R crash course

Arnaud Legrand and Jean-Marc Vincent

Scientific Methodology and Performance Evaluation
ENS Lyon, November 2016

Outline

Why R?

R is a great language for data analysis and statistics

- Open-source and multi-platform
- Very expressive with high-level constructs
- Excellent graphics
- Widely used in academia and business
- Very active community
  - Documentation, FAQ on http://stackoverflow.com/questions/tagged/r
- Great integration with other tools

Why is such R a pain for computer scientists?

- R is not really a programming language
- Documentation is for statisticians
- Default plots are ~~cumbersome~~ (meaningful)
- Summaries are ~~cryptic~~ (precise)
- Steep learning curve even for us, computer scientists whereas we generally switch seamlessly from a language to another! That's frustrating! ;)

Do's and dont's

~~R is high level, I'll do everything myself~~

- CTAN comprises 4,334 TEX, LATEX, and related packages and tools. Most of you do not use plain TEX.
- Currently, the CRAN package repository features 4,030 available packages.
- How do you know which one to use??? Many of them are highly exotic (not to say useless to you).

  I learnt with http://www.r-bloggers.com/
- Lots of introductions but not necessarily what you're looking for so I'll give you a short tour.

  You should quickly realize though that you need proper training in statistics and data analysis if you do not want tell nonsense.
- Again, you should read Jain's book on The Art of Computer Systems Performance Analysis
- You may want to follow online courses:
  - https://www.coursera.org/course/compdata
  - https://www.coursera.org/course/repdata

Install and run R on debian

```
apt-cache search r
```

Err, that's not very useful :) It's the same when searching on google but once the filter bubble is set up, it gets better...

```
sudo apt-get install r-base
```

```
R
```

```
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

```

Install a few cool packages

R has it's own package management mechanism so just run R and type the following commands:

- `ddply`, `reshape` and `ggplot2` by Hadley Wickham (http://had.co.nz/)

```
install.packages("plyr")
  # or better: install.packages("dplyr")
install.packages("reshape")
  # or better; install.packages("tidyr")
install.packages("ggplot2")
```

- `knitR` by (Yihui Xie) http://yihui.name/knitr/

```
install.packages("knitr")
```

IDE

Using R interactively is nice but quickly becomes painful so at some point, you'll want an IDE.

Emacs is great but you'll need *Emacs Speaks Statistics*

```
sudo apt-get install ess
```

In this tutorial, I will briefly show you rstudio (https://www.rstudio.com/) and later how to use `org-mode`

Outline

Rstudio screenshot

RStudio

File Edit Code View Project Workspace Plots Tools Help

markdown-introduction.rmd | example-r-markdown.rmd

Knit HTML | Run | Chunks

```
28
29 ```{r basicconsole}
30 x <- 1:10
31 y <- round(rnorm(10, x, 1), 2)
32 df <- data.frame(x, y)
33 df
34 ```
35
36 ## Plots
37 Images generated by `knitr` are saved in a figures folder. However,
   they also appear to be represented in the HTML output using a [data
   URI scheme]( http://en.wikipedia.org/wiki/Data_URI_scheme). This
   means that you can paste the HTML into a blog post or discussion
   forum and you don't have to worry about finding a place to store the
   images; they're embedded in the HTML.
38
39 ### Simple plot
40 Here is a basic plot using base graphics:
41
42     ```{r simpleplot}
43     plot(x)
44     ```
45
46 ```{r simpleplot}
47 plot(x)
```

47:1 | Chunk 3: simpleplot | R Markdown

Workspace | History

Load | Save | Import Dataset | Clear All

Data	
df	10 obs. of 2 variables
Values	
x	integer[10]
y	numeric[10]

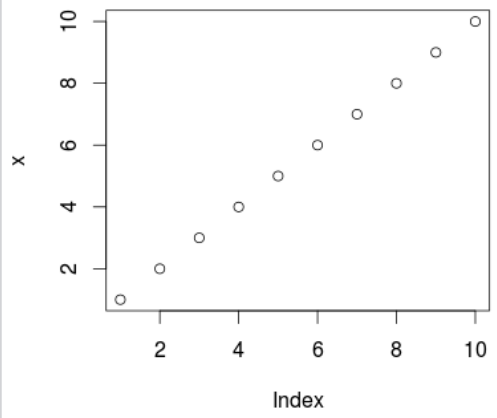
Console ~/research/statistics/rmarkdown-rmeetup-2012/

```
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> set.seed(1234)
> library(ggplot2)
> library(lattice)
> x <- 1:10
> y <- round(rnorm(10, x, 1), 2)
> df <- data.frame(x, y)
> df
    x    y
1   1 1.31
2   2 2.31
3   3 3.36
4   4 3.27
5   5 5.04
6   6 6.11
7   7 8.43
8   8 8.98
9   9 8.38
10 10 9.27
> plot(x)
```

Files | Plots | Packages | Help

Zoom | Export | Clear All



Reproducible analysis in Markdown + R

- Create a new R Markdown document (Rmd) in rstudio
- R chunks are interspersed with ```` ```{r} ```` and ```` ``` ````
- Inline R code: `` `r sin(2+2)` ``
- You can knit the document and share it via rpubs
- R chunks can be sent to the top-level with `Alt-Ctrl-c`
- I usually work mostly with the current environment and only knit in the end
- Other engines can be used (use rstudio completion)

````
```{r engine='sh'}
ls /tmp/
```
````

- Makes reproducible analysis as simple as one click
- Great tool for quick analysis for self and colleagues, homeworks, ...

Reproducible articles with LaTeX + R

- Create a new R Sweave document (Rnw) in rstudio
- R chunks are interspersed with <<>>= and @
- You can knit the document to produce a pdf
- You'll probably quickly want to change default behavior (activate the cache, hide code, ...). In the preembule:

```
<<echo=FALSE>>=
opts_chunk$set(cache=TRUE,dpi=300,echo=FALSE,fig.width=7,
               warning=FALSE,message=FALSE)
@
```

- Great for journal articles, theses, books, ...

Outline

Data frames

A data frame is a data tables (with columns and rows). `mtcars` is a built-in data frame that we will use in the sequel

```
head(mtcars);
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

You can also load a data frame from a CSV file:

```
df <- read.csv("http://foo.org/mydata.csv", header=T,
               strip.white=TRUE);
```

You will get help by using `?`:

```
?data.frame
?rbind
?cbind
```

Exploring Content (1)

```
names(mtcars);
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
[11] "carb"
```

```
str(mtcars);
```

```
'data.frame':         32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Exploring Content (2)

```
dim(mtcars);
length(mtcars);
```

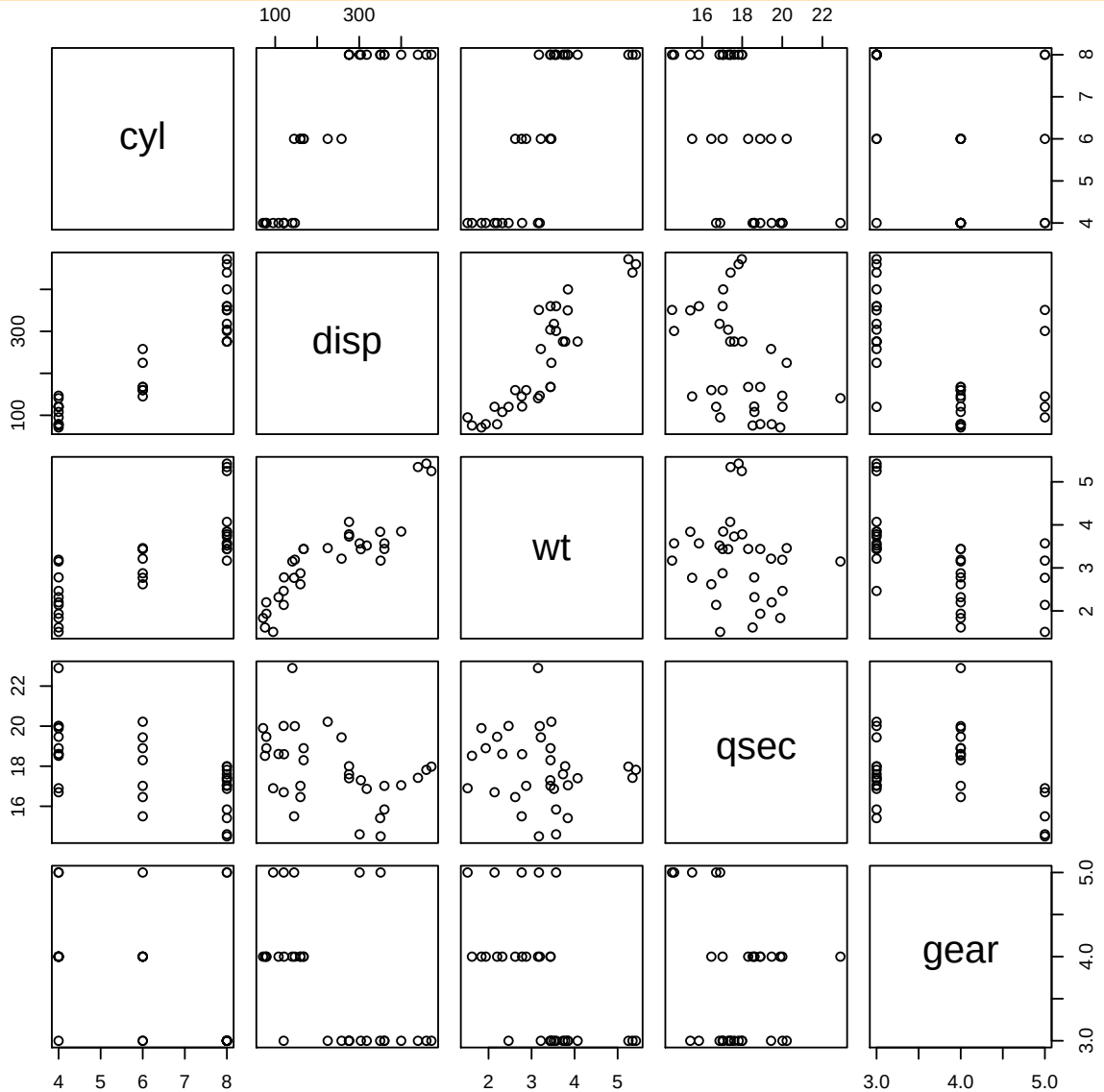
```
[1] 32 11
[1] 11
```

```
summary(mtcars);
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
```

