Tutoriais | Intro

+

28 / 09 (Hoje)

+ Observable

01 / 10 (segunda-feira)

python

Bokeh

08 / 10

Data Analysis and Visualization

Research

Universität Konstanz

News | Persons | Research | Teaching | Publications

Analysis of Geographic Movement

Interactive analysis of movement data (e.g., animal movement, flight path analysis)

FGV EMAp

AG Keim - Datenanalyse und Visualisierung

Bruno Schneider

Universität Konstanz

Aktuelles | Personen | Forschung | Lehre | Publikationen

Bruno Schneider

Research Interests

- Information Visualization
- Classification Algorithms
- Interactive Machine Learning

Teaching Assistance

WS 2015/2016	Information Visualization I (Assistant - Exercises)
WS 2016/2017	Information Visualization I (Assistant - Exercises)
SS 2017	Information Visualization II (Assistant - Exercises)
WS 2017/2018	Data Mining: Basic Concepts (Assistant - Exercises)

Kontakt

Bruno Schneider
Universität Konstanz
Datenanalyse und Visualisierung
Postfach 78

Universität Konstanz
Universitätsstraße 10
78457 Konstanz
Deutschland

E-Mail schreiben

FGV EMAp

- Open source programming language
  - https://www.r-project.org/
- Software environment for statistical computing and graphics
- Huge Community
  - 13089 packages (September 2018)

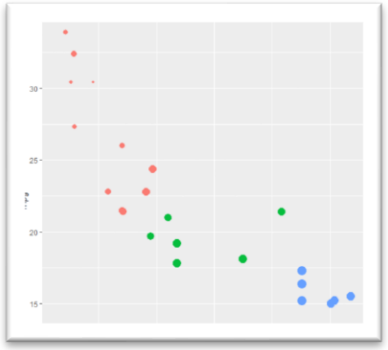
What is R?

FGV EMAp

- Statistical Analysis
- Data Preprocessing & Manipulation
- Data Visualization

What can I do with R?

Sumário

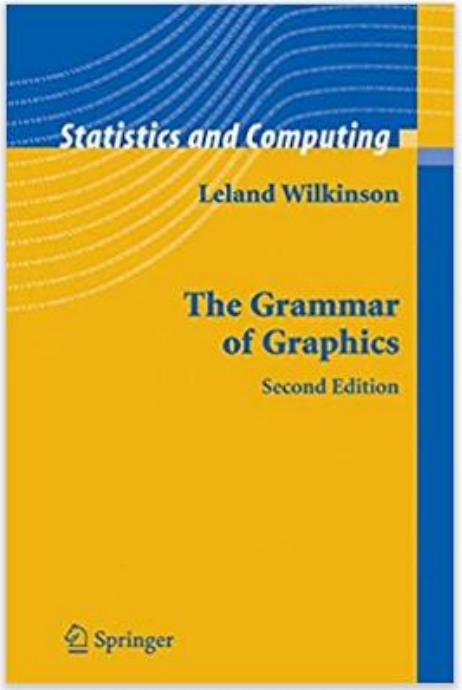


1 Começando pelo final: um exemplo de visualização no ggplot2

2 R: Introdução básica

3 ggplot2: explorando outros recursos e visualizações

www.rstudio.com



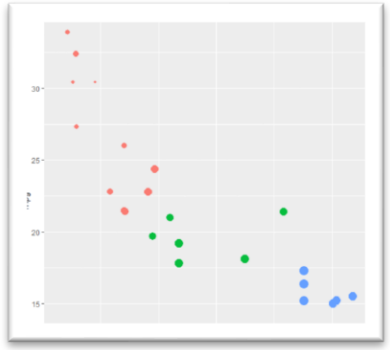
What Is The Grammar Of Graphics?

The basic idea: independently specify plot building blocks and combine them to create just about any kind of graphical display you want. Building blocks of a graph include:

- data
- aesthetic mapping
- geometric object
- statistical transformations
- scales
- coordinate system
- position adjustments
- faceting

Sumário

https://github.com/brunoschneider-de/emap



1 Começando pelo final: um exemplo de visualização no ggplot2

2 R: Introdução básica

3 ggplot2: explorando outros recursos e visualizações

```
> library(ggplot2)
```

```
> install.packages("ggplot2")
```

What Is The Grammar Of Graphics?

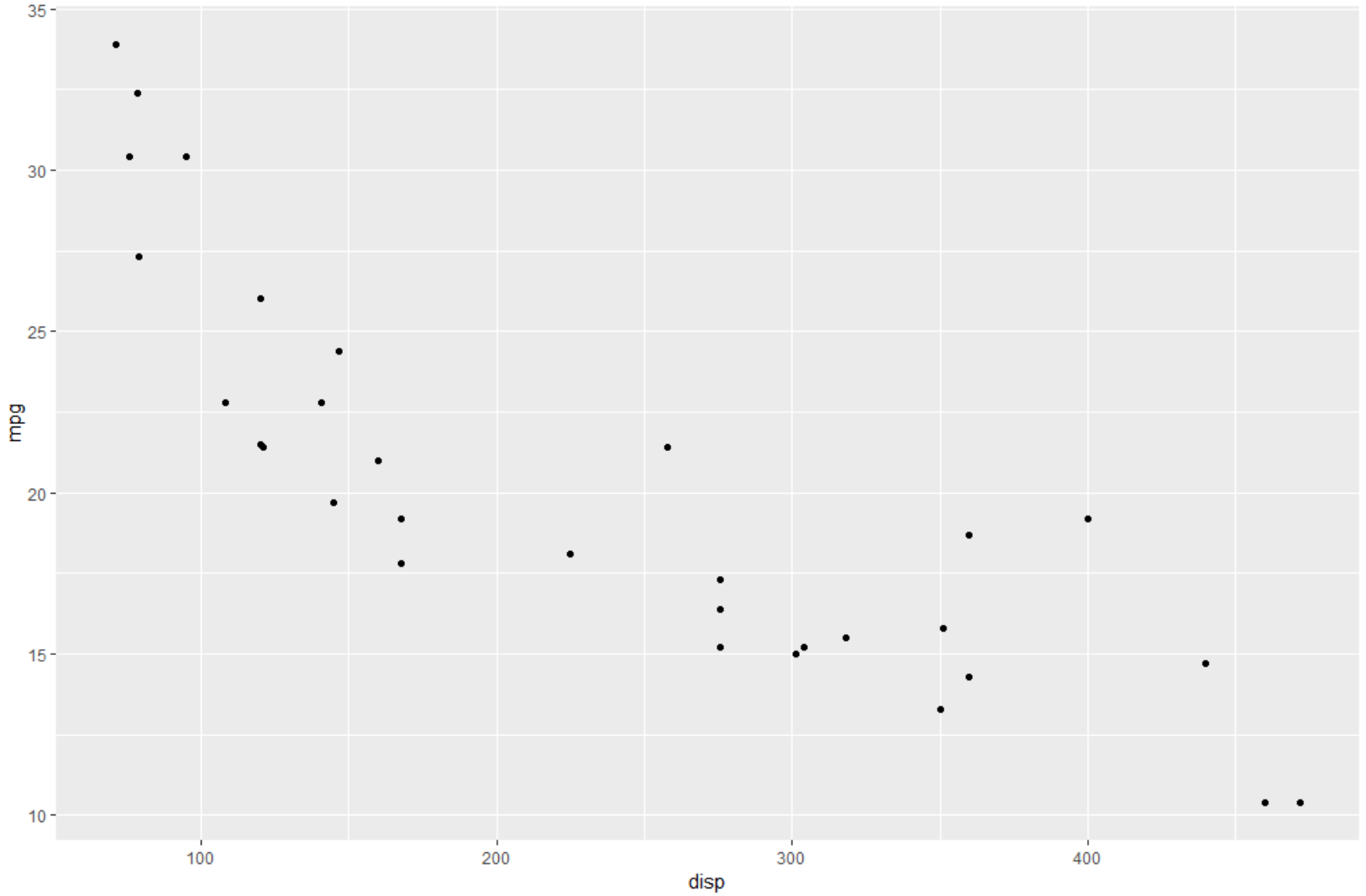
The basic idea: independently specify plot building blocks and combine them to create just about any kind of graphical display you want. Building blocks of a graph include:

- data
- aesthetic mapping
- geometric object
- statistical transformations
- scales
- coordinate system
- position adjustments
- faceting

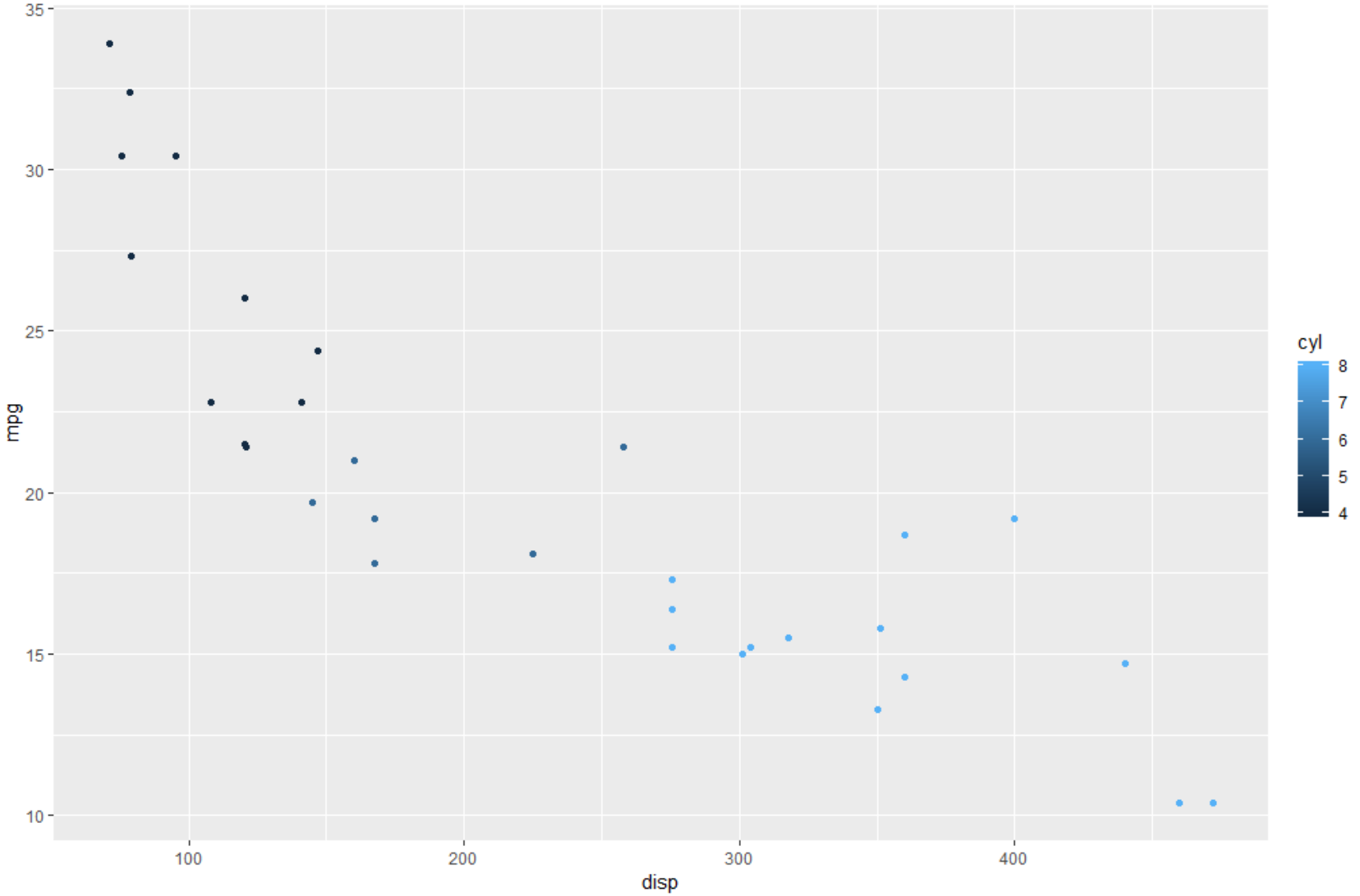
```
> ggplot(data = mtcars, aes(x = disp, y = mpg))
                + geom_point()
```

```
> mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic       30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona     21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin       15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28        13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9         27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2     26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa      30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E        21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
>
```

