The tidyverse

September 2016

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio
Import

Tidy → Transform
Consistent way of storing data
Create new variables & new summaries

Visualise
Surprises, but doesn’t scale

Model
Scales, but doesn’t (fundamentally) surprise

Communicate

Program
No matter how complex and polished the individual operations are, it is often the quality of the glue that most directly determines the power of the system.

— Hal Abelson
The tidy tools manifesto
tidyverse

Import
- readr
- readxl
- haven
- httr
- jsonlite
- DBI
- rvest
- xml2

Tidy
- tibble
- tidyr

Program
- purrr
- magrittr

Transform
- dplyr
- forcats
- hms
- lubridate
- stringr

Visualise
- ggplot2

Model
- broom
- modelr

http://r4ds.had.co.nz
1. Share data structures.
2. Compose simple pieces.
3. Embrace FP.
4. Write for humans.
Share data structures
Tidy data

1. Put each **dataset** in a data frame.

2. Put each **variable** in a column.
Messy data has a varied “shape”

# A tibble: 5,769 × 22

| iso2 | year | m04 | m514 | m014 | m1524 | m2534 | m3544 | m4554 | m5564 | m65 | mu | f04 | f514 | f014 | f1524 | f2534 | f3544 | f4554 | f5564 | f65 | fu |
|------|------|-----|------|------|-------|-------|-------|-------|-------|-----|----|-----|-----|------|-----|-------|-------|-------|-------|------|----|-----|
| AD   | 1989 | NA  | NA   | NA   | NA    | NA    | NA    | NA    | NA    | NA  | NA| NA  | NA  | NA   | NA  | NA    | NA    | NA    | NA    | NA  | NA |
| AD   | 1990 | NA  | NA   | NA   | NA    | NA    | NA    | NA    | NA    | NA  | NA| NA  | NA  | NA   | NA  | NA    | NA    | NA    | NA    | NA  | NA |
| AD   | 1991 | NA  | NA   | NA   | NA    | NA    | NA    | NA    | NA    | NA  | NA| NA  | NA  | NA   | NA  | NA    | NA    | NA    | NA    | NA  | NA |
| AD   | 1992 | NA  | NA   | NA   | NA    | NA    | NA    | NA    | NA    | NA  | NA| NA  | NA  | NA   | NA  | NA    | NA    | NA    | NA    | NA  | NA |
| AD   | 1993 | NA  | NA   | NA   | NA    | NA    | NA    | NA    | NA    | NA  | NA| NA  | NA  | NA   | NA  | NA    | NA    | NA    | NA    | NA  | NA |
| AD   | 1994 | NA  | NA   | NA   | NA    | NA    | NA    | NA    | NA    | NA  | NA| NA  | NA  | NA   | NA  | NA    | NA    | NA    | NA    | NA  | NA |
| AD   | 1996 | NA  | NA   | 0    | 0     | 0     | 4     | 1     | 0     | 0    | NA| NA  | NA  | NA   | 0    | 1     | NA    | NA    | NA    | 0    | 1  |
| AD   | 1997 | NA  | NA   | 0    | 0     | 1     | 2     | 2     | 1     | 6    | NA| NA  | NA  | NA   | 0    | 1     | NA    | NA    | NA    | 0    | 1  |
| AD   | 1998 | NA  | NA   | 0    | 0     | 0     | 1     | 0     | 0     | 0    | NA| NA  | NA  | NA   | 0    | 0     | NA    | NA    | NA    | 0    | 0  |
| AD   | 1999 | NA  | NA   | 0    | 0     | 0     | 1     | 1     | 1     | 0    | 0 | NA  | NA  | NA   | 0    | 0     | NA    | NA    | NA    | 0    | 0  |
| AD   | 2000 | NA  | NA   | 0    | 0     | 1     | 0     | 0     | 0     | 0    | NA| NA  | NA  | NA   | 0    | 0     | NA    | NA    | NA    | 0    | 0  |
| AD   | 2001 | NA  | NA   | 0    | NA    | NA   | 2     | 1     | NA    | NA   | NA| NA  | NA  | NA   | NA   | NA    | NA    | NA    | NA    | NA  | NA |
| AD   | 2002 | NA  | NA   | 0    | 0     | 0     | 1     | 0     | 0     | 0    | NA| NA  | NA  | NA   | 0    | 1     | NA    | NA    | NA    | 0    | 1  |
| AD   | 2003 | NA  | NA   | 0    | 0     | 0     | 1     | 2     | 0     | 0    | NA| NA  | NA  | NA   | 0    | 1     | NA    | NA    | NA    | 0    | 1  |
| AD   | 2004 | NA  | NA   | 0    | 0     | 0     | 1     | 1     | 0     | 0    | NA| NA  | NA  | NA   | 0    | 0     | NA    | NA    | NA    | 0    | 0  |
| AD   | 2005 | 0   | 0    | 0    | 0     | 1     | 1     | 0     | 0     | 0    | 0 | 0   | 0    | 0     | 0     | 0     | 0     | 0     | 0    | 0  |
| AD   | 2006 | 0   | 0    | 0    | 0     | 1     | 1     | 2     | 0     | 1    | 1 | 0   | 0    | 0     | 0     | 0     | 0     | 0     | 0    | 0  |

