Hands-on data exploration using R

Sebastian Sauer

last update: 2018-11-12

Setup

Overview

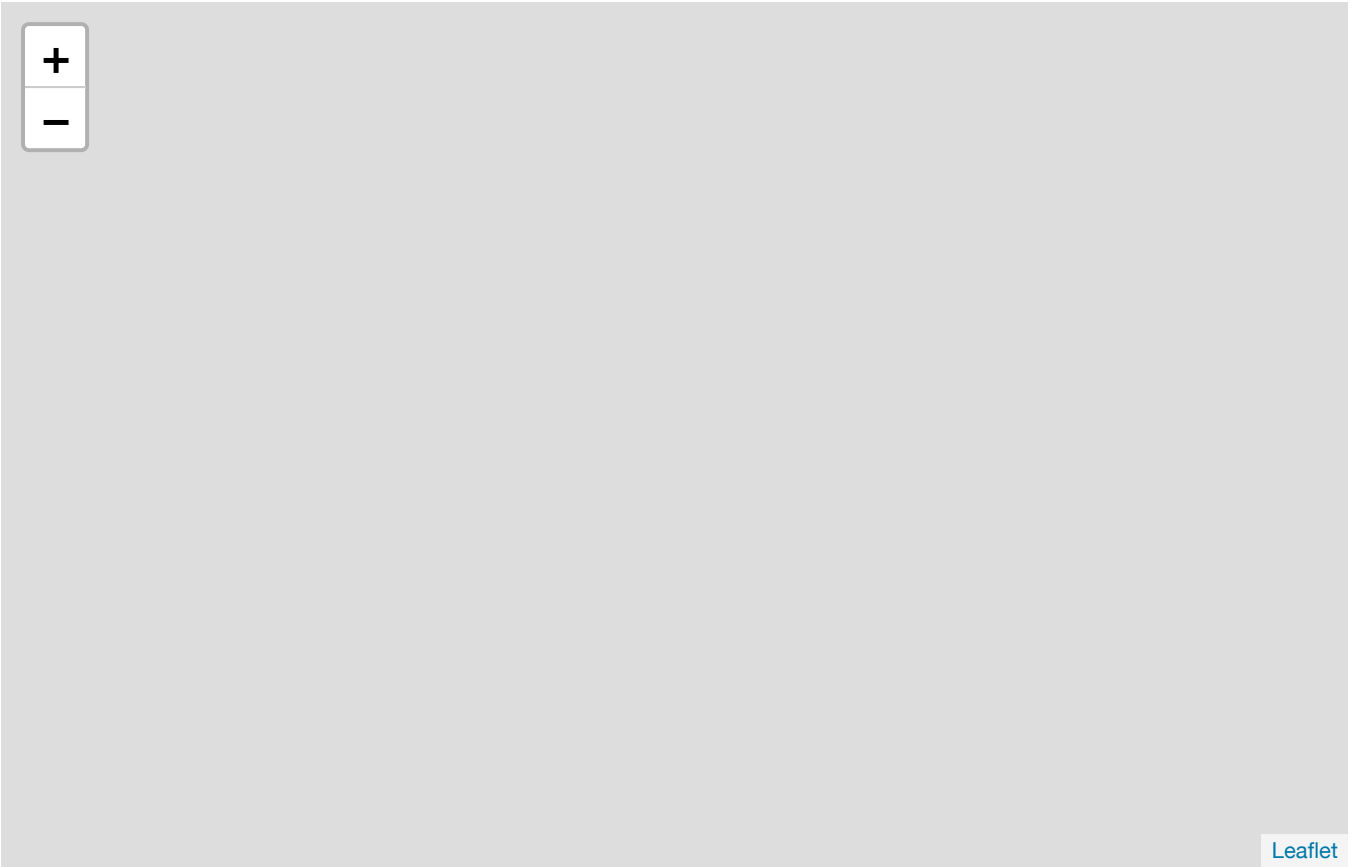
- Setup
- Tidyverse 101
- Data diagrams 101
- Case study

whoami

```
system("whoami")
```

- R enthusiast
- Data analyst/scientist
- Professor at FOM Hochschule

The lights are on



Upfront preparation

Please install the following software upfront:

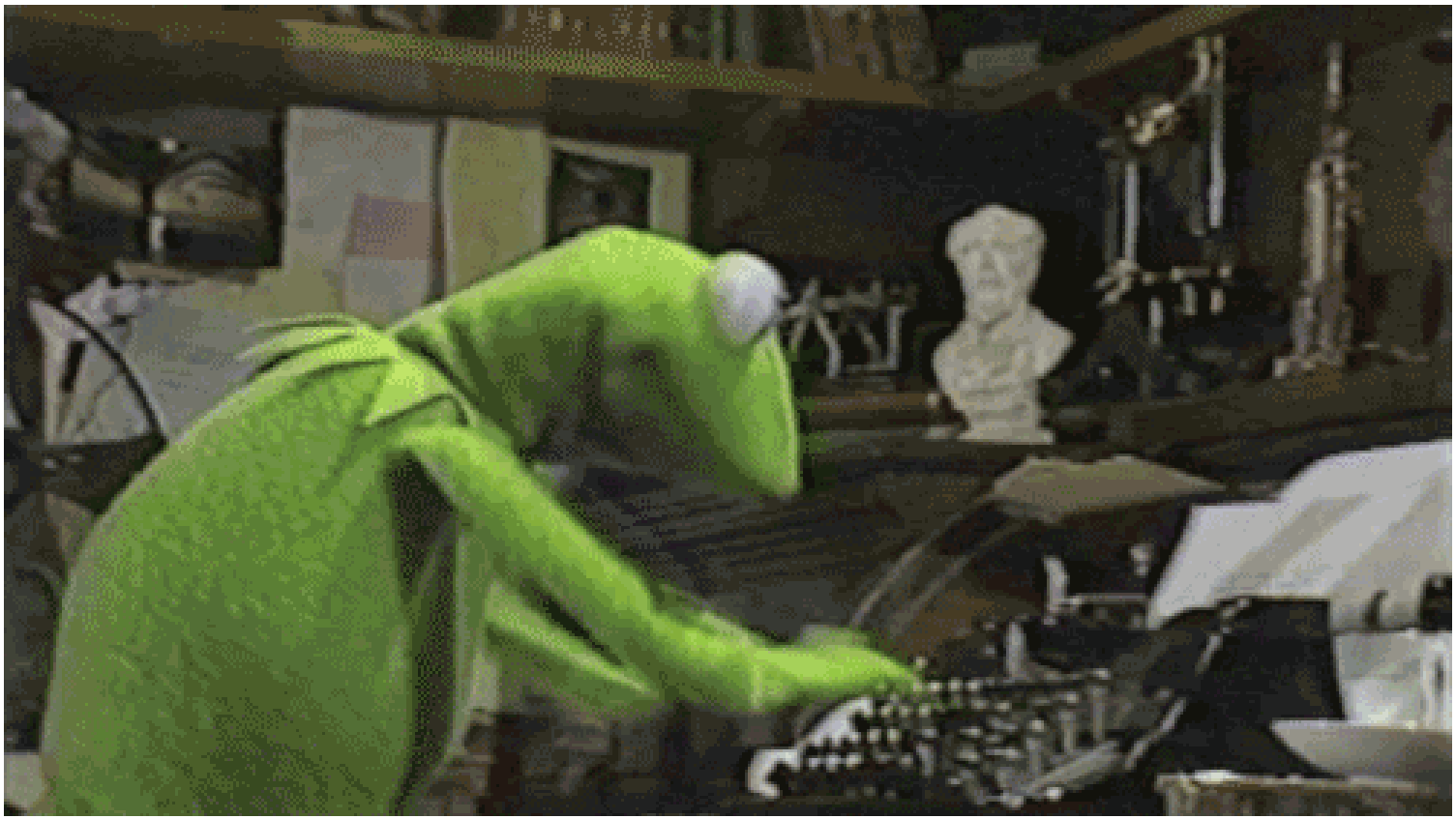
- R
- RStudio Desktop

Starting RStudio will start R automatically.

