

Thinking with Data Using R and RStudio: Powerful Idioms for Analysts

Nicholas J. Horton

Amherst College, Amherst, MA, USA

June 3, 2018

nhorton@amherst.edu

Acknowledgements

- joint work with Danny Kaplan (Macalester College), Randy Pruim (Calvin College), Ben Baumer (Smith College), and Hadley Wickham (Rice/RStudio)
- supported by NSF grant 0920350 (building a community around modeling, statistics, computation and calculus)

Real acknowledgements

- R Core team
- R Foundation (<https://www.r-project.org/foundation>)
- Hadley Wickham and RStudio
- many others...

Intended audience

- Analysts who are new to R and RStudio
- Analysts who have used R somewhat, but it's been a while
- Statistical educators working to reinvigorate their curricula

- Motivational sermon: statistics and data science
- Brief history and overview of R and RStudio
- Key idioms for data wrangling
- Case study: airline delays
- Closing thoughts

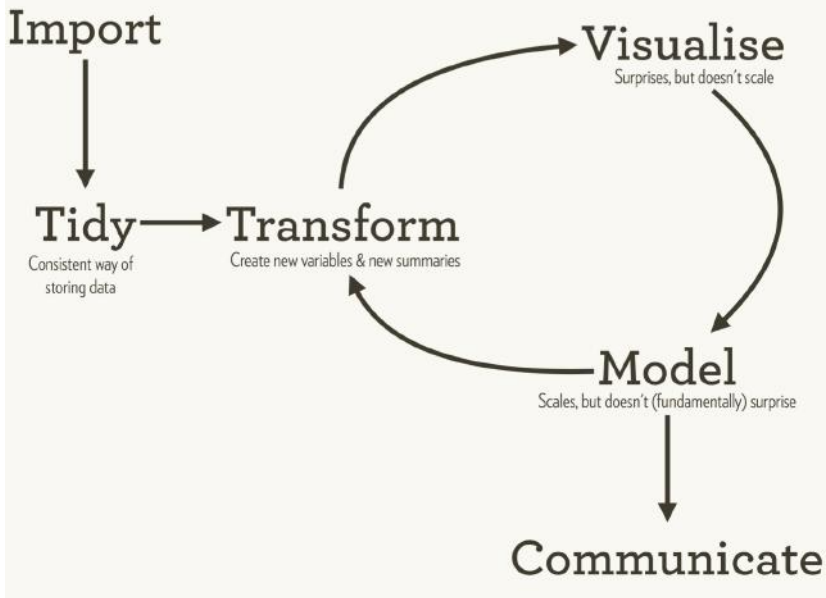
At its August 1992 meeting in Boston, the Committee on Applied and Theoretical Statistics (CATS) noted widespread sentiment in the statistical community that upper-level undergraduate and graduate curricula for statistics majors ... are currently structured in ways that do not provide sufficient exposure to modern statistical analysis, computational and graphical tools, communication skills, and the ever growing interdisciplinary uses of statistics.

Approaches and materials once considered standard are being rethought. The growth that statistics has undergone is often not reflected in the education that future statisticians receive. There is a need to incorporate more meaningfully into the curriculum the computational and graphical tools that are today so important to many professional statisticians. There is a need for improved training of statistics students in written and oral communication skills, which are crucial for effective interaction with scientists and policy makers. More realistic experience is needed in various application areas for which statistics is now a key to further progress.

We are concerned that many of our graduates do not have sufficient skills to be effective in the modern workforce. Thomas Lumley (personal communication) has stated that our students know how to deal with $n \rightarrow \infty$, but cannot deal with a million observations. If statistics is the science of learning from data, then our students need to be able to “think with data” (as Diane Lambert of Google has so elegantly described).

- Horton and Hardin (TAS, 2015)

Statistical analysis cycle (Wickham)



- Increased importance of data science
- Need for real applications
- Need for more diverse models and approaches
- Ability to communicate

- Increased importance of data science
- Need for real applications
- Need for more diverse models and approaches
- Ability to communicate

Good news: orders of magnitude decrease in degree of difficulty of data analysis

Why R and RStudio?

- 40+ years of S: Becker, Chambers, many others
- key role of “Interfaces”

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).

Why R and RStudio?

