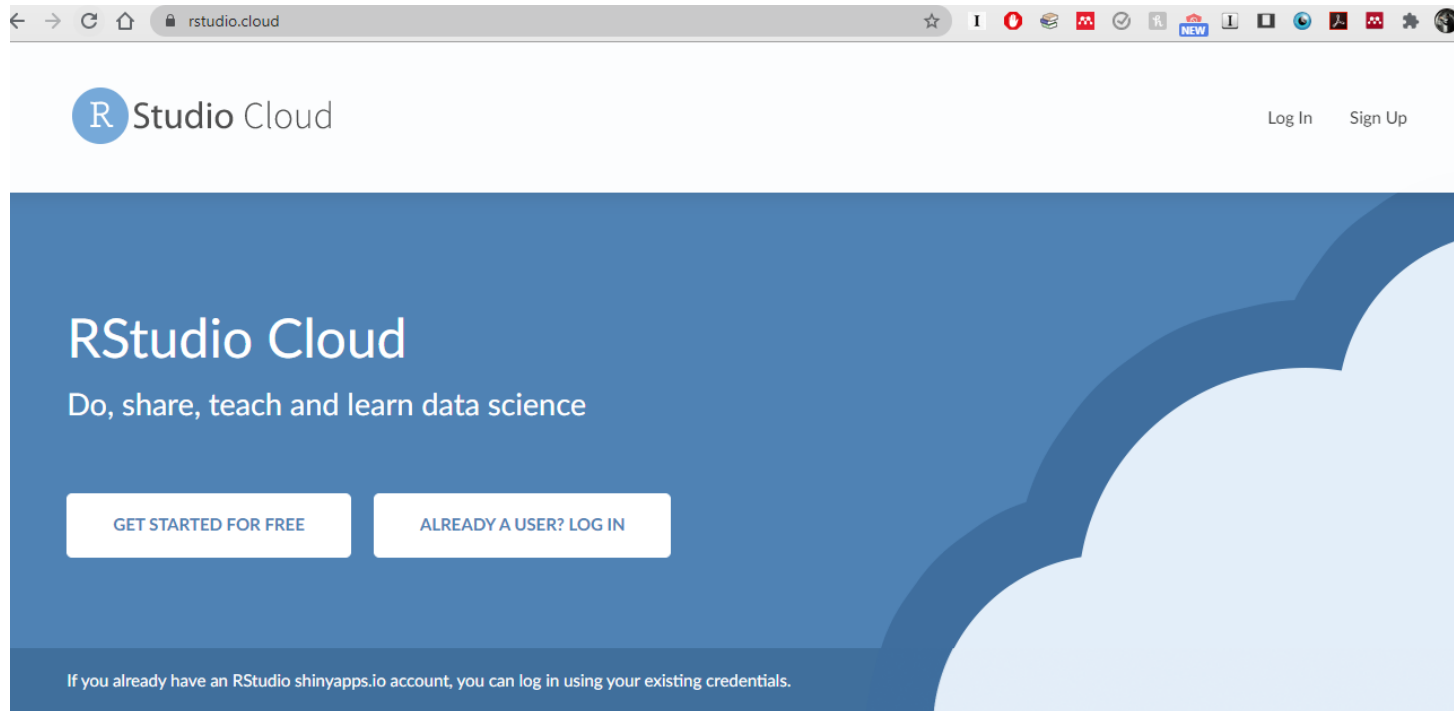
1 # -----
2 # Setup
3 # -----
4 # Please download and install:
5 #   - R 4.1.1
6 #   - RStudio Desktop 2021.09.1+372
7 # -----
8 # Packages to Install|
9 # -----
10 # -----
11 packages <- c(
12   'ggthemes',

```

The console at the bottom shows the R version (R 4.1.1) and the current directory (C:/Users/jherish/GitHub/DCR2021_ML_PubPol/). The file explorer on the right shows the project structure, including a 'code' directory with several R scripts.

All links to files,
datasets are
relative to this
root folder

R Studio Cloud



- Go to rstudio.cloud if your version of R is ever not working

Data science without the hardware hassles

RStudio Cloud is a lightweight, cloud-based solution that allows anyone to do, share, teach and learn data science online.

- Analyze your data using the RStudio IDE, directly from your browser.
- Share projects with your team, class, workshop or the world.
- Teach data science with R to your students or colleagues.
- Learn data science in an instructor-led environment or with interactive tutorials.

\$ AVAILABLE PRICING PLANS

🔗 RSTUDIO CLOUD GUIDE

📄 RSTUDIO.COM

R Studio Cloud

The screenshot displays the RStudio Cloud web interface in a browser window. The address bar shows the URL `rstudio.cloud/project/118272`. The main workspace is titled "Your Workspace / Untitled Project" and includes a menu bar with options like File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar is a toolbar with icons for file operations and a "Go to file/function" search bar. The central pane shows an R script with the following code:

