Modeling big data with R, sparklyr, and Apache Spark

1:30pm–5:00pm Tuesday, March 14, 2017
Data science & advanced analytics
Location: LL21 C/D
Level: Intermediate
Secondary topics: R

John Mount *(Win-Vector LLC)*
Steve Nolen *(RStudio)*
Edgar Ruiz   *(RStudio)*

url: https://github.com/WinVector/BigDataRStrata2017

RStudio Shiny Server Pro Accounts

- Distribute credentials and get everyone started with RStudio Server Pro.

- Server time generously donated by RStudio.

RStudio Server Pro

RStudio for the Enterprise

RStudio is the premier IDE for R. RStudio Server lets you access RStudio from anywhere using a web browser. RStudio Server Pro delivers the team productivity, security, centralized management, metrics, and commercial support that professional data science teams need to develop at scale.

From: https://www.rstudio.com/products/rstudio-server-pro/

Win-Vector LLC

What are we going to do?

- Become `dplyr` masters.

Win-Vector LLC

Work through markdowns in a bit

- Exercises/01-Universal-tools.Rmd

- Exercises/02-Big-Data.Rmd

- slides/lazyeval.Rmd

If you have trouble: ask your neighbors, flag me and the TAs, or peek in Exercises/solutions. These exercises are a memory aid, not a test.

Win-Vector LLC

Exercises/01-Universal-tools.Rmd Exercise 1

- Let's start with Exercises/01-Universal-tools.Rmd
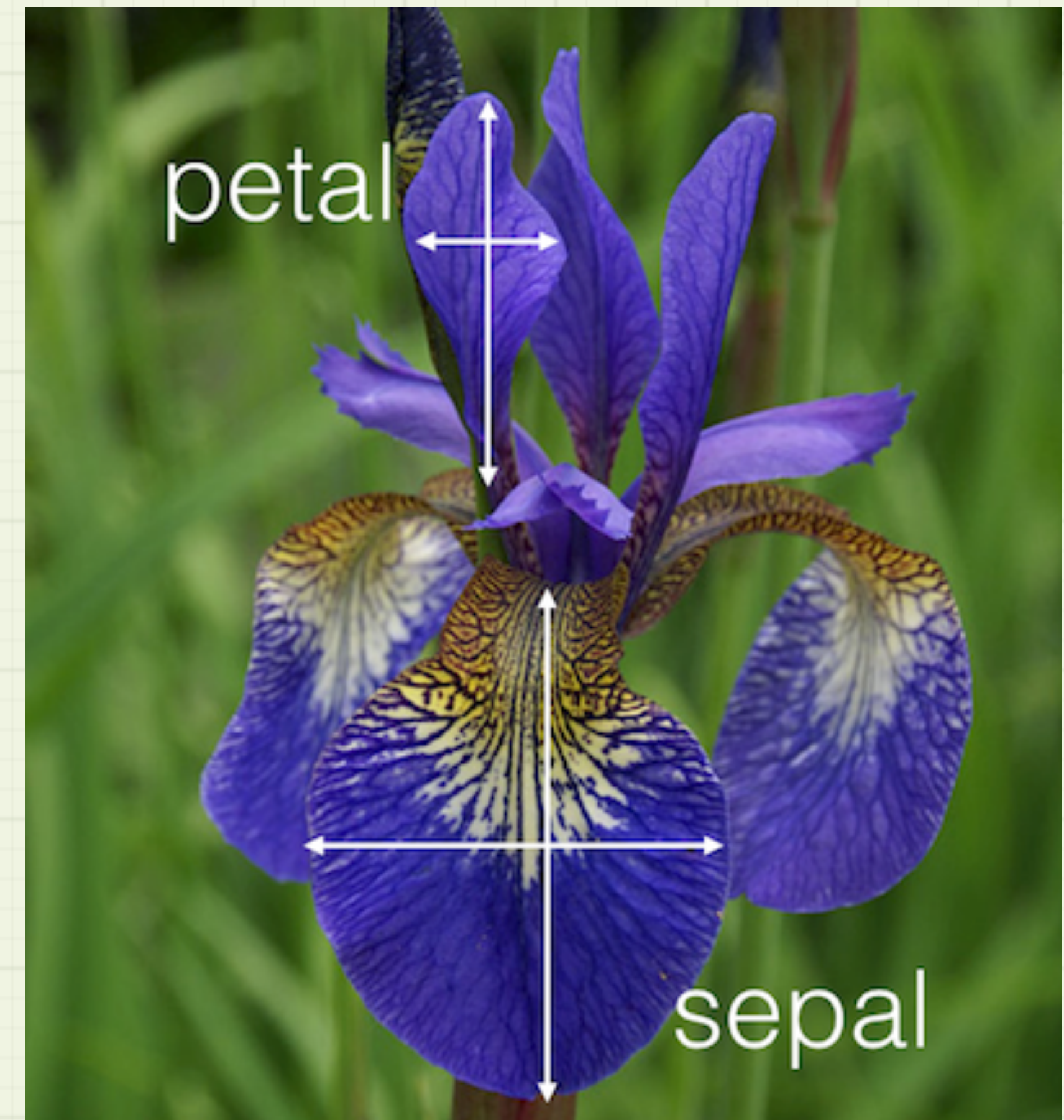
- We will just read through "Exercise 1" a bit.