- 20+ years of R: Ihaka and Gentleman
- amazing efforts of the R Core and others
- open source with large community of developers and users
- more than 8,000 packages (up from 500 in 2005, 2000 in 2010)
- base R static (by design: too large an installed base to do more than tweak)

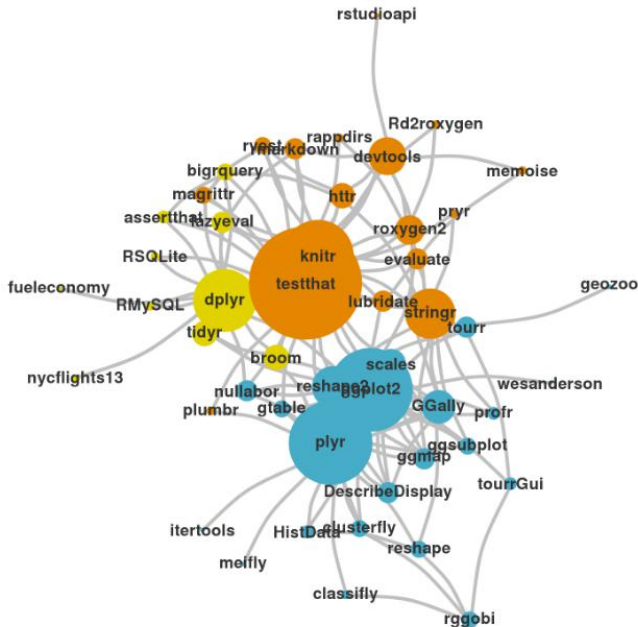
Why R and RStudio?

- 5+ years of RStudio
- designed for experts: great for novice users
- incremental improvement and punctuated equilibrium via new packages with professional polish
- development of “Hadleyverse” (83 co-authored packages)

Why R and RStudio?

- 5+ years of RStudio
- designed for experts: great for novice users
- incremental improvement and punctuated equilibrium via new packages with professional polish
- development of **tidyverse** (83 very useful packages)

More on the tidyverse (display by Adolfo Álvarez)



Design goals of tidyverse

- tools that work well together
- each one designed for a particular task
- if you don't succeed at first, try, try again (CS prototyping)
 - 1 `stats::reshape()`
 - 2 reshape package
 - 3 reshape2 package
 - 4 tidyr package
- compose simple steps with the pipe (`%>%`) operator
- connects output from one function to input of another
- clarifies complex data wrangling workflow

Life without the pipe operator

```
foo_foo <- little_bunny()
bop_on(
  scoop_up(
    hop_through(foo_foo, forest),
    field_mouse
  ),
  head
)
```

Life with the pipe operator

```
foo_foo %>%  
  hop_through(forest) %>%  
  scoop_up(field_mouse) %>%  
  bop_on(head)
```

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)**

Airline delays: better traveling via data

- Collected by the Bureau of Transportation Statistics since 1987
- All commercial flights within the US (more than 180 million records)
- Easily motivated: have you ever been stuck in an airport because your flight was delayed or cancelled and wondered if you could have predicted it if you'd had more data? (Wickham, JCGS, 2011)
- Details at <http://stat-computing.org/dataexpo/2009>
- Anyone flying home at the end of the JSM?

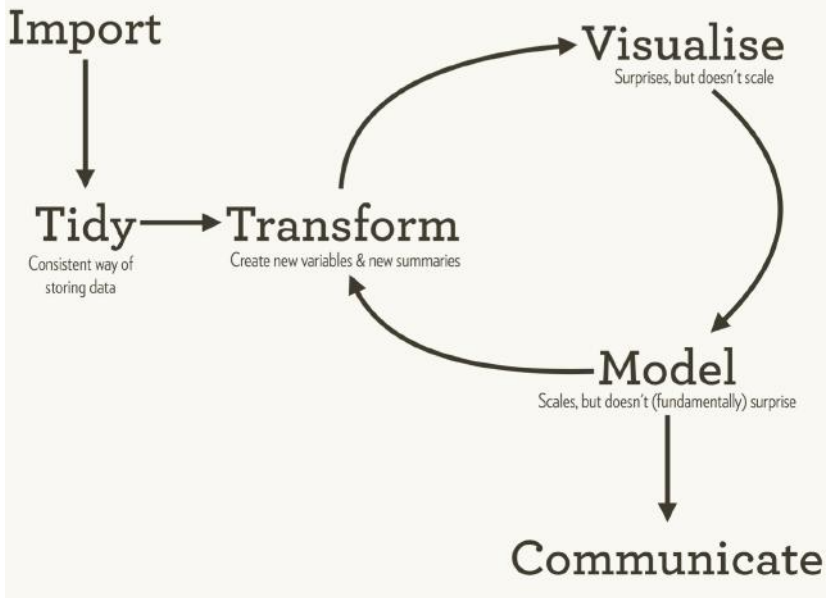
Airline Delays Codebook (flights table: abridged)

<code>year</code>	1987, 1998, ..., 2015
<code>month</code>	1 through 12
<code>day</code>	1 through 31
<code>dep_time</code>	departure time
<code>carrier</code>	OH = Comair, DL = Delta, etc.
<code>tailnum</code>	plane identifier
<code>arr_delay</code>	arrival delay, in minutes
<code>origin</code>	BOS, MDW, ORD, SFO, etc.
<code>dest</code>	

Full details at

http://www.transtats.bts.gov/Fields.asp?Table_ID=236

Statistical analysis cycle (Wickham)



Revolutions

Daily news about using open source R for big data analysis, predictive modeling, data visualization, since 2008