Exploring Content (3)

```
1 plot(mtcars[names(mtcars) %in% c("cyl","wt","disp","qsec","gear")])
```



Accessing Content

```
mtcars$mpg
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.
[31] 15.0 21.4
```

```
mtcars[2:5,]$mpg
```

```
[1] 21.0 22.8 21.4 18.7
```

```
mtcars[mtcars$mpg == 21.0,]
```

```
                mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4        21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag    21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```
mtcars[mtcars$mpg == 21.0 & mtcars$wt > 2.7,]
```

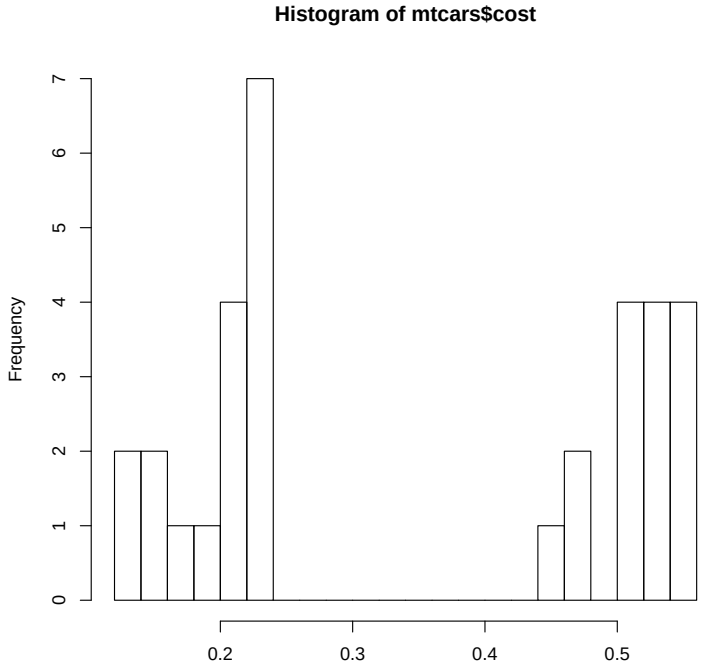
```
                mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4 Wag    21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

Extending Content

```
mtcars$cost = log(mtcars$hp)*atan(mtcars$disp)/
                  sqrt(mtcars$gear**5);
mean(mtcars$cost);
summary(mtcars$cost);
```

```
[1] 0.345994
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1261  0.2038  0.2353  0.3460  0.5202  0.5534
```

```
hist(mtcars$cost,breaks=20);
```



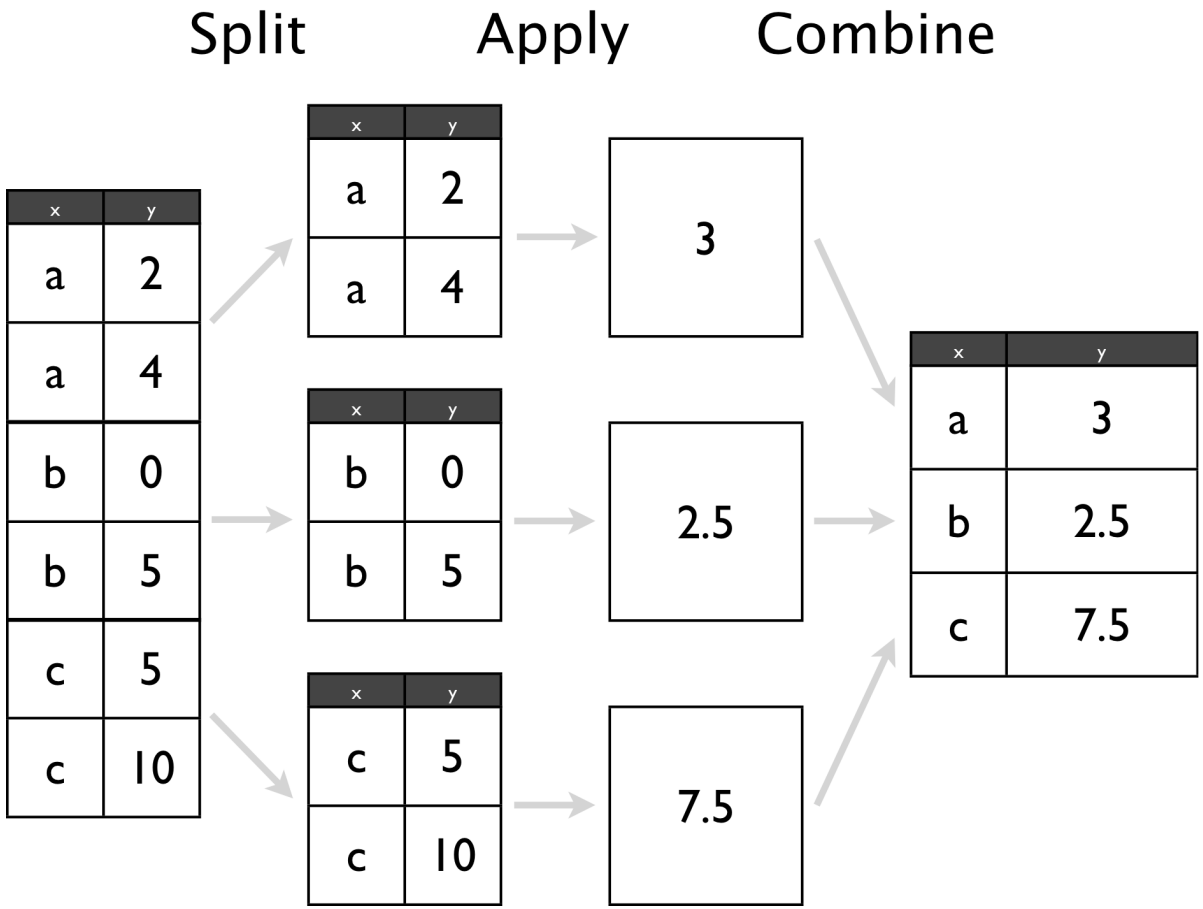
Take away Message

- R is a great tool but is only a tool. There is no magic. You need to understand what you are doing and get a minimal training in statistics
- It is one of the building block of reproducible research (the *reproducible analysis* block) and will save you a lot of time
- It provides you an access to any statistical method you ever dreamt of
- Read at least Jain's book: The Art of Computer Systems Performance Analysis
- There are introductory online courses (from John Hopkins university) on coursera which you may want to follow

Outline

plyr: the Split-Apply-Combine Strategy

Have a look at http://plyr.had.co.nz/09-user/ for a more detailed introduction.



plyr: Powerful One-liners

```
library(plyr)
mtcars_summarized = ddply(mtcars,c("cyl","carb"), summarize,
      num = length(wt), wt_mean = mean(wt), wt_sd = sd(wt),
      qsec_mean = mean(qsec), qsec_sd = sd(qsec));
mtcars_summarized
```

```
  cyl carb num  wt_mean     wt_sd qsec_mean   qsec_sd
1   4    1   5 2.151000 0.2627118  19.37800 0.6121029
2   4    2   6 2.398000 0.7485412  18.93667 2.2924368
3   6    1   2 3.337500 0.1732412  19.83000 0.5515433
4   6    4   4 3.093750 0.4131460  17.67000 1.1249296
5   6    6   1 2.770000        NA  15.50000        NA
6   8    2   4 3.560000 0.1939502  17.06000 0.1783255
7   8    3   3 3.860000 0.1835756  17.66667 0.3055050
8   8    4   6 4.433167 1.0171431  16.49500 1.4424112
9   8    8   1 3.570000        NA  14.60000        NA
```

plyr next generation = dplyr

It's much much faster and more readable. The *tutorial* is great...

```
library(dplyr)
mtcars %>% group_by(cyl,carb) %>%
           select(wt,qsec) %>%
           summarise(num = n(),
        wt_mean = mean(wt), wt_sd = sd(wt),
        qsec_mean = mean(qsec), qsec_sd = sd(qsec)) %>%
           filter(num>=1)
```

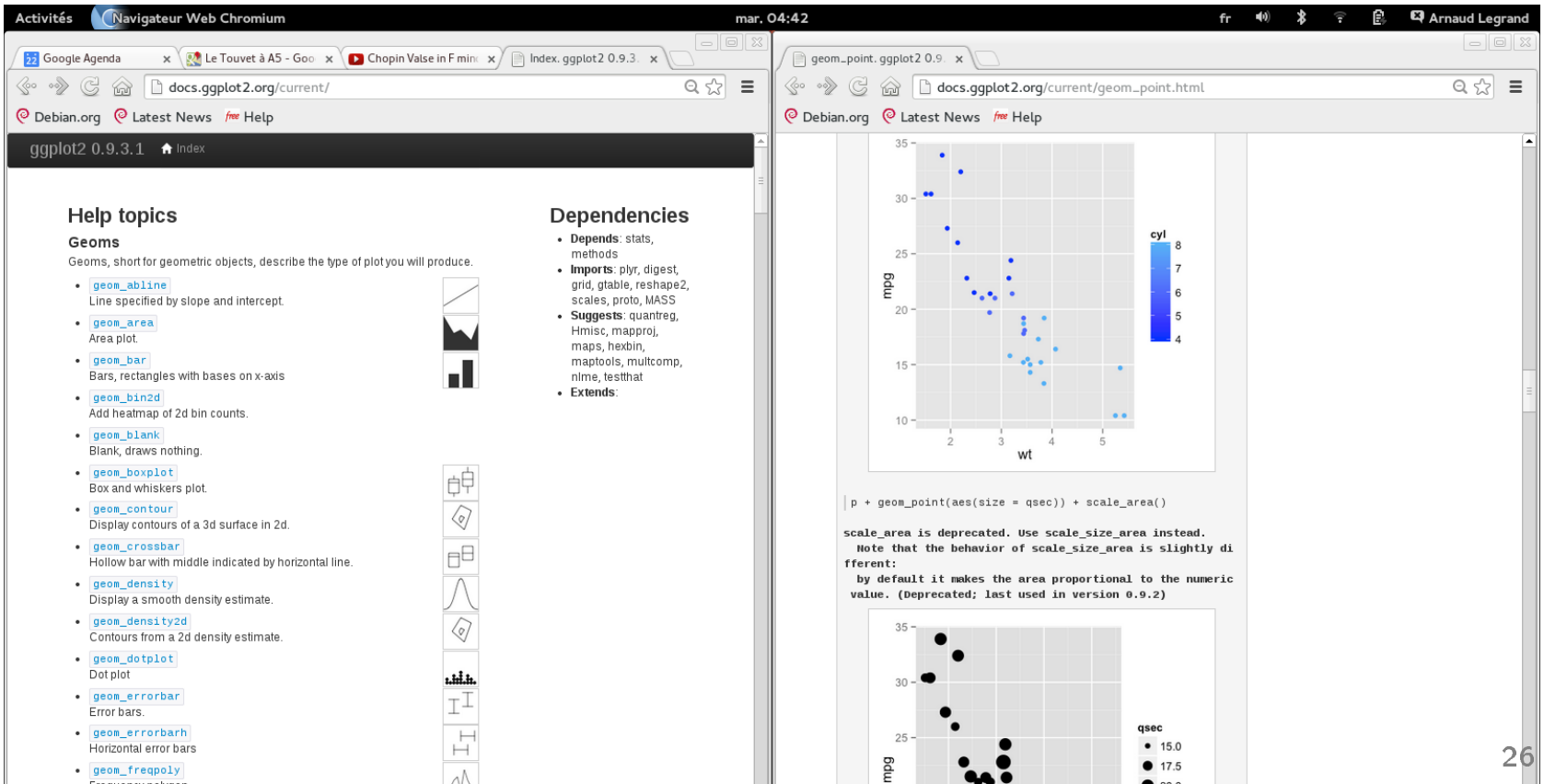
```
 Source: local data frame [9 x 7]
Groups: cyl