Win-Vector LLC

Exercises/01-Universal-tools.Rmd Exercise 2

1. **Group** iris by Species
2. For each group, return:

   • avg_width = mean Petal.Width

   • avg_length = mean Petal.Length

3. For each row, calculate diff = avg_length - avg_width
4. Return the **row** whose diff == the max diff
5. Return the **columns** above

Hint 1: save each results in a different variable such as iris1, iris2, … and examine your results before moving to the next step.

Hint 2: Consider save the max diff *value* into a temp table before trying to find which row has that value.

05:00

Exercises/01-Universal-tools.Rmd
Exercise 2 solution

```r
library(dplyr)

iris1 <- group_by(iris, Species)
iris2 <- summarise(iris1,
  avg_width = mean(Petal.Width),
  avg_length = mean(Petal.Length))
iris3 <- mutate(iris2, diff = avg_length – avg_width)
iris4 <- filter(iris3, diff == max(diff))
select(iris4, Species, avg_width, avg_length)
```

Win-Vector LLC

Mini-topic
pipes and pipelines

Win-Vector LLC

%>%

Ceci n'est pas une pipe.

Win-Vector LLC

The pipe operator %>%

These all do the same thing
Try it!

```
filter(iris, Sepal.Length == max(Sepal.Length))

iris %>% filter(., Sepal.Length == max(Sepal.Length))

iris %>% filter(Sepal.Length == max(Sepal.Length))
```

%>%

```
iris          filter(_____, Sepal.Length == max(Sepal.Length))
```

Win-Vector LLC

Shortcut to type %>%

Cmd + Shift + M   (Mac)

Ctrl + Shift + M   (Windows)