```
1 library('ggplot2')
2 install.packages('glmnet')
3 install.packages('glmnetUtils')
4 library('glmnet')
5 library('glmnetUtils')
6 library(ggplot2)
7 data(mpg)
8
9 movies <- read.csv("movie_metadata.csv")
10 head(movies)
11 ridge <- cv.glmnet(gross ~ duration + num_voted_users +
12                   budget, data = movies, alpha = 0)
13 plot(ridge)
14 coef(ridge, s = ridge$lambda.min)
15
16 lasso_mod <- cv.glmnet(gross ~ duration + num_voted_users +
```

The right-hand side of the interface features several panels: "Environment" (showing "Global Environment" and "Environment is empty"), "History", "Connections", and "Tutorials". Below these is a "Files" panel with tabs for "Files", "Plots", "Packages", "Help", and "Viewer". The "Files" panel shows a file tree with the following items:

Name	Size
..	
.Rhistory	24.2 KB
movie_metadata.csv	1.4 MB
project.Rproj	205 B

The bottom of the interface includes a "Console" panel with the following output:

```
* installing *binary* package 'ipred' ...
* DONE (ipred)
* installing *binary* package 'recipes' ...
* DONE (recipes)
* installing *binary* package 'caret' ...
* DONE (caret)

The downloaded source packages are in
  '/tmp/RtmpPdRM1A/downloaded_packages'
R version change [3.6.0 -> 3.6.3] detected when restoring session;
search path not restored
> |
```

- R Studio Cloud is a full featured version of R in your browser!

glimpse() to summarize the data

```
# -----  
# GLIMPSE to summarize data  
# -----  
# let's summarize the IDB poverty data using the glimpse function  
  
glimpse(CR_dat)
```

```
> glimpse(CR_dat)  
Rows: 5,940  
Columns: 19  
$ household_ID <chr> "21eb7fcc1", "0e5d7a658", "2c7317ea8", "2b~  
$ poor_stat    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~  
$ num_rooms    <int> 3, 4, 8, 5, 2, 3, 4, 2, 4, 3, 1, 5, 4, 5, ~  
$ bathroom     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, ~  
$ refrig       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~  
$ no_elect     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~  
$ no_toilet    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ~  
$ comp        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~  
$ dep_rate     <dbl> 0.0000000, 1.0000000, 1.0000000, 1.0000000~  
$ tv          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ~  
$ mobile      <int> 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~  
$ num_hh      <int> 1, 1, 1, 4, 4, 2, 2, 4, 2, 2, 3, 1, 3, 2, ~  
$ urban       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~  
$ mean_educ    <dbl> 10.000000, 12.000000, 11.000000, 11.000000~  
$ num_children <int> 0, 0, 0, 2, 2, 1, 0, 2, 0, 0, 0, 0, 0, 0, ~  
$ num_adults   <int> 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 3, 1, 3, 2, ~  
$ num_elderly  <int> 0, 1, 1, 0, 0, 0, 1, 1, 0, 2, 0, 0, 1, 0, ~  
$ disabled     <int> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~  
$ mar_stat     <fct> divorced, divorced, widowed, other, other,~
```

pipe operator %>%

```
# -----  
# Pipe Operator!  
# -----  
# The pipe operator "%>%" is super useful!  
# It allows us to execute a series of functions on an object in stages  
# The general recipe is Data_Frame %>% function1() %>% function2() etc  
# Functions are applied right to left  
  
CR_dat %>% glimpse()  
glimpse(CR_dat)  
  
# cmd shift  
CR_dat %>% glimpse()  
glimpse(CR_dat)
```

slice() to select rows

```
# -----  
# Slice function: to select ROWS  
# -----  
# SLICE: slice to view only the first 10 rows  
CR_dat %>% slice(1:10)  
  
# SLICE to view only rows 300 to 310  
CR_dat %>% slice(300:310)
```

arrange() to order dataset

```
# -----  
# Arrange function: to ORDER dataset  
# -----  
  
# arrange the dataframe in descending order by mean_educ  
CR_dat %>%  
  arrange(desc(mean_educ)) %>%  
  head()  
  
# arrange the dataframe in ascending order by mean_educ  
CR_dat %>%  
  arrange(mean_educ) %>%  
  head()
```

select() to select columns in a dataset

```
# -----  
# SELECT columns of the dataset using the 'select' function  
# -----  
# select then pass to table function  
CR_dat %>% select(poor_stat) %>% table()  
  
# select only columns starting with particular characters  
CR_dat %>%  
  select(starts_with("num")) %>%  
  head()  
  
# remove variables using - operator  
CR_dat %>% |  
  select(-num_rooms) %>%  
  head()
```

rename() to rename variables in a dataset