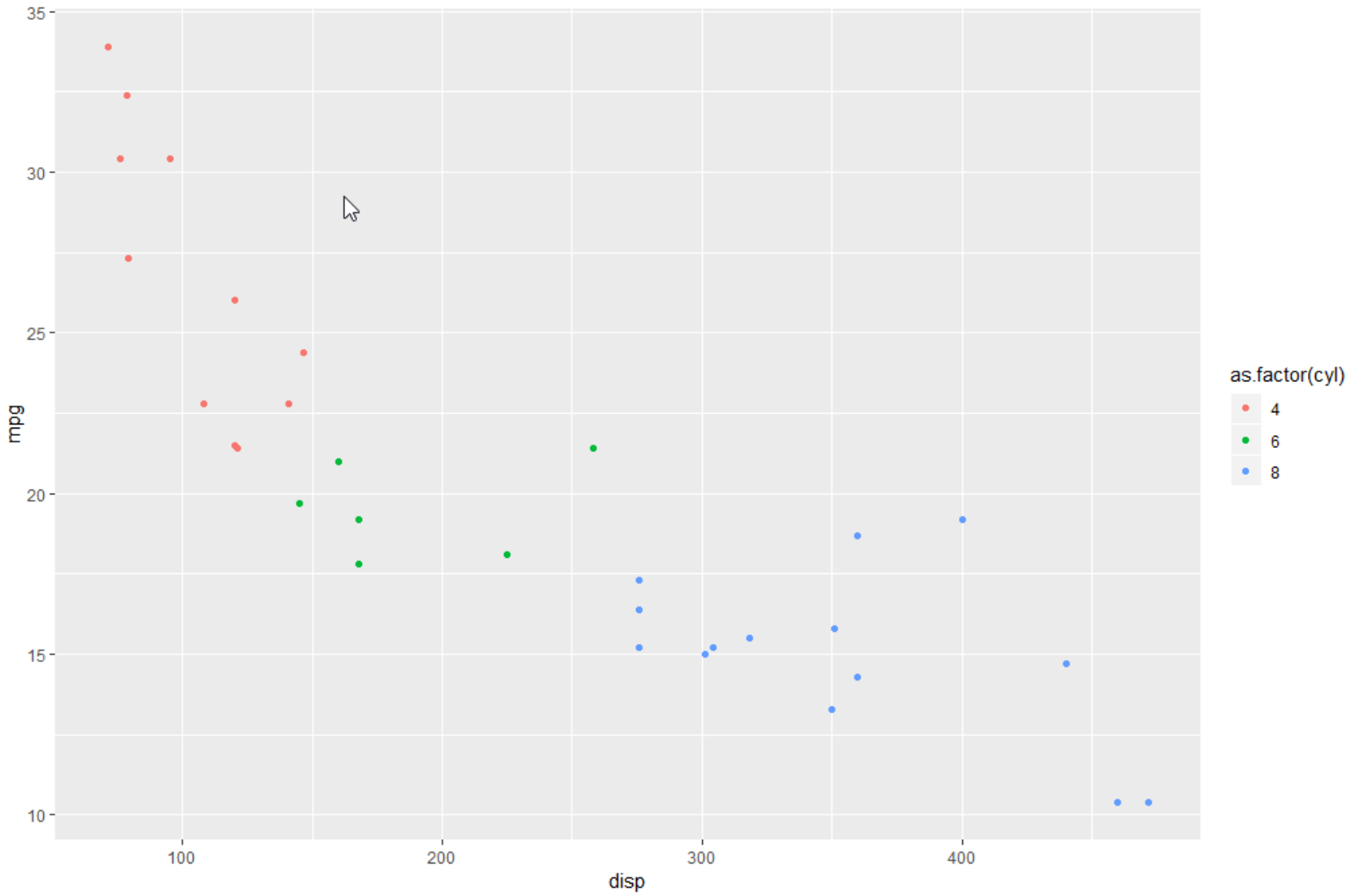
```
> ggplot(data = mtcars, aes(x = disp, y = mpg))
                + geom_point()
```



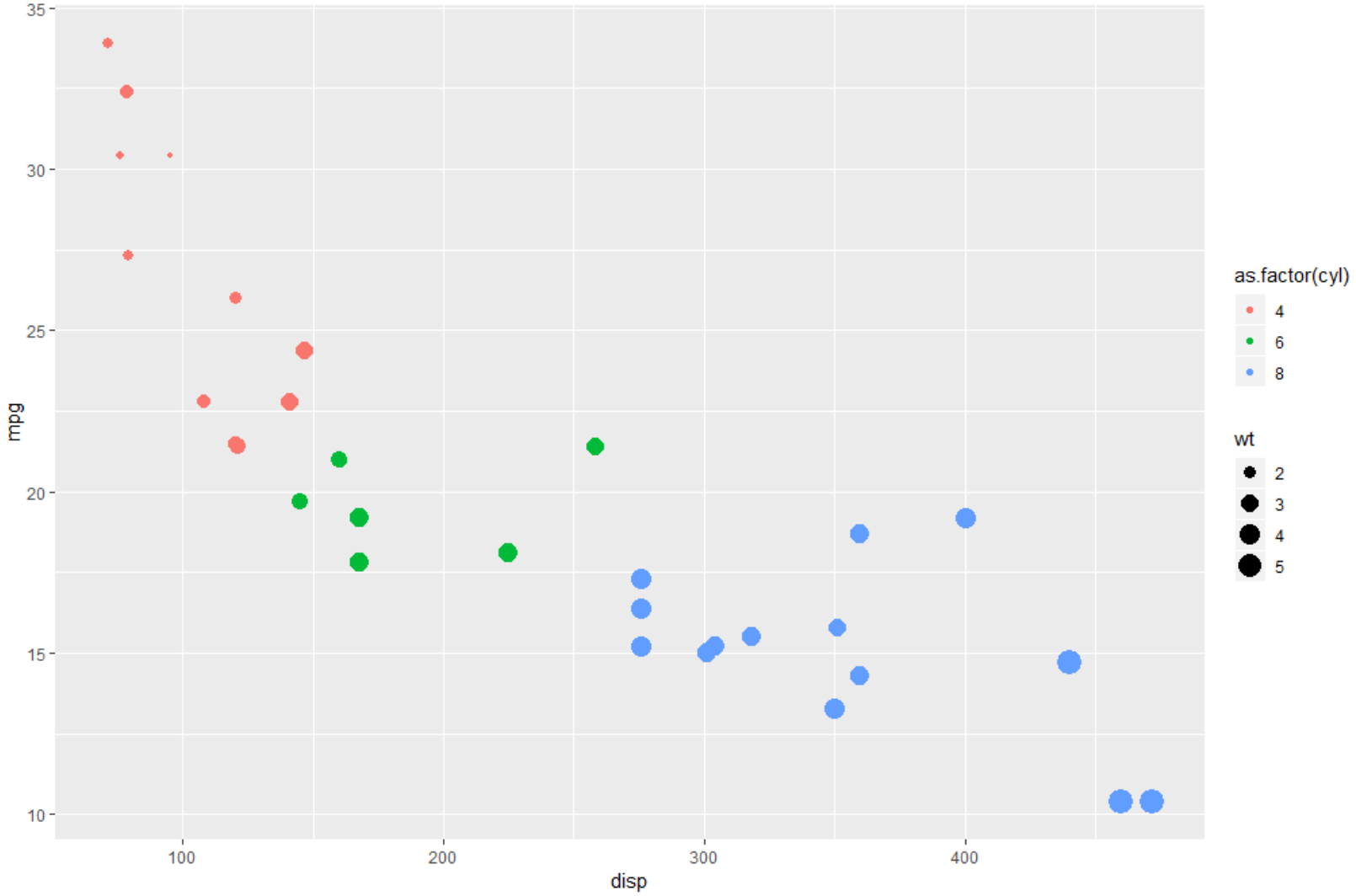
> ggplot(mtcars, aes(x = disp, y = mpg, **colour = cyl**)) + geom_point()



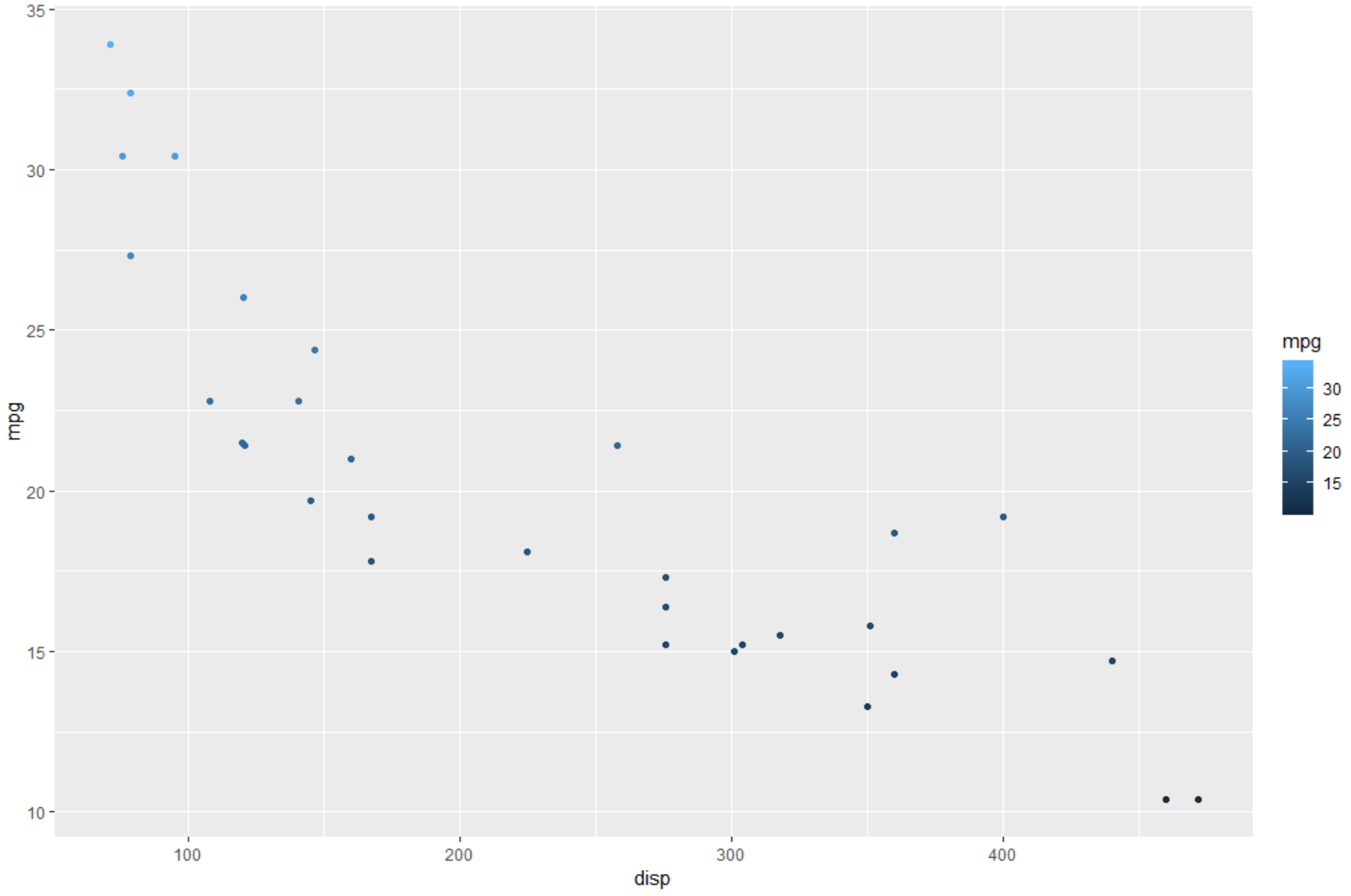
> ggplot(mtcars, aes(x = disp, y = mpg, colour = **as.factor**(cyl)))
+ geom_point()



```
> ggplot(mtcars, aes(x = disp, y = mpg, colour = as.factor(cyl), size = wt))
                + geom_point()
```

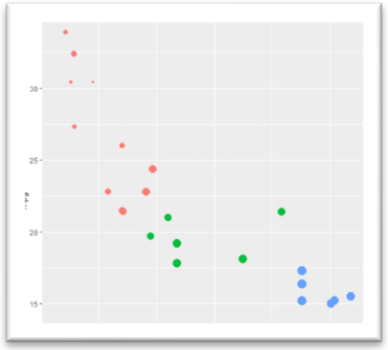


```
> ggplot(data = mtcars, aes(x = disp, y = mpg, colour = mpg))
      + geom_point()
```



 # double-encoding

Sumário



1 Começando pelo final: um exemplo de visualização no ggplot2

2 R: Introdução básica

3 ggplot2: explorando outros recursos e visualizações

How can I get R?