Win-Vector LLC
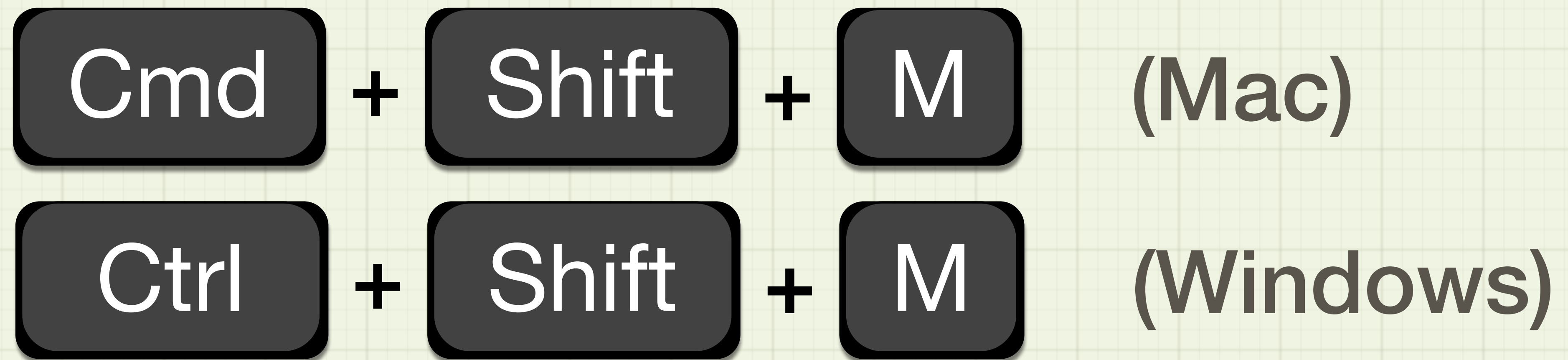
Exercises/01-Universal-tools.Rmd
Exercise 3

Use %>% to turn your code from that last exercise into a single long pipe.

03:00

Exercise 3 solution:
Take Exercise 2 solution

```r
library(dplyr)

iris1 <- group_by(iris, Species)
iris2 <- summarise(iris1,
  avg_width = mean(Petal.Width),
  avg_length = mean(Petal.Length))
iris3 <- mutate(iris2, diff = avg_length – avg_width)
iris4 <- filter(iris3, diff == max(diff))
select(iris4, Species, avg_width, avg_length)
```

Win-Vector LLC

Exercise 3 solution:
and *mechanically* translate it into a pipeline

```
iris %>%
  group_by(Species) %>%
  summarise(
    avg_width = mean(Petal.Width),
    avg_length = mean(Petal.Length)) %>%
  mutate(diff = avg_length – avg_width) %>%
  filter(diff == max(diff)) %>%
  select(Species, avg_width, avg_length)
```

Win-Vector LLC

pipeline debugging hint

- Break the pipeline early and look at intermediate results.

- Use "->" ("write arrow") to save result to "." (dot).

  - Since right arrow is forbidden by most style guides you can search for it to make sure you have not left debugging code in!

Win-Vector LLC

Debugging example

```r
iris %>%
  group_by(Species) %>%
  summarise(
    avg_width = mean(Petal.Width),
    avg_length = mean(Petal.Length)) ->.

print(.)

. %>% mutate(diff = avg_length - avg_width) %>%
filter(diff == max(diff)) %>%
select(Species, avg_width, avg_length)
```

Bizarro
pipe

Ceci n'est
pas %>%.

-> ;

Image:
KayPikeFashion
CC BY-NC-SA 3.0

Challenge Project: Parameterized Code

- If you find these exercises are too basic, please consider the following challenge project instead.

    - Research the dplyr "underbar operators"
        - summarize_
        - mutate_
        - filter_
        - select_

    - Re-write the above pipeline assuming the names of the length and width columns are not known to the analyst, but are instead in two variables WIDTH_COLUMN_NAME and LENGTH_COLUMN_NAME.
        - WIDTH_COLUMN_NAME = 'Petal.Width'
        - LENGTH_COLUMN_NAME = 'Petal.Length'

- Some references:

    - "Non-standard-evaluation and standard evaluation in `dplyr`"

        - http://rmhogervorst.nl/cleancode/blog/2016/06/13/NSE_standard_evaluation_dplyr.html

    - "Rewiring `replyr` with `dplyr`"