  cyl carb num  wt_mean      wt_sd qsec_mean   qsec_sd
1   4    1   5 2.151000 0.2627118  19.37800 0.6121029
2   4    2   6 2.398000 0.7485412  18.93667 2.2924368
3   6    1   2 3.337500 0.1732412  19.83000 0.5515433
```

Outline

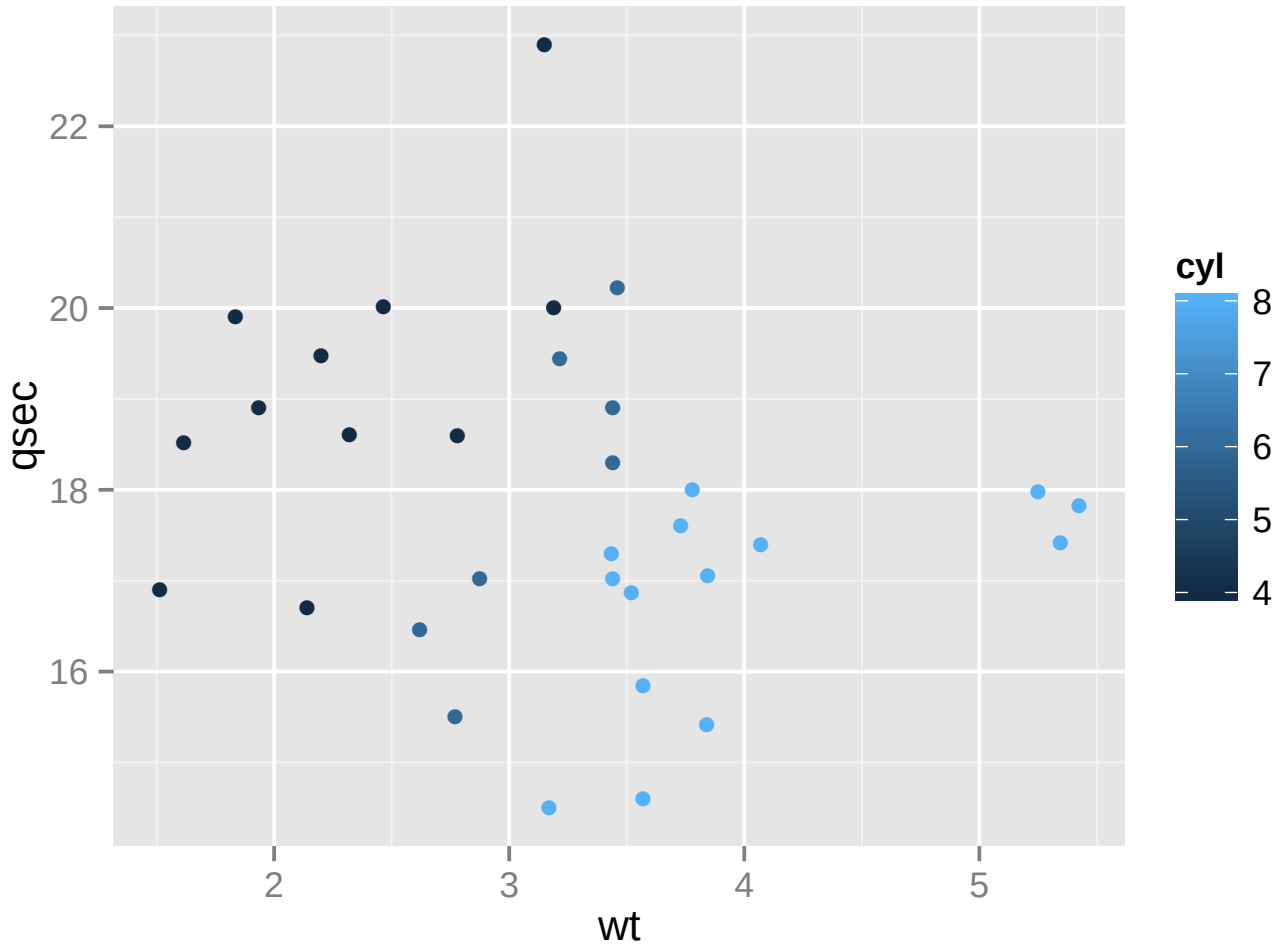
ggplot2: Modularity in Action

- `ggplot2` builds on plyr and on a modular grammar of graphics
- ~~obnoxious function with dozens of arguments~~
- combine small functions using layers and transformations
- aesthetic mapping between observation characteristics (data frame column names) and graphical object variables
- an incredible documentation: http://docs.ggplot2.org/current/



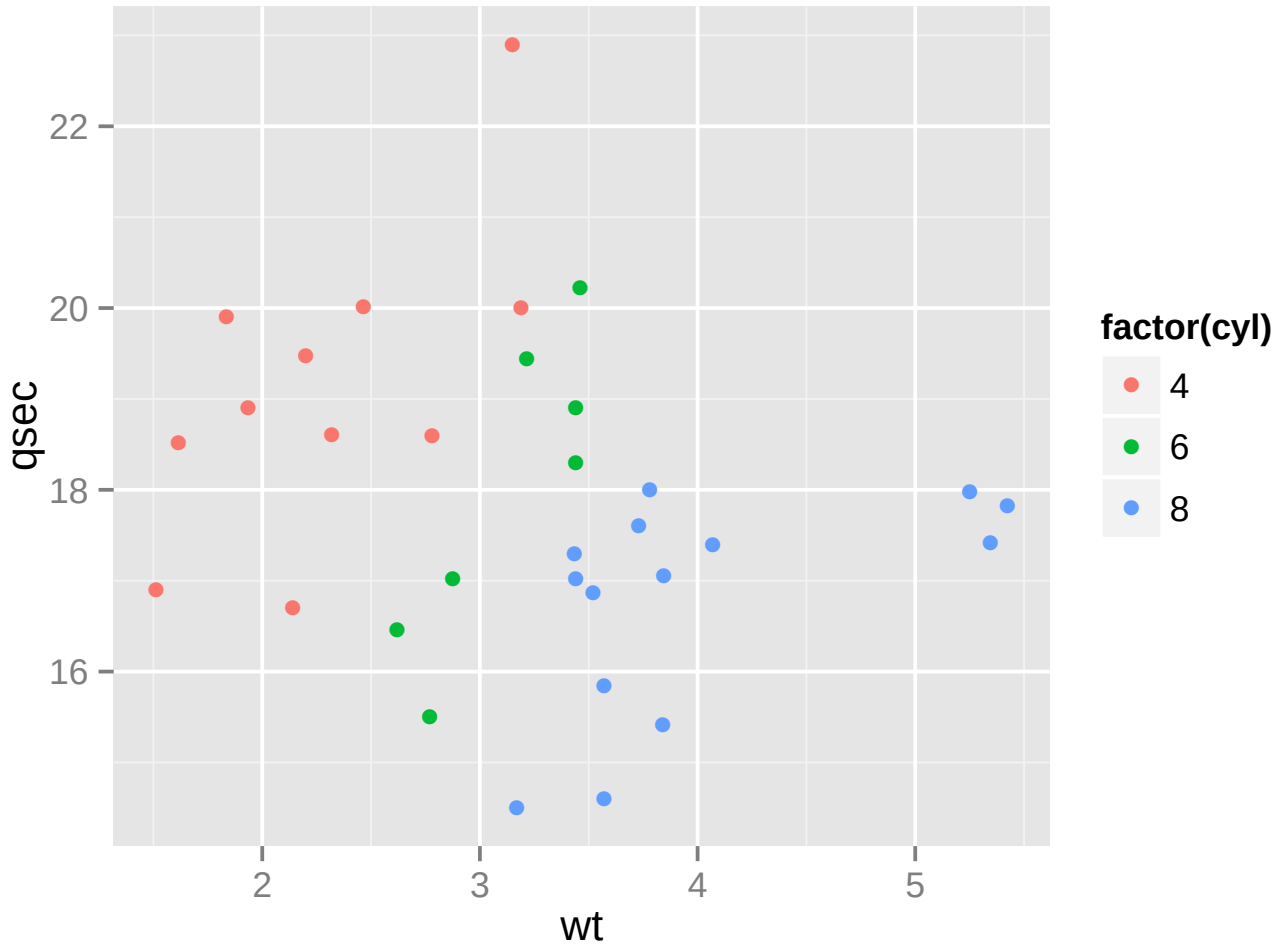
ggplot2: Illustration (1)