- Windows & Mac OS X
  - Download binaries from https://cran.r-project.org/
- Linux
  - Download binaries or use package manager *(sudo apt-get install r-base)*

- Recommendation: RStudio (www.rstudio.com)
  - IDE for R with many useful features (console, syntax-highlighting, plotting, etc.)

Getting help:

- ?functionname

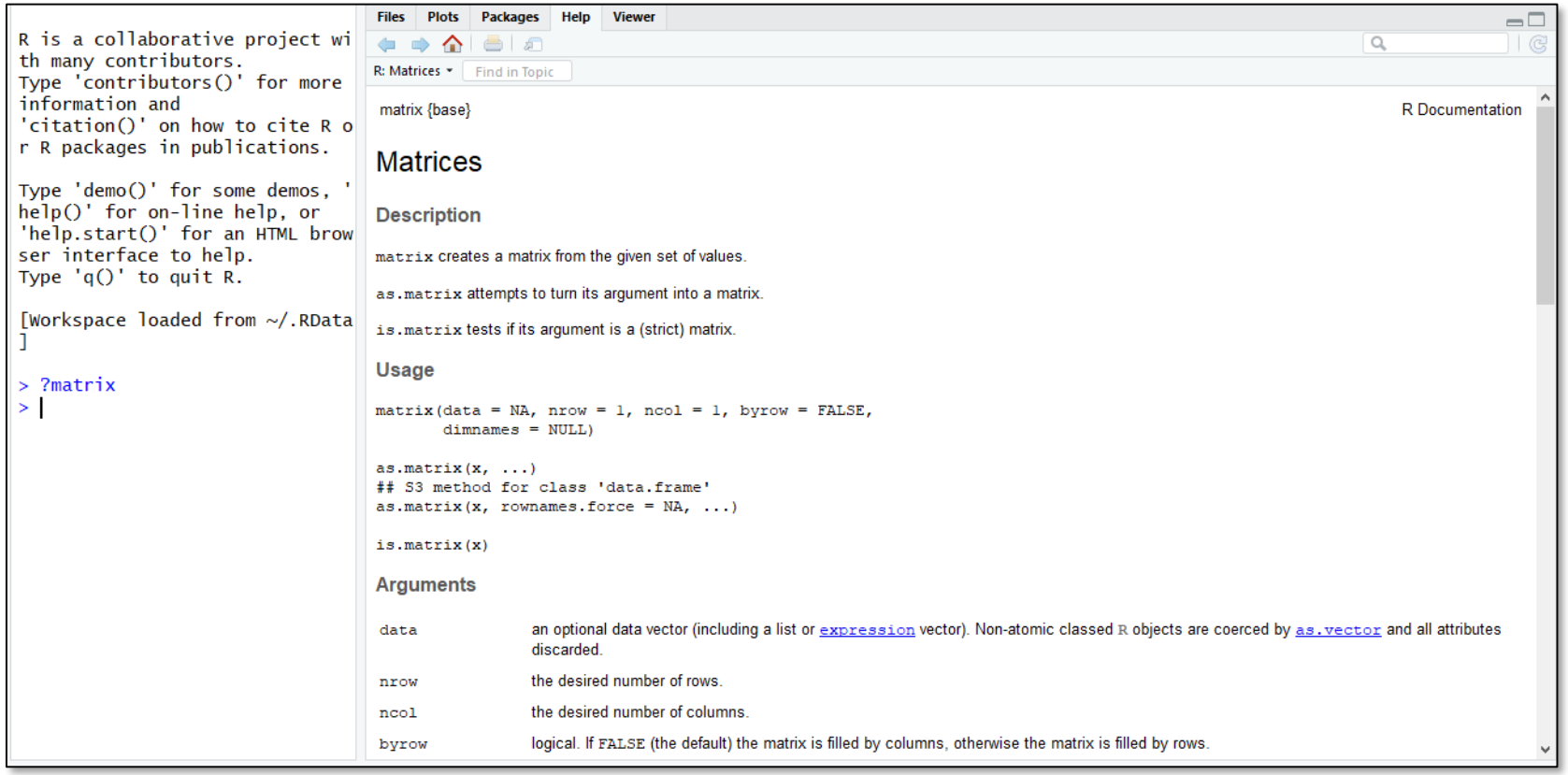
OR

- help(functionname)

> ?mean

> ?matrix

> help(read.table)



R Introduction

FGV EMAp

Assignment Operators:

- (<- or =)

```
> x <- 4  #assign value 4 to variable x
> x = 4   #assign value 4 to variable x
```

FGV EMAp

Calculations:

```
> x <- 4 * 5 - 5^2 # result is stored in variable x
```

Remove object:

```
> remove(x)
> remove(list = ls())
```

FGV EMAp

Read data to work with:

- Get your working directory:

```
> getwd()
```

- Set a new working directory:

```
> setwd(…)
```

FGV EMAp

Read data to work with:

- Example:

```
Degree Rank Sex Year YSdeg Salary
1 3 0 25 35 36350
1 3 0 13 22 35350
1 3 0 10 23 28200
1 3 1 7 27 26775
0 3 0 19 30 33696
1 3 0 16 21 28516
0 3 1 0 32 24900
1 3 0 16 18 31909
0 3 0 13 30 31850
0 3 0 13 31 32850
1 3 0 12 22 27025
1 2 0 15 19 24750
1 3 0 9 17 28200
0 2 0 9 27 23712
1 3 0 9 24 25748
1 3 0 7 15 29342
1 3 0 13 20 31114
0 2 0 11 14 24742
0 2 0 10 15 22906
0 3 0 6 21 24450
0 1 0 16 23 19175
0 2 0 8 31 20525
1 3 0 7 13 27959
1 3 1 8 24 38045
```

```
> salary <- read.table("salary.txt", header=TRUE)
# Read the file salary; assume the data contains header information
```

FGV EMAp

Read data to work with:

- Example:

```
Degree;Rank;Sex;Year;YSdeg;Salary
1;3;0;25;35;36350
1;3;0;13;22;35350
1;3;0;10;23;28200
1;3;1;7;27;26775
0;3;0;19;30;33696
1;3;0;16;21;28516
0;3;1;0;32;24900
1;3;0;16;18;31909
0;3;0;13;30;31850
0;3;0;13;31;32850
1;3;0;12;22;27025
1;2;0;15;19;24750
1;3;0;9;17;28200
0;2;0;9;27;23712
1;3;0;9;24;25748
1;3;0;7;15;29342
1;3;0;13;20;31114
0;2;0;11;14;24742
0;2;0;10;15;22906
0;3;0;6;21;24450
0;1;0;16;23;19175
0;2;0;8;31;20525
1;3;0;7;13;27959
1;3;1;8;24;38045
```

```
> salary <- read.table("salary-semicolon.txt", header=TRUE, sep = ";")
# Read the file salary; assume the data contains header information
and the semi-colon to separate columns
```

FGV EMAp

Read data to work with:

- Example:

```
1;3;0;25;35;36350
1;3;0;13;22;35350
1;3;0;10;23;28200
1;3;1;7;27;26775
0;3;0;19;30;33696
1;3;0;16;21;28516
0;3;1;0;32;24900
1;3;0;16;18;31909
0;3;0;13;30;31850
0;3;0;13;31;32850
1;3;0;12;22;27025
1;2;0;15;19;24750
1;3;0;9;17;28200
0;2;0;9;27;23712
1;3;0;9;24;25748
1;3;0;7;15;29342
1;3;0;13;20;31114
0;2;0;11;14;24742
0;2;0;10;15;22906
0;3;0;6;21;24450
0;1;0;16;23;19175
0;2;0;8;31;20525
1;3;0;7;13;27959
```

```
> salary <- read.table("salary-noheader.txt", header=FALSE, sep = ";")
# Read the file salary; assume the data contains NO header information
> names(salary) <- c("Degree","Rank","Sex","Year","YSdeg","Salary")
```

FGV EMAp

Read data to work with:

- Online:

```
> data <- read.table ("http://faculty.washington.edu/tlumley/data/FLvote.dat", header=TRUE)
# Read some online available data
```

FGV EMAp

Write data to local storage:

- Example:

```
> write.table(data, file = "data.txt", sep=";")
```

Data Structures:

- **Vectors**: either numerical, logical, or character
- **Matrices**: multidimensional generalization of vectors.
- **Lists**: general form of vectors, can contain various data types.
- **Data frame**: matrix-like structure; Columns can be of different types
- **Factors**: provide compact ways to handle categorical data

FGV EMAp

Creating Vectors:

Use the function c (combine or concatenate)

```
> c(1,4,3,2)                                  # numeric
> c("hallo", "servus", "guten Tag")           # character
> c(TRUE, FALSE, FALSE, TRUE)                 # logical
```

However, this also works:

```
> c(1, 4, TRUE, FALSE)
> c(1, 5.4, TRUE, "hello")
```

Vectors

FGV EMAp

Creating Vectors:

Sequences and repetitions

```
> 1:15                    #seq(1, 15)
> seq(1, 15, 2)           #seq(from, to, by)
> rep(1:3, each=2)        #1 1 2 2 3 3
> rep(1:3, times=2)       #1 2 3 1 2 3
```

Vectors

FGV EMAp

Creating Vectors:

Sequences and repetitions

```
> 3:4 - 1:6                 # 2  2  0  0 -2 -2

> rev(1:4)                  #4 3 2 1
> rep(1:3, each=2)          #1 1 2 2 3 3
> rep(1:3, times=2)         #1 2 3 1 2 3
```

Vectors

FGV EMAp

Operations on Vectors:

- +, -, *, /, ^ (are performed for each element)
- log, exp, sin, cos, tan, sqrt
- max(x), min(x)
- range(x) (= c(min(x), max(x))
- length(x)
- sum(x)
- prod(x)
- sort(x)

Vectors

FGV EMAp

Creating a Matrix:

matrix(data, nrow, ncol, byrow = F)

```
> matrix(1:6, nrow =2)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(1:6, nrow =2, byrow  = TRUE)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Matrices

FGV EMAp

Creating a Matrix:

matrix(data, nrow, ncol, byrow = F)

```
> matrix(1:10, nrow = 4, ncol = 4)
     [,1] [,2] [,3] [,4]
[1,]    1    5    9    3
[2,]    2    6   10    4
[3,]    3    7    1    5
[4,]    4    8    2    6
```

There will be a warning, however, the matrix will be created anyway.

Matrices

FGV EMAp

Creating a Matrix:

Can make use of existing vectors

```
> x <- c(1:4)
> y <- c(2:5)
> z <- c(3:6)
> cbind(x, y, z)        # bind by columns
> rbind(x, y, z)        # bind by rows
```

Matrices

FGV EMAp

Creating Lists:

Lists are like vectors with elements of various data types. Each element can have a different data type.

```
> x<-list("a", 1, TRUE)
[[1]]
[1]       "a”
[[2]]
[1]       1
[[3]]
[1]       TRUE
```

Lists

FGV EMAp

Creating Lists:

Lists are like vectors with elements of various data types. Each element can have a different data type.