Please also make sure:

- Your OS is up to date
- You have internet access during the course
- You reach the next power socket (maybe better bring a power cable)

You, after this workshop



Well, kinda off...

Learning goals

- Understanding basic tidyverse goals
- Applying tidyverse tools
- Visualizing data
- Basic modeling

We'll use the following R packages

```
pckgs <- c("nycflights13", "mosaic",
           "broom", "corrr", "lubridate", "viridis",
           "GGally", "ggmap", "pacman", "sjmisc",
           "leaflet", "knitr",  "tidyverse")
```

Please install them prior to the workshop from within R:

Install each missing package like this:

```
install.packages("nycflights13")
```

Load each package after each start of Rstudio:

```
library(pacman)
p_load(pckgs, character.only = TRUE)
```

Data we'll use: `mtcars`

- `mtcars` is a toy dataset built into R (no need for installing).
- Data come from 1974 motorsports magazine describing some automobiles.
- Columns: e.g., horsepower, weight, fuel consumption

Load the dataset:

```
data(mtcars)
```

Get help:

```
?mtcars
```

Data we'll use: `flights`

- `flights` is a dataset from R package `nycflights13` (package must be installed).
- Data come from flights leaving the NYC airports in 2013.
- Coumns: eg., delay, air time, carrier name

Load the dataset:

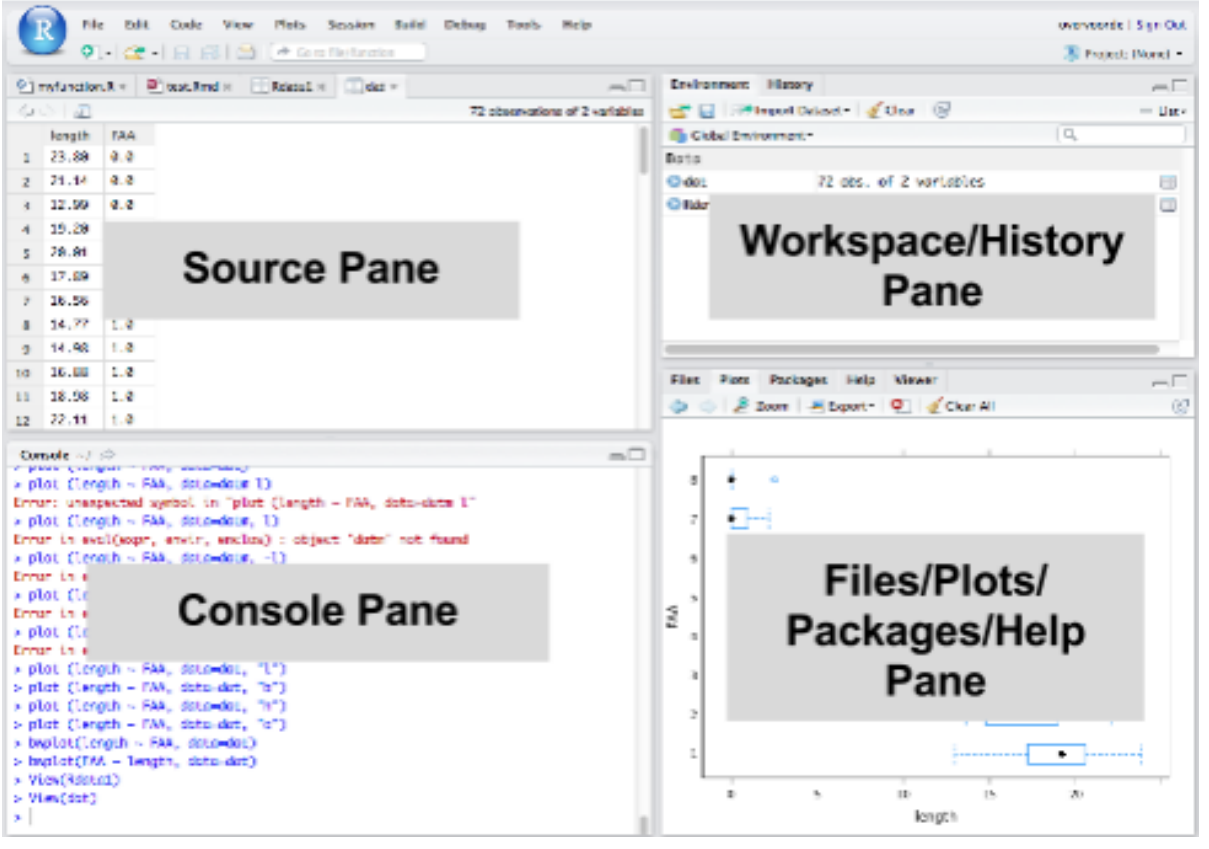
```
data(flights, package = "nycflights13")
```

Get help:

```
?flights
```

Load the data each time you open RStudio (during this workshop).

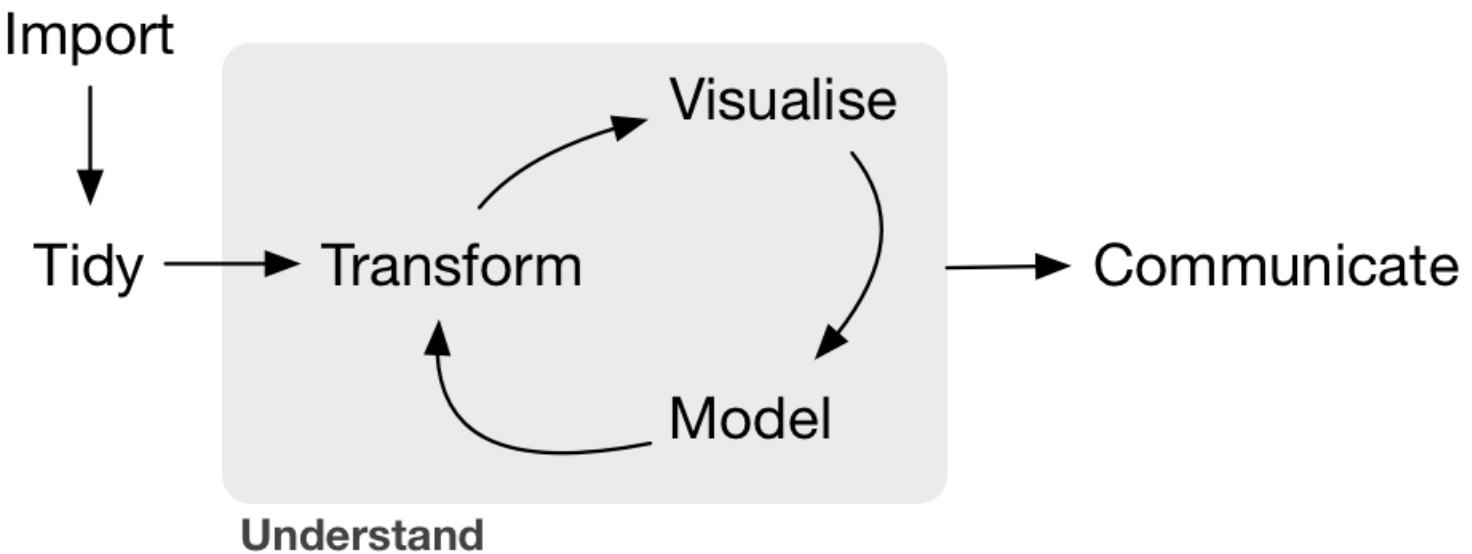
RStudio running



The tidyverse


broom
dplyr
reshape2
%>%
Ceci n'est pas un pipe.
test that
purrr
rmarkdown
RStudio
knitr
Shiny
packrat
lubridate
ggplot2
haven
stringr
tidyr
readr