[« Where are the R users? | Main | Because it's Friday: Boy and Bear in the snow »](#)

April 10, 2015

New packages for reading data into R — fast

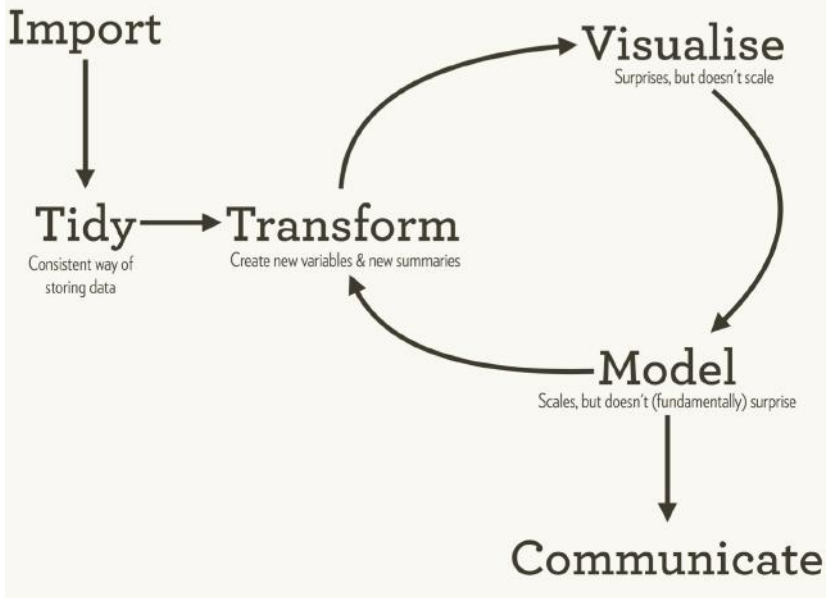
Hadley Wickham and the RStudio team have created some new packages for R, which will be very useful for anyone who needs to read data into R (that is, everyone). The [readr package](#) provides functions for reading text data into R, and the [readxl package](#) provides functions for reading Excel spreadsheet data into R. Both are much faster than the functions you're probably using now.

The readr package provides several functions for reading tabular text data into R. This is a task normally accomplished with the `read.table` family of functions in R, and readr provides a number of replacement functions that provide additional functionality and are *much* faster.

Importing the airline delays dataset (August, 2015)

- half a million domestic commercial flights last August
- ingest using `read.csv()` in around 30 seconds
- ingest using `readr::read_csv()` in about 8 seconds
- optimized for reading vast quantities of data *and no factors!*
- full dataset on the order of 40GB (not a problem for a modern laptop) using a database system
- improved functionality in `haven` package (import from SPSS, Stata, and SAS)

Statistical analysis cycle (Wickham)



A framework for tidy data

- 80% of data analysis spend on cleaning and preparation?
- we don't teach this effectively (or research how to teach!)
- some principles help make cleaning easier and more reliable
 - 1 Each variable forms a column
 - 2 Each observation forms a row
 - 3 Each type of observational unit forms a table

“Tidy Data” Wickham *JSS*, 2014, [https:](https://www.jstatsoft.org/article/view/v059i10/v59i10.pdf)

[//www.jstatsoft.org/article/view/v059i10/v59i10.pdf](https://www.jstatsoft.org/article/view/v059i10/v59i10.pdf)

Key idioms for dealing with big(ger) data

`select`: subset variables

`filter`: subset rows

`mutate`: add new columns

`summarize`: reduce to a single row

`group-by`: aggregate

`join`: merge tables

`gather/spread`: transpose (e.g., wide to tall)

Hadley Wickham, bit.ly/bigrdata4 and “Building precursors to data science” (CHANCE, 2015,

<https://nhorton.people.amherst.edu/precursors>)

Data wrangling idioms

- tidy data: each variable in its own column and observation in its row
- focus on Chicago area flights (`dplyr::filter()`)
- focus on desired variables (`dplyr::select()`)
- correct odd variable names (`dplyr::rename()`)
- efficiency lazy evaluation: only evaluated when and if used
- key resource: RStudio cheatsheets

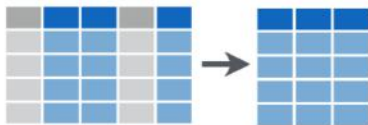
Subset Observations (Rows)



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

Extract rows that meet logical criteria.