```
1 ggplot(data = mtcars, aes(x=wt, y=qsec, color=cyl)) +
2         geom_point();
```



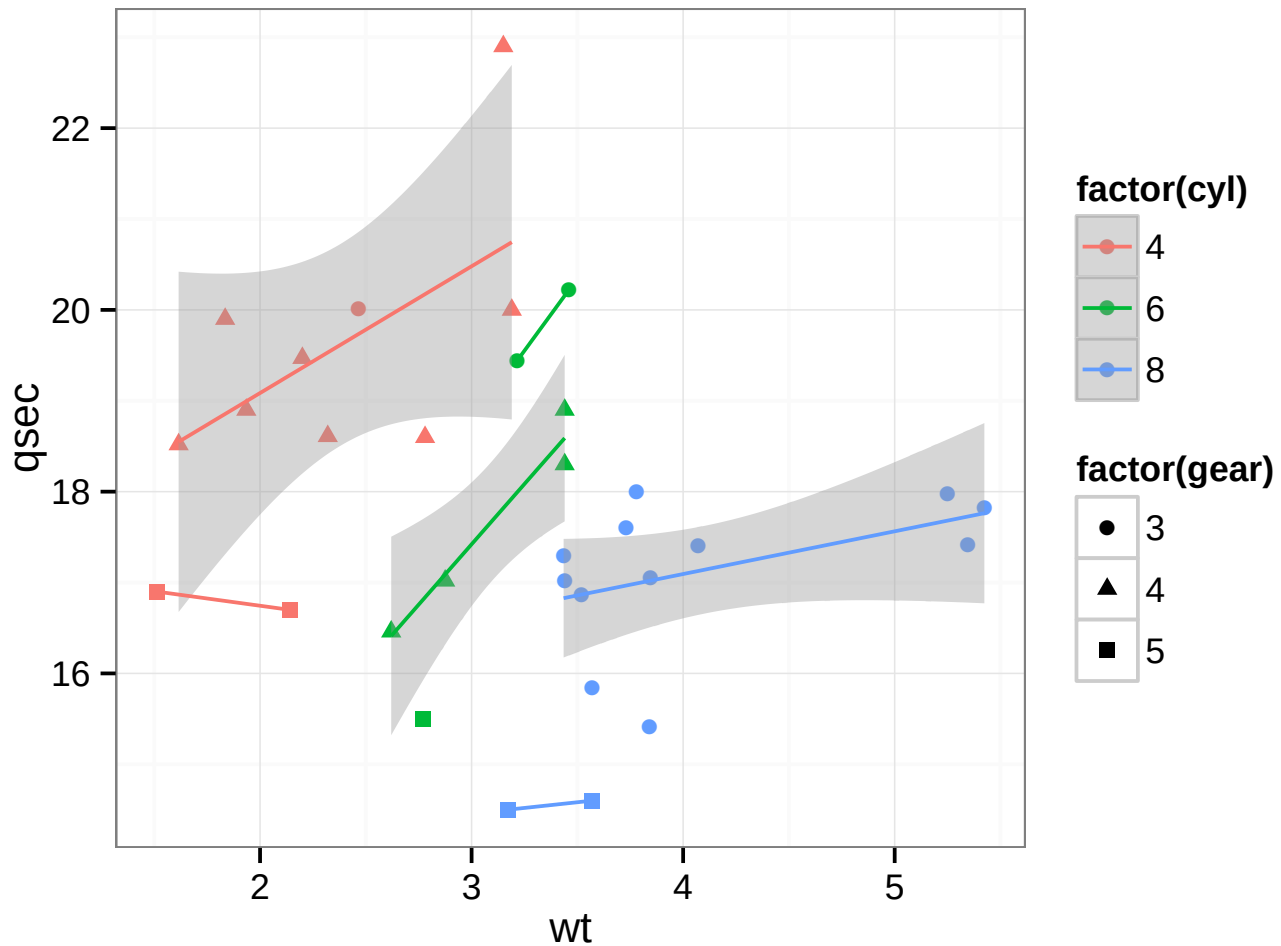
ggplot2: Illustration (2)

```
1 ggplot(data = mtcars, aes(x=wt, y=qsec, color=factor(cyl))) +
2         geom_point();
```



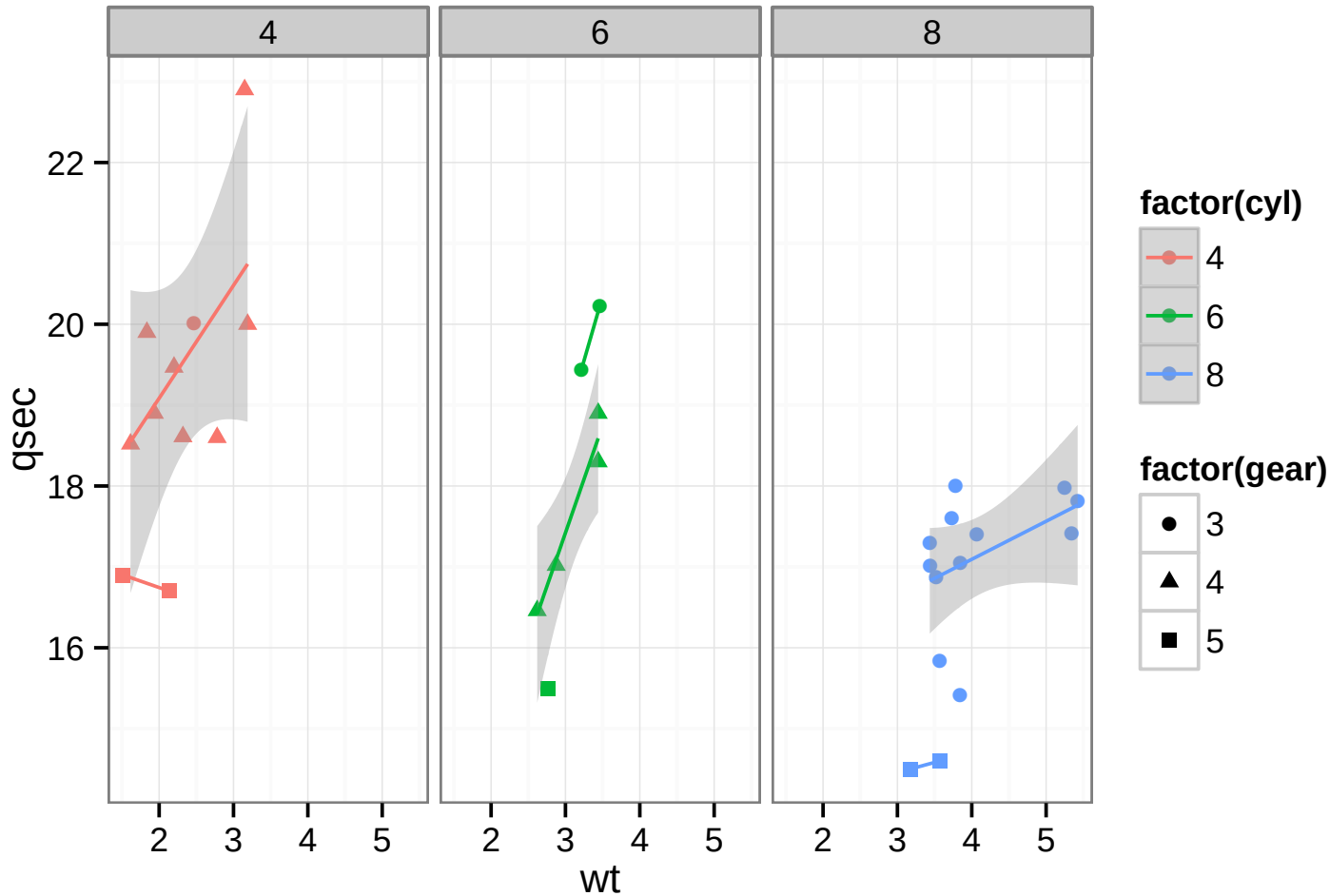
ggplot2: Illustration (3)

```
ggplot(data = mtcars, aes(x=wt, y=qsec, color=factor(cyl),
       shape = factor(gear))) +  geom_point() + theme_bw() +
       geom_smooth(method="lm");
```



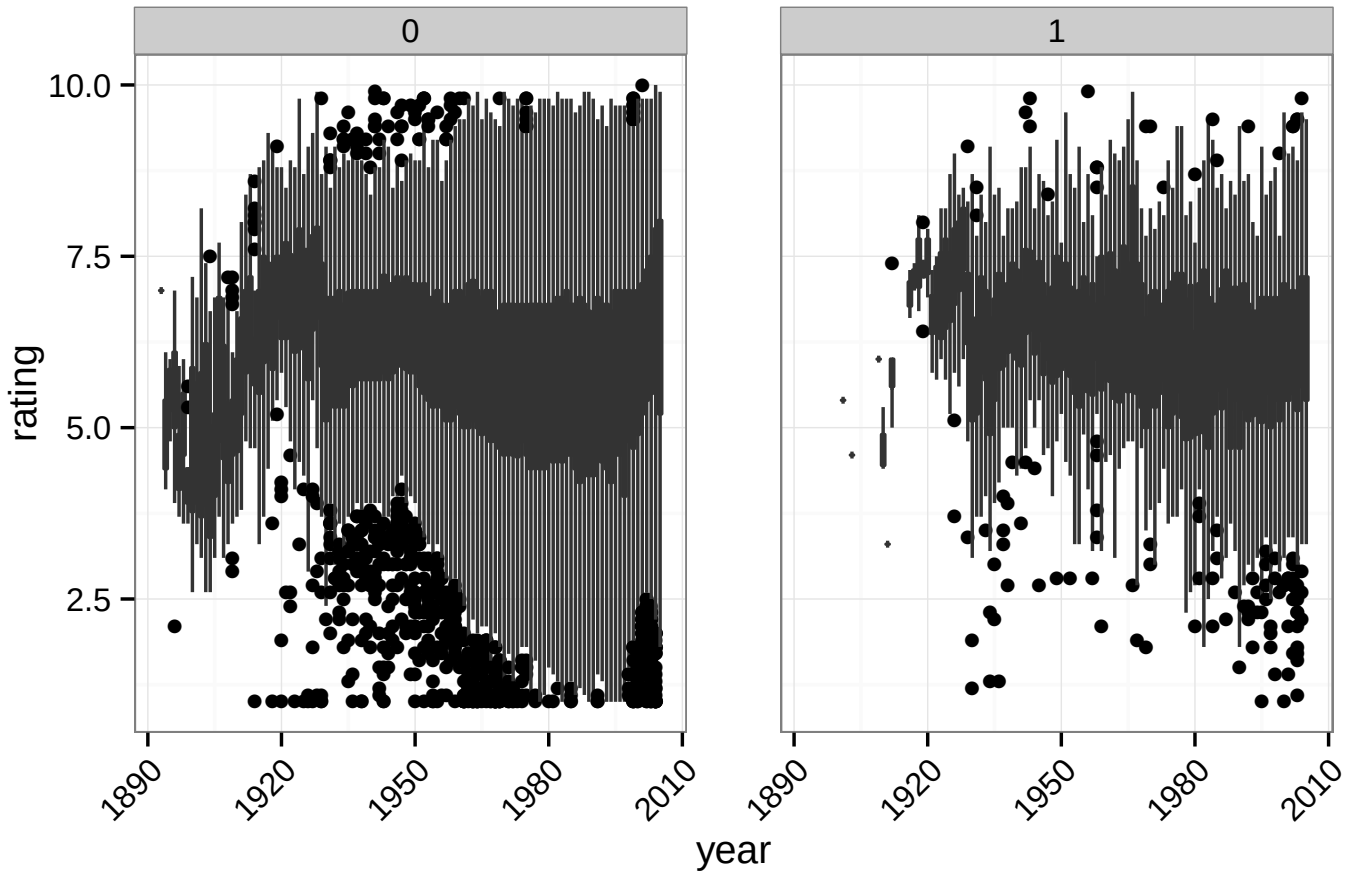
ggplot2: Illustration (4)