```
> x<-list(c("a","b","c"), c(1,3,4), TRUE)
[[1]]
[1]         "a" "b" "c”
[[2]]
[1]         1 3 4
[[3]]
[1]         TRUE
```

Lists

FGV EMAp

Creating Lists:

Lists are like vectors with elements of various data types. Each element can have a different data type.

```
> x<-list(c("a","b","c"), c(1,3,4), TRUE)
> mean(x[[2]])  # 2.666667
> median(x[[1]])  # “b”
```

Lists

FGV EMAp

Creating Data Frames:

Matrices whose columns may have different data types.

```
> shoppinglist<-data.frame(
        product=c("water", "ham", "cheese", "milk", "beer", "apple", "orange", "steak"),
        department=c("drinks", "meat", "milk products", "milk products", "drinks", "fruit",
                "fruit", "meat" ),
        amount=c(10,2,1,5,12,6,6,2))
```

	product	department	amount
1	water	drinks	10
2	ham	meat	2
3	...		

Data Frames

FGV EMAp

Selecting Single Data Entries:

Vector:

```
> data <- c(1:10)
> data[2]          # 2
> data[-2]         # 1  3  4  5  6  7  8  9 10 (all except the 2nd value)
> data[c(1,5)]     # 1  5
> data[3:6]        # 3  4  5  6
> data[c(1:3, 9)]# 1  2  3  9
```

Data Access

FGV EMAp

Selecting Single Data Entries:

Lists:

```
                                                    # giving names to list elements
> data <- list(name="Fred", wife="Mary", noChildren =3, childAges=c(4,7,9))
> data$name              # “Fred”
> data$childAges         # 4  7  9
> data$childAges[1]      # 4
```

FGV EMAp

R – data():

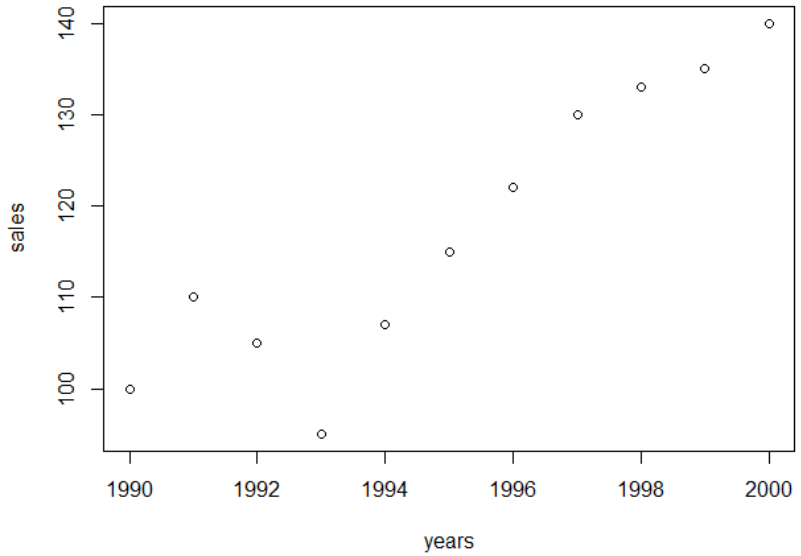
R already comes with some nice data tables.

```
> data()
> data(AirPassengers)          # load data
> AirPassengers                # show data


>  plot(AirPassengers)
```

FGV EMAp

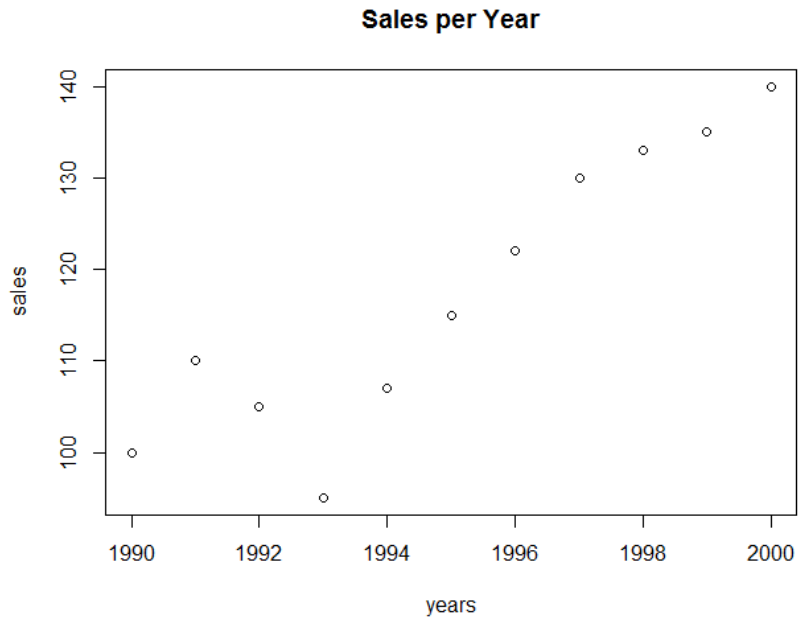
```
> years <- c(1990:2000)
> sales <- c(100, 110, 105, 95, 107, 115, 122, 130, 133, 135, 140)
> plot(years, sales)
```



Visualizing Data

FGV EMAp

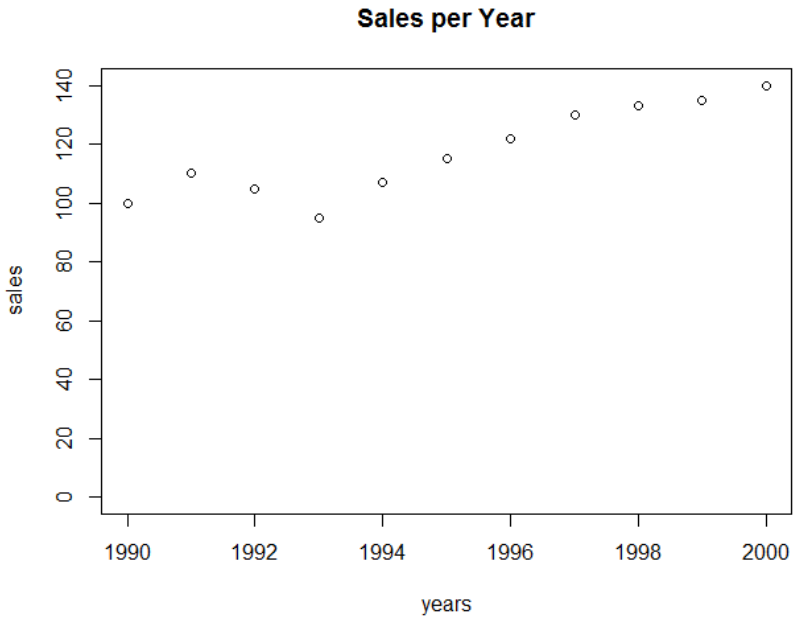
```
> years <- c(1990:2000)
> sales <- c(100, 110, 105, 95, 107, 115, 122, 130, 133, 135, 140)
> plot(years, sales, main="Sales per Year")
```



Visualizing Data

FGV EMAp

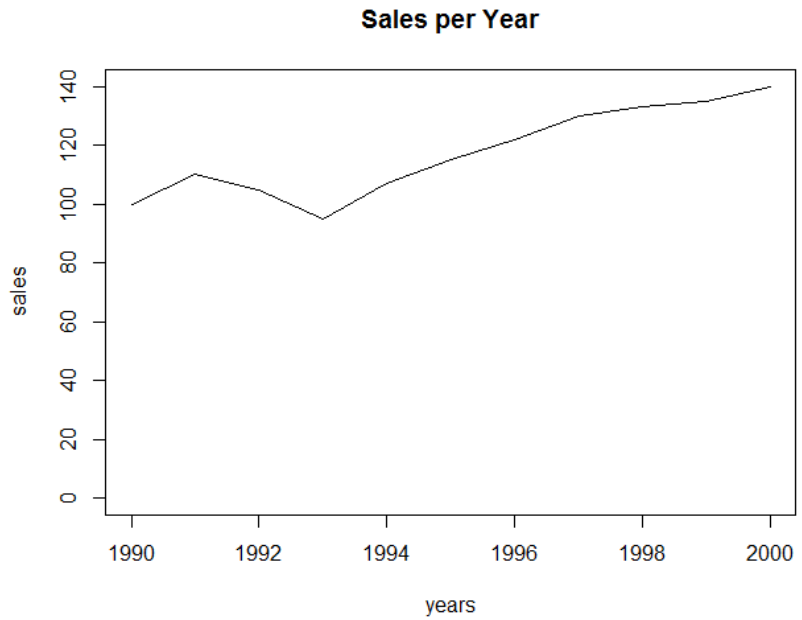
```
> years <- c(1990:2000)
> sales <- c(100, 110, 105, 95, 107, 115, 122, 130, 133, 135, 140)
> plot(years, sales, main="Sales per Year", ylim=c(0, max(sales)))
```



Visualizing Data

FGV EMAp