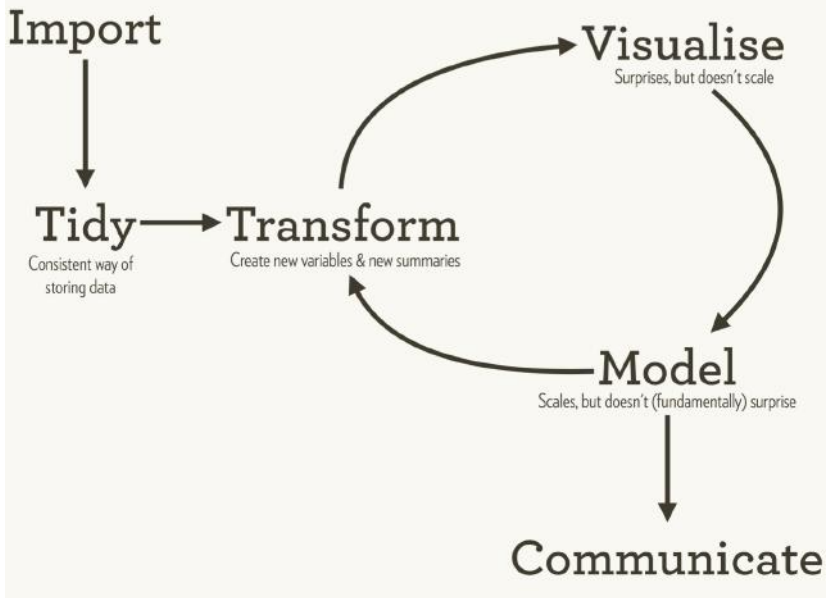
Subset Variables (Columns)



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

Select columns by name or helper function.

Statistical analysis cycle (Wickham)



Transform

```
> flights
  year month   day dep_time dep_delay arr_time arr_delay
<int> <int> <int>   <int>   <int>   <int>   <int>   <int>
1  1987    10     1     3       318     41     321
2  1987    10     1     3        -2     38     -2
3  1987    10     1     3         4    728     16
4  1987    10     1     4        -5    607     -2
5  1987    10     1     5         0    119      5
6  1987    10     1     5         0     52     17
7  1987    10     1     5        -5     41    -9
8  1987    10     1     6        -2    534     17
9  1987    10     1     6         26     34     19
10 1987    10     1     7        -8     39     -8
# ... with more rows, and 8 more variables: tailnum <chr>,
#   flight <int>, origin <chr>, dest <chr>, air_time <int>,
#   distance <int>, cancelled <int>, diverted <int>
```

Make New Variables



```
mutate(iris, sepal = Sepal.Length + Sepal.Width)
```

Transform

```
library(dplyr) # data wrangling for fun and profit
library(mosaic) # add support for modeling
flights %>%
  mutate(incoming = dest=="ORD" | dest=="MDW") %>%
  filter(year==2015, month==8, incoming) %>%
  favstats(arr_delay ~ dest, data=.)
```

```
##   dest min  Q1 median Q3  max mean  sd      n missing
## 1  MDW -43 -14     -7  5  604 2.15 34.2 7833      95
## 2  ORD -69 -16     -7  7 1134 5.00 46.3 27343     785
```

What's with the missing values?

```
flights %>%  
  filter(year==2015, month==8, is.na(arr_delay)) %>%  
  select(dest, origin, carrier, cancelled) %>%  
  head(3)
```

```
## # A tibble: 3 x 4  
##   dest  origin carrier cancelled  
##   <chr> <chr>  <chr>      <dbl>  
## 1 ALB    ORD     UA          1  
## 2 IAD    ORD     UA          1  
## 3 ORD    BWI     UA          1
```

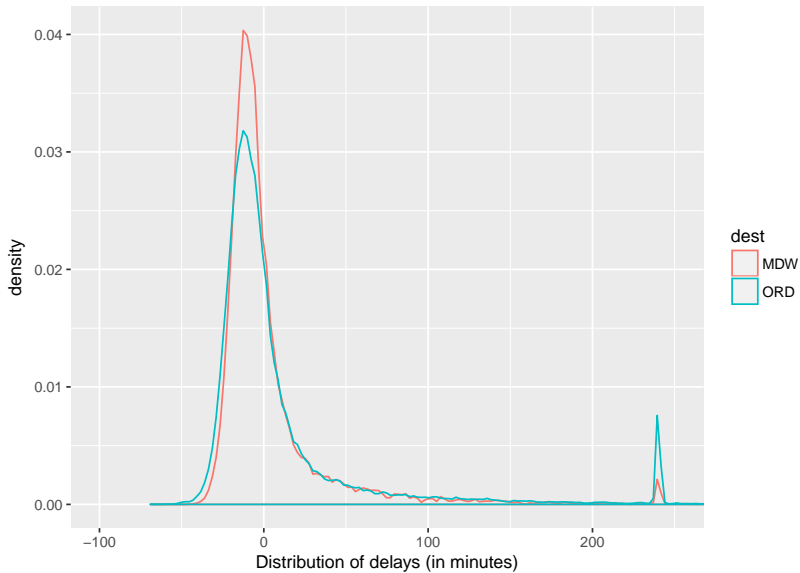
How to handle cancellations?

