RStudio®

The Little R Workshop: A Primer

12/29/16

250 Northern Ave, Boston, MA 02210
Phone: 844-448-1212

Email: info@rstudio.com
Web: http://www.rstudio.com

All Training materials are provided "as is" and without warranty and RStudio disclaims any and all express and implied warranties including without limitation the implied warranties of title, fitness for a particular purpose, merchantability and noninfringement.

The Training Materials are licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License. To view a copy of this license, visithttp://creativecommons.org/licenses/by-nc/3.0/us/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

R 4 DS

How we think about Data Science

R for Data Science, Grolemund & Wickham

R for Data Science Toolchain

Examining data frames in base R can be a challenge…

```
dim(ToothGrowth)
str(ToothGrowth)
Class(ToothGrowth)
ToothGrowth
```

Coerce a data frame to a tibble, now it’s pretty!...

```
tibble_ToothGrowth <- tbl_df(ToothGrowth)
tibble_ToothGrowth
```

To complement tibble() , tibble provides **as_tibble**() to coerce objects into tibbles. Generally, **as_tibble**() methods are much simpler than as.data.frame() methods

https://stackoverflow.com/questions/43942328/what-is-the-difference-between-as-tibble-as-data-frame-and-tbl-df

CC by RStudio

Tibbles – Quick Intro

- as_tibble()
- as_tibble(ToothGrowth)
- This will work for reasonable inputs that are already data.frames, lists, matrices, or tables.
- There are two main differences in the usage of a tibble vs. a classic data.frame: printing and subsetting
- Tibbles show only the first 10 rows
- Each column reports its type, a nice feature borrowed from str()
- package?tibble

RStudio

Tibbles – What to Know

- tibble() does much less than data.frame():
- A. it never changes the type of the inputs (e.g. it never converts strings to factors!)
- B. it never changes the names of variables, and it never creates row names
- Cdata.frame() vs. modern data_frame():
- Base R has a burning desire to turn character information into factor...via read.table() & data.frame() and other functions are also eager
- To shut this down, use stringsAsFactors = FALSE in read.table() and data.frame() or – even better – use the tidyverse!
- readr::read_csv(), readr::read_tsv(), etc. For data frame creation, use tibble::tibble()

RStudio

TIBBLES - EXAMPLE

- ToothGrowth
- str(ToothGrowth)
- str(ToothGrowth$supp)
- levels(ToothGrowth$supp)
- nlevels(ToothGrowth$supp)
- class(ToothGrowth$supp)
- summary(ToothGrowth$supp)
- 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. Tibbles are data.frames that are lazy and surly: they do less (i.e. they don't change variable names or types, and don't do partial matching) and complain more (e.g. when a variable does not exist). This forces you to confront problems earlier, typically leading to cleaner, more expressive code. Tibbles also have an enhanced print method() which makes them easier to use with large datasets containing complex objects.
- Notice how data.frame converts characters into factors, whereas tibble does not. No more stringsAsFactors = FALSE.
- SW <- data.frame(id = 1:3,trilogies = trilogies)
- SW2 <- tibble(id = 1:3,trilogies = trilogies)

RStudio

tidyverse
www.rstudio.com

Import
readr
readxl
haven
xml2
Tidy
tibble
tidyr
Transform
dplyr
forcats
hms
lubridate
stringr
Visualise
ggplot2
Model
broom
modelr
purrr
magrittr
Program

Masters of the Tidyverse

Art by Dan Mumford

https://github.com/rstudio/RStartHere

CC by RStudio

Tidyverse

A collection of R packages that share common philosophies and are designed to work together.

tidyverse.org

install.packages("tidyverse")
library(tidyverse)

RStudio

Tidy data

A data set is tidy iff:

1. It is a data frame (e.g. tibble)
2. Each variable is in its own column
3. Each case is in its own row

Standardized framework allows your data to fit nicely into the APIs provided by base R and other key packages

CC by RStudio

Tidyverse

A collection of R packages that share common philosophies and are designed to work together.

tidyverse.org

install.packages("tidyverse")
library(tidyverse)

tidyverse

An R package that serves as a short cut for installing and loading the components of the tidyverse.

CC by RStudio

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

does the equivalent of

```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
```

CC by RStudio

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

does the equivalent of

```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
```

```
library("tidyverse")
```

does the equivalent of

```
library("ggplot2")
library("dplyr")
library("tidyr")
library("readr")
library("purrr")
library("tibble")
```

CC by RStudio

tibbles

A type of data frame common throughout tidyverse packages. Tibbles enhance data frames in three ways:

1. Subsetting - [always returns a new tibble, [[and $ always return a new vector
2. No partial matching - You must use full column names when subsetting
3. Display - When you print a tibble, R provides a concise view of the data that fits on one screen

CC by RStudio

tibble

A package with several helper functions for tibbles:

- as_tibble() - convert a data frame to a tibble
- as.data.frame() - convert a tibble to a data frame
- tribble() - make a tibble (transversed)

```
tribble(
   ~x,  ~y,
    1, "a",
    2, "b",
    3, "c")
```

x	y
1	a
2	b
3	c

CC by RStudio

babynames

Names of male and female babies born in the US from 1880 to 2008. 1.8M rows.

```
# install.packages("babynames")
library(babynames)
```

CC by RStudio

```
View(babynames)
```

	year	sex	name	n	prop
1	1880	F	Mary	7065	0.0723835869
2	1880	F	Anna	2604	0.0266789611
3	1880	F	Emma	2003	0.0205214897
4	1880	F	Elizabeth	1939	0.0198657856
5	1880	F	Minnie	1746	0.0178884278
6	1880	F	Margaret	1578	0.0161672045
7	1880	F	Ida	1472	0.0150811946
8	1880	F	Alice	1414	0.0144869628
9	1880	F	Bertha	1320	0.0135238973
10	1880	F	Sarah	1288	0.0131960453
11	1880	F	Annie	1258	0.0128886840

CC by RStudio

tibbles

CC by RStudio

Transforming Data & Data Visualization

Art by Lou Pimentel

CC by RStudio

dplyr

A package that transforms data.

dplyr implements a *grammar* for transforming tabular data.