# ... with 5,752 more rows, and 6 more variables: f2534 <int>, f3544 <int>, f4554 <int>, f5564 <int>, f65 <int>, fu <int>

What are the variables in this dataset?
(Hint: f = female, u = unknown, 1524 = 15-24)
library(tidyr)

read_csv("tb.csv") %>%
gather(
m04:fu, key = demo, value = n,  
na.rm = TRUE
) %>%
separate(demo, c("sex", "age"), 1) %>%
arrange(iso2, year, sex, age) %>%
rename(country = iso2)
Tidy data has a uniform “shape”

# A tibble: 35,750 × 5

  country year sex  age  n
  <chr> <int> <chr> <chr> <int>
1   AD    1996 f   014   0
2   AD    1996 f  1524   1
3   AD    1996 f  2534   1
4   AD    1996 f  3544   0
5   AD    1996 f  4554   0
6   AD    1996 f   65   0
7   AD    1996 m   014   0
8   AD    1996 m  1524   0
9   AD    1996 m  2534   0
10  AD    1996 m  3544   0

# ... with 35,740 more rows
Sometimes you don’t have variables & cases

matrices

http://simplystatistics.org/2016/02/17/non-tidy-data/

dates

vectors

strings

factors

xml

HTTP requests

HTTP response
What if you have a mix of object types?

Cross-validation

Training data  Model  Predictions  RMSE
  data frame   lm          vector    scalar

Test data
  data frame
Use a tibble with list-columns!

# A tibble: 100 x 5

<table>
<thead>
<tr>
<th>train</th>
<th>test</th>
<th>.id</th>
<th>mod</th>
<th>rmse</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>001</td>
<td>&lt;S3: lm&gt;</td>
<td>0.5661605</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>002</td>
<td>&lt;S3: lm&gt;</td>
<td>0.2399357</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>003</td>
<td>&lt;S3: lm&gt;</td>
<td>3.5482986</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>004</td>
<td>&lt;S3: lm&gt;</td>
<td>0.2396810</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>005</td>
<td>&lt;S3: lm&gt;</td>
<td>0.1591336</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>006</td>
<td>&lt;S3: lm&gt;</td>
<td>0.1934869</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>007</td>
<td>&lt;S3: lm&gt;</td>
<td>0.2697834</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>008</td>
<td>&lt;S3: lm&gt;</td>
<td>0.4910886</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>009</td>
<td>&lt;S3: lm&gt;</td>
<td>1.7002645</td>
</tr>
<tr>
<td>&lt;S3: resample&gt;</td>
<td>&lt;S3: resample&gt;</td>
<td>010</td>
<td>&lt;S3: lm&gt;</td>
<td>0.2047787</td>
</tr>
</tbody>
</table>

... with 90 more rows
Your turn!

def <- data.frame(xyz = "a")
# What does this return?
df$x
df <- data.frame(xyz = "a")
# What does this return?
df$x
 #=> [1] a
 #=> Levels: a

Two surprises
partial name matching &
stringsAsFactors
Two important tensions for understanding base R

Interactive exploration

Programming

Conservative

Utopian
Tibbles are data frames that are lazy & surly

df <- tibble(xyz = "a")

df$xyz

#> [1] "a"

is.data.frame(df[, "xyz"])

#> [1] TRUE

df$x

#> Warning: Unknown column 'x'
#> NULL
And work better with list-columns

data.frame(x = list(1:2, 3:5))

#> Error: arguments imply differing number
#> of rows: 2, 3
And work better with list-columns

data.frame(x = list(1:2, 3:5))
#> Error: arguments imply differing number
#> of rows: 2, 3

data.frame(x = I(list(1:2, 3:5)))
#>    x
#> 1 1, 2
#> 2 3, 4, 5
And work better with list-columns

data.frame(x = list(1:2, 3:5))
#> Error: arguments imply differing number
#> of rows: 2, 3

data.frame(x = I(list(1:2, 3:5)))
#> x
#> 1 1, 2
#> 2 3, 4, 5

tibble(x = list(1:2, 3:5))
#> # A tibble: 2 x 1
#> x
#> <list>
#> 1 <int [2]>
#> 2 <int [3]>
Compose simple pieces
Goal: Solve complex problems by combining uniform pieces.
magrittr::

%>%

%>%
foo_foo <- little_bunny()

bop_on(
  scoop_up(
    hop_through(foo_foo, forest),
    field_Mouse
  ),
  head
)