        - http://blog.eighty20.co.za//package%20exploration/2017/02/16/replyr-dplyr/

wrapr
sweet R code

Exercises/01-Universal-tools.Rmd
Exercise 4

- Try joins on the "band data."
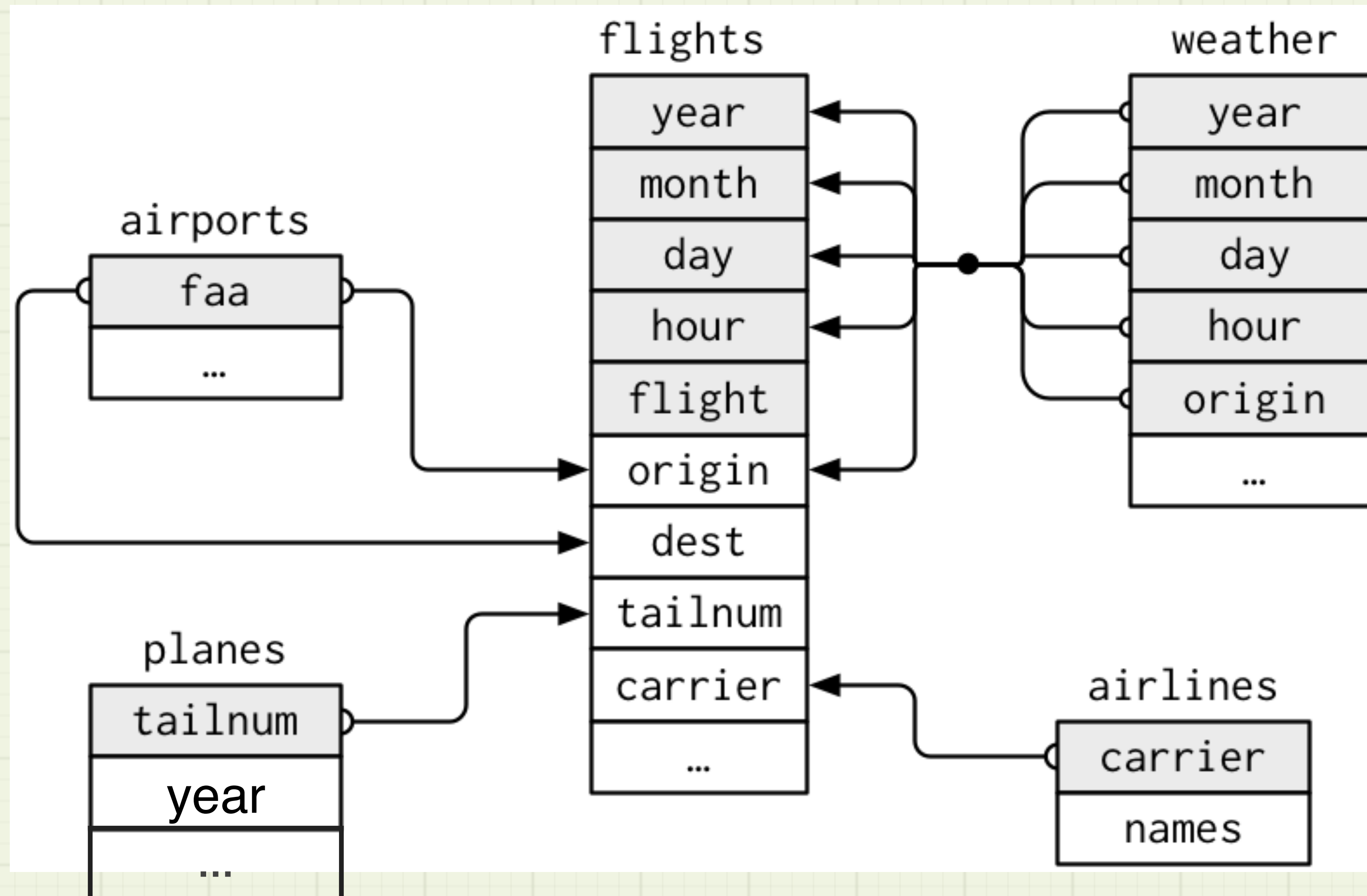
`03:00`

Exercises/01-Universal-tools.Rmd