Data transformation toolbox

CC by RStudio

Power of dplyr - sparklyr

http://spark.rstudio.com/dplyr.html

http://spark.rstudio.com/

http://spark.rstudio.com/examples-emr.html

It is often said that 80% of data analysis is spent on the process of cleaning and preparing the data. (Dasu and Johnson, 2003)

CC by RStudio

Scaling with R

1. Extract data
Sample into R Memory

2. Compute on the parts
HPC, R parallel, etc.

3. Compute on the whole
sparklyr, MR, etc.

CC by RStudio

Sparklyr

Makes it easy to use R with Spark

Spark with AWS EMR
Spark and the data lake

CC by RStudio

single table verbs

filter() - extract cases
arrange() - reorder cases
group_by() - group cases
select() - extract variables
mutate() - create new variables
summarise() - summarise variables / create cases

CC by RStudio

common syntax

Each function take a data frame / tibble as its first argument and returns a data frame / tibble.

```
filter(.data, … )
```

dplyr function

data frame to transform

specific arguments

CC by RStudio

two table verbs

bind_rows() - adds one table to another as new cases
union(), intersect(), setdiff() - set operations for cases
semi_join(), anti_join() - filters cases in one table against another
bind_cols() - adds one table to another as new variables
left_join(), right_join(), full_join(), inner_join() - mutates one table by matching values from another as new variables

CC by RStudio

Toy data

```
storms <- tribble(
     ~storm, ~wind, ~pressure,        ~date,
  "Alberto",   110,      1007, "2000-08-12",
     "Alex",    45,      1009, "1998-07-30",
  "Allison",    65,      1005, "1995-06-04",
      "Ana",    40,      1013, "1997-07-01",
   "Arlene",    50,      1010, "1999-06-13",
   "Arthur",    45,      1010, "1996-06-21"
)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

CC by RStudio

filter()

CC by RStudio

filter()

Extract rows that meet logical criteria.

```
filter(.data, …)
```

data frame to transform

one or more logical tests (filter returns each row for which the test is TRUE)

CC by RStudio

filter()

Extract rows that meet logical criteria.

```
filter(storms, wind == 45)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Alex	45	1009	1998-07-30
Arthur	45	1010	1996-06-21

= sets

(returns nothing)
== tests if equal
(returns TRUE or FALSE)

CC by RStudio

filter()

Extract rows that meet logical criteria.

```
filter(storms, wind == 45)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	wind	pressure	date
Alex	45	1009	1998-07-30
Arthur	45	1010	1996-06-21

CC by RStudio

filter()

Extract rows that meet logical criteria.

```
filter(storms, wind == 45, pressure == 1009)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	wind	pressure	date
Alex	45	1009	1998-07-30

CC by RStudio

babynames

Names of male and female babies born in the US from 1880 to 2008. 1.8M rows.

```
# install.packages("babynames")
library(babynames)
```

CC by RStudio

```
View(babynames)
```

	year	sex	name	n	prop
1	1880	F	Mary	7065	0.0723835869
2	1880	F	Anna	2604	0.0266789611
3	1880	F	Emma	2003	0.0205214897
4	1880	F	Elizabeth	1939	0.0198657856
5	1880	F	Minnie	1746	0.0178884278
6	1880	F	Margaret	1578	0.0161672045
7	1880	F	Ida	1472	0.0150811946
8	1880	F	Alice	1414	0.0144869628
9	1880	F	Bertha	1320	0.0135238973
10	1880	F	Sarah	1288	0.0131960453
11	1880	F	Annie	1258	0.0128886840

CC by RStudio

```
Phil <- filter(babynames, name == "Phil", sex == "M")
P <- ggplot(data = Phil, mapping = aes(year, prop)) +
  geom_line()
require(scales)
p + scale_y_continuous(labels = percent)
```

CC by RStudio

Logical tests

?Comparison

<	Less than
>	Greater than
==	Equal to
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to
%in%	Group membership
is.na()	Is NA
!is.na()	Is not NA

?base::Logic

&	and
|	or
xor()	exactly or
!	not
any()	any true
all()	all true

CC by RStudio

```
gnames <- c("Nathan", "Jonathan", "Phil")
group_names <- filter(babynames, name %in% gnames)
ggplot(group_names, aes(year, prop, color = name)) +
  geom_line()
```

CC by RStudio

arrange()

CC by RStudio

arrange()

Order rows from smallest to largest values.

```
arrange(.data, …)
```

data frame to transform

one or more columns to order by (additional columns will be used as tie breakers)

CC by RStudio

arrange()

Order rows from smallest to largest values.

```
arrange(storms, wind)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	wind	pressure	date
Ana	40	1013	1997-07-01
Alex	45	1009	1998-07-30
Arthur	45	1010	1996-06-21
Arlene	50	1010	1999-06-13
Allison	65	1005	1995-06-04
Alberto	110	1007	2000-08-12

CC by RStudio

desc()

Changes ordering to largest to smallest.