```
> years <- c(1990:2000)
> sales <- c(100, 110, 105, 95, 107, 115, 122, 130, 133, 135, 140)
> plot(years, sales, main="Sales per Year", ylim = c(0, max(sales)), type = "l")
```



Visualizing Data

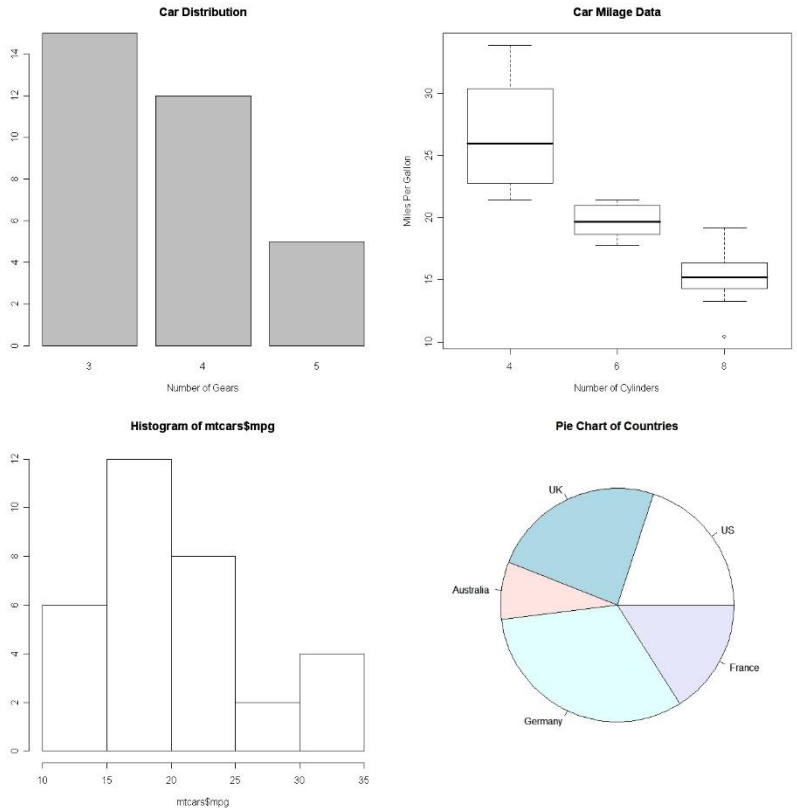
Many different plot types available

> barplot()

> boxplot()

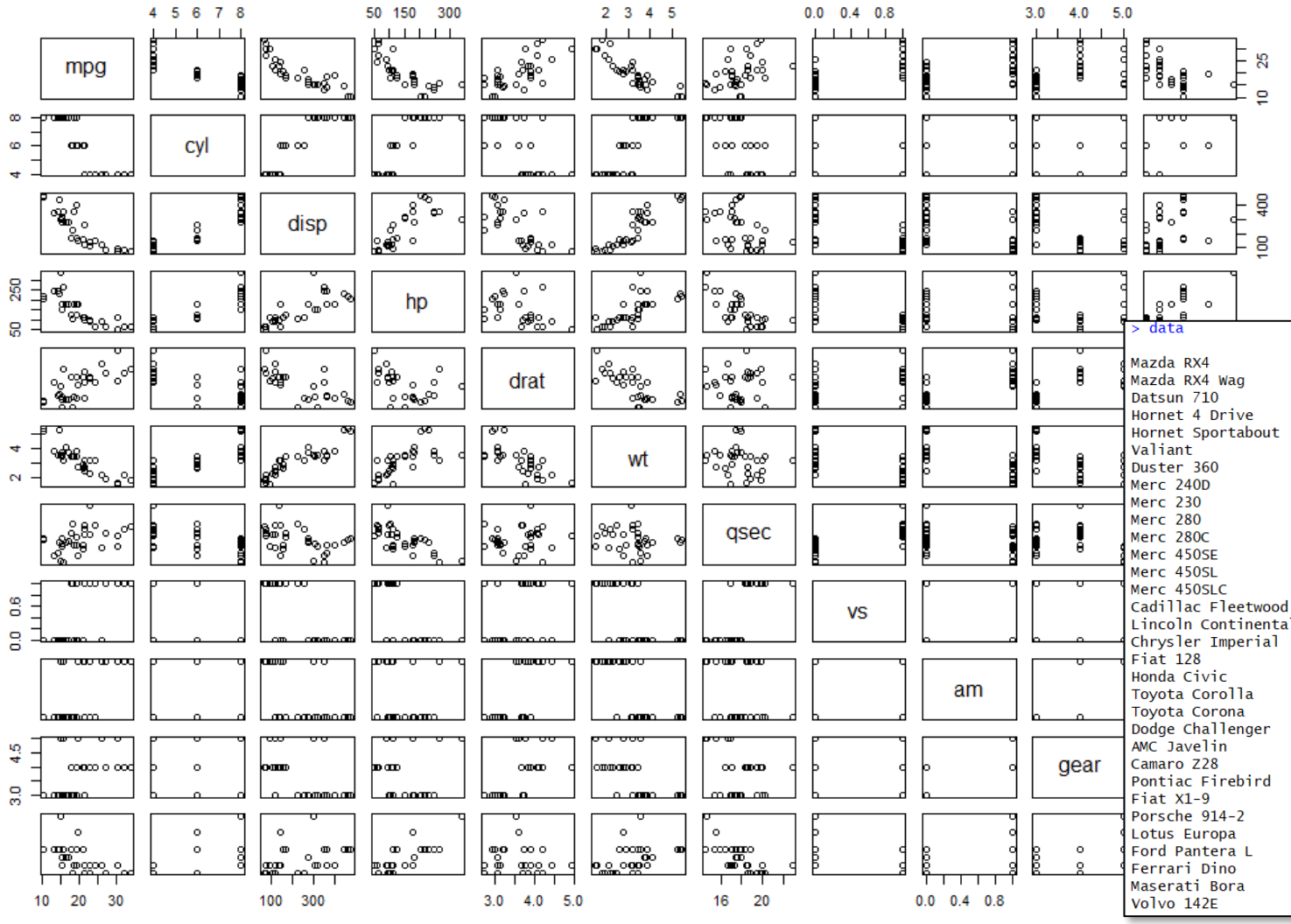
> hist()

> pie()



Visualizing Data

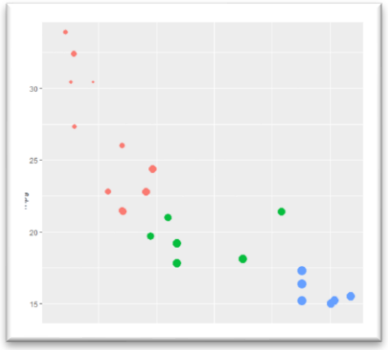
> plot(data)




```
> data
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Visualizing Data

Sumário



1 Começando pelo final: um exemplo de visualização no ggplot2

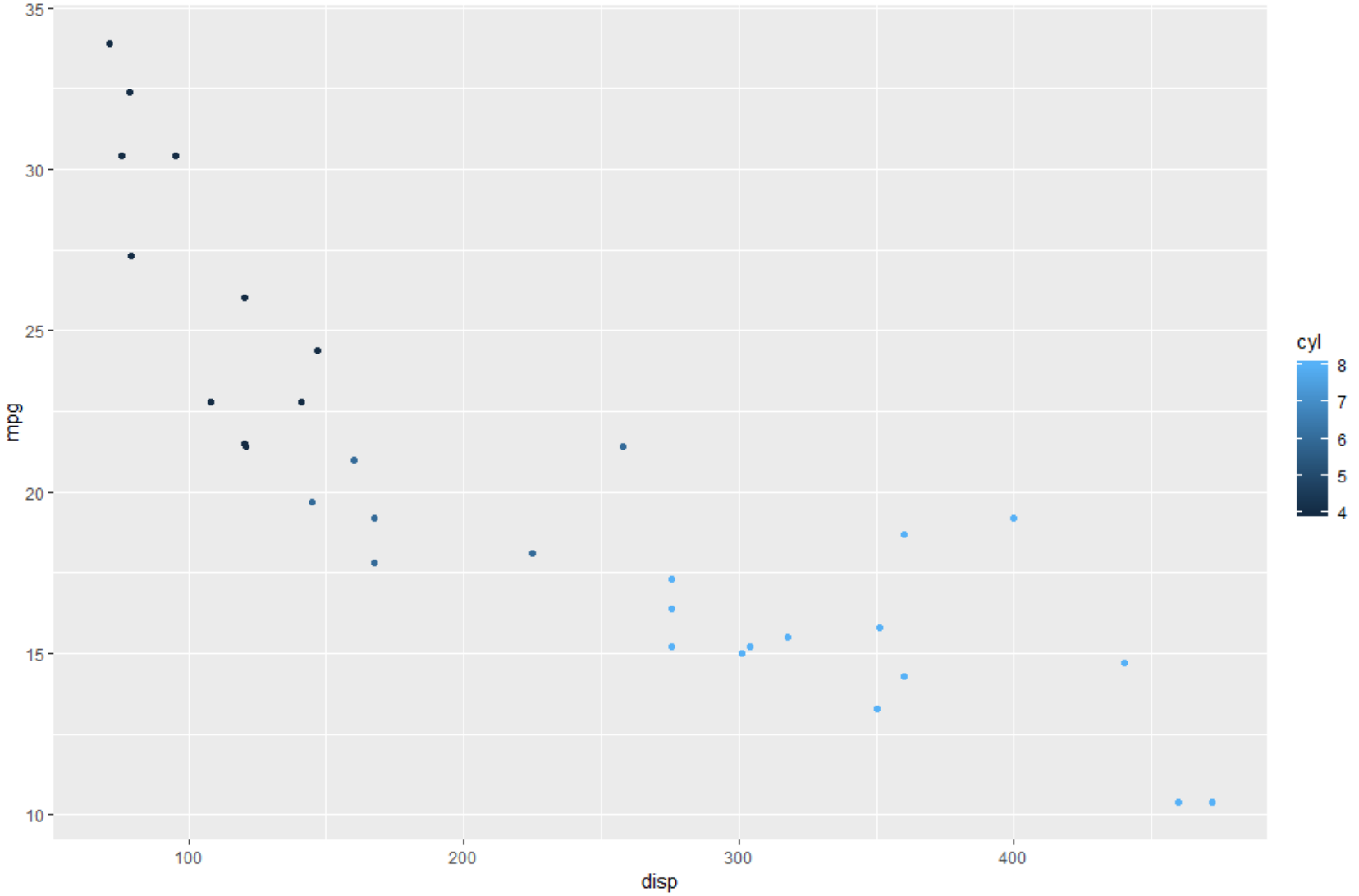
2 R: Introdução básica

3 ggplot2: explorando outros recursos e visualizações

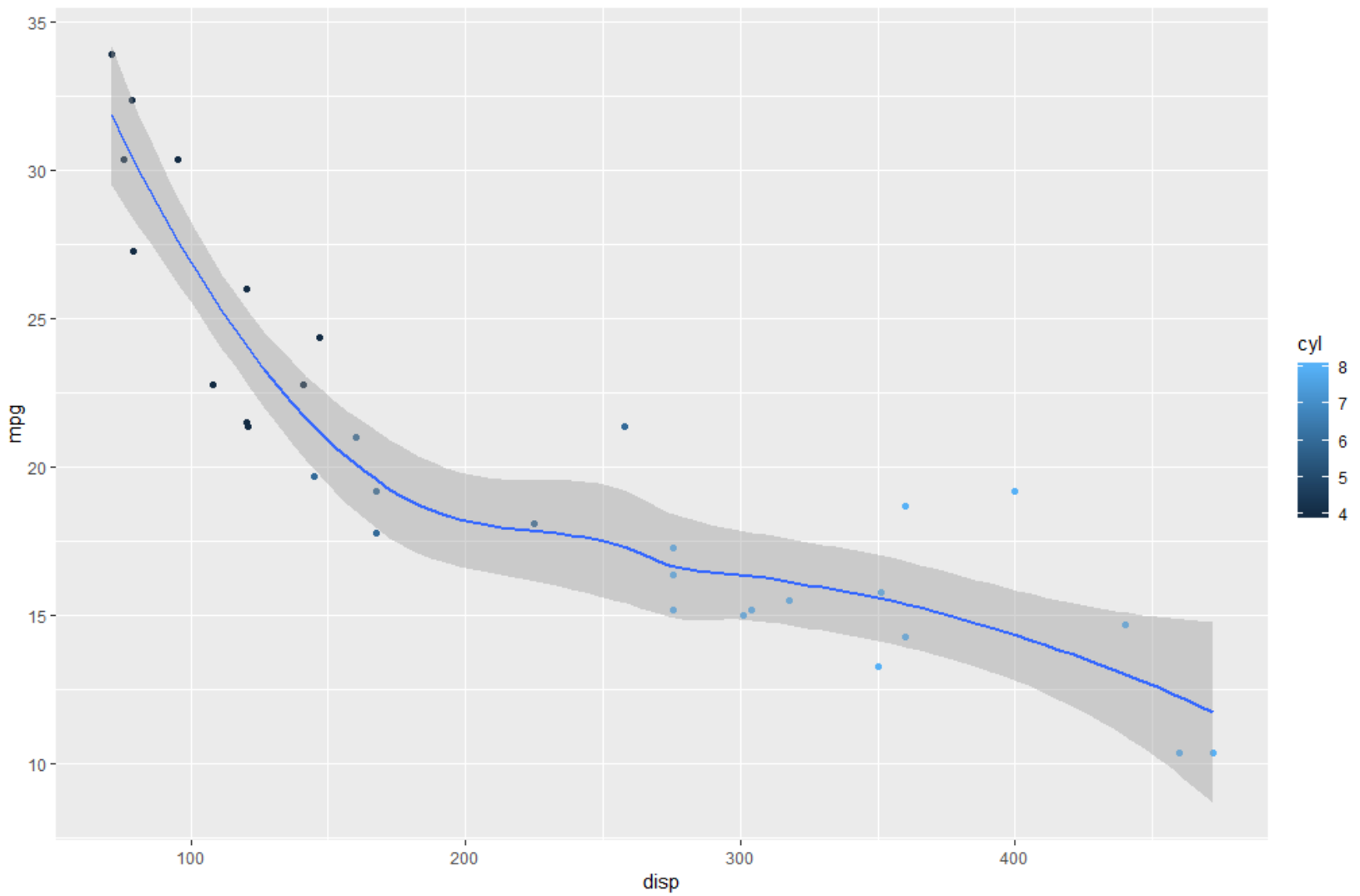
```
> library(ggplot2)
```

```
> install.packages("ggplot2")
```

