

Learning R packages

Cristian Villegas

2023-05-11

Contents

1	Intro	5
1.1	Carrega pacotes a serem usados	5
1.2	Alguns atalhos no Rstudio	5
2	dplyr (60 minutos)	7
2.1	Carrega pacotes a serem usados	7
2.2	Descrição dos dados mpg	7
2.3	Lista de funções do pacote dplyr	9
2.4	Operador Pipe	11
2.5	select() para colunas	11
2.6	rename()	15
2.7	mutate() para colunas	17
2.8	transmute()	18
2.9	filter() para linhas	19
2.10	slice() para linhas	24
2.11	arrange() para linhas	25
2.12	distinct() para linhas	28
2.13	summarise()	30
2.14	group_by()	31
2.15	count()	33
2.16	sample_n()	34
2.17	sample_frac()	35
3	ggplot2 (60 minutos)	37
3.1	Carrega pacotes a serem usados	37
3.2	Lista de funções do pacote ggplot2	38
3.3	Primeiros passos usando geom_point	38
3.4	smooth, boxplot, histogram	55
3.5	gridExtra e patchwork	59
3.6	bar, col, density, density2d	66
3.7	facet_grid, facet_wrap	73
3.8	stat_function	80
3.9	stat_summary	80

3.10	theme_*()	81
3.11	Gráfico de perfis (Spagueti plot)	82
3.12	plotly	84
3.13	esquisse	84
3.14	Exemplo esquisse	85
4	purrr	87
4.1	Apply a function to each element of a list or atomic vector	88
4.2	Examples	90
4.3	map functions	100
4.4	map2 functions	101
5	tidyr	103
5.1	nest()	103
5.2	unnest()	108
5.3	Exemplos da ajuda do R	108
5.4	Mais detalhes	113
6	broom and dplyr	115
6.1	Tidy bootstrapping	122
6.2	links	129

Chapter 1

Intro

1.1 Carrega pacotes a serem usados

```
#install.packages("tidyverse")
#install.packages("dplyr")
#install.packages("tidyr")
#install.packages("ggplot2")

library(tidyverse)
# Manipulação de dados
#library(dplyr)

# Visualização de gráficos
library(ggplot2)
library(gridExtra)
library(patchwork)
library(plotly)
library(esquisse)

# Para dados gráfico de perfis
library(nlme)
```

Ver como citar referências Wickham et al. [2019], Wickham [2023], Wickham et al. [2023c], Wickham et al. [2023a], Wickham and Henry [2023], Wickham et al. [2023b], Xie [2023b], Xie [2023a]

1.2 Alguns atalhos no Rstudio

Para considerar

Operador Pipe (%>%): Ctrl + Shift + M (Windows) ou Cmd + Shift + M (Mac).

Criar novos chunks: Ctrl + Alt + I (Windows) ou Cmd + Option + I (Mac).

Chapter 2

dplyr (60 minutos)

2.1 Carrega pacotes a serem usados

```
#install.packages("tidyverse")
#install.packages("dplyr")
#install.packages("tidyr")
#install.packages("ggplot2")

library(tidyverse)
# Manipulação de dados
#library(dplyr)

# Visualização de gráficos
library(ggplot2)
library(gridExtra)
library(patchwork)
library(plotly)
library(esquisse)

# Para dados gráfico de perfis
library(nlme)
```

2.2 Descrição dos dados mpg

Dados de economia de combustível de 1999 a 2008 para *38 modelos populares de carros*. Este conjunto de dados contém um subconjunto dos dados de economia de combustível que a EPA disponibiliza em <https://fuelconomy.gov/>. Ele contém apenas modelos que tiveram um novo lançamento a cada ano entre 1999 e 2008 - isso foi usado como um substituto para a popularidade do carro. Um

data frame com 234 linhas e 11 variáveis:

- *manufacturer* nome do fabricante
- *model* nome do modelo
- *displ* cilindrada do motor, em litros
- *year* ano de fabricação
- *cyl* número de cilindros
- *trans* tipo de transmissão
- *drv* o tipo de trem de força, onde **f** = **tração dianteira**, **r** = **tração traseira** e **4** = **4wd**
- *cty* milhas urbanas por galão
- *hwy* milhas rodoviárias por galão
- *fl* tipo de combustível
- *class* “tipo” de carro

```
#help("mpg")
```

```
library(tidyverse)
```

```
dados <- mpg
```

```
glimpse(dados)
```

```
## Rows: 234
```

```
## Columns: 11
```

```
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
```

```
## $ model <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
```

```
## $ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
```

```
## $ year <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
```

```
## $ cyl <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
```

```
## $ trans <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
```

```
## $ drv <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
```

```
## $ cty <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
```

```
## $ hwy <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
```

```
## $ fl <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
```

```
## $ class <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