Transform

```
flights %>%  
  mutate(  
    incoming = dest=="ORD" | dest=="MDW",  
    true_delay = ifelse(cancelled, 240, arr_delay)) %>%  
  filter(year==2015, month==8, incoming) %>%  
  favstats(true_delay ~ dest, data=.)
```

##	dest	min	Q1	median	Q3	max	mean	sd	n	missing
## 1	MDW	-43	-14	-7	5	604	3.72	39.1	7885	43
## 2	ORD	-69	-16	-6	9	1134	10.42	57.8	27989	139

Visualize



Summarise Data



```
summarise(iris, avg = mean(Sepal.Length))
```

Group Data

dplyr::group_by(iris, Species)

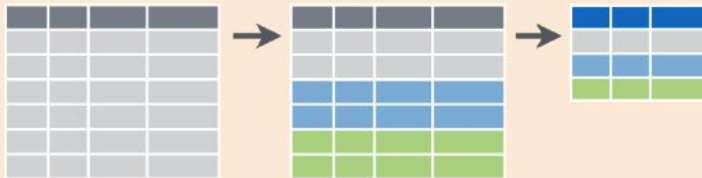
Group data into rows with the same value of Species.

dplyr::ungroup(iris)

Remove grouping information from data frame.

iris %>% group_by(Species) %>% summarise(...)

Compute separate summary row for each group.



Summarize and aggregate

```
flights %>%
  filter(year==2015, origin=="ORD" | origin=="MDW"
         | dest=="ORD" | dest=="MDW") %>%
  group_by(incoming) %>%
  summarize(
    count = n(),
    meandelay = mean(true_delay, na.rm=TRUE))

## # A tibble: 2 x 3
##   incoming count meandelay
##   <lgl>     <int>     <dbl>
## 1 FALSE     36077     9.63
## 2 TRUE      36056     8.95
```

Summarize and aggregate

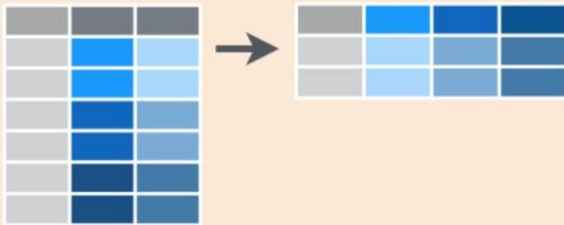
```
aug2015 <- flights %>%  
  filter(year==2015, origin=="ORD" | origin=="MDW"  
         | dest=="ORD" | dest=="MDW") %>%  
  group_by(airport, day) %>%  
  summarize(  
    count = n(), year = first(year),  
    month = first(month),  
    meandelay = mean(true_delay, na.rm=TRUE)) %>%  
  arrange(day) %>% data.frame()  
head(aug2015, 4)
```

##	airport	day	count	year	month	meandelay
## 1	MDW	1	478	2015	8	13.64
## 2	ORD	1	1607	2015	8	3.22
## 3	MDW	2	533	2015	8	15.68
## 4	ORD	2	1852	2015	8	46.29



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

Gather columns into rows.



tidyr::spread(pollution, size, amount)

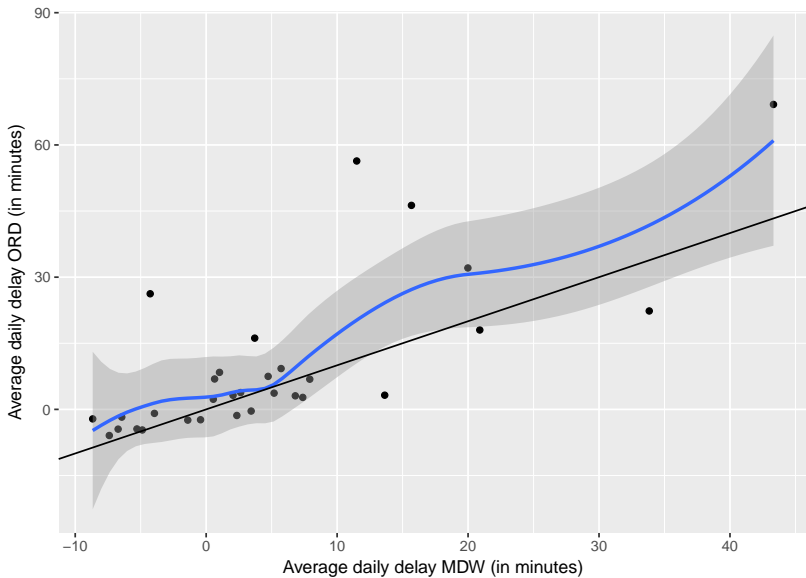
Spread rows into columns.