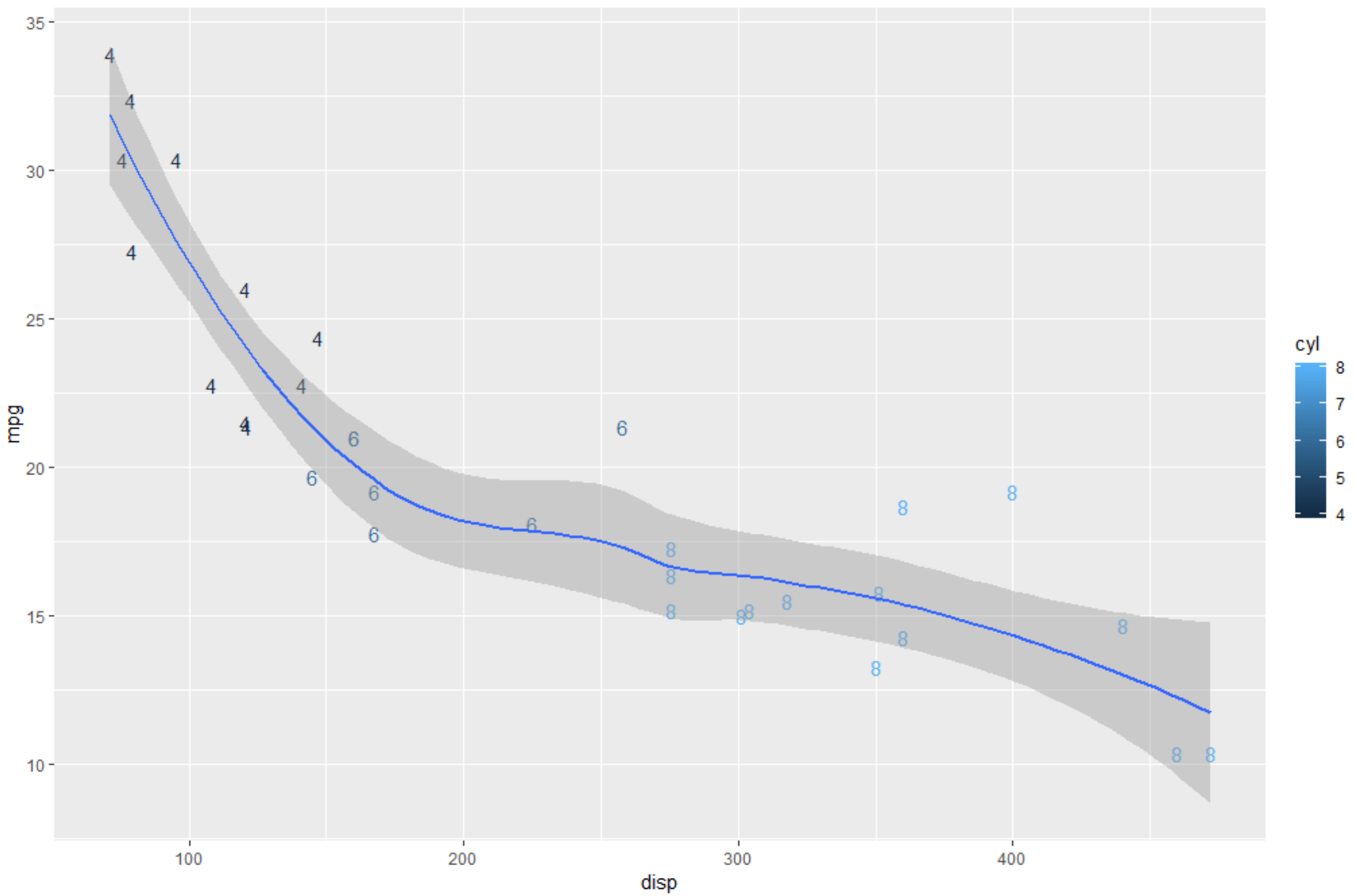
> ggplot(mtcars, aes(x = disp, y = mpg, **colour = cyl**)) + geom_point()



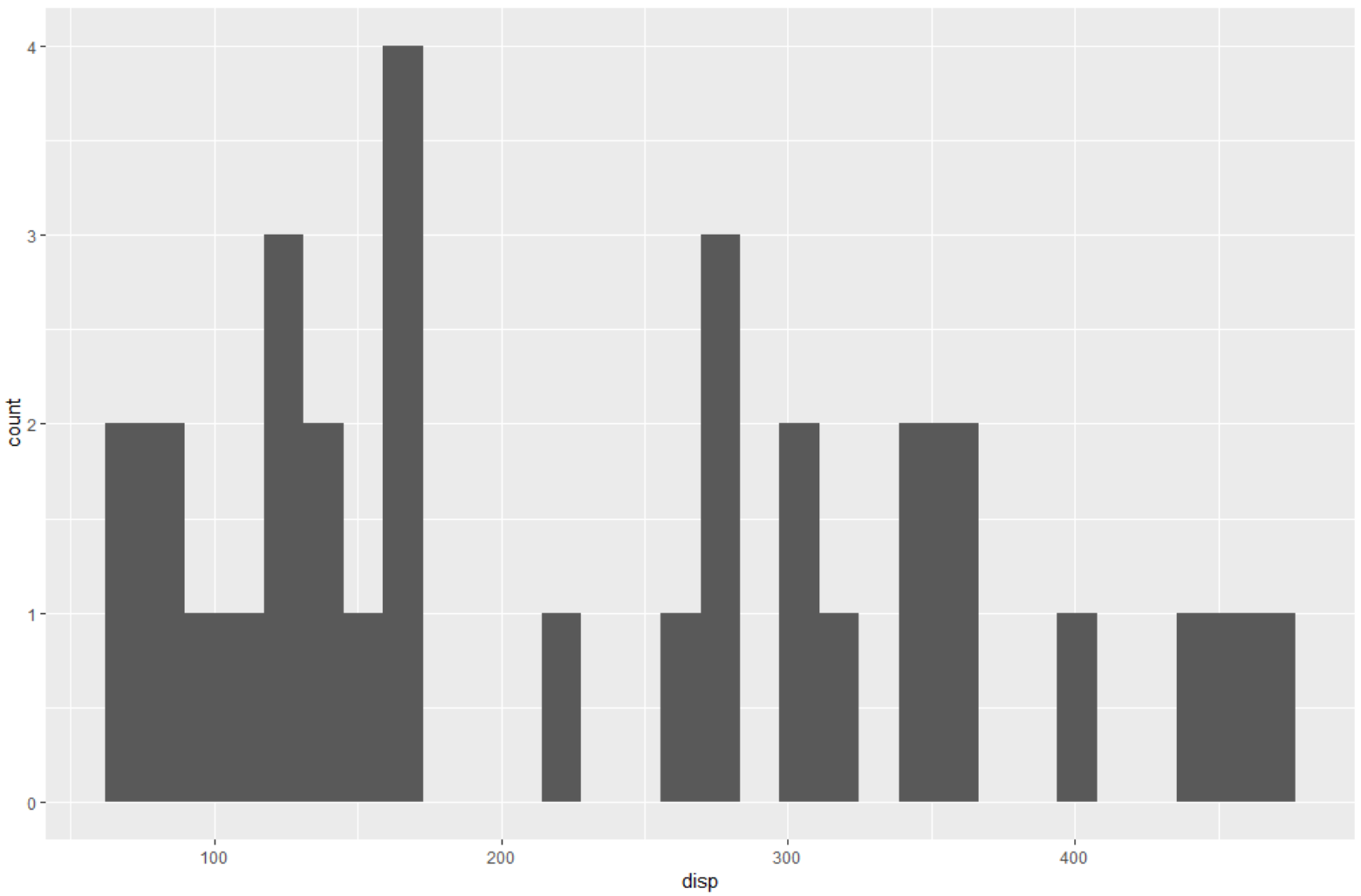
> ggplot(mtcars, aes(x = disp, y = mpg, **colour = cyl**)) + geom_point()
+ geom_smooth()



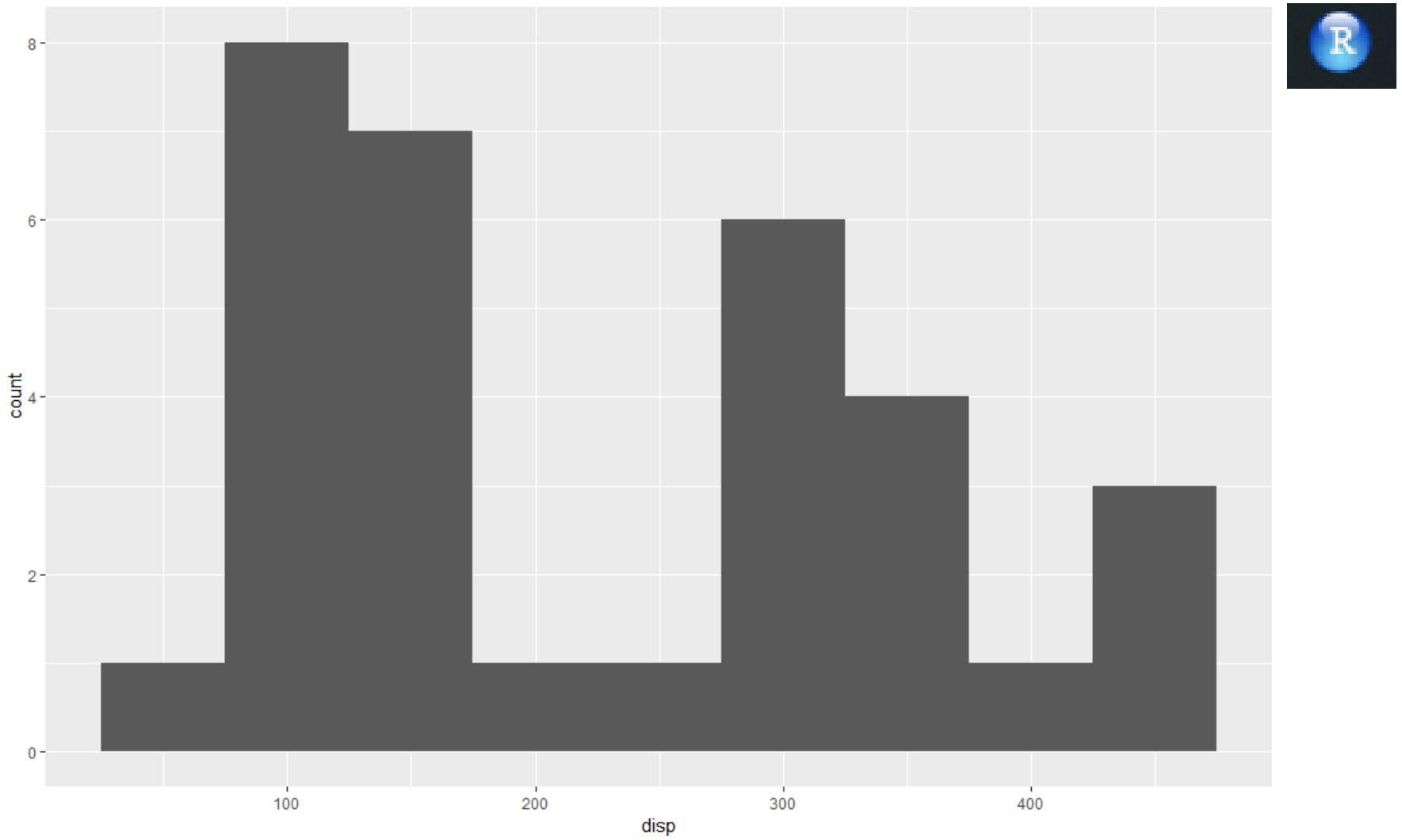
> ggplot(mtcars, aes(x = disp, y = mpg, **colour = cyl**))
+ **geom_text(aes(label=cyl))** + geom_smooth()