```
arrange(storms, desc(wind))
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21
Alex	45	1009	1998-07-30
Ana	40	1013	1997-07-01

CC by RStudio

select()

CC by RStudio

select()

Extract columns by name.

```
select(.data, …)
```

data frame to transform

name(s) of columns to extract
(or a select helper function)

CC by RStudio

select()

Extract columns by name.

```
select(storms, storm, pressure)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

CC by RStudio

select() helpers

: - Select range of columns

```
select(storms, storm:pressure)
```

- - Select every column but

```
select(storms, -c(storm, pressure))
```

starts_with() - Select columns that start with…

```
select(storms, starts_with("w"))
```

ends_with() - Select columns that end with…

```
select(storms, ends_with("e"))
```

CC by RStudio

select() helpers

contains() - Select columns whose names contain…

```
select(storms, contains("d"))
```

matches() - Select columns whose names match regular expression

```
select(storms, matches("^.{4}$"))
```

one_of() - Select columns whose names are one of a set

```
select(storms, one_of(c("storm", "storms", "Storm")))
```

num_range() - Select columns named in prefix, number style

```
select(storms, num_range("x", 1:5))
```

CC by RStudio

mutate()

CC by RStudio

mutate()

Create new columns.

```
mutate(.data, …)
```

data frame to transform

named argument
(that consists of the name of column to create set equal to an expression that creates it)

CC by RStudio

mutate()

Create new columns.

```
mutate(storm, ratio = pressure / wind)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	wind	pressure	date	ratio
Alberto	110	1007	2000-08-12	9.15
Alex	45	1009	1998-07-30	22.42
Allison	65	1005	1995-06-04	15.46
Ana	40	1013	1997-07-01	25.32
Arlene	50	1010	1999-06-13	20.20
Arthur	45	1010	1996-06-21	22.44

CC by RStudio

summarise()

CC by RStudio

summarise()

Compute table of summaries.

```
summarise(.data, …)
```

data frame to transform

named argument
(that consists of the name of column to create set equal to an expression that creates it)

CC by RStudio

summarise()

Compute table of summaries.

```
summarise(storms, avg_wind = mean(wind), n = n())
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	[illegible]
Allison	65	1005	[illegible]
Ana	40	1013	[illegible]
Arlene	50	1010	[illegible]
Arthur	45	1010	1996-06-21

→

avg_wind	n
59.17	6

Returns the number of cases (rows) Very useful!

CC by RStudio

n() and n_distinct()

Two helper functions for summarise.

```
summarise(storms,
  n = n(), # Number of cases / rows
  n_wind = n_distinct(wind) # number of unique values
)
```

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

n	n_wind
6	5

CC by RStudio

%>%

Little bunny Foo Foo
Went hopping through the forest
Scooping up the field mice
And bopping them on the head

CC by RStudio

```
# Every intermediate as a new object
foo_foo <- little_bunny()
foo_foo_1 <- hop(foo_foo, through = forest)
foo_foo_2 <- scoop(foo_foo_1, up = field_mice)
foo_foo_3 <- bop(foo_foo_2, on = head)

# Overwrite the original object
foo_foo <- hop(foo_foo, through = forest)
foo_foo <- scoop(foo_foo, up = field_mice)
foo_foo <- bop(foo_foo, on = head)

# Nested function calls
bop(
  scoop(
    hop(foo_foo, through = forest),
    up = field_mice
  ),
  on = head
)
```

CC by RStudio

```
foo_foo %>%
  hop(through = forest) %>%
  scoop(up = field_mouse) %>%
  bop(on = head)
```

CC by RStudio

```
head(filter(select(iris, Species,
Petal.Length), Species ==
"setosa"))
```

is clearer when written:

```
iris %>% select(Species,
Petal.Length) %>%
filter(Species == "setosa") %>%
head()
```

CC by RStudio

```
mutate(babynames, percent = prop * 100)
```

```
summarize(babynames, n = n_distinct(name))
```

```
phil <- filter(babynames, name == "Phil", sex == "M")
summarise(phil, min = min(prop), mean = mean(prop),
  max = max(prop))
```

```
babynames2 <- mutate(babynames, percent = prop * 100)
babynames3 <- filter(babynames2, percent > 1)
summarise(babynames3, nn = n())
```

CC by RStudio

The pipe operator %>%

babynames mutate(___, percent = prop * 100)

Passes result on left into first argument of function on right.
So, for example, these do the same thing. Try it.

```
mutate(babynames, percent = prop * 100)
babynames %>% mutate(percent = prop * 100)
```

CC by RStudio

```
Phil <- filter(babynames, name == "Phil", sex == "M")
summarise(Phil, min = min(prop), mean = mean(prop),
  max = max(prop))
```

```
filter(babynames, name == "Phil", sex == "M") %>%
  summarise(min = min(prop), mean = mean(prop),
    max = max(prop))
```

CC by RStudio

```
Phil <- filter(babynames, name == "Phil", sex == "M")
summarise(Phil, min = min(prop), mean = mean(prop),
  max = max(prop))
```

```
babynames %>%
  filter(name == "Phil", sex == "M") %>%
  summarise(min = min(prop), mean = mean(prop),
    max = max(prop))
```

CC by RStudio

Shortcut to type %>%

Cmd + Shift + M (Mac)

Ctrl + Shift + M (Windows)

CC by RStudio

Grouping cases

group_by()

Groups cases by common values.