Reshape and transpose

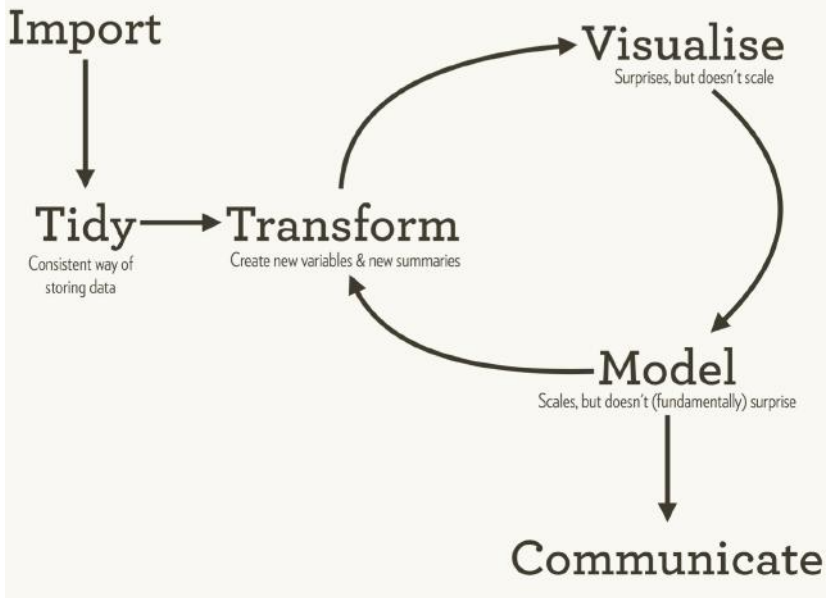
```
oneline <- aug2015 %>%  
  select(airport, year, month, day, meandelay) %>%  
  spread(airport, meandelay)  
head(oneline)
```

```
##   year month day   MDW   ORD  
## 1 2015     8   1 13.64  3.22  
## 2 2015     8   2 15.68 46.29  
## 3 2015     8   3  5.72  9.27  
## 4 2015     8   4  6.81  3.08  
## 5 2015     8   5  2.05  3.20  
## 6 2015     8   6  5.19  3.69
```

Visualize



Statistical analysis cycle (Wickham)

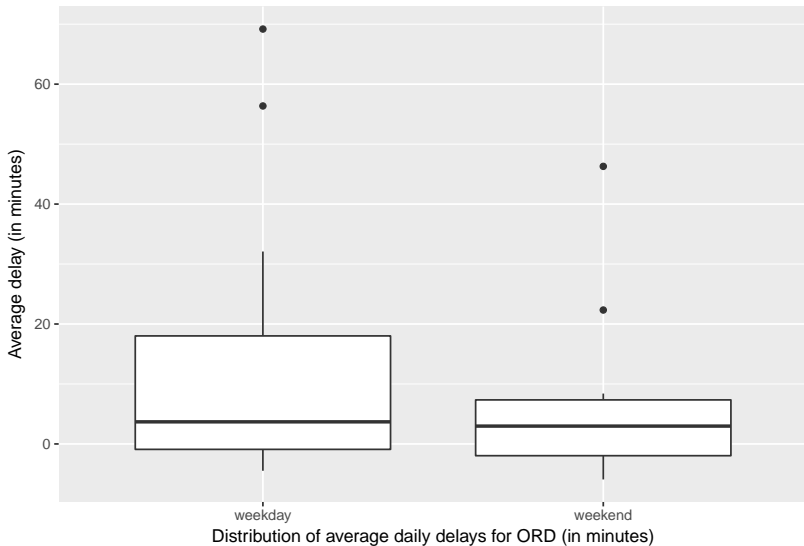


Airline Delays

```
library(lubridate)
oneline <- oneline %>%
  mutate(dow = wday(ymd(paste(year, month, day))),
         daytype = ifelse(dow %in% 2:5,
                          "weekday", "weekend"))
head(oneline, 4)
```

##	year	month	day	MDW	ORD	dow	daytype
## 1	2015	8	1	13.64	3.22	7	weekend
## 2	2015	8	2	15.68	46.29	1	weekend
## 3	2015	8	3	5.72	9.27	2	weekday
## 4	2015	8	4	6.81	3.08	3	weekday

Airline Delays



Airline Delays

```
oneline %>%  
  filter(ORD > 40)
```

```
##   year month day  MDW  ORD dow  daytype  
## 1 2015     8   2  15.7 46.3   1 weekend  
## 2 2015     8  10  43.3 69.2   2 weekday  
## 3 2015     8  17  11.5 56.4   2 weekday
```

Combine Data Sets

a

x1	x2
A	1
B	2
C	3

b

x1	x3
A	T
B	F
D	T

+ =

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

`dplyr::left_join(a, b, by = "x1")`
Join matching rows from b to a.

```
glimpse(airports)
```

```
## Observations: 1,458
```

```
## Variables: 9
```

```
## $ faa      <chr> "04G", "06A", "06C", "06N", "09J", "0A9"
```

```
## $ name     <chr> "Lansdowne Airport", "Moton Field Munic"
```

```
## $ lat      <dbl> 41.1, 32.5, 42.0, 41.4, 31.1, 36.4, 41.5
```

```
## $ lon      <dbl> -80.6, -85.7, -88.1, -74.4, -81.4, -82.2
```

```
## $ alt      <int> 1044, 264, 801, 523, 11, 1593, 730, 492
```

```
## $ tz       <dbl> -5, -6, -6, -5, -5, -5, -5, -5, -5, -8,
```

```
## $ dst      <chr> "A", "A", "A", "A", "A", "A", "A", "A",
```

```
## $ city     <chr> "Youngstown", "Tuskegee", "Schaumburg",
```

```
## $ country  <chr> "United States", "United States", "Unite
```