```
ggplot(data = mtcars, aes(x=wt, y=qsec, color=factor(cyl),
       shape = factor(gear))) + geom_point() + theme_bw() +
       geom_smooth(method="lm") + facet_wrap(~ cyl);
```



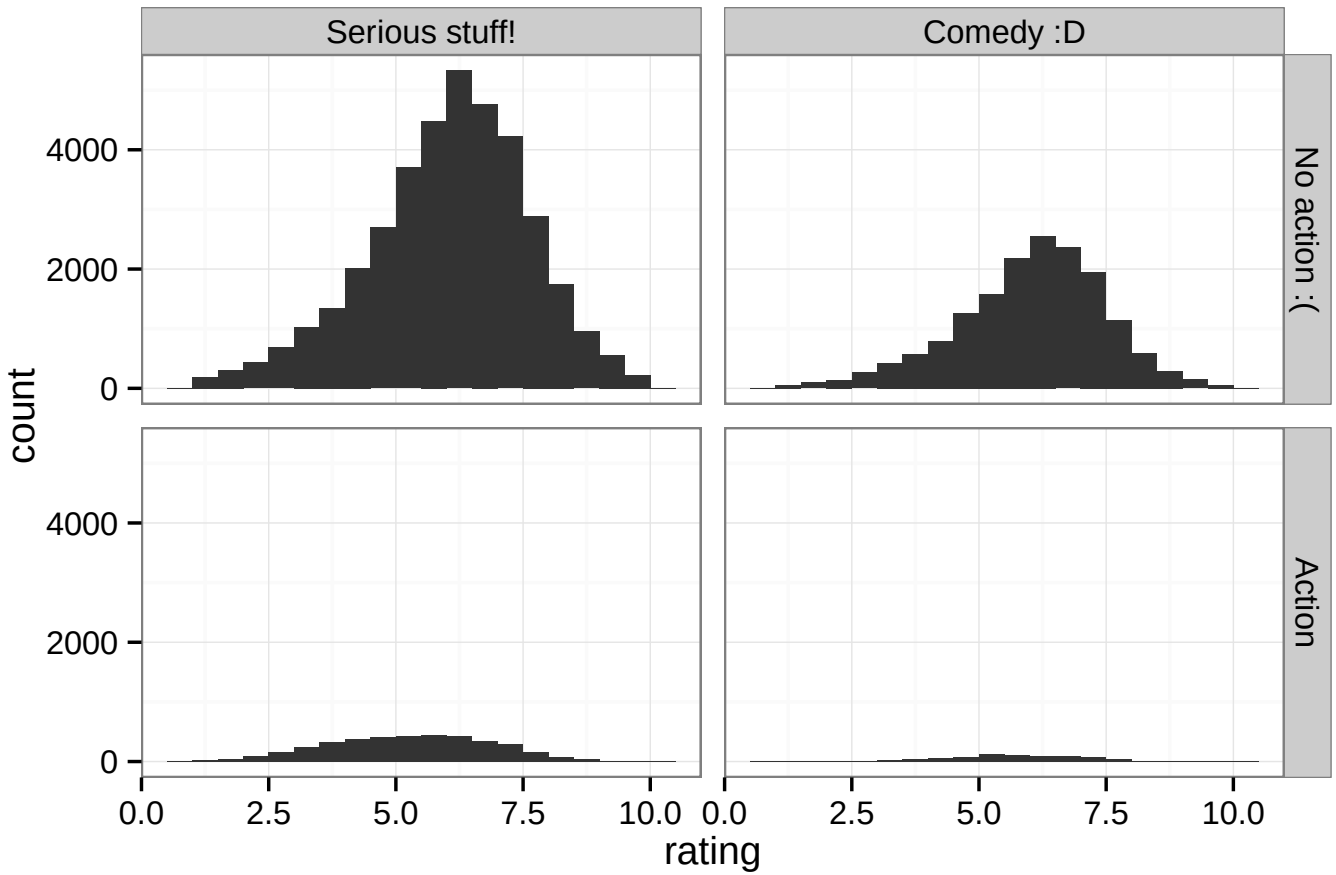
ggplot2: Illustration (5)

```
ggplot(data = movies, aes(x=year,y=rating,group=factor(year))) +
       geom_boxplot() + facet_wrap(~Romance) + theme_bw() +
       theme(axis.text.x = element_text(angle = 45, hjust = 1),
             panel.margin = unit(2, "lines"));
```



ggplot2: Illustration (6)

```
ggplot(movies, aes(x = rating)) + geom_histogram(binwidth = 0.5)+
       facet_grid(Action ~ Comedy, labeller=mf_labeller) +
       theme_bw() + theme(panel.margin = unit(.5, "lines"));
```



Outline

"Messy" data

When using ggplot or plyr, your data may not in the right shape, in which case you should give a try to reshape/melt

```
messy <- data.frame(
  name = c("Wilbur", "Petunia", "Gregory"),
  a = c(67, 80, 64),
  b = c(56, 90, 50)
)
messy
```

```
     name  a  b
1  Wilbur 67 56
2 Petunia 80 90
3 Gregory 64 50
```

- a and b are two different types of drugs and the values correspond to heart rate
- ggplot faceting or coloring based on the drug type is a pain
- we need a way to make "wide" data longer

Reshape

```
library(reshape)
cleaner = melt(messy,c("name"))
names(cleaner)=c("name","drug","heartrate")
cleaner
```

```
     name drug heartrate
1  Wilbur    a        67
2 Petunia    a        80
3 Gregory    a        64
4  Wilbur    b        56
5 Petunia    b        90
6 Gregory    b        50
```

Tidyr

Just like plyr, reshape is a little magical. tidyr is the new generation (faster, more coherent). Again, the *tutorial* is great.

```
library(tidyr)
library(dplyr)
messy %>% gather(drug, heartrate, -name)
```

```
     name drug heartrate
1  Wilbur    a        67
2 Petunia    a        80
3 Gregory    a        64
4  Wilbur    b        56
5 Petunia    b        90
6 Gregory    b        50
```

Hint: Avoid mixing old-generation with new-generation as it overrides some function names and leads to weird behaviors

Outline

Summarizing information

You may like these cheat sheets:

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