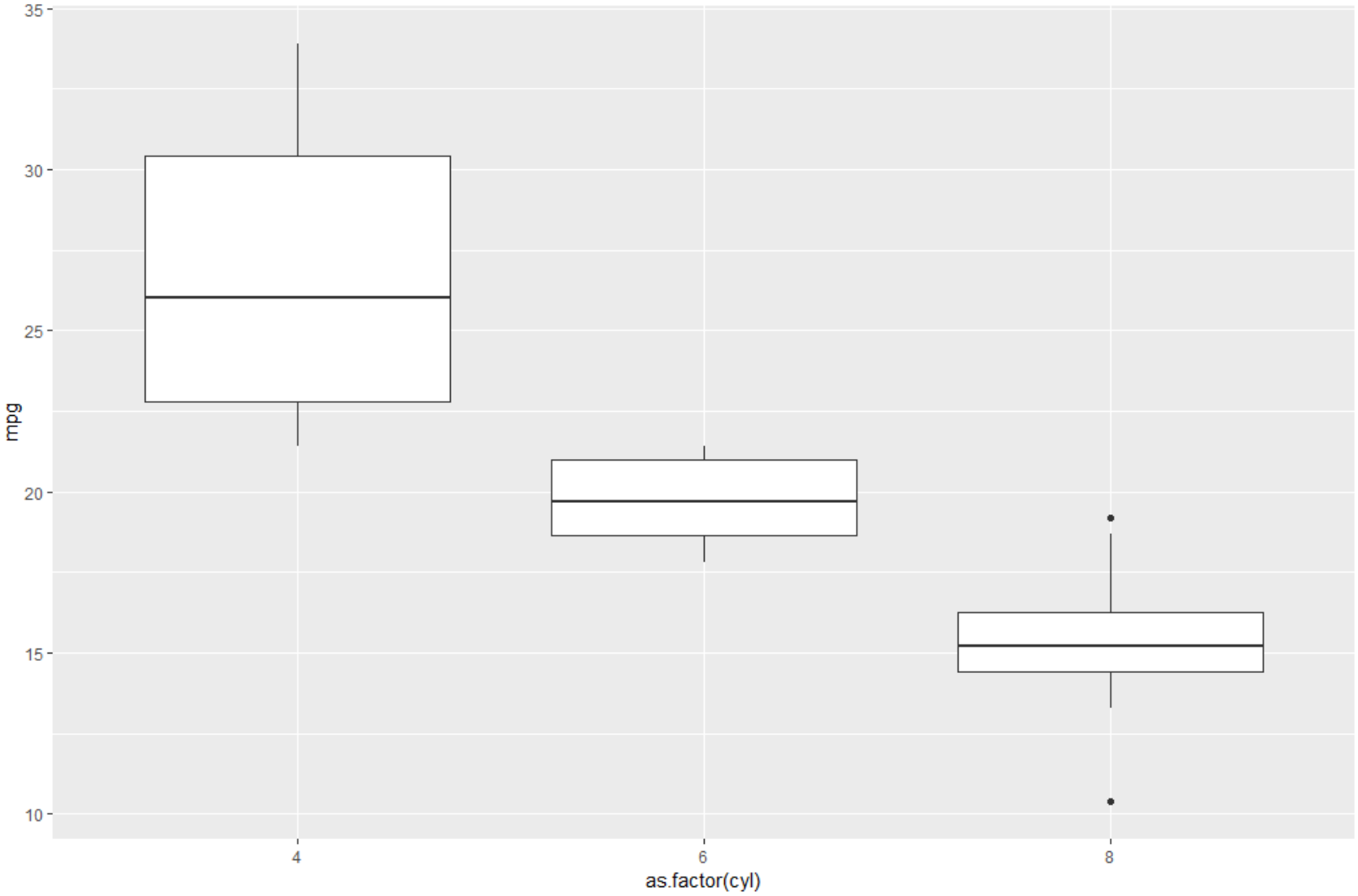
> ggplot(mtcars, aes(x = disp)) + **geom_histogram()**



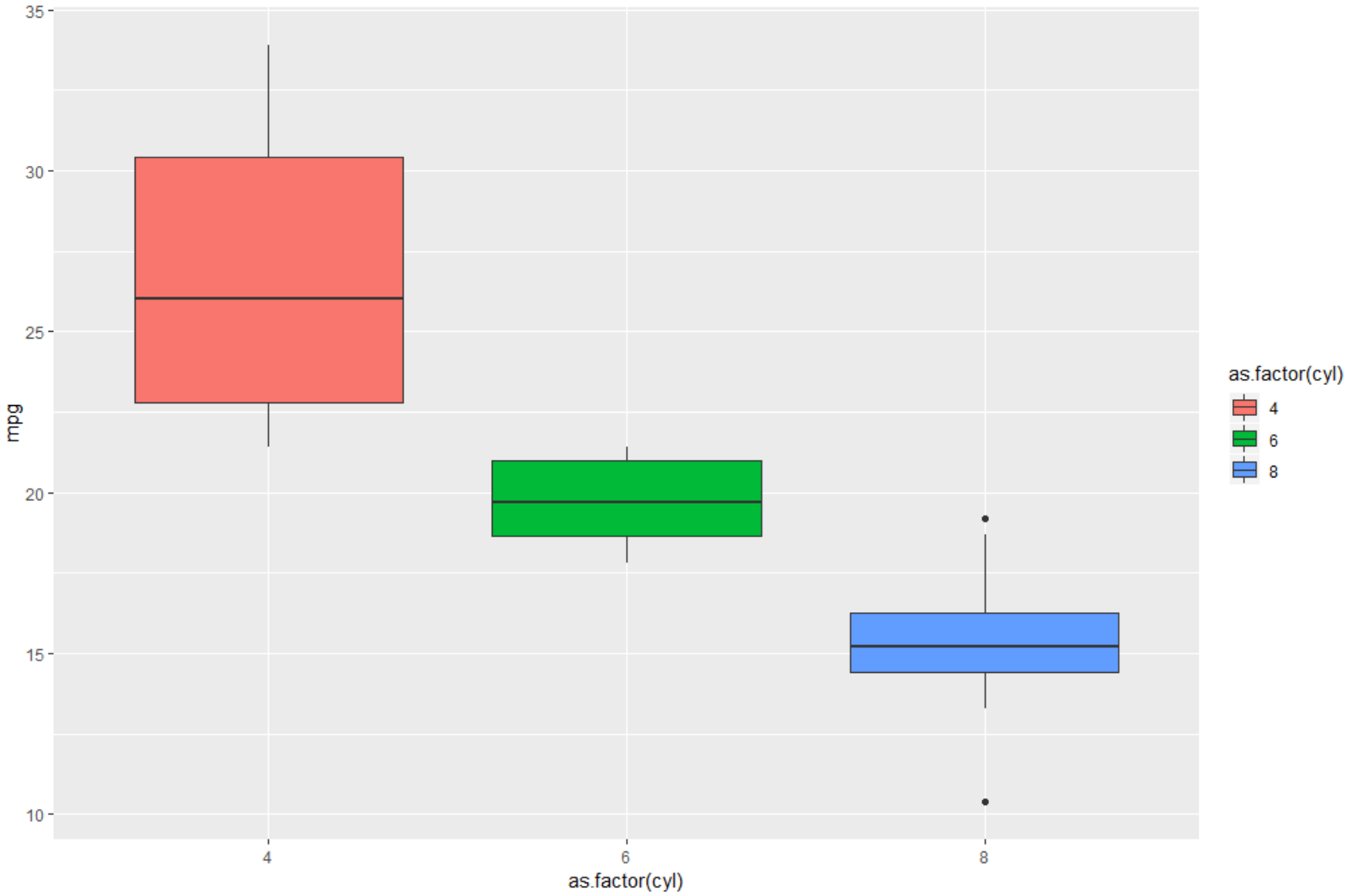
> ggplot(mtcars, aes(x = disp)) + geom_histogram**(binwidth = 50)**



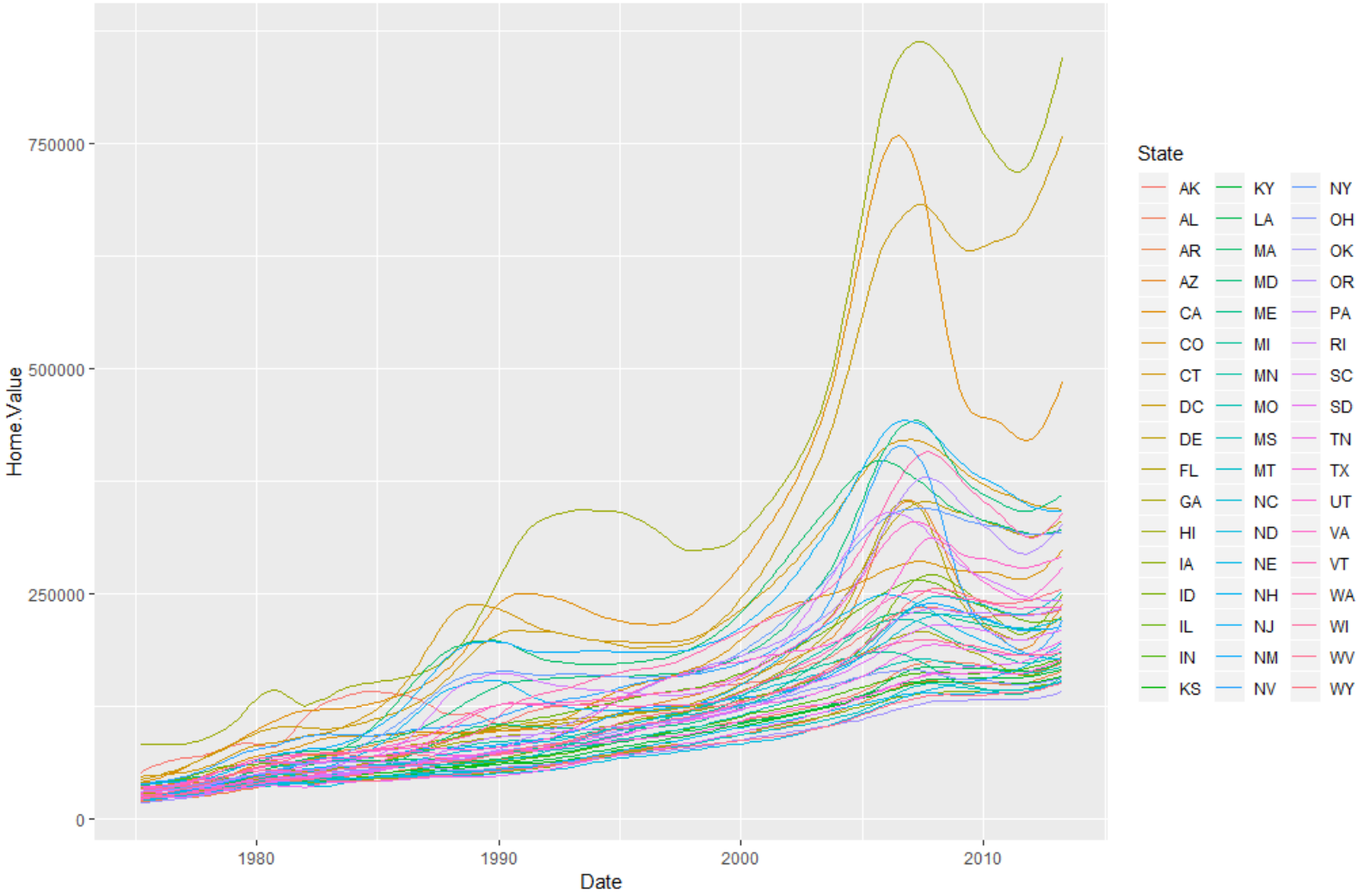
> ggplot(mtcars, aes(x = as.factor(cyl), y = mpg))
+ **geom_boxplot()**



> ggplot(mtcars, aes(x = as.factor(cyl), y = mpg, **fill = as.factor(cyl)**))
+ geom_boxplot()