Join/merge

```
outgoing <- flights %>%  
  filter(incoming==FALSE) %>%  
  rename(faa=dest)  
merged <- left_join(outgoing, airports) %>%  
  select(faa, tz, true_delay, year, month, day,  
         carrier) %>%  
  arrange(true_delay)  
  
## Joining, by = "faa"
```

Join/merge

```
data.frame(merged) %>% head(2)
```

```
##   faa tz true_delay year month day carrier
## 1 PHX -7          -58 2015     8  30      AA
## 2 LAX -8          -55 2015     8  29      AA
```

```
favstats(true_delay ~ tz, data=merged)
```

```
##   tz min      Q1 median      Q3 max  mean  sd      n missing
## 1 -10 -17  -0.75    9.5 22.2 559 24.95 85.5    44
## 2  -9 -46 -12.00    5.0 28.0 152 10.85 33.4   109
## 3  -8 -55 -17.00   -5.0 13.0 600  6.75 43.2  4621
## 4  -7 -58 -15.00   -4.0 13.0 783  8.97 48.5  2480
## 5  -6 -51 -15.00   -6.0  9.0 609  8.81 49.3 11152
## 6  -5 -45 -13.00   -4.0 12.0 614 10.99 49.9 17473
```

Join/merge

```
merged %>%  
  filter(tz=="-10") %>%  
  head(5)
```

A tibble: 5 x 7

##	faa	tz	true_delay	year	month	day	carrier
##	<chr>	<dbl>	<dbl>	<int>	<int>	<int>	<chr>
## 1	HNL	-10	-17	2015	8	31	UA
## 2	HNL	-10	-16	2015	8	21	UA
## 3	HNL	-10	-13	2015	8	30	UA
## 4	HNL	-10	-12	2015	8	22	UA
## 5	OGG	-10	-8	2015	8	14	UA

- what if we wanted to analyze all of the flight data?
- how to access 180 million records in R?
- answer: use a database that talks to `dplyr` (e.g., MySQL, SQLite, PostgreSQL)
- see the ETL package to facilitate setup (<http://github.com/beanumber/airlines>)

Airline delays: access via a database

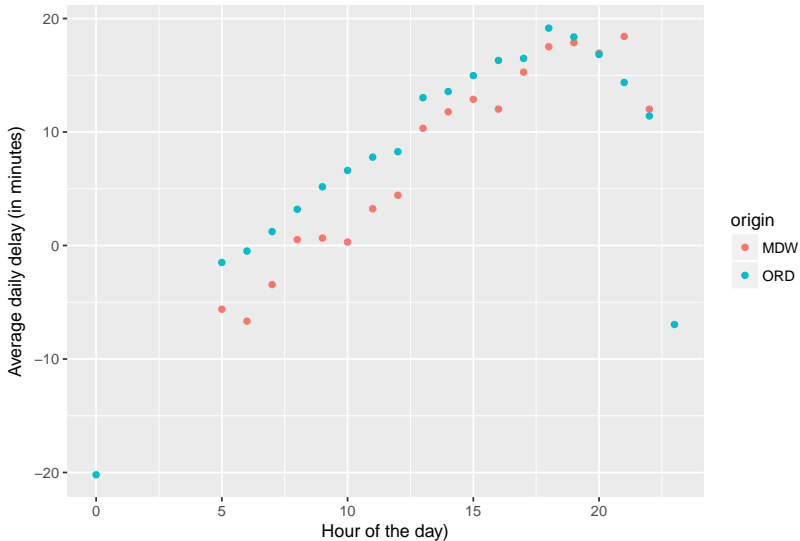
```
library(RMySQL)
db <- src_mysql(host="dbhost.mysite.com",
  user="r-user", password="XX",
  dbname = "airlines")
flights <- tbl(db, "flights")
airports <- tbl(db, "airports")
carriers <- tbl(db, "carriers")
planes <- tbl(db, "planes")
```

Airline Delays

```
sumhours <-  
flights %>%  
  filter(month==8, year > 2010,  
         origin=="MDW" | origin=="ORD") %>%  
  group_by(hour, origin) %>%  
  summarize(count=n(), avgdelay=mean(arr_delay)) %>%  
  collect() # forces computation of lazy tbls
```