# vs

foo_foo %>%
  hop_through(forest) %>%
  scoop_up(field_mouse) %>%
  bop_on(head)
library(nycflights13)
library(dplyr)
library(ggplot2)

flights %>%
group_by(date) %>%
summarise(n = n()) %>%
ggplot(aes(date, n)) +
geom_line()
And ggplot2 is not even internally consistent

ggplot(mtcars, aes(mpg, wt)) + geom_point() + geom_line() + ggsave("mtcars.pdf")
And ggplot2 is not even internally consistent

ggsave(
    "mtcars.pdf",
    ggplot(mtcars, aes(mpg, wt)) +
    geom_point() +
    geom_line() +
)

😱
ggplot1 had a tidier API than ggplot2!

```r
# devtools::install_github("hadley/ggplot1")
library(ggplot1)

ggsave(
  ggpoint(
    ggplot(
      mtcars,
      list(x = mpg, y = wt)
    ),
  ),
  "mtcars.pdf", width = 8, height = 6
)
```
library(ggplot1)

mtcars %>%
  ggplot(list(x = mpg, y = wt)) %>%
  ggpoint() %>%
  ggsave("mtcars.pdf", width = 8, height = 6)
library(rvest)
library(purrr)
library(readr)
library(dplyr)
library(lubridate)

read_html("https://www.massshootingstracker.org/data") %>%
  html_nodes("a[href^='https://docs.google']") %>%
  html_attr("href") %>%
  map_df(read_csv) %>%
  mutate(date = mdy(date)) -> shootings
Embrace FP
Why are for loops “bad”? Answered with cupcakes
Vanilla cupcakes

1 cup flour  
a scant ¾ cup sugar  
1 ½ t baking powder  
3 T unsalted butter  
½ cup whole milk  
1 egg  
¼ t pure vanilla extract

Preheat oven to 350°F.
Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.
Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.
Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.
Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.
Chocolate cupcakes

Preheat oven to 350°F.

Put the flour, cocoa, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.
Chocolate cupcakes

Preheat oven to 350°F.

Put the flour, cocoa, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps. Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.
Vanilla cupcakes

1 cup flour
a scant ¾ cup sugar
1 ½ t baking powder
3 T unsalted butter
½ cup whole milk
1 egg
¼ t pure vanilla extract

Preheat oven to 350°F.
Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.
Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.
Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.
Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.
Vanilla cupcakes

Preheat oven to 170°C.

Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.
Vanilla cupcakes

Beat flour, sugar, baking powder, salt, and butter until sandy.

Whisk milk, egg, and vanilla. Mix half into flour mixture until smooth (use high speed). Beat in remaining half. Mix until smooth.

Bake 20-25 min at 170°C.
Vanilla cupcakes

120g flour
140g sugar
1.5 tsp baking powder
40g butter
120ml milk
1 egg
0.25 tsp vanilla

Beat dry ingredients + butter until sandy.

Whisk together wet ingredients. Mix half into dry until smooth (use high speed). Beat in remaining half. Mix until smooth.

Bake 20-25 min at 170°C.
Cupcakes

Beat dry ingredients + butter until sandy.
Whisk together wet ingredients. Mix half into dry until smooth (use high speed). Beat in remaining half. Mix until smooth.
Bake 20-25 min at 170°C.

Vanilla
- 120g flour
- 140g sugar
- 1.5t baking powder
- 40g butter
- 120ml milk
- 1 egg
- 0.25 t vanilla

Chocolate
- 100g flour
- 20g cocoa
- 140g sugar
- 1.5t baking powder
- 40g butter
- 120ml milk
- 1 egg
- 0.25 t vanilla

4. Extract out common code
What do these for loops do?

```r
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```
For loops emphasise the objects

```r
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
    out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
    out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
Functional programming emphasises the actions

library(purrr)

means <- map_dbl(mtcars, mean)
medians <- map_dbl(mtcars, median)
Teaser: simulation

```r
sim <- tribble(~f, ~params,
               "runif", list(min = -1, max = 1),
               "rnorm", list(sd = 5),
               "rpois", list(lambda = 10))
sim %>%
    mutate(sim = invoke_map(f, params, n = 10))
```
reports <- tibble(
    class = unique(mpg$class),
    filename = paste0("fuel-economy-", class, ".html"),
    params = map(class, ~ list(my_class = .))
)

reports %>%
    select(output_file = filename, params) %>%
pwalk(rmarkdown::render, input = "fuel-economy.Rmd")
Write for humans
Programs must be written for people to read, and only incidentally for machines to execute.

— Hal Abelson
Conclusion
1. Share data structures.
2. Compose simple pieces.
3. Embrace FP.
4. Write for humans.
My goal is to make a pit of success

Gotta install them all

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

Import
- readr
- readxl
- haven
- httr
- jsonlite
- DBI
- rvest
- xml2

Tidy
- tibble
- tidyr

Program
- purrr
- magrittr

Transform
- dplyr
- forcats
- hms
- lubridate
- stringr

Visualise
- ggplot2

Model
- broom
- modelr
- ???

http://r4ds.had.co.nz
This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License.

To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/us/