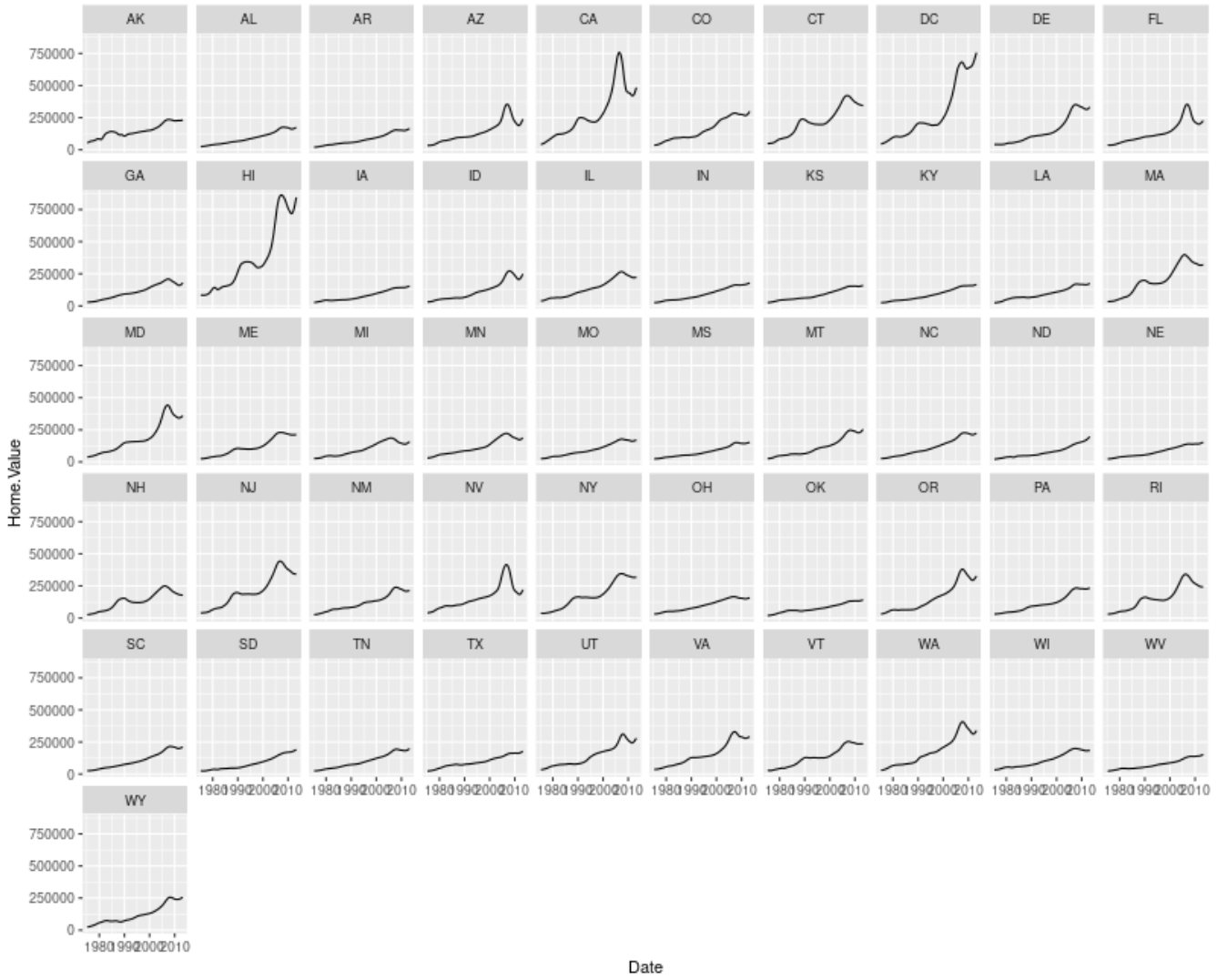


```
> p5 <- ggplot(housing, aes(x = Date, y = Home.Value))
  p5 + geom_line(aes(color = State))
```



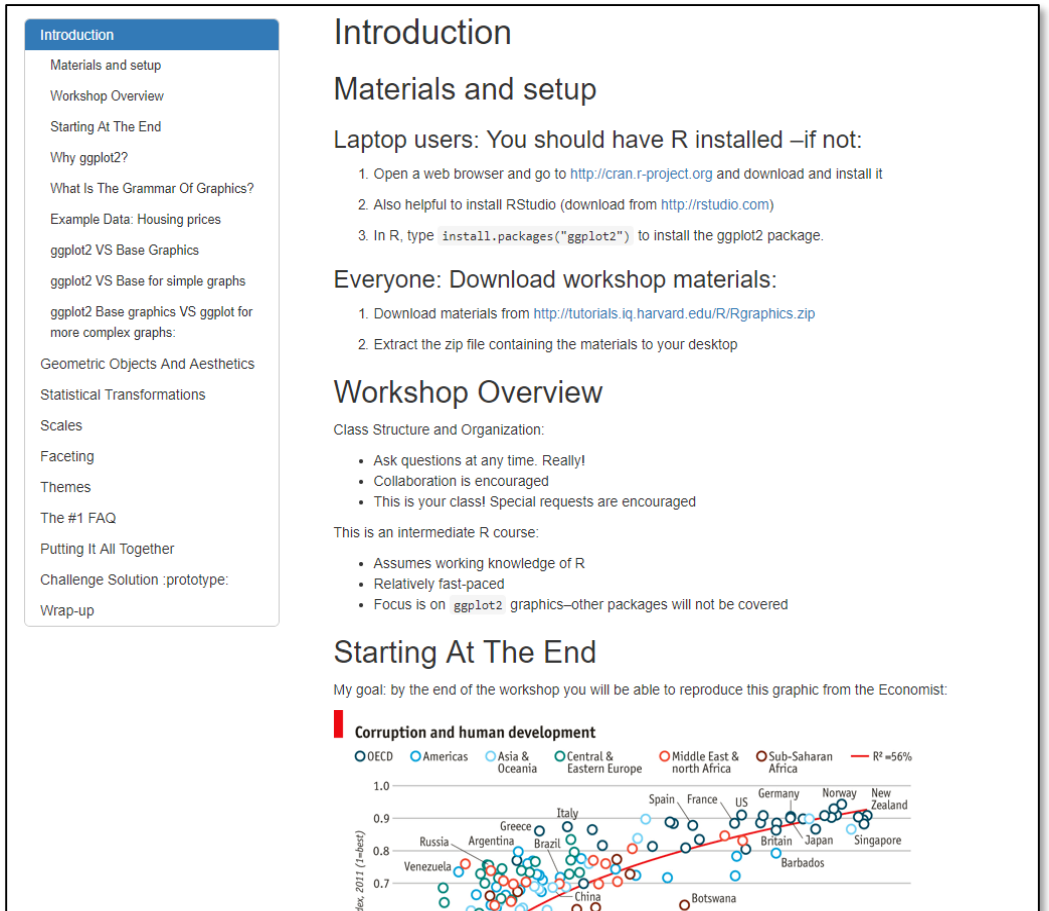
Facets

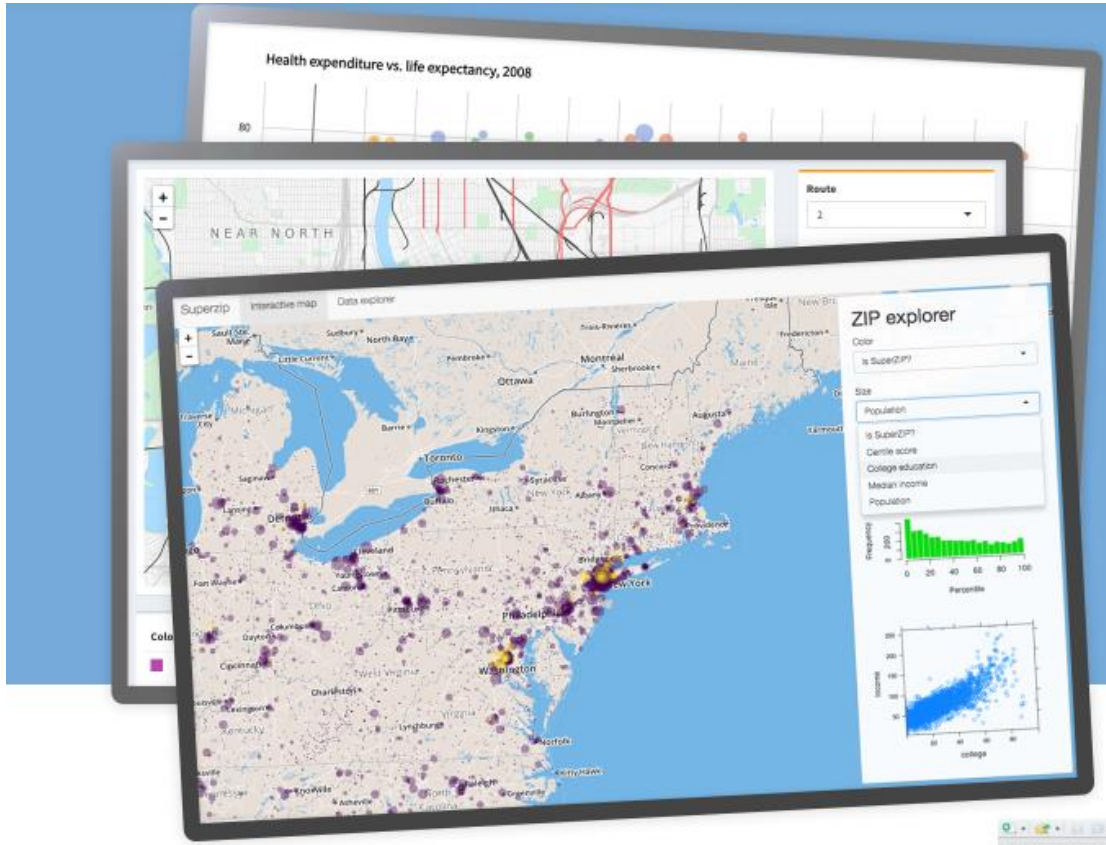
> p5 + geom_line() + **facet_wrap(~State, ncol = 10)**



Referência

https://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

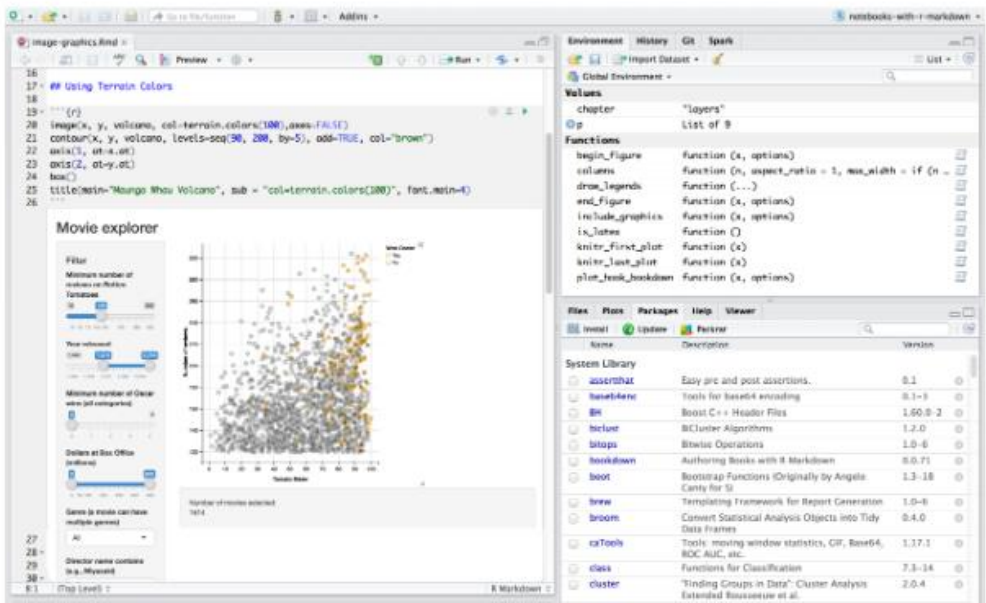




Interact. Analyze. Communicate.

Take a fresh, interactive approach to telling your data story with Shiny. Let users interact with your data and your analysis. And do it all with R.

Shiny is an R package that makes it easy to build interactive web apps straight from R. You can host standalone apps on a webpage or embed them in R Markdown documents or build dashboards. You can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions.



Muito obrigado pela atenção...!

01 / 10 (segunda-feira) **08 / 10**