Airline Delays

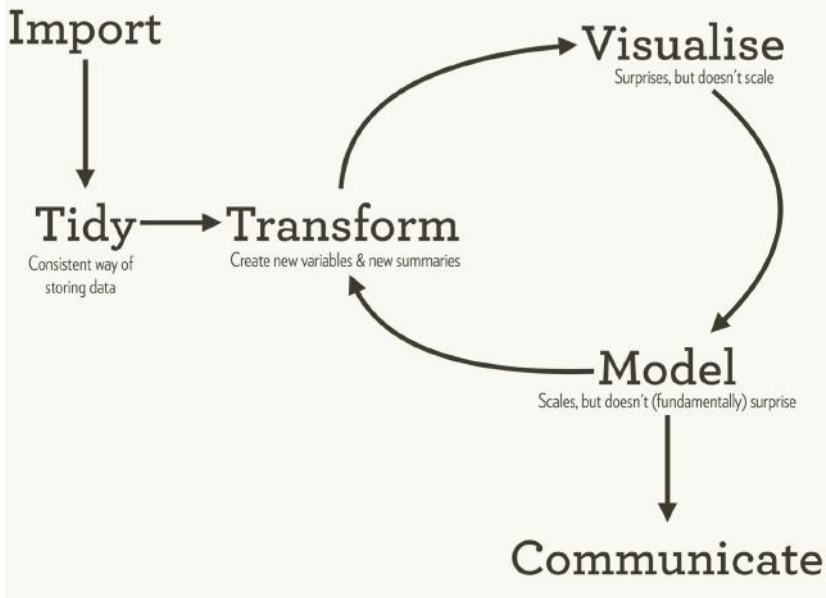
Delays leaving Chicago airports in August since 2011



Next steps

- restrict to only recent Thursdays (use lubridate package)
- account for airline and destination in addition to time of day
- incorporate external information (weather?)
- identify padded schedules
- predictors of cancellations?

Statistical analysis cycle (Wickham)



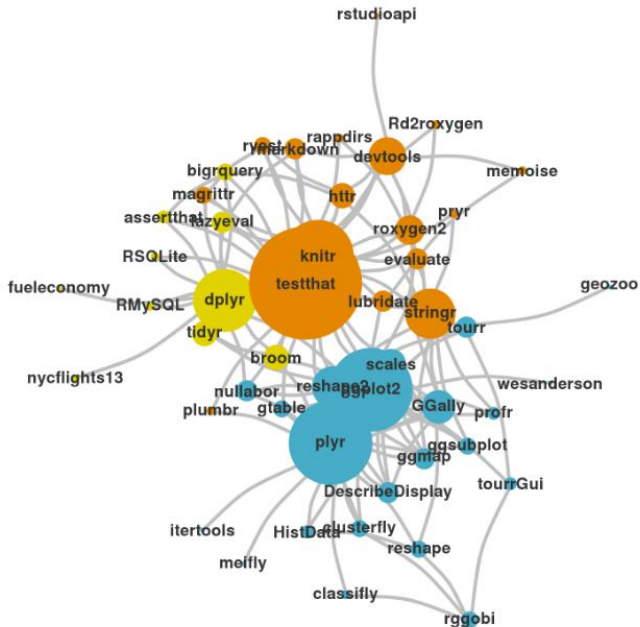
Modeling with broom (David Robinson et al)

- after fitting a regression model then tidy it
- `tidy(mod)`: construct data frame with model's statistical findings
- `augment(mod)`: add columns (predictions, residuals, cluster assignments)
- `glance(mod)`: construct one row summary of the model

Other noteworthy developments

- `rvest` package (easily harvest (scrape) web pages)
- `httr` package (work with URLs and HTTP)
- `stringr` simple wrappers for string operations
- `testthat` unit testing for fun and profit
- `ggplot2` implementation of the grammar of graphics

More on the tidyverse (display by Adolfo Álvarez)



Additional resources to help with wrangling in R

- DataCamp (free for certain educational use)
 - Importing and cleaning data
 - Data manipulation
 - Intro to R (free)
 - R for SAS Users
- learnr (more later)

Version control is the only reasonable way to keep track of changes in code, manuscripts, presentations, and data analysis projects.

Karl Broman,

http://kbroman.org/github_tutorial/pages/why.html

If you need to collaborate on data analysis or code development, then all involved should use Git.

Jenny Bryan, <http://happygitwithr.com>

Linkage to remainder of workshop

- R Markdown and reproducible analysis
- dplyr and friends
- Shiny and dynamic visualizations (briefly)
- lots of time for Question and Answer (see Google doc)

Thinking with Data Using R and RStudio: Powerful Idioms for Analysts

Nicholas J. Horton

Amherst College, Amherst, MA, USA

June 3, 2018

`nhorton@amherst.edu`

github repo for the workshop at [https://github.com/
Amherst-Statistics/TeachingStatsDataScience2018-06](https://github.com/Amherst-Statistics/TeachingStatsDataScience2018-06)