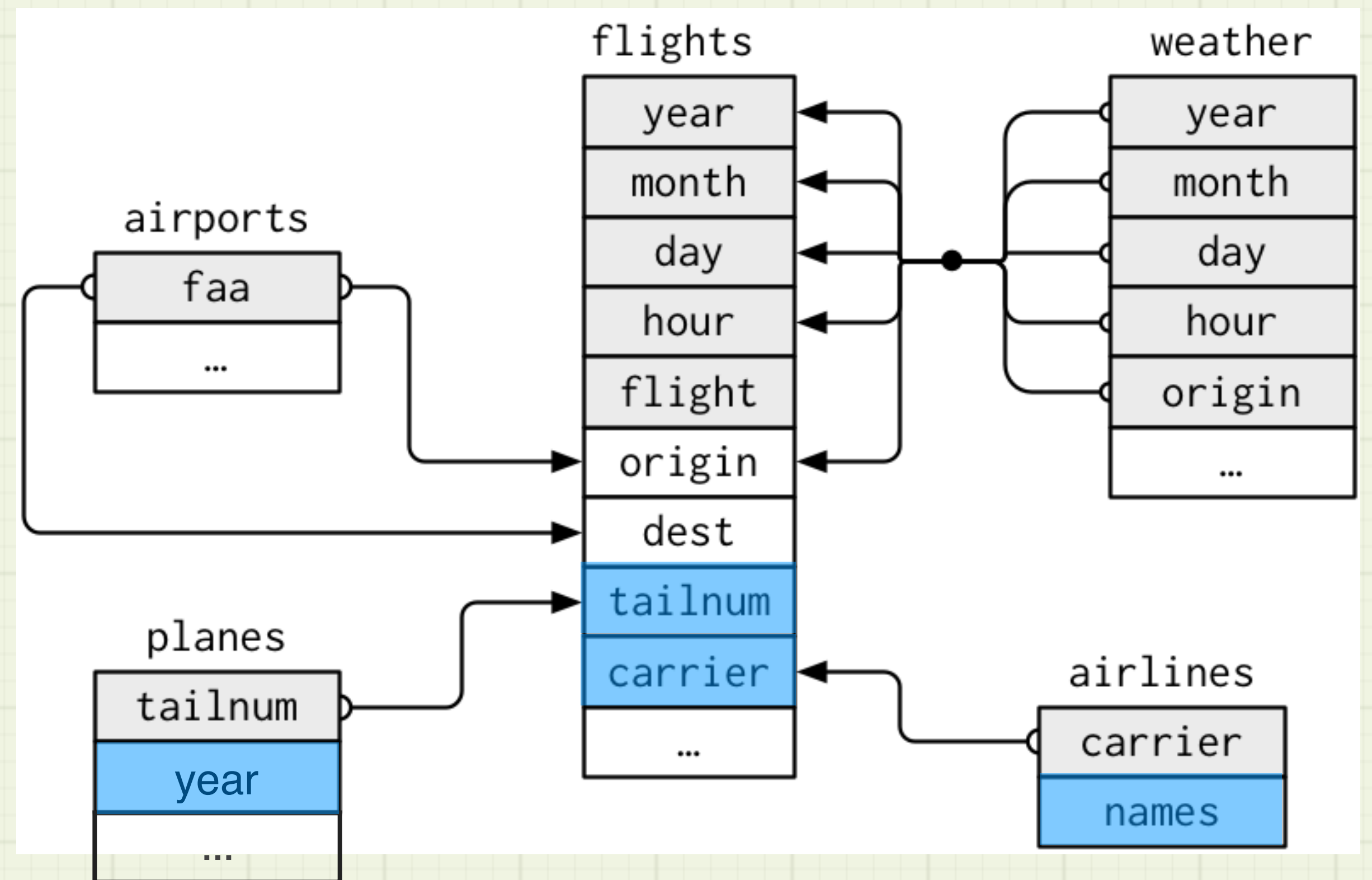# Exercise 5

nycflights13

Data on every flight that departed La Guardia, JFK, or Newark airports in 2013

nycflights13

nycflights13

On average, which airline
has the newest planes
(assigned to the NYC area)?