```
group_by(.data, …)
```

data frame to transform

column(s) to group by
(will group by combinations of values if more than one column is specified)

CC by RStudio

```
babynames %>%
  group_by(year, sex) %>%
  summarise(total = sum(n)) %>%
  ggplot(aes(x = year, y = total, color = sex)) +
    geom_line()
```

CC by RStudio

Example

- 1_dplyr_tidyr_r4ds_tidyverse.Rmd

CC by RStudio

Tidy Data with

tidyr

A package that reshapes the layout of tabular data.

CC by RStudio

Reshaping tables

spread() is one of a family of functions for reshaping tables.

- spread() - move values into column names
- gather() - move column names into values
- separate() - separate variables that share a column
- unite() - unite a variable that is split across several columns

Most data comes to you untidy

CC by RStudio

spread() and gather()

CC by RStudio

You can represent any group of values as **key:value pairs**

key	value
2011	7000
2011	5800
2011	15000
2012	6900
2012	6000
2012	14000
2013	7000
2013	6200
2013	13000

CC by RStudio

You can represent any group of values as **key:value pairs** or a **group of columns** with column names.

key	value
2011	7000
2011	5800
2011	15000
2012	6900
2012	6000
2012	14000
2013	7000
2013	6200
2013	13000

spread()

gather()

2011	2012	2013
7000	6900	7000
5800	6000	6200
15000	14000	13000

CC by RStudio

spread()

```
pollution %>% spread(key = size, value = amount)
```

data frame to reshape

column to use for keys (becomes new column names)

column to use for values (becomes new column cells)

CC by RStudio

gather()

```
cases %>% gather(key = "year", value = "n", 2:4)
```

data frame to reshape

name of the new key column (a character string)

name of the new value column (a character string)

numeric indexes of columns to collapse (or names)

CC by RStudio

separate()

CC by RStudio

separate()

Splits a column by dividing values at a specific character.

```
separate(storms, date, c("year","month","day"), sep = "-")
```

data frame to reshape

a column to split

names of new columns to make

string to split on
(Defaults to any non_alpha-numeric character)

CC by RStudio

```
separate(storms, date, c("year","month","day"), sep = "-",
  convert = TRUE)
```

	storm	wind	pressure	date
1	Alberto	110	1007	2000-08-03
2	Alex	45	1009	1998-07-27
3	Allison	65	1005	1995-06-03
4	Ana	40	1013	1997-06-30
5	Arlene	50	1010	1999-06-11
6	Arthur	45	1010	1996-06-17

	storm	wind	pressure	year	month	day
1	Alberto	110	1007	2000	08	03
2	Alex	45	1009	1998	07	27
3	Allison	65	1005	1995	06	03
4	Ana	40	1013	1997	06	30
5	Arlene	50	1010	1999	06	11
6	Arthur	45	1010	1996	06	17

CC by RStudio

unite()

CC by RStudio

unite()

Unites columns into single column by combining cells.

```
unite(data, col, …, sep = "")
```

name of new column to make (in quotes)

two or more columns to combine

separator to place between elements in new column (Defaults to an underscore)

CC by RStudio

```
storms %>%
  separate(date, c("year", "month", "day"), sep = "-") %>%
  unite("date", month, day, year, sep = "/")
```

CC by RStudio

Example

- 1_dplyr_tidyr_r4ds_tidyverse.Rmd
- At Bottom

CC by RStudio

Data Visualization with

CC by RStudio

ggplot2

A package that visualizes data.

ggplot2 implements the *grammar of graphics*, a system for building visualizations that is built around cases and variables.

CC by RStudio

ggplot2

CC by RStudio

ggplot2

CC by RStudio

ggplot2

CC by RStudio

ggplot2

CC by RStudio

ggplot2

CC by RStudio

ggplot2

CC by RStudio

To plot

1. "Initialize" a plot with ggplot()
2. Add layers with geom_ functions

Pro tip: Always put the + at the end of a line, Never at the beginning

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

CC CC by RStudio

To plot

1. "Initialize" a plot with ggplot()
2. Add layers with geom_ functions

data

+ before new

```
ggplot(mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

type of layer

aes()

x variable

y variable

A ggplot2 template

Make any plot by filling in the parameters of this template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

geom

The visual objects that represents the cases. geom functions add layers to the graph.