```
# -----  
# RENAME variables using the RENAME function  
# -----  
# note we must pass the DF back to the original data  
CR_dat <- CR_dat %>%  
  rename(HH_ID = household_ID)  
  
CR_dat %>% names()  
  
# change it back!  
CR_dat <- CR_dat %>%  
  rename(household_ID = HH_ID)
```


filter() to remove rows you don't want

```
# -----  
# FILTER and ONLY allow certain rows using the FILTER function  
# -----  
# only select households with poverty status  
# and see # of rows  
CR_dat %>%  
  filter(poor_stat == 1) %>%  
  head()  
  
CR_dat %>%  
  filter(mar_stat == "divorced") %>%  
  count()  
  
CR_dat %>%  
  filter(comp == 1 & num_hh > 3) %>%  
  count()
```

mutate() to create new variables

```
# -----  
# MUTATE to Transform variables in your dataset  
# -----  
# adding new variables using mutate()  
CR_dat <- CR_dat %>%  
  mutate(mean_educ_sq = mean_educ * mean_educ,  
         mean_educ_log = log(mean_educ + 1))  
  
# see average education and educ squared  
CR_dat %>% select(matches("educ")) %>% colMeans()  
  
# Same thing, but using the package purrr to "map"  
# the function mean to all the columns of the data frame  
CR_dat %>% select(matches("educ")) %>% map_df(mean)
```

group_by() and summarize() to create group variables

```
# -----  
# Create summary statistics by GROUP using group_by()  
# -----  
CR_dat <- CR_dat %>%  
  # group by urban rural status  
  group_by(urban)  
  
glimpse(CR_dat)  
  
# calculate average and sd of poverty by group  
CR_urb <- CR_dat %>%  
  # calculate average poor status  
  summarize(pov_avg = mean(poor_stat),  
            pov_sd = sd(poor_stat)) %>%  
  print()
```

Data Wrangling with dplyr and tidyr

Cheat Sheet



Syntax - Helpful conventions for wrangling

dplyr::tbl_df(iris)

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1           5.1           3.5           1.4
2           4.9           3.0           1.4
3           4.7           3.2           1.3
4           4.6           3.1           1.5
5           5.0           3.6           1.4
...
Variables not shown: Petal.Width (dbl),
Species (fctr)
```

dplyr::glimpse(iris)

Information dense summary of tbl data.

utils::View(iris)

View data set in spreadsheet-like display (note capital V).

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::%>%

Passes object on left hand side as first argument (or argument) of function on righthand side.

$x \%>\% f(y)$ is the same as $f(x, y)$
 $y \%>\% f(x, ., z)$ is the same as $f(x, y, z)$

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - A foundation for wrangling in R



In a tidy data set:

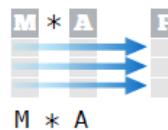
Each **variable** is saved in its own **column**

&

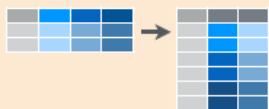


Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

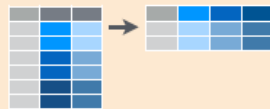


Reshaping Data - Change the layout of a data set



tidyr::gather(cases, "year", "n", 2:4)

Gather columns into rows.



tidyr::spread(pollution, size, amount)

Spread rows into columns.



tidyr::separate(storms, date, c("y", "m", "d"))

Separate one column into several.



tidyr::unite(data, col, ..., sep)

Unite several columns into one.

dplyr::data_frame(a = 1:3, b = 4:6)

Combine vectors into data frame (optimized).

dplyr::arrange(mtcars, mpg)

Order rows by values of a column (low to high).

dplyr::arrange(mtcars, desc(mpg))

Order rows by values of a column (high to low).

dplyr::rename(tb, y = year)

Rename the columns of a data frame.

Subset Observations (Rows)



dplyr::filter(iris, Sepal.Length > 7)

Extract rows that meet logical criteria.

dplyr::distinct(iris)

Remove duplicate rows.

dplyr::sample_frac(iris, 0.5, replace = TRUE)

Randomly select fraction of rows.

dplyr::sample_n(iris, 10, replace = TRUE)

Randomly select n rows.

dplyr::slice(iris, 10:15)

Select rows by position.

dplyr::top_n(storms, 2, date)

Select and order top n entries (by group if grouped data).

Subset Variables (Columns)



dplyr::select(iris, Sepal.Width, Petal.Length, Species)

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))

Select columns whose name contains a character string.

select(iris, ends_with("Length"))

Select columns whose name ends with a character string.

select(iris, everything())

Select every column.

select(iris, matches("t."))

Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))

Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))

Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))

Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)

Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)

Select all columns except Species.