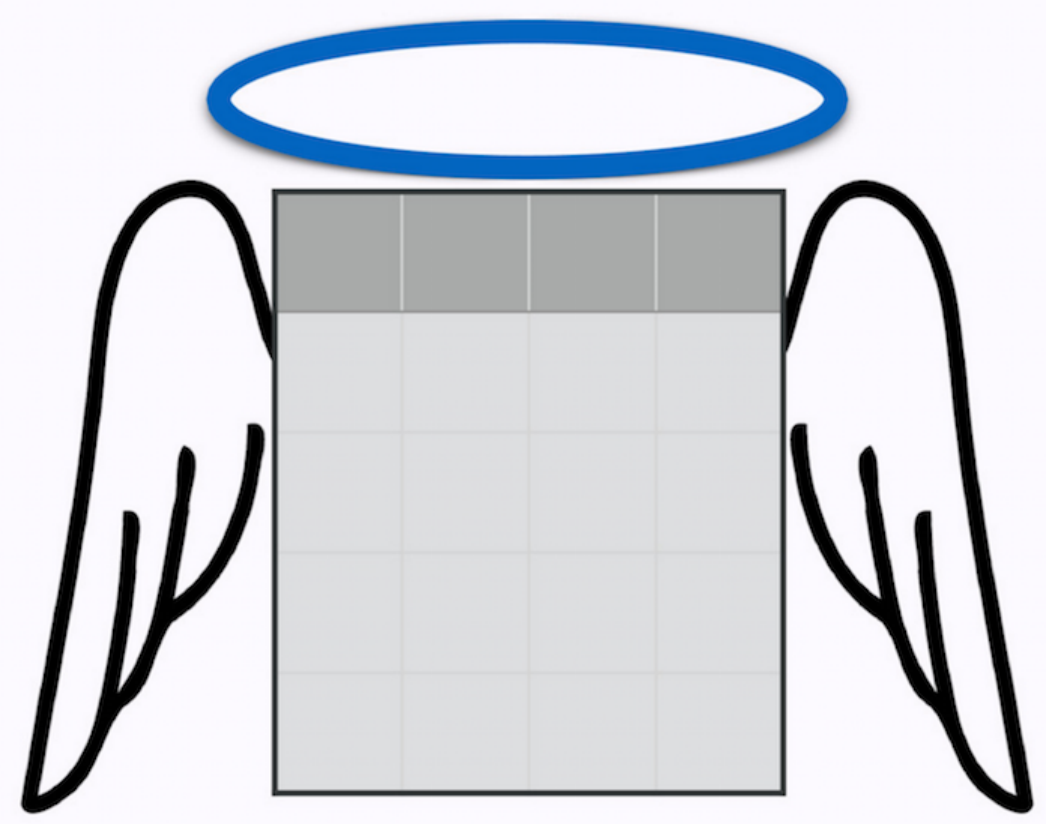
The data analysis (science) pipeline



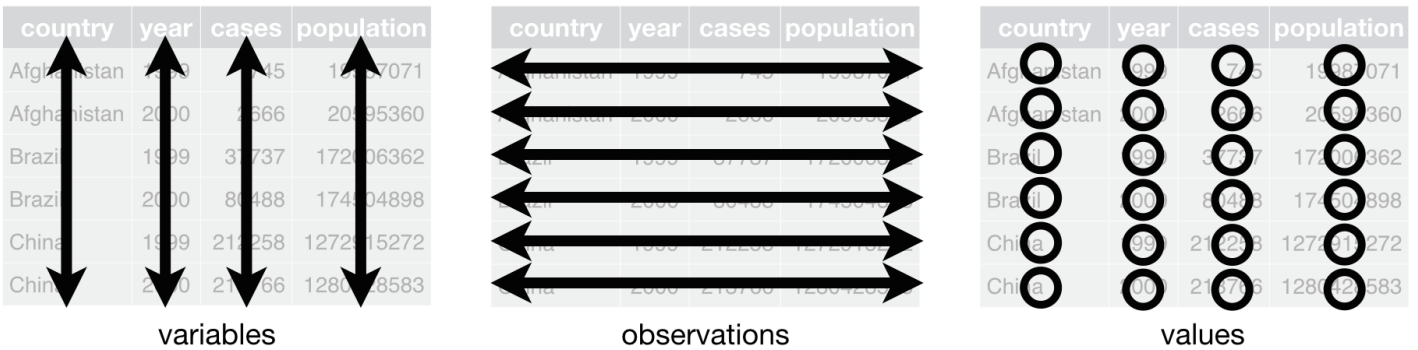
Get the power of the ~~uni~~ tidyverse

But I love the old way ...

Nice data



Tidy data



More Details

Dataset `mtcars`

```
glimpse(mtcars)
#> Observations: 32
#> Variables: 11
#> $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8,
#> $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4,
#> $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.
#> $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 1
#> $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92,
#> $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.19
#> $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.0
#> $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,
#> $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
#> $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4,
#> $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1,
```

Data wrangling

Two tidyverse principles

Knock-down principle

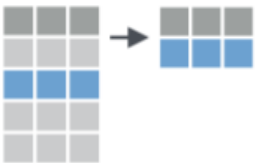
Pipe princriple

Atoms of the knock-down principle

- `filter()`
- `select()`
- `mutate()`
- `group_by()`
- …

Filtering rows with `filter()`

Extract rows that meet logical criteria.



Filter table `mtcars` such that only rows remain where `cols` equal `6`

```
filter(mtcars, cyl == 6)
#>    mpg cyl  disp  hp drat
#> 1 21.0   6 160.0 110 3.90 2.6
#> 2 21.0   6 160.0 110 3.90 2.8
#> 3 21.4   6 258.0 110 3.08 3.2
#> 4 18.1   6 225.0 105 2.76 3.4
#> 5 19.2   6 167.6 123 3.92 3.4
#> 6 17.8   6 167.6 123 3.92 3.4
#> 7 19.7   6 145.0 175 3.62 2.7
```

`filter()` - exercises

- Filter the automatic cars.
- Filter the automatic cars with more than 4 cylinders.
- Filter cars with either low consumption or the the super. thirsty ones

`filter()` - solutions to exercises

```
data(mtcars)  # only if dataset is not yet loaded
filter(mtcars, am == 1)
filter(mtcars, cyl > 4)
filter(mtcars, mpg > 30 | mpg < 12)
```

Select columns with `select()`

Extract columns by name.

Select the columns `cyl` and `hp`. Discard the rest.

```
select(mtcars, cyl, hp)
```

```
#>                   cyl  hp
#> Mazda RX4           6 110
#> Mazda RX4 Wag       6 110
#> Datsun 710          4  93
#> Hornet 4 Drive      6 110
#> Hornet Sportabout   8 175
#> Valiant             6 105
```

`select()` - exercises

- Select the first three columns.
- Select the first and third column.
- Select all columns containing the letter "c".

`select()` - solutions to exercises

```
select(mtcars, 1:3)
select(mtcars, 1, disp)
select(mtcars, contains("c"))  # regex supported
```

Add or change a column with `mutate`

Apply vectorized functions to columns to create new columns.

Define weight in kg for each car.

```
mtcars <- mutate(mtcars,
                 weight_kg = wt
head(select(mtcars, wt, weight_k
#>      wt weight_kg
#> 1 2.620      5.24
#> 2 2.875      5.75
#> 3 2.320      4.64
#> 4 3.215      6.43
#> 5 3.440      6.88
#> 6 3.460      6.92
```

`mutate()` - exercises

- Compute a variable for consumption (gallons per 100 miles).
- Compute two variables in one mutate-call.

`mutate()` - solutions to exercises

```
mtcars <- mutate(mtcars, consumption = (1/mpg) * 100 * 3.8 / 1.6)
mtcars <- mutate(mtcars,
                 consumption_g_per_m = (1/mpg),
                 consumption_l_per_100_k = consumption_g_per_m  * 3.8
```

Summarise a column with `summarise()`

Apply function to summarise column to single value.

Summarise the `values` to their mean.

```
summarise(mtcars,
          mean_hp = mean(hp))
#>    mean_hp
#> 1 146.6875
```

`summarise()` - exercises

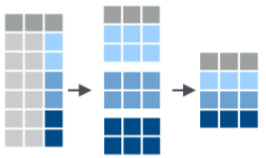
- Compute the median of consumption.
- Compute multiple statistics at once.

`summarise()` - solution to exercises

```
summarise(mtcars, median(consumption))
summarise(mtcars,
          consumption_md = median(consumption),
          consumption_avg = mean(consumption)
          )
```

Group with `group_by()`

Create "gruoped" copy of table. dplyr functions will manipulate each group separately and then combine the results.