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

The geom

```
ggplot(mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

CC by RStudio

A ggplot2 template

Make any plot by filling in the parameters of this template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
                  stat = <STAT>) +
  <FACET_FUNCTION>
```

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut), stat = "count")
```

CC by RStudio

Example

- 2_ggplot2_r4ds_tidyverse

CC by RStudio

Data types with

CC by RStudio

Iteration with

CC by RStudio

Modeling with

CC by RStudio

htmlwidgets

htmlwidgets

htmlwidgets for R:

- R bindings to JavaScript libraries
- Used to create interactive visualizations
- A line or two of R code is all it takes to produce an example

Use htmlwidgets in:

- RStudio viewer pane
- R Markdown files
- Shiny Apps

www.htmlwidgets.org

© 2015 RStudio, Inc. CC BY 4.0 © 2015 RStudio, Inc.

htmlwidgets gallery

http://gallery.htmlwidgets.org/

© CC 2015 RStudio, Inc.

R Graphics:

Four Main Graphical Systems in R:

- R's Base Graphics
- Grid Graphics System
- The lattice Package
- The ggplot2 Package – Created by Hadley Wickham

Why ggplot2?

- Consistent underlying - Grammar of Graphics (Wilkinson, 2005)
- Very flexible
- Mature and complete graphics system
- Many users, active mailing list
- http://www.cookbook-r.com & http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

© CC 2015 RStudio, Inc.

https://beta.rstudioconnect.com/content/3066/#1

http://www.wthr.com/article/watch-video-shows-tornado-destroying-kokomo-starbucks

© CC 2015 RStudio, Inc.

Example

- 1_htmlwidgets_tornadoes.R
- 2_htmlwidgets_Bike_Stations.R
- 3_htmlwidgets_Unemployment.R

© CC 2015 RStudio, Inc.

R Markdown

The Magic of R Markdown…:

IDE - R Markdown & Notebooks…

http://r4ds.had.co.nz/intro.html

© CC 2015 RStudio, Inc.

Reproducible Research via R Markdown

Can we do better?

Ctrl + C (Copy)

Ctrl + V (Paste)

© 2015 RStudio, Inc. CC BY 4.0 RStudio, Inc.

R Markdown

© 2015 RStudio, Inc. CC BY 4.0 RStudio, Inc.

demo

1. Open a new `.Rmd` file
2. Edit text
3. Click "Knit HTML"
4. Preview output
5. Access output file
6. Publish

© 2015 RStudio, Inc. CC BY 4.0 © 2015 RStudio, Inc.

Parameters

A list of values that you can call in R code chunks

params list

elements and values

```
---
title: "Untitled"
output: html_document
params:
  filename: "data.csv"
  symbol: "GOOG"
---
```

Access as params$filename and params$symbol

© CC 2015 RStudio, Inc.

Recap: R Markdown

Reproducible

Automatic

Flexible

Reusable

params:

Parameterizable

© CC 2015 RStudio, Inc.

Teach yourself R Markdown

The R Markdown Development Center

rmarkdown.rstudio.com

R Markdown — Dynamic D

Garrett

rmarkdown.rstudio.com

R Markdown v2 | Home | Authoring | Formats | Developer

R Markdown — Dynamic Documents for R

R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R. It combines the core syntax of markdown (an easy-to-write plain text format) with embedded R code chunks that are run so their output can be included in the final document. R Markdown documents are fully *reproducible* (they can be automatically regenerated whenever underlying R code or data changes).

This website describes R Markdown v2, a next generation implementation of R Markdown based on knitr and pandoc. This implementation brings many enhancements to R Markdown, including:

- Many available output formats including HTML, PDF, and MS Word.
- Support for creating Beamer, ioslides, and Slidy presentations.
- New markdown syntax including expanded support for tables and bibliographies.
- Hooks for customizing HTML and PDF output (include CSS, headers, and footers).
- Include raw LaTeX within markdown for advanced customization of PDF output.
- Compile HTML, PDF, or MS Word notebooks from R scripts.
- Extensiblity: create custom templates and even entirely new output formats.
- Create interactive R Markdown documents using Shiny.

Note that PDF output (including Beamer slides) requires an installation of TeX.

Quick Tour

Installation

You can install the R Markdown package from CRAN as follows:

```
install.packages("rmarkdown")
```

© CC 2015 RStudio, Inc.

The R Markdown Reference Guide

www.rstudio.com/resources/cheatsheets/

R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com

Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

1. Markdown Syntax
2. **Knitr chunk options**
3. Pandoc options

Syntax	Becomes

Make a code chunk with three back ticks followed by an r in braces. End the chunk with three back ticks:
``{r}
paste("Hello", "World!")
``
	Make a code chunk with three back ticks followed by an r in braces. End the chunk with three back ticks:
	`paste("Hello", "World!")`
	`## [1] "Hello World!"`

Place code inline with a single back ticks. The first back tick must be followed by an R, like this `r paste("Hello", "World!")`.	Place code inline with a single back ticks. The first back tick must be followed by an R, like this Hello World!.

Add chunk options within braces. For example, `echo=FALSE` will prevent source code from being displayed:
``{r eval=TRUE, echo=FALSE}
paste("Hello", "World!")
``
	Add chunk options within braces. For example, `echo=FALSE` will prevent source code from being displayed:
	`## [1] "Hello World!"`

Learn more about chunk options at http://yihui.name/knitr/options

Chunk options

option	default value	description
Code evaluation		
child	NULL	A character vector of filenames. Knitr will knit the files and place them into the main document.
code	NULL	Set to R code. Knitr will replace the code in the chunk with the code in the code option.
engine	'R'	Knitr will evaluate the chunk in the named language, e.g. **engine = 'python'**. Run **names(knitr::knit_engines$get())** to see supported languages.
eval	TRUE	If **FALSE**, knitr will not run the code in the code chunk.
include	TRUE	If **FALSE**, knitr will run the chunk but not include the chunk in the final document.
purl	TRUE	If **FALSE**, knitr will not include the chunk when running **purl()** to extract the source code.
Results		
collapse	FALSE	If **TRUE**, knitr will collapse all the source and output blocks created by the chunk into a single block.
echo	TRUE	If **FALSE**, knitr will not display the code in the code chunk above it's results in the final document.
		If **'hide'**, knitr will not display the code's results in the final document. If **'hold'**, knitr will delay displaying all output

© CC 2015 RStudio, Inc.

The R Markdown Cheat Sheet

www.rstudio.com/resources/cheatsheets/

R Markdown Cheat Sheet

learn more at rmarkdown.rstudio.com

rmarkdown 0.2.50 Updated: 8/14

1. Workflow

R Markdown is a format for writing reproducible, dynamic reports with R. Use it to embed R code and results into slideshows, pdfs, html documents, Word files and more. To make a report:

i. Open - Open a file that uses the .Rmd extension.

ii. Write - Write content with the easy to use R Markdown syntax

iii. Embed - Embed R code that creates output to include in the report

iv. Render - Replace R code with its output and transform the report into a slideshow, pdf, html or ms Word file.

2. Open File

Start by saving a text file with the extension .Rmd, or open an RStudio Rmd template

- In the menu bar, click **File ▸ New File ▸ R Markdown...**
- A window will open. Select the class of output you would like to make with your .Rmd file
- Select the specific type of output to make with the radio buttons (you can change this later)
- Click OK

3. Markdown

Next, write your report in plain text. Use markdown syntax to describe how to format text in the final report.

syntax