Exercise 5

Determine which airline has the newest planes. Please start with the code in the notebook.

```
flights %>%
  distinct(carrier, tailnum) %>%
  _____ %>%
  _____ %>%
  _____ ...
```

10:00

Exercises/01-Universal-tools.Rmd
Exercise 5 answer

| name	avg	n	nas
Hawaiian Airlines Inc.	2011.77	14	1
Virgin America	2008.71	53	1
Frontier Airlines Inc.	2008.00	26	3
Alaska Airlines Inc.	2007.84	84	1
JetBlue Airways	2006.50	193	6
SkyWest Airlines Inc.	2005.86	28	0
Endeavor Air Inc.	2004.71	204	2
Mesa Airlines Inc.	2003.56	58	1
ExpressJet Airlines Inc.	2002.44	316	8
AirTran Airways Corporation	2002.21	129	17

Win-Vector LLC

Exercises/01-Universal-tools.Rmd
Exercise 5 solution

```r
flights %>%
  # selects distinct combinations of carrier and tailnum
  distinct(carrier, tailnum) %>%
  # join to planes to get year manufactured
  # (which column should you join on?)
  left_join(planes, by = "tailnum") %>%
  # group by carrier (e.g. the airline)
  group_by(carrier) %>%
  # calculate by carrier:
  #    1. avg - the mean year (with na.rm = TRUE)
  #    2. n - the total number of planes
  #    3. nas - the number of planes with unknown year (year == NA)
  summarise(avg = mean(year, na.rm = TRUE),
            n = n(), nas = sum(is.na(year))) %>%
  # join to airlines to get full airline name
  # (which column should you join on?)
  left_join(airlines, by = "carrier") %>%
  # select just the name, avg, n, and nas variables in that order
  select(name, avg, n, nas) %>%
  # order the results by avg with the newest planes at the top
  arrange(desc(avg))
```

Win-Vector LLC

Mini topic databases

User Browser   **Server**   **Database**
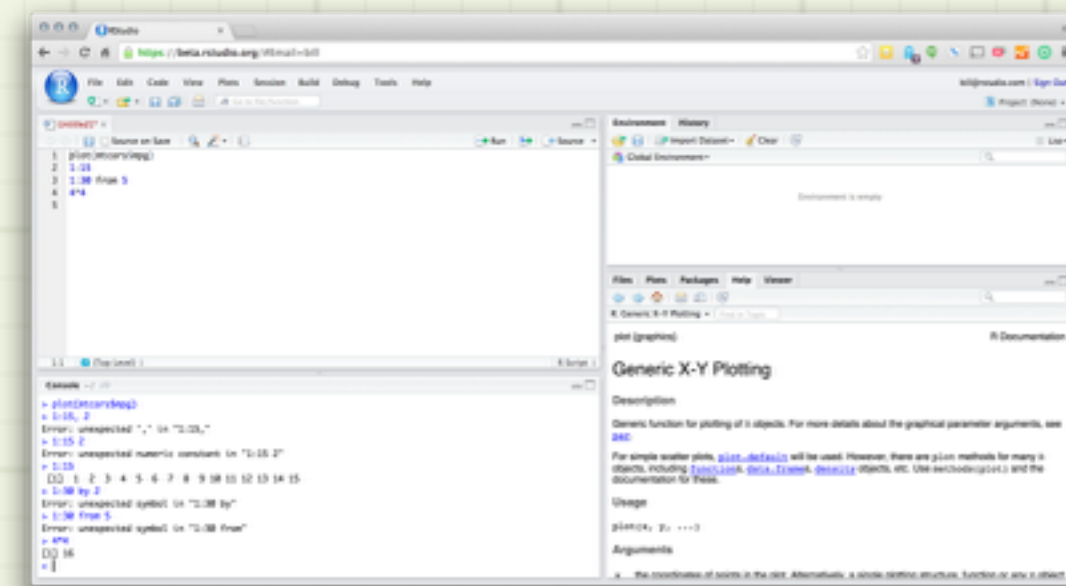
RStudio Server
Pro

Redshift

? Airlines

Win-Vector LLC

dplyr driver functions

Package	DBMS
src_sqlite()	SQLite
src_mysql()	MySQL, MariaDB
src_postgres()	PostgreSQL
library(bigrquery) src_bigquery()	Google BigQuery

https://cran.r-project.org/web/packages/dplyr/vignettes/databases.html

Win-Vector LLC

`dplyr` adapts to databases

- Many common base-R task must be translated `dplyr` to work on databases.