Group cars by `am` (automatic vs. manual). Then summarise to mean in each group.

```
mtcars_grouped <- group_by(mtcar
summarise(mtcars_grouped, mean_h
#> # A tibble: 2 x 2
#>      am mean_hp
#>   <dbl>   <dbl>
#> 1     0    160.
#> 2     1    127.
```

`group_by()` - exercises

- Compute the median consumption, grouped by cylinder.
- Compute the median consumption, grouped by cylinder and `am`.

`group_by()` - exercises

```
#> # A tibble: 3 x 2
#>     cyl mean_hp
#>   <dbl>   <dbl>
#> 1     4    9.14
#> 2     6   12.1
#> 3     8   16.2
```

Enter the pipe


%>%
Ceci n'est pas un pipe.

Life without the pipe operator

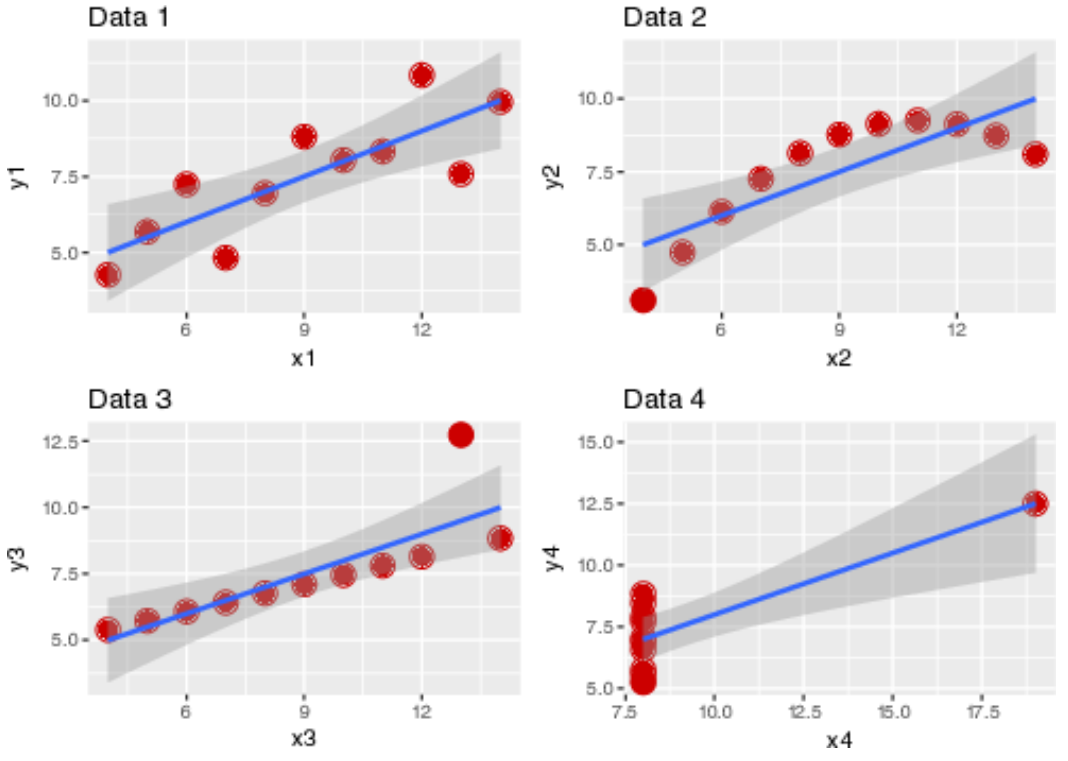
```
summarise(
  raise_to_power(
    compute_differences(data, mean),
    2
  ),
  mean
)
```

Life with the pipe operator

```
data %>%
  compute_differences(mean) %>%
  raise_to_power(2) %>%
  summarise(mean)
```

Data diagrams

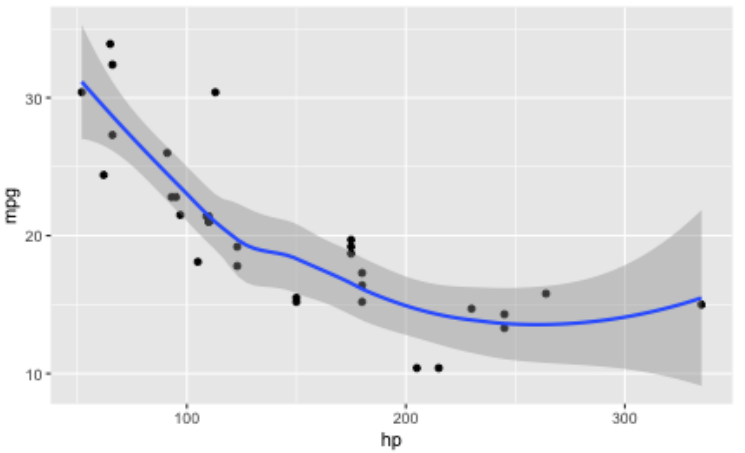
Why we need diagrams



Anatomy of a diagram

First plot with `ggplot`

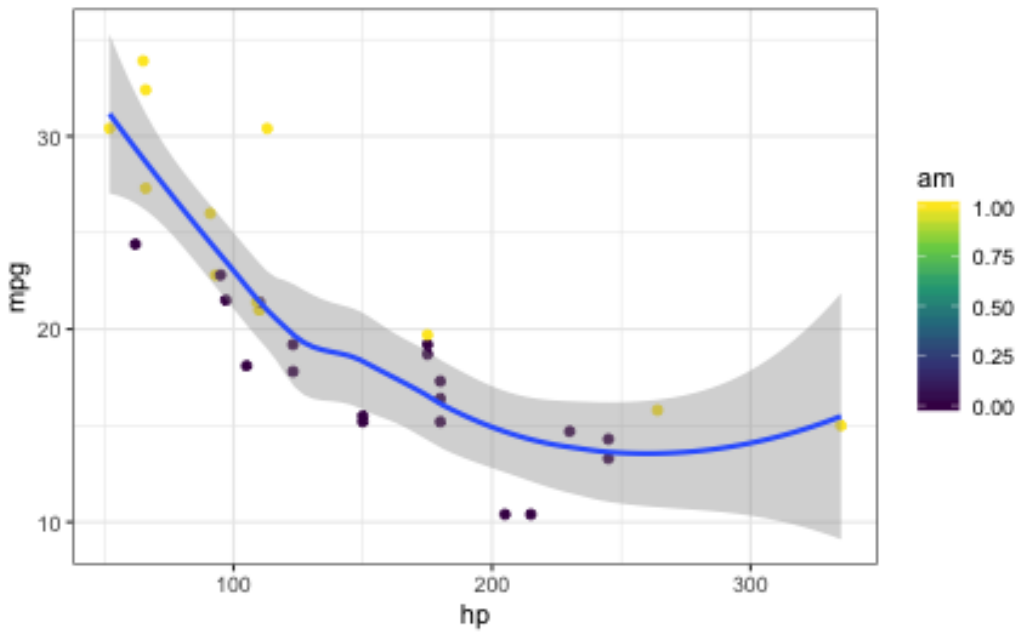
```
mtcars %>%
  ggplot() +  # initialize plot
  aes(x = hp, y = mpg) +  # define axes etc.
  geom_point() +  # graw points
  geom_smooth()  # draw smoothing line
```



Notice the + in contrast to the pipe %>%.