```
Plain text
End a line with two spaces to start a new paragraph.
*italics* and _italics_
**bold** and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image: ![](path/to/smallorb.png)

horizontal rule (or slide break):

***

> block quote

* unordered list
* item 2
    + sub-item 1
    + sub-item 2

1. ordered list
2. item 2
    + sub-item 1
    + sub-item 2

Table Header   | Second Header
-------------- | -------------
Table Cell     | Cell 2
Cell 3         | Cell 4
```

becomes

Plain text
End a line with two spaces to start a new paragraph.
italics and *italics*
bold and **bold**
superscript^2
~~strikethrough~~
link

Header 1
Header 2
Header 3
Header 4
Header 5
Header 6

endash: –
emdash: —
ellipsis: ...
inline equation: $A = \pi * r^2$
image:
horizontal rule (or slide break):

> block quote

- unordered list
- item 2
  - sub-item 1
  - sub-item 2

1. ordered list
2. item 2
   - sub-item 1
   - sub-item 2

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4

4. Choose Output

Write a YAML header that explains what type of document to build from your R Markdown file.

YAML

A YAML header is a set of key: value pairs at the start of your file. Begin and end the header with a line of three dashes (- - -)

```
---
title: "Untitled"
author: "Anonymous"
output: html_document
---

This is the start of my
report. The above is metadata
saved in a YAML header.
```

The RStudio template writes the YAML header for you

The output value determines which type of file R will build from your .Rmd file (in Step 6)

- **output: html_document** ····· html file (web page)
- **output: pdf_document** ····· pdf document
- **output: word_document** ····· Microsoft Word .docx
- **output: beamer_presentation** ····· beamer slideshow (pdf)
- **output: ioslides_presentation** ····· ioslides slideshow (html)

RStudio® is a trademark of RStudio, Inc. • All rights reserved • info@rstudio.com • 844-448-1212 • rstudio.com

© CC 2015 RStudio, Inc.

R Markdown Render Function

rmarkdown::**render**

Render at the command line with YAML options

```
> render("doc.Rmd")
```

Render at the command line, override output format.

```
> render("doc.Rmd", "html_document")
```

Render at the command line to multiple formats.

```
> render("doc.Rmd", c("html_document", "pdf_document"))
```

© CC 2015 RStudio, Inc.

rmarkdown::**render**

Render at the command line with YAML options

```
> render("doc.Rmd")
```

Render at the command line, set parameters.

```
> render("doc.Rmd", params = list(
      filename = "other_data.csv",
      symbol = "AAPL")
```

© CC 2015 RStudio, Inc.

Examples

- 1_RMD_Stocks.Rmd
- 2_RMD_parameterized-html-stock.Rmd
- 3_RMD_parameterized-html-stock-many-inputs.Rmd

© CC 2015 RStudio, Inc.

Notebooks

Leonardo da Vinci…Page from the Codex Atlanticus shows notes and images about water wheels and Archimedean Screws

© CC 2015 RStudio, Inc.

Notebooks

- Number 3: Notebooks are for doing science
- Number 2: R Notebooks have great features
- Number 1: R Notebooks make it easy to create and share reports

https://rviews.rstudio.com/2017/03/15/why-i-love-r-notebooks/
http://r4ds.had.co.nz/r-markdown-workflow.html

© CC 2015 RStudio, Inc.

Notebooks

Combine in a single document:

- Narrative
- Code
- Output

Then Render to HTML

Useful websites

The Shiny development center:
shiny.rstudio.com

The R Markdown development center:
rmarkdown.rstudio.com

Shiny and R Markdown cheat sheets:
www.rstudio.com/resources/cheatsheets

RStudio products:
www.rstudio.com/products/shiny

© CC 2015 RStudio, Inc.

Examples

- 4_RMD_Model_notebook_tidyverse.Rmd
- 5_RMD_Model_notebook_tidyverse.Rmd

© CC 2015 RStudio, Inc.

Shiny

Motivation

- R is a powerful platform for data analysis
- State of the art statistical power
- A massive set of packages for statistical modelling, machine learning, visualization, and importing and manipulating data
- Large and enthusiastic community

© CC 2015 RStudio, Inc.

Shiny

R + Reactive Programming + HTML 5 + CSS 3

Web based User Interface

© 2015 RStudio, Inc. All rights reserved.

© CC 2015 RStudio, Inc.

Recap: UI

```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

Begin each app with the template

Add elements as arguments to **fluidPage()**

Create reactive inputs with an ***Input()** function

Display reactive results with an ***Output()** function

© CC 2015 RStudio, Inc.