```
df = read.csv("data/set1.csv",header=T)
# Alternatively: read.csv("https://raw.githubusercontent.com/
#                alegrand/SMPE/master/lectures/data/set1.csv")
head(df,n=2)
```

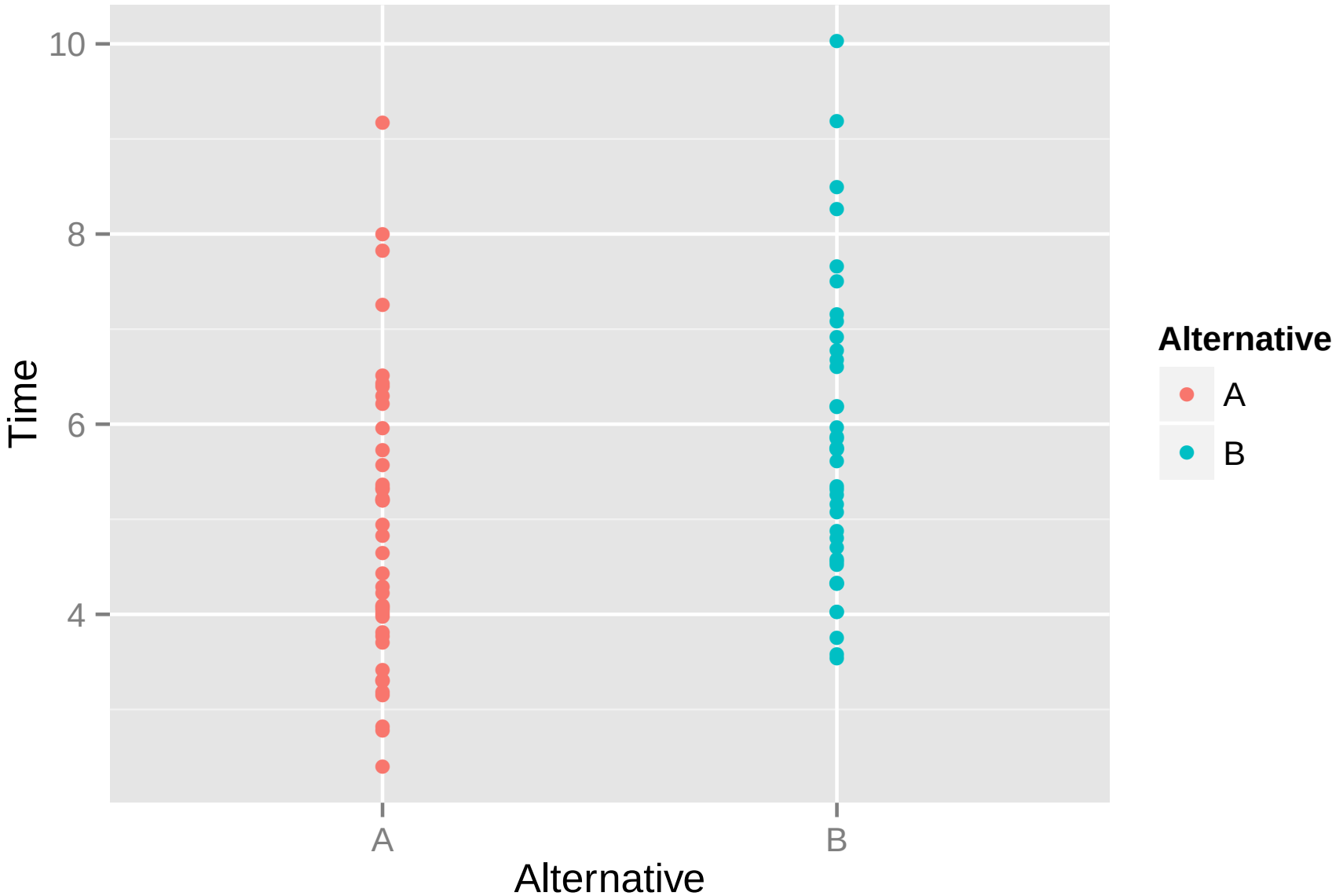
```
          A         B
1 7.256717 8.261171
2 3.813100 4.335301
```

Get the following summary using plyr/reshape or dplyr/tydir:

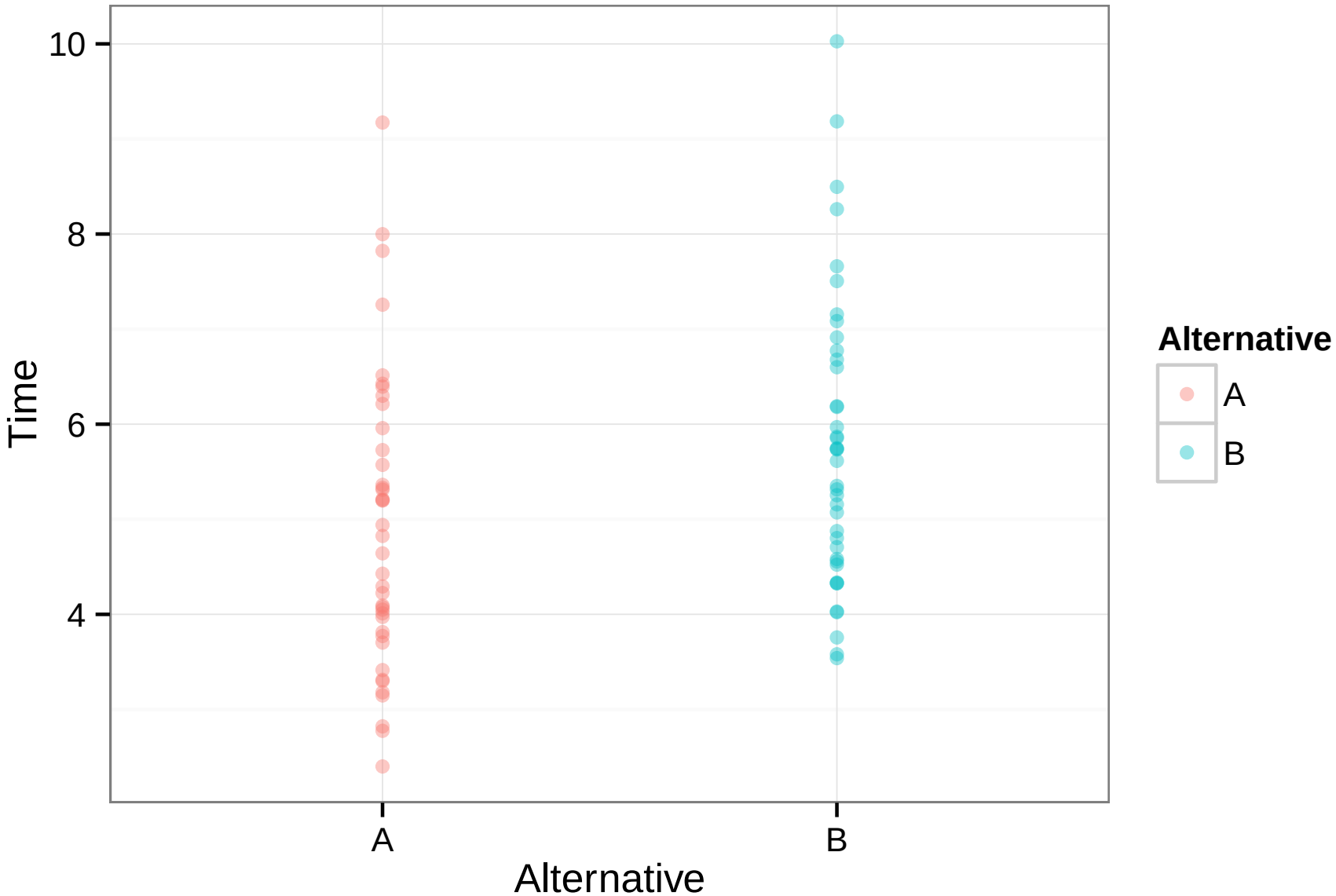
```
Source: local data frame [2 x 6]