Groups and colors

```
mtcars %>%
  ggplot() +
  aes(x = hp, y = mpg, color = am) +
  geom_point() +
  geom_smooth() +
  scale_color_viridis_c() +
  theme_bw()
```

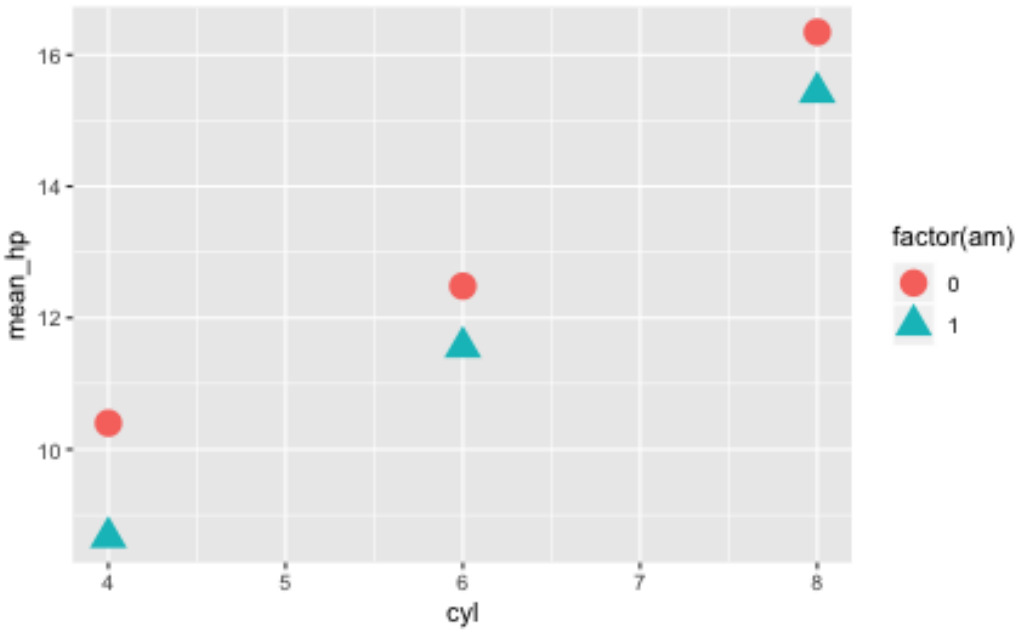


Diagrams - exercises

- Plot the mean and the median for each cylinder group (dataset `mtcars`).

Diagrams - solutions to exercises

```
mtcars_summarized %>%
  ggplot() +
  aes(x = cyl, y = mean_hp, color = factor(am),
      shape = factor(am)) +
  geom_point(size = 5)
```



Case study Why are flights delayed?

Know thy data

Don't forget to load it from the package via:

```
data(flights)
```

A look to the help page:

```
?flights
```

Glimpse data

```
glimpse(flights)
#> Observations: 336,776
#> Variables: 19
#> $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2
#> $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
#> $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
#> $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557
#> $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600
#> $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2
#> $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 83
#> $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 84
#> $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -
#> $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "
#> $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708
#> $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N6
#> $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "
#> $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "
#> $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140,
#> $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229
#> $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6
#> $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0,
#> $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00,
```

Data sanity - quantitative variables

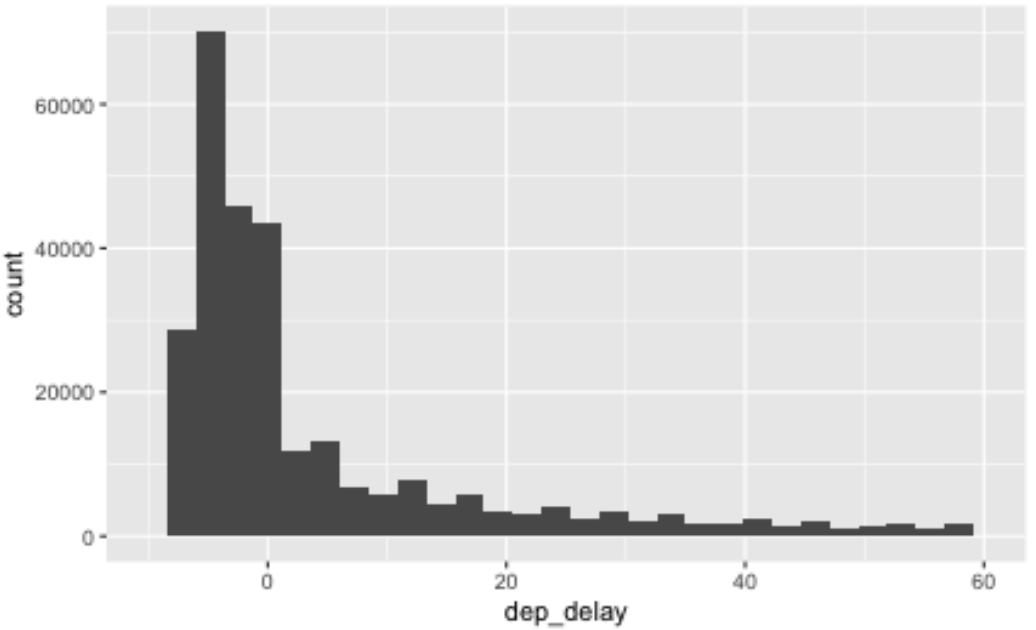
```
flights %>% descr()
#>
#> ## Basic descriptive statistics
#>
#>              var    type          label      n NA.prc    mean
#>             year integer           year 336776   0.00 2013.00
#>            month integer          month 336776   0.00    6.55
#>    md trimmed           range  skew
#>  2013 2013.00   0 (2013-2013)   NaN
#>     7    6.56       11 (1-12) -0.01
#>  [ reached getOption("max.print") -- omitted 12 rows ]
```

Data sanity - qualitative variables

```
flights %>%
  select_if(is.character) %>%
  inspect()
#>
#> categorical variables:
#>      name     class levels      n missing
#> 1 carrier character     16 336776       0
#> 2 tailnum character   4043 334264    2512
#> 3  origin character      3 336776       0
#> 4    dest character    105 336776       0
#>                                   distribution
#> 1 UA (17.4%), B6 (16.2%), EV (16.1%) ...
#> 2 N725MQ (0.2%), N722MQ (0.2%) ...
#> 3 EWR (35.9%), JFK (33%), LGA (31.1%)
#> 4 ORD (5.1%), ATL (5.1%), LAX (4.8%) ...
```

Distribution -- quantitative variables

```
flights %>%
  ggplot(aes(x = dep_delay)) +
  geom_histogram() +
  scale_x_continuous(limits = c(-10, 60))
```



Descriptive statistics for delay

```
flights %>%
  drop_na() %>%
  summarise(mean(dep_delay), median(dep_delay),
            sd(dep_delay), iqr(dep_delay))
#> # A tibble: 1 x 4
#>   `mean(dep_delay)` `median(dep_delay)` `sd(dep_delay)` `iqr(dep_d
#>               <dbl>               <dbl>           <dbl>
#> 1              12.6                  -2            40.1
```

Descriptive statistics by origin

```
flights %>%
  drop_na() %>%
  group_by(origin) %>%
  summarise(mean(dep_delay), median(dep_delay),
            sd(dep_delay), iqr(dep_delay))
#> # A tibble: 3 x 5
#>   origin `mean(dep_delay…` `median(dep_de…` `sd(dep_delay)` `iqr(dep
#>   <chr>              <dbl>            <dbl>            <dbl>
#> 1 EWR                15.0               -1             41.2
#> 2 JFK                12.0               -1             38.8
#> 3 LGA                10.3               -3             39.9
```