Recap: Server

Use the server function to assemble inputs into outputs. Follow 3 rules:

output$hist <-

1. Save the output that you build to **output$**

```
renderPlot({
  hist(rnorm(input$num))
})
```

2. Build the output with a **render*()** function

input$num

3. Access input values with **input$**

© CC 2015 RStudio, Inc.

Recap

Open a new Shiny app with
File ▶ New File ▶ Shiny Web App…

Launch the app by opening app.R and clicking **Run App**

Close app by clicking the stop sign icon

Select view mode in the drop down menu next to Run App

© 2014 RStudio, Inc. All rights reserved.

© CC 2015 RStudio, Inc.

Warm up

1. Go to File ▶ New File ▶ Shiny Web App ▶ Single File
2. Click Run App at the top of the file to launch your first Shiny App.

© CC 2015 RStudio, Inc.

runApp

You can launch any app from the command line with `runApp`

```
runApp("~/Documents/App-1")
```

File path to app directory.
R will append the file path to the working directory, if path does not begin at the home directory

© 2016 RStudio, Inc. All rights reserved.

© CC 2015 RStudio, Inc.

INTRO PROCESS

- https://github.com/rstudio/shiny-examples/
- http://shiny.rstudio.com/gallery/single-file-shiny-app.html
- Panels
- ToothGrowth
- http://shiny.rstudio.com/gallery/basic-datatable.html
- http://shiny.rstudio.com/gallery/reactivity.html

© CC 2015 RStudio, Inc.

SHINY DEVELOPMENT EXAMPLE

- Go - http://shiny.rstudio.com/gallery/basic-datatable.html
- Create a new multi file shiny app in the IDE by going to 'File' and select 'New File' and select 'Shiny Web App…' and create a new multiple file app and title it 'clustering'
- Copy over the ui.R and app.R files with the code found: https://github.com/rstudio/shiny-examples/tree/master/030-basic-datatable
- Click 'Run App' in the IDE when done

© CC 2015 RStudio, Inc.

SHINY DEVELOPMENT EXAMPLE CONT.

- http://shiny.rstudio.com/gallery/plot-interaction-exclude.html
- http://shiny.rstudio.com/gallery/plot-plus-three-columns.html
- http://shiny.rstudio.com/gallery/plot-interaction-selecting-points.html

© CC 2015 RStudio, Inc.

Example

- 1_Shiny_HelloWorld
- 1_Shiny_ToothGrowth_HelloWorld
- 1_Shiny_ToothGrowth_Basic
- 2_Shiny_Basic_Plots
- 3_Shiny_ToothGrowth_Plots
- 4_Shiny_shinydashboard_basic
- 5_Shiny_htmlwidgets_Leaflet_Starbucks
- 6_Shiny_htmlTemplates

© CC 2015 RStudio, Inc.

What can an app do?

What can a Shiny app do?

Make R analysis accessible to **non-programmers**

Highly **customizable**, highly **shareable** HTML front end

Read and write to **databases**

Monitor **streaming data**

Require and use **authentication**

Ideal for Exploratory Data Analysis

Ideal Data Portal / Results Explorer / Simulation API / Dashboard

© CC 2015 RStudio, Inc.

Learn more

shiny.rstudio.com/articles

Interactive plots

Create interactive plots with base and ggplot2 graphics

Interactive plots

Selecting rows of data

Interactive plots - advanced

© 2015 RStudio, Inc. CC BY 4.0 RStudio, Inc.

Teach yourself Shiny

The Shiny Development Center

shiny.rstudio.com

© CC 2015 RStudio, Inc.

The Shiny Cheat Sheet

www.rstudio.com/resources/cheatsheets/

© 2015 RStudio, Inc. CC BY RStudio, Inc.

Other Shiny…

© CC 2015 RStudio, Inc.

Other Shiny…

- Shiny Dashboard
- HTML Templates
- https://channel9.msdn.com/Events/useR-international-R-User-conferences/useR-International-R-User-2017-Conference/Developing-and-deploying-large-scale-Shiny-applications-for-non-life-insurance
- https://beta.rstudioconnect.com/content/2771/

© CC 2015 RStudio, Inc.

Example

- 4_Shiny_shinydashboard_basic
- 5_Shiny_htmlwidgets_Leaflet_Starbucks
- 6_Shiny_htmlTemplates

© CC 2015 RStudio, Inc.

What Shiny functions are being used below?

https://beta.rstudioconnect.com/content/2976/#51

© CC 2015 RStudio, Inc.

Example

- 4_htmlwidgets_crosstalk_Cars
- 5_htmlwidgets_crosstalk_ToothGrowth
- 6_htmlwidgets_Crosstalk_Shiny

© CC 2015 RStudio, Inc.