- Most `dplyr` verbs work with many data sources.

Base R	dplyr
x$col	select(x, col)

dplyr database workflow

1. **Create a connection**

   con <- dplyr::src_sqlite(":memory:", create = TRUE)

2. **Create a reference**

   tab <- copy_to(con, data, 'tablename')

3. **Manipulate the reference**

   query <- tab %>% filter(x > 1) %>% select(x, y, z)

4. **Collect the results**

   results <- collect(query)

5. **Close the connection**

   rm(con); gc()

Win-Vector LLC

1. **Create a connection**

```
con <- dplyr::src_sqlite(":memory:", create = TRUE)
```

Save
to use

src_ driver
function

driver
specific args

Lists tables
in DB

```
src_tbls(con)
## "iris"   "iris2"   "iris3"
```

Win-Vector LLC

2. **Create a table reference**

connection
to DB

name of
table in DB

```
tab <- tbl(con, "table_name")
```

Use `tbl()` to create objects that refer to tables
in the database

Win-Vector LLC

3. **Manipulate the reference**

Treat the reference as if it were a table in R. dplyr will translate your code to SQL and execute it in the DBMS.*

```r
flights <- tbl(air, "flights")
flights %>%
  distinct(uniquecarrier, tailnum) %>%
  …
```

Win-Vector LLC

4. **Collect the results**

Use collect() to import the entire set of results into R.

```
q6 <- flights %>%
    filter(year > 2007, depdelay > 15) %>%
    filter(depdelay == 240) %>%
    collect()
```

Win-Vector LLC

5. **Close the connection**

```
rm(air)
gc()
```

dplyr automatically closes connections when you remove the connection object *and then run the garbage collector, gc().*

Exercises/02-Big-Data.Rmd

Determine which airline has the newest planes using `dplyr` to control data in our practice database.

```
flights %>%
    distinct(carrier, tailnum) %>%
    _____ %>%
    _____ %>%
    _____ …
```

Hint: **CHEAT!!!**

03:00

Mini-topic: lazy evaluation

"Just in Time" delivery of results.

slides/lazyeval.Rmd

Lazy Execution 1

```
q1 <- filter(flights, year < 2007)
q2 <- filter(q1, depdelay > 15)
q3 <- filter(q2, depdelay < 240)
q4 <- select(q3, arrdelay, depdelay, year)
q4
```

Win-Vector LLC

Lazy Execution 1

```
q1 <- filter(flights, year < 2007)
q2 <- filter(q1, depdelay > 15)
q3 <- filter(q2, depdelay < 240)
q4 <- select(q3, arrdelay, depdelay, year)
q4
```

dplyr will not retrieve data until last possible moment. It combines all necessary work into a single optimized query.

Win-Vector LLC

```
show_query(q4)
##  <SQL>
##  SELECT "arrdelay" AS "arrdelay",
       "depdelay" AS "depdelay",
       "year" AS "year"
##  FROM "flights"
##  WHERE "year" > 2007.0
       AND "depdelay" > 15.0
       AND "depdelay" < 240.0
```

Win-Vector LLC

collapse()

Forces execution in DBMS

```
q5 <- flights %>%
    mutate(adjdelay = depdelay - 15) %>%
    collapse() %>%
    filter(adjdelay > 0)
```

collapse() turns the preceding queries into a table expression

remaining queries are run against the table described in the collapsed expression

Next:
Spark and sparklyr

Win-Vector LLC