Start modeling

Delay as a function of origin?

```
delay = f(origin)
```

More Rish:

```
dep_delay ~ origing
```

Linear models

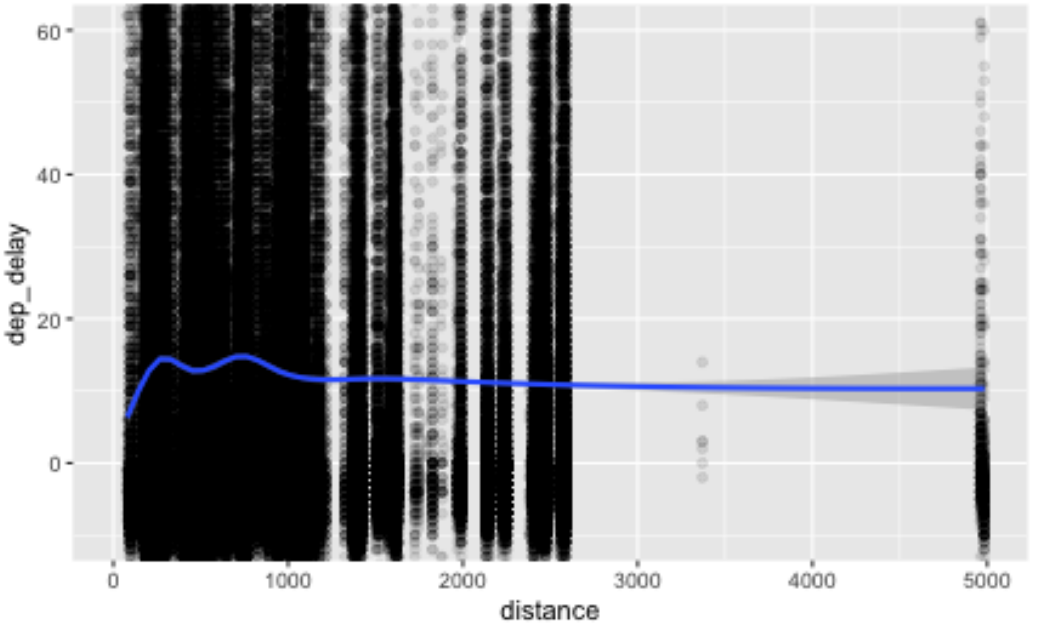
```
lm(dep_delay ~ origin, data = drop_na(flights)) %>% tidy()
#> # A tibble: 3 x 5
#>   term        estimate std.error statistic   p.value
#>   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept)    15.0      0.117     128.  0.
#> 2 originJFK      -2.99     0.168     -17.7 2.65e- 70
#> 3 originLGA      -4.72     0.172     -27.5 3.33e-166
```

Some as above, stated differently.

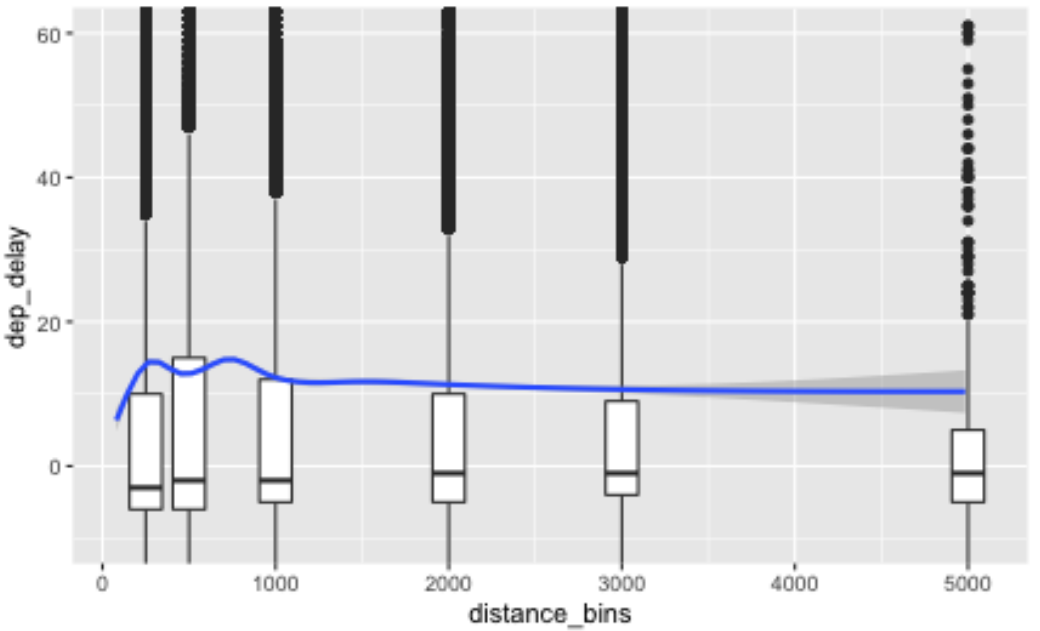
Does `distance` predicts `dep_delay`?

```
flights %>%
  ggplot(aes (x = distance, y = dep_delay)) +
  geom_point(alpha = .1) +
  geom_lm() +
  coord_cartesian(ylim(-10, 60))
```

Does `distance` predicts `dep_delay`?



Alternative visualiziation



Correlation of distance and delay

```
flights %>%
  select(distance, dep_delay, origin) %>%
  group_by(origin) %>%
  drop_na() %>%
  summarise(cor_delay_dist = cor(dep_delay, distance))
#> # A tibble: 3 x 2
#>   origin cor_delay_dist
#>   <chr>           <dbl>
#> 1 EWR           -0.0361
#> 2 JFK           -0.0398
#> 3 LGA            0.0114
```

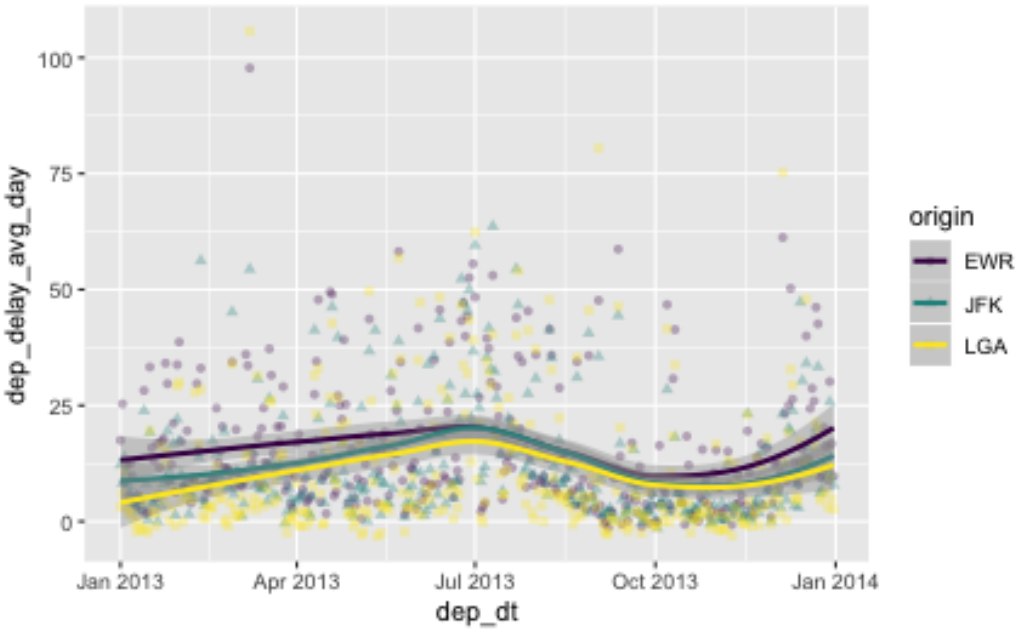
Delay as a function of distance

```
lm(dep_delay ~ I(distance/1000) + origin, data = flights) %>%
  tidy()
#> # A tibble: 4 x 5
#>   term              estimate std.error statistic   p.value
#>   <chr>                <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept)          16.8     0.157      107.  0.
#> 2 I(distance/1000)     -1.60    0.0988     -16.2 9.43e- 59
#> 3 originJFK            -2.66    0.170      -15.7 3.09e- 55
#> 4 originLGA            -5.21    0.174      -29.9 3.80e-196
```

Delay per month (code)

```
flights %>%
  group_by(origin, month, day) %>%
  summarise(dep_delay_avg_day = mean(dep_delay, na.rm = TRUE)) %>%
  ungroup %>%
  mutate(dep_dt = make_date(2013, month, day)) %>%
  ggplot(aes(x = dep_dt, y = dep_delay_avg_day, shape = origin, color
  geom_point(alpha = .3) +
  geom_smooth() +
  scale_color_viridis_d()
```