Deployment

What is the difference between Shiny and Shiny Server?

Shiny

Free and open source R Package

Makes it incredibly easy to
build interactive web applications with R

Automatic reactive binding between inputs and outputs and extensive pre-built widgets make it possible to build beautiful, responsive, and powerful applications.

Shiny Server

Software you install on your server

Enable users to
host and managing Shiny applications

Scale a Shiny application to support many users
Protect and secure your applications
Manage the user experience

© CC 2015 RStudio, Inc.

Shiny Showcase

www.rstudio.com/products/shiny/shiny-user-showcase/

© 2015 RStudio, Inc. All rights reserved.

Every Shiny app is maintained by a computer running R

© 2015 RStudio, Inc. CC BY 4.0 RStudio, Inc.

Every Shiny app is maintained by a computer running R

© 2015 RStudio, Inc. CC BY 4.0 Creative Commons Attribution 4.0 International License

How do I deploy apps?

Develop and deploy on the same machine

Deploy Apps, R Markdown, and HTML

Scaling and Tuning Shiny Server Pro

RSTUDIO CONNECT

A new publishing platform for the work your teams create in R.

Share Shiny applications, R Markdown reports, dashboards, plots, and more in one convenient place.

RStudio

RStudio Connect

A publishing platform for all the work your team creates in R

Content Creators

Publish

Push button

RStudio Connect

Create variants
Schedule jobs
Distribute reports

Self-service

Content Consumers

Make decisions
Take actions
See results

Connect is Built for Data Scientists and IT

	Features	Solutions
Analyst	Push Button Deployment	Rapid Iteration
	Web UI for Managing Content	Self Service
	Email Integration	Content Sharing
	Job Scheduler	Automatic Updates
IT	User Roles	Access Control
	Authentication: AD / PAM / LDAP	Secure & Integrated
	Metrics and Logging	Scalable & Reliable

R Markdown

Other R Markdown Output Types

- Blogdown
- RMD Websites - Example
- Bookdown
- Presentations
- Flexdashboard – Let’s get back to Shiny

© CC 2015 RStudio, Inc.

Example

- 6_RMD_Flex_Dygraphs.Rmd
- 7_RMD_Flex_ToothGrowth
- 8_RMD_Flex_Perams
- 9_RMD_stock_flexdashboard
- 10_RMD_EconIndicators_Flexdashboard
- 11_RMD_DC_Home_Prices
- 12_RMD_Stocks_RMarkdown_Website

© CC 2015 RStudio, Inc.

Worlds Collide

- https://beta.rstudioconnect.com/content/2671/Combining-Shiny-R-Markdown.html
- https://beta.rstudioconnect.com/content/2792/Interactive%20Dashboards%20Shiny.nb.html

© CC 2015 RStudio, Inc.

Example

- 13_RMD_DB_Interactive.Rmd
- 14_RMD_Shiny_Flex_RMD_To_Shiny_Intro
- 15_Shiny_RMD_Flex_Shiny_ToothGrowth

© CC 2015 RStudio, Inc.

Other Goodies

- Cheatsheets
- Webinars & Recordings

Latest Webinars

Getting your data into R
You can't use R for data analysis unless you can get your data into R. Getting your data into R can be a major hassle, so in the last few months Hadley Wickham has been working hard to make it easier.

In this webinar Hadley will discuss the places you most often find data (databases, excel, text files, other statistical packages, web apis, and web pages) and the packages (DBI, xml2, jsonlite, haven, readr, exel) that make it easy to get your data into R.

GitHub Webinar Repository
Materials for this specific webinar

How to start with Shiny – Part 3
In this talk (Part 3 of 3), Garrett Grolemund will show you how to customize the appearance of your app. You will learn how to arrange the components of your app into an attractive layout, as well as how to change the appearance of text, images, and other HTML elements in your app.

GitHub Webinar Repository
Materials for this specific webinar
Alternate Link: bit.ly/shiny-quickstart-3

RStudio

Useful Websites

The Shiny development center:
shiny.rstudio.com

The R Markdown development center:
rmarkdown.rstudio.com

Shiny and R Markdown cheat sheets:
www.rstudio.com/resources/cheatsheets

RStudio products:
www.rstudio.com/products/shiny

RStudio®

RStudio®

Open Source & Free

Desktop: http://www.rstudio.com/products/rstudio/download/
RStudio Server: http://www.rstudio.com/products/rstudio/download-server/
Shiny Server: http://www.rstudio.com/products/shiny/download-server/
shinyapps.io beta: https://www.shinyapps.io/admin/#/signup

45 Day Evaluation of Pro Products

RStudio Server Pro: http://www.rstudio.com/products/rstudio-server-pro/evaluation/
Shiny Server Pro: http://www.rstudio.com/products/shiny-server-pro/evaluation/

PLEASE STAY IN TOUCH

Blog - http://blog.rstudio.org/

Twitter - @rstudio #rstats http://twitter.com/rstudio/

GitHub - https://github.com/rstudio/

LinkedIn - https://linkedin.com/company/rstudio-inc

Facebook - https://www.facebook.com/pages/RStudio-inc

Google+ - https://plus.google.com/110704473211154995841/posts

RStudio

Community Activities

R Consortium

Supports the R foundation

RStudio

Community

rstudio::conf

Annual conference

2016 - Palo Alto
2017 - Orlando Florida
2018 - San Diego

R Views

Community blog

Q/A…