  Alternative num     mean       sd      min       max
1           A  40 4.903817 1.544423 2.400016  9.172525
2           B  40 5.783643 1.542987 3.539874 10.027147
```

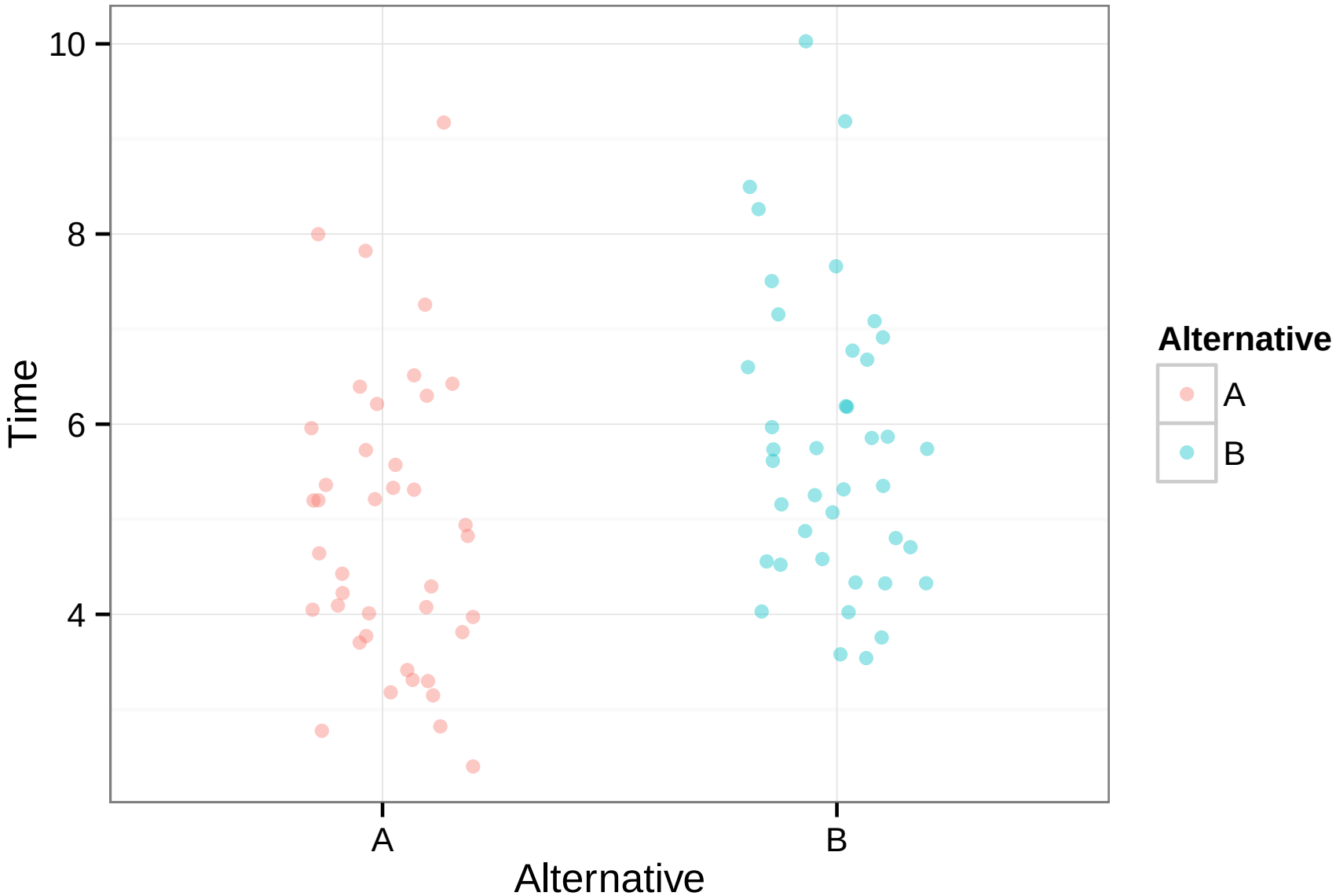
Plot the data



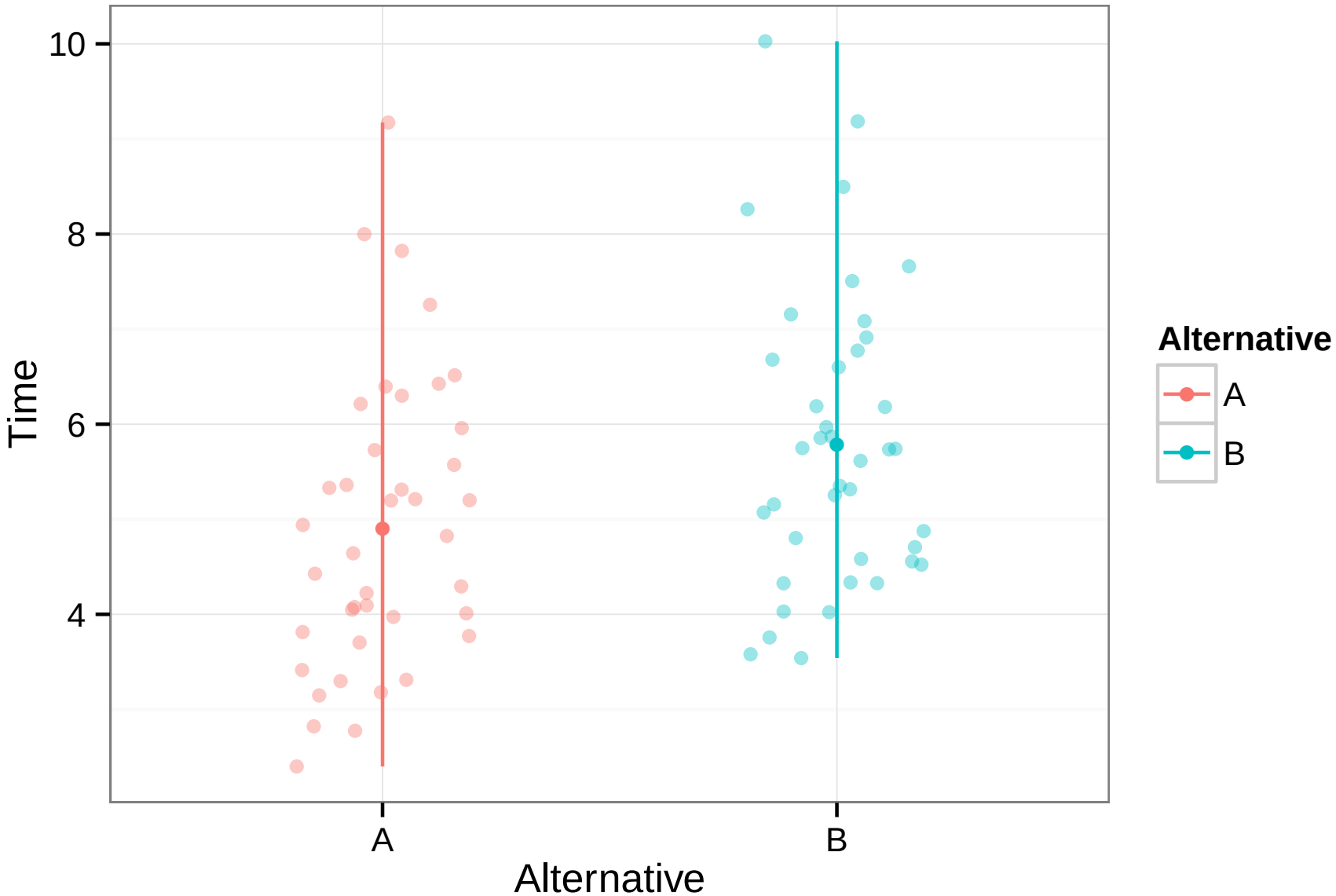
Alleviate over-plotting



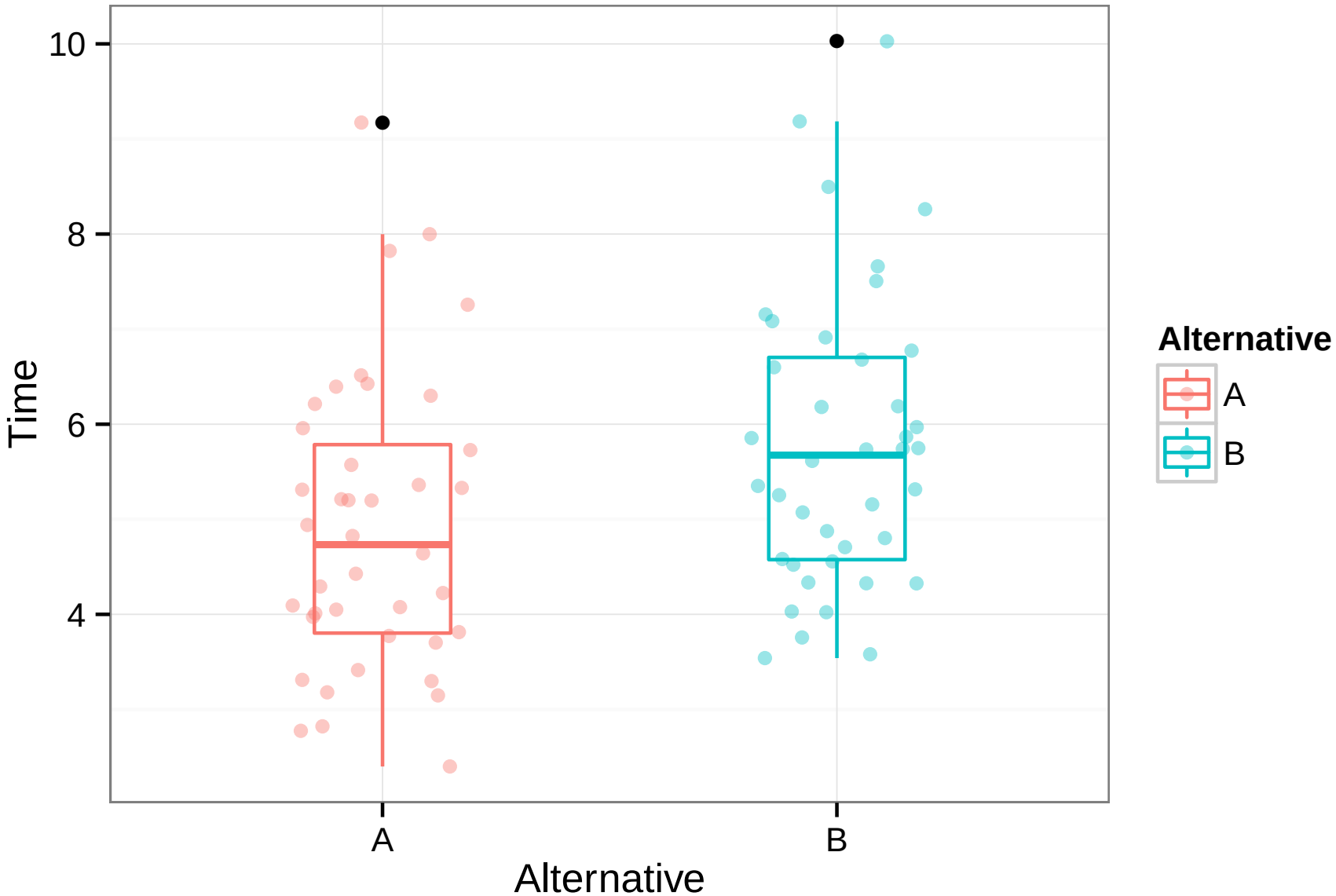
Avoid over-plotting



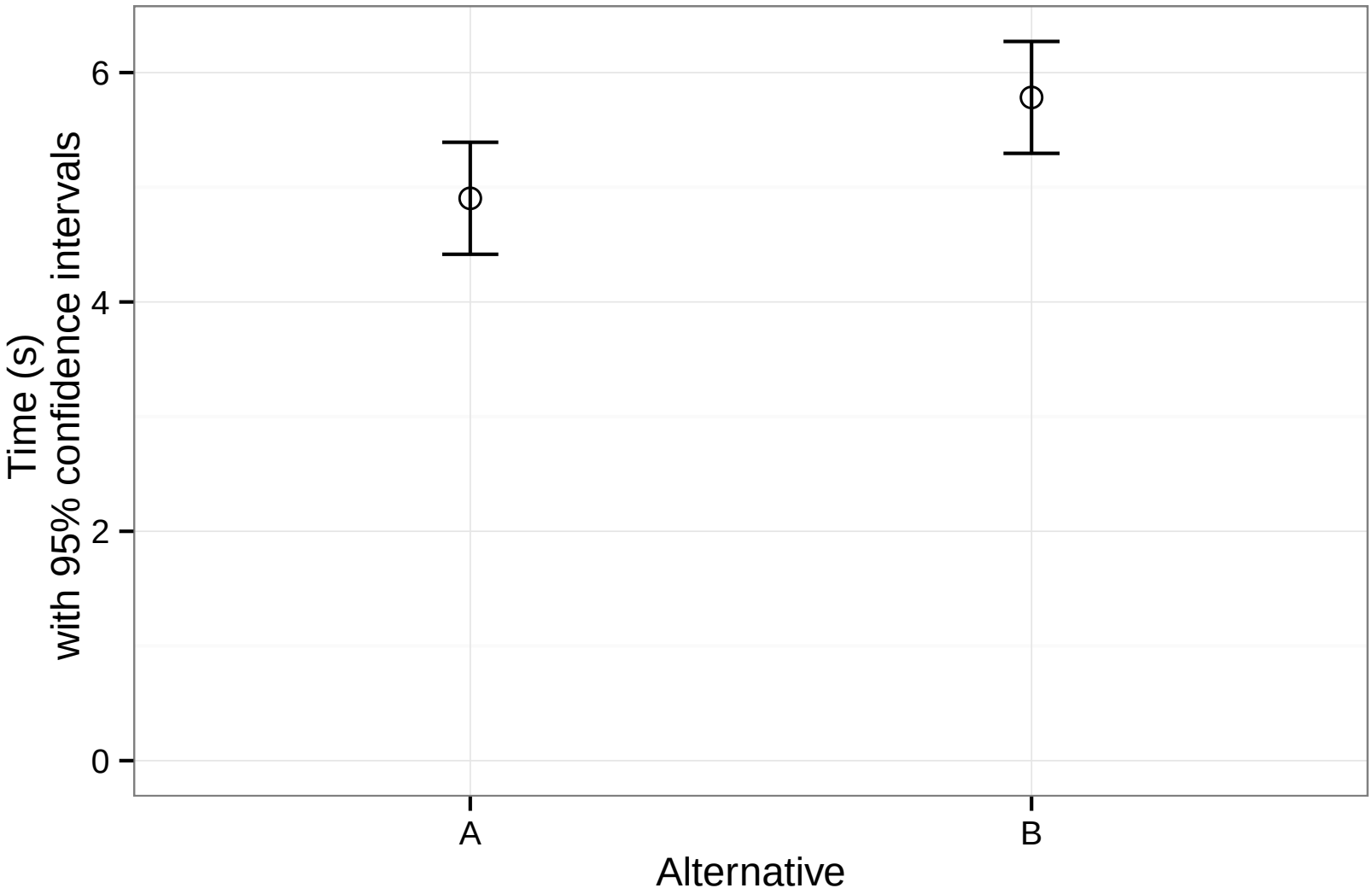
Add summary information



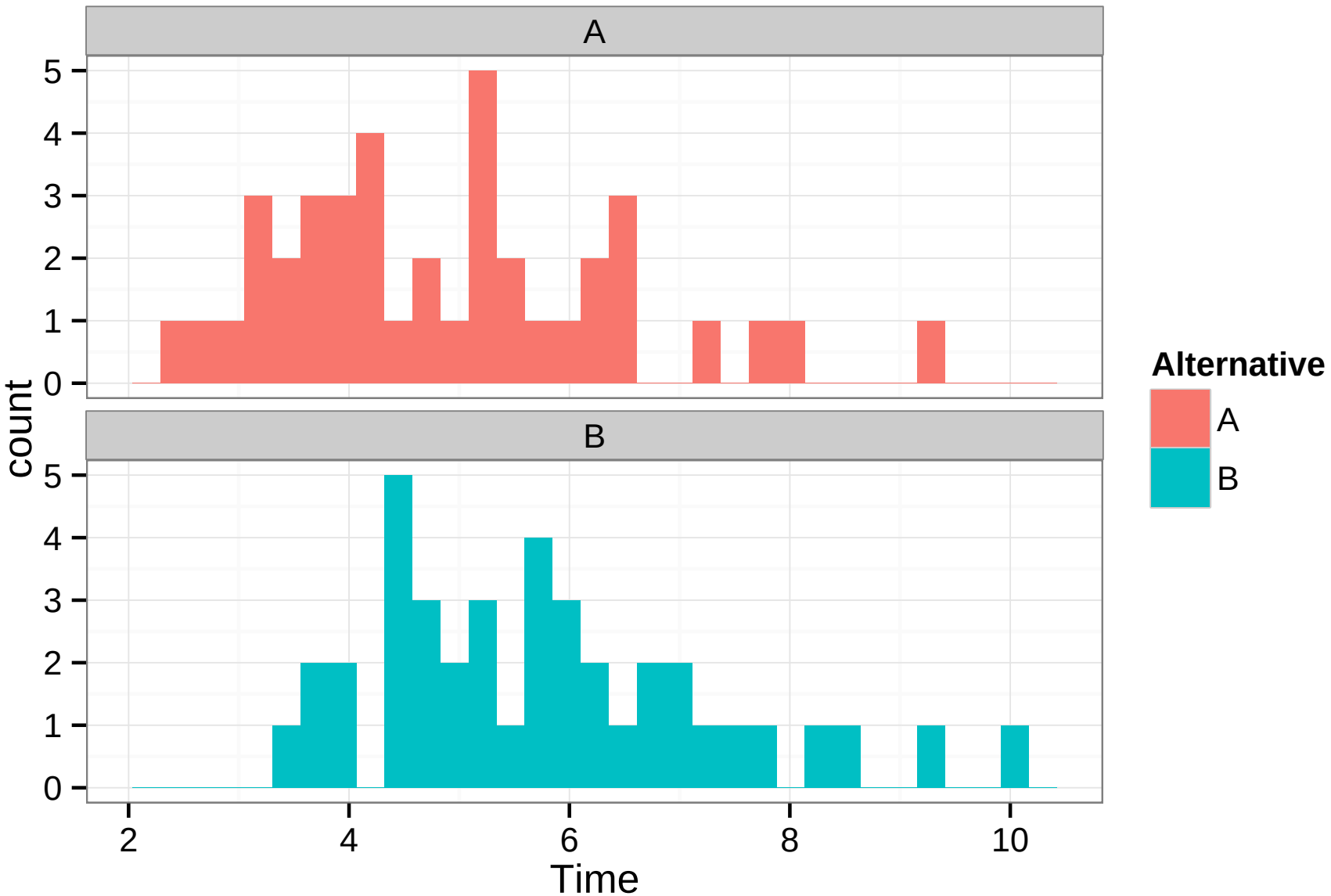
Add more standard summaries



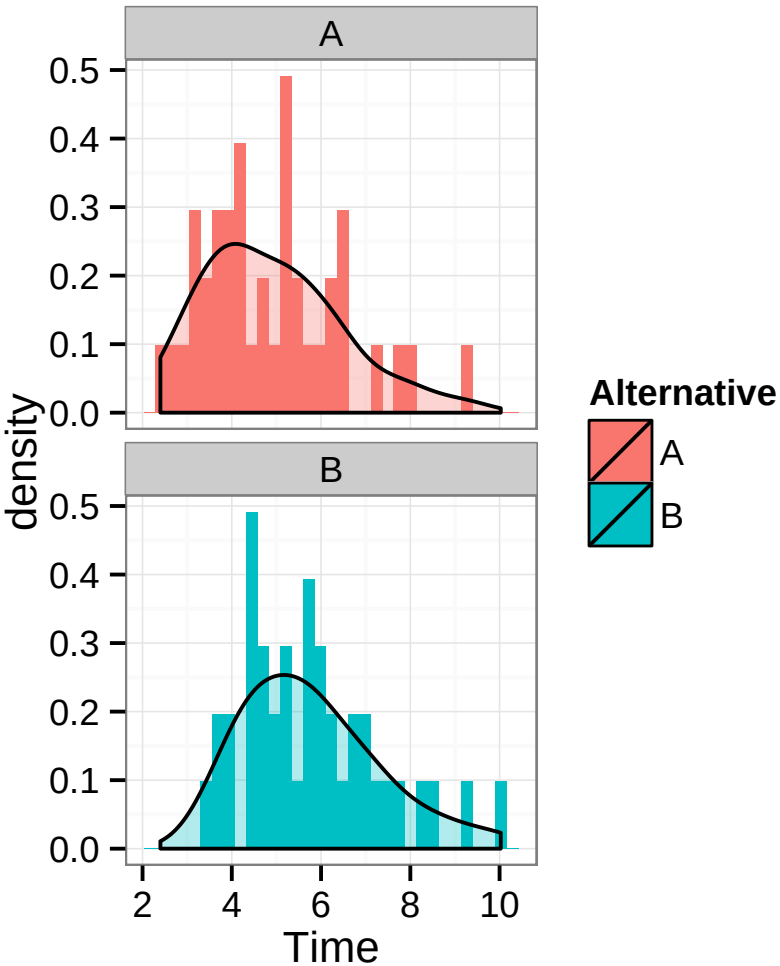
Or depict confidence intervals

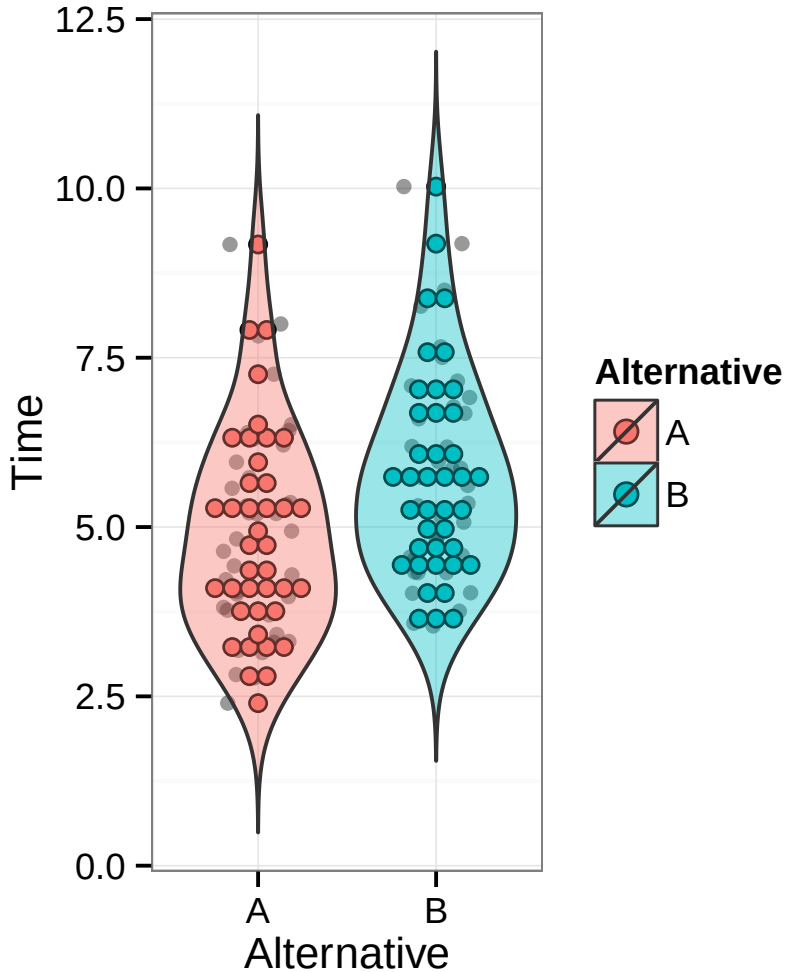


Or use histograms...



Be careful with fancy plots you do not fully understand!





Outline

Take away Message

- R, ggplot and other such tools are incredibly powerful for presenting data. They are much more high level than any other tools I have seen so far.
- Mastering it will save you a lot of time as it will allow to look at your data through different angles and thus check many hypothesis and present them in the best possible way
- Read at least Jain's book: The Art of Computer Systems Performance Analysis
- However, you may have started understanding that a visualization is meant to check or to illustrate one particular aspect and that this "aspect" relies on a mathematical model. I will thus explain you in the next lecture what this model is.

To do for the Next Time: Use what you just learned to improve your data analysis, the article you're currently writing, ...