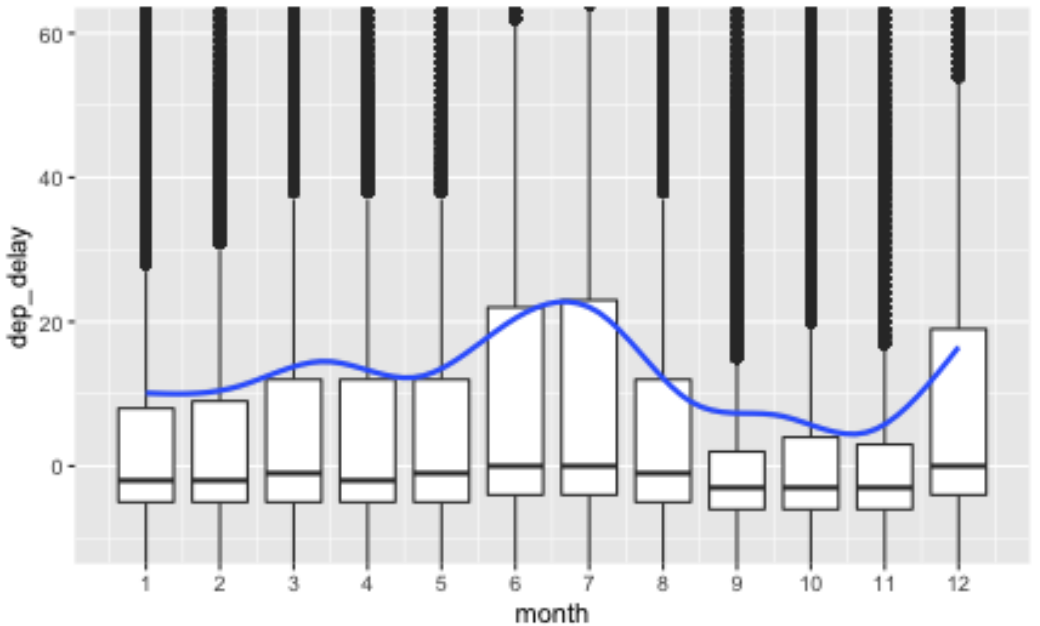
Delay per month (output)



Delay per month - boxplot (code)

```
flights %>%
  ggplot(aes(x = month, y = dep_delay)) +
  geom_boxplot(aes(group = month)) +
  geom_smooth() +
  coord_cartesian(ylim = c(-10, 60)) +
  scale_x_continuous(breaks = 1:12)
```

Delay per month - boxplot (output)



Is it the weekends? (code)

```
flights <- flights %>%
  mutate(dow = wday(time_hour))

delay_dow <-
  flights %>%
  group_by(dow) %>%
  drop_na() %>%
  summarise(delay_m = mean(dep_delay),
            delay_md = median(dep_delay),
            q_05 = quantile(x = dep_delay, prob = .05),
            q_95 = quantile(x = dep_delay, prob = .95))
```

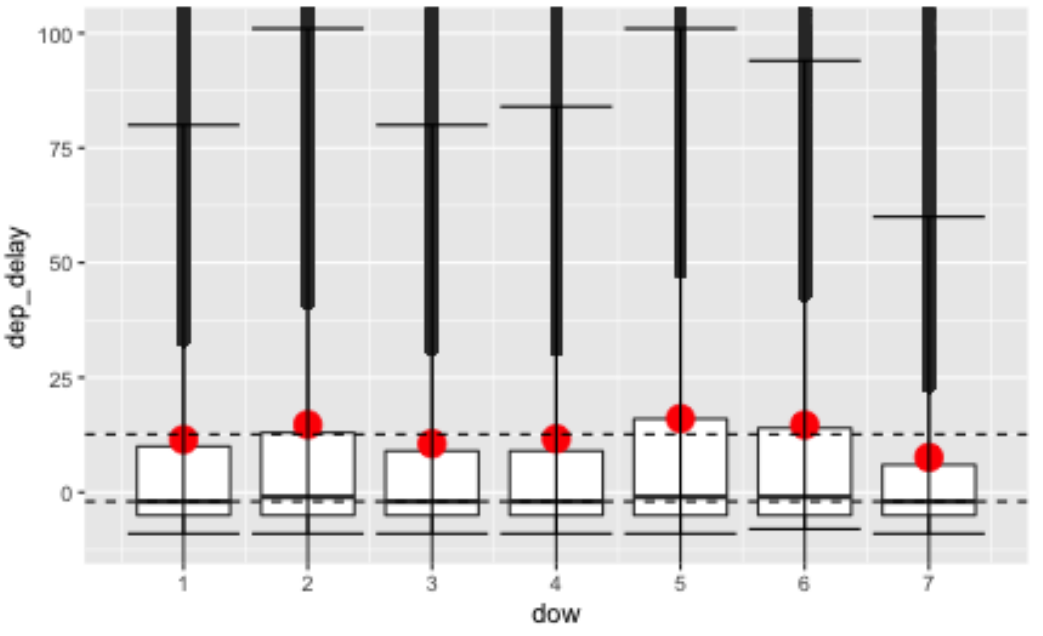
Is it the weekends? (output)

```
delay_dow
#> # A tibble: 7 x 5
#>     dow delay_m delay_md  q_05  q_95
#>   <dbl>   <dbl>    <dbl> <dbl> <dbl>
#> 1     1   11.5        -2    -9    80
#> 2     2   14.7        -1    -9   101
#> 3     3   10.6        -2    -9    80
#> 4     4   11.6        -2    -9    84
#> 5     5   16.0        -1    -9   101
#> 6     6   14.7        -1    -8    94
#> 7     7    7.59       -2    -9    60
```

Diagram of day of week delays (code)

```
flights %>%
  ggplot(aes(x = dow)) +
  geom_boxplot(aes(group = dow, y = dep_delay)) +
  geom_point(data = delay_dow, aes(y = delay_m), color = "red",
             size = 5) +
  coord_cartesian(ylim = c(-10, 100)) +
  scale_x_continuous(breaks = 1:7) +
  geom_hline(yintercept = mean(flights$dep_delay, na.rm = TRUE),
             linetype = "dashed") +
  geom_hline(yintercept = median(flights$dep_delay, na.rm = TRUE),
             linetype = "dashed") +
  geom_errorbar(aes(ymin = q_05, ymax = q_95), data = delay_dow)
```

Diagram of day of week delays (output)

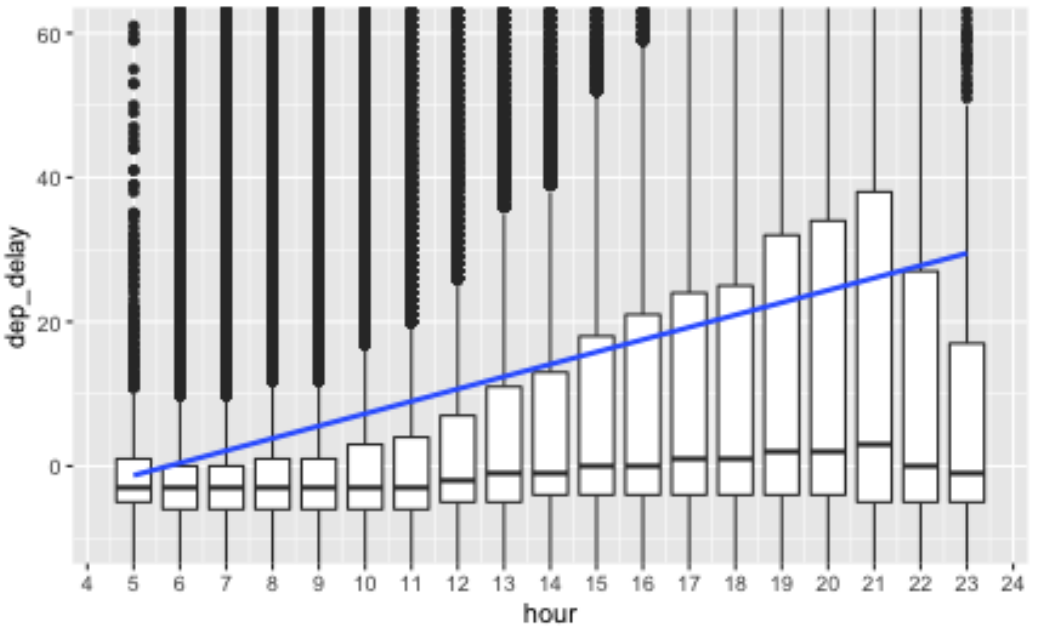


Delay per time of the day (code)

Let's check whether delays add up during the day, a popular opinion among travellers.

```
flights %>%
  select(dep_delay, hour) %>%
  ggplot(aes(x = hour, y = dep_delay)) +
  geom_boxplot(aes(group = hour)) +
  geom_smooth(method = "lm") +
  coord_cartesian(ylim = c(-10, 60))  +
  scale_x_continuous(breaks = 1:24)
```

Delay per time of the day (output)



Delay as function of month, hour, origin, and weekday

```
lm_hour <- lm(dep_delay ~ hour + month + origin + I(dow == 7),
              data = flights)

rsquared(lm_hour)
#> [1] 0.04408207
```

Geoplotting

Join airport data

```
data("airports")

flights_airports <-  # join destination long/lat
  flights %>%
  left_join(airports, by = c("dest" = "faa")) %>%
  rename(long = lon)

origin_latlong <-
  airports %>%
  filter(faa %in% c("LGA", "JFK", "EWR")) %>%
  rename(lat_origin = lat,
         long_origin = lon)

flights_airports <-  # join origin long/lat
  flights_airports %>%
  left_join(origin_latlong, by = c("origin" = "faa"))
```

Dataframe for plotting (code)

```
flights_airports_sum <- flights_airports %>%
  group_by(dest, origin) %>%
  summarise(n = n(),
            long = max(long),
            lat = max(lat),
            long_origin = max(long_origin),
            lat_origin = max(lat_origin))
```

Dataframe for plotting (output)

```
head(flights_airports_sum)
#> # A tibble: 6 x 7
#> # Groups:   dest [5]
#>   dest  origin     n   long   lat long_origin lat_origin
#>   <chr> <chr>  <int>  <dbl> <dbl>       <dbl>      <dbl>
#> 1 ABQ   JFK      254 -107.   35.0       -73.8       40.6
#> 2 ACK   JFK      265  -70.1  41.3       -73.8       40.6
#> 3 ALB   EWR      439  -73.8  42.7       -74.2       40.7
#> 4 ANC   EWR        8 -150.   61.2       -74.2       40.7
#> 5 ATL   EWR     5022  -84.4  33.6       -74.2       40.7
#> 6 ATL   JFK     1930  -84.4  33.6       -73.8       40.6
```

Geo plot flights (code)

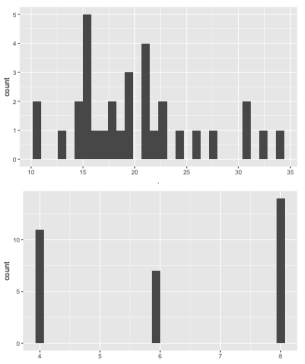
```
ggplot(data = map_data("usa")) +
  aes(x = long, y = lat, group = group) +
  geom_path(color = "grey40", size = .1) +
  geom_point(data = flights_airports_sum,
             aes(size = n, color = n, group = NULL)) +
  geom_segment(data = flights_airports_sum,
               aes(color = n, group = NULL,
                   x = long_origin, y = lat_origin,
                   xend = long, yend = lat)) +
  geom_text(data = flights_airports_sum %>% filter(n > 6000),
            aes(x = long, y = lat, label = dest, group = NULL),
            color = "grey40") +
  theme_map() +
  xlim(-130, -70) + ylim(+20, +50) +
  scale_color_viridis()
```

Geo plot flights (output)

More advanced stuff

Map columns to function with `map()`

```
data(mtcars)
purrr::map(select(mtcars, 1:2), ~ {ggplot(mtcars, aes(x = .)) +
    geom_histogram()})
#> $mpg
#>
#> $cyl
```



Map TWO columns to function with `map2()`

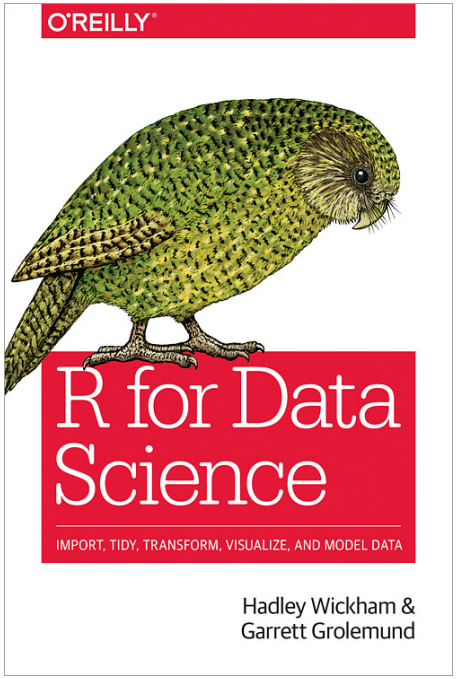
```
flights %>%
  select_if(~!is.numeric(.)) %>%
  map2(., names(.), ~ {ggplot(data = flights, aes(x = .x)) + geom_bar
```

Resources

Modern Dive

Modern Dive -- An Introduction to Statistical and Data Sciences via R Chester Ismay and Albert Y. Kim

R for Data Science

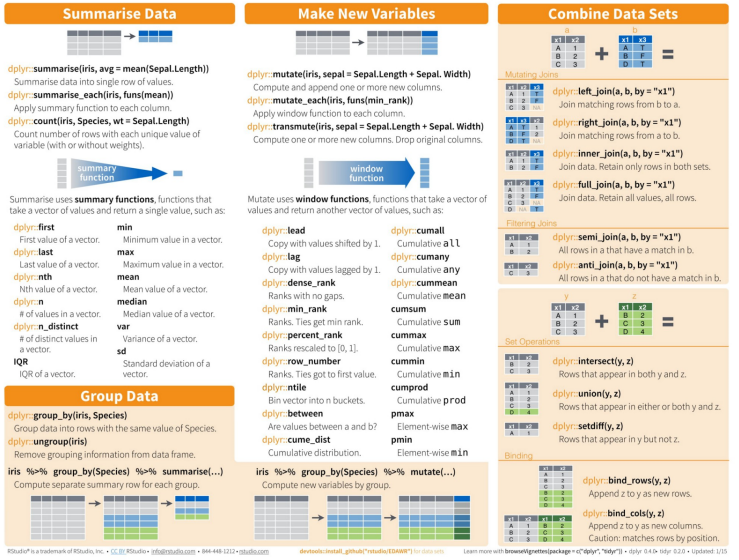


R for Data Science

Moderne Datenanalyse mit R

Moderne Datenanalyse mit R

Cheatsheets



https://www.rstudio.com/resources/cheatsheets/

Disclaimer: There may be issues at times

StackOverflow is your friend

Wrap-up

That was quick, but it was a start

Thank you

Sebastian Sauer

sebastiansauer

https://data-se.netlify.com/

ssauer@posteo.de

Sebastian Sauer

Get slides here:

Credit to

Built using R, RMarkdown, Xaringan. Thanks to the R community and the tidyverse developers.

Thanks to Yihui Xie and Antoine Bichat, among others, for Xaringan inspiration.

Thanks to R Hochschule for supporting me.

Images:

- Data Transformation with dplyr Cheat Sheet, by RStudio
- Modern Dive, Chester Ismay and Albert Y. Kim

SessionInfo

```
R.Version()$version.string
#> [1] "R version 3.5.1 (2018-07-02)"
search()
#>  [1] ".GlobalEnv"           "package:maps"         "package:bindro
#>  [4] "package:forcats"      "package:stringr"      "package:purrr
#>  [7] "package:readr"        "package:tidyr"        "package:tibble
#> [10] "package:tidyverse"    "package:knitr"        "package:sjmisc
#> [13] "package:ggmap"        "package:GGally"       "package:viridi
#> [16] "package:viridisLite"  "package:lubridate"    "package:corrr
#> [19] "package:broom"        "package:mosaic"       "package:Matrix
#> [22] "package:mosaicData"   "package:ggformula"    "package:ggstat
#> [25] "package:ggplot2"      "package:lattice"      "package:dplyr
#> [28] "package:nycflights13" "package:pacman"       "package:leafle
#>  [ reached getOption("max.print") -- omitted 8 entries ]
```