```
dados <- mutate(.data = dados,
```

```
  across(where(is.character),
```

```
  as.factor))
```

```
#View(df)
```

```
glimpse(dados)
```

```
## Rows: 234
```

```
## Columns: 11
```


## [88] "distinct_if"	"distinct_prepare"	"do"
## [91] "do_"	"dplyr_col_modify"	"dplyr_reconstruct"
## [94] "dplyr_row_slice"	"ends_with"	"enexpr"
## [97] "enexprs"	"enquo"	"enquos"
## [100] "ensym"	"ensyms"	"eval_tbls"
## [103] "eval_tbls2"	"everything"	"explain"
## [106] "expr"	"failwith"	"filter"
## [109] "filter_"	"filter_all"	"filter_at"
## [112] "filter_if"	"first"	"full_join"
## [115] "funs"	"funs_"	"glimpse"
## [118] "group_by"	"group_by_"	"group_by_all"
## [121] "group_by_at"	"group_by_drop_default"	"group_by_if"
## [124] "group_by_prepare"	"group_cols"	"group_data"
## [127] "group_indices"	"group_indices_"	"group_keys"
## [130] "group_map"	"group_modify"	"group_nest"
## [133] "group_rows"	"group_size"	"group_split"
## [136] "group_trim"	"group_vars"	"group_walk"
## [139] "grouped_df"	"groups"	"id"
## [142] "ident"	"if_all"	"if_any"
## [145] "if_else"	"inner_join"	"intersect"
## [148] "is_grouped_df"	"is.src"	"is.tbl"
## [151] "is_grouped_df"	"join_by"	"lag"
## [154] "last"	"last_col"	"last_dplyr_warnings"
## [157] "lead"	"left_join"	"location"
## [160] "lst"	"make_tbl"	"matches"
## [163] "min_rank"	"mutate"	"mutate_"
## [166] "mutate_all"	"mutate_at"	"mutate_each"
## [169] "mutate_each_"	"mutate_if"	"n"
## [172] "n_distinct"	"n_groups"	"na_if"
## [175] "near"	"nest_by"	"nest_join"
## [178] "new_grouped_df"	"new_rowwise_df"	"nth"
## [181] "ntile"	"num_range"	"one_of"
## [184] "order_by"	"percent_rank"	"pick"
## [187] "progress_estimated"	"pull"	"quo"
## [190] "quo_name"	"quos"	"recode"
## [193] "recode_factor"	"reframe"	"relocate"
## [196] "rename"	"rename_"	"rename_all"
## [199] "rename_at"	"rename_if"	"rename_vars"
## [202] "rename_vars_"	"rename_with"	"right_join"
## [205] "row_number"	"rows_append"	"rows_delete"
## [208] "rows_insert"	"rows_patch"	"rows_update"
## [211] "rows_upsert"	"rowwise"	"same_src"
## [214] "sample_frac"	"sample_n"	"select"
## [217] "select_"	"select_all"	"select_at"
## [220] "select_if"	"select_var"	"select_vars"
## [223] "select_vars_"	"semi_join"	"setdiff"

## [226] "setequal"	"show_query"	"slice"
## [229] "slice_"	"slice_head"	"slice_max"
## [232] "slice_min"	"slice_sample"	"slice_tail"
## [235] "sql"	"sql_escape_ident"	"sql_escape_string"
## [238] "sql_join"	"sql_select"	"sql_semi_join"
## [241] "sql_set_op"	"sql_subquery"	"sql_translate_env"
## [244] "src"	"src_df"	"src_local"
## [247] "src_mysql"	"src_postgres"	"src_sqlite"
## [250] "src_tbls"	"starts_with"	"starwars"
## [253] "storms"	"summarise"	"summarise_"
## [256] "summarise_all"	"summarise_at"	"summarise_each"
## [259] "summarise_each_"	"summarise_if"	"summarize"
## [262] "summarize_"	"summarize_all"	"summarize_at"
## [265] "summarize_each"	"summarize_each_"	"summarize_if"
## [268] "sym"	"syndiff"	"syms"
## [271] "tally"	"tally_"	"tbl"
## [274] "tbl_df"	"tbl_nongroup_vars"	"tbl_ptype"
## [277] "tbl_vars"	"tibble"	"top_frac"
## [280] "top_n"	"transmute"	"transmute_"
## [283] "transmute_all"	"transmute_at"	"transmute_if"
## [286] "tribble"	"type_sum"	"ungroup"
## [289] "union"	"union_all"	"validate_grouped_df"
## [292] "validate_rowwise_df"	"vars"	"where"
## [295] "with_groups"	"with_order"	"wrap_dbplyr_obj"

2.4 Operador Pipe

```
sqrt(log(44))
```

```
## [1] 1.945299
```

```
44 %>% log %>% sqrt
```

```
## [1] 1.945299
```

2.5 select() para columnas

```
select(dados, manufacturer, model, year)
```

```
## # A tibble: 234 x 3
```

##	manufacturer	model	year
##	<fct>	<fct>	<int>
##	1 audi	a4	1999
##	2 audi	a4	1999
##	3 audi	a4	2008

```
## 4 audi      a4      2008
## 5 audi      a4      1999
## 6 audi      a4      1999
## 7 audi      a4      2008
## 8 audi      a4 quattro 1999
## 9 audi      a4 quattro 1999
## 10 audi     a4 quattro 2008
## # ... with 224 more rows
```

```
select(dados, starts_with("m"))
```

```
## # A tibble: 234 x 2
##   manufacturer model
##   <fct>          <fct>
## 1 audi          a4
## 2 audi          a4
## 3 audi          a4
## 4 audi          a4
## 5 audi          a4
## 6 audi          a4
## 7 audi          a4
## 8 audi          a4 quattro
## 9 audi          a4 quattro
## 10 audi         a4 quattro
## # ... with 224 more rows
```

```
select(dados, contains("r"))
```

```
## # A tibble: 234 x 4
##   manufacturer year trans      drv
##   <fct>         <int> <fct>   <fct>
## 1 audi          1999 auto(l5)  f
## 2 audi          1999 manual(m5) f
## 3 audi          2008 manual(m6) f
## 4 audi          2008 auto(av)   f
## 5 audi          1999 auto(l5)  f
## 6 audi          1999 manual(m5) f
## 7 audi          2008 auto(av)   f
## 8 audi          1999 manual(m5) 4
## 9 audi          1999 auto(l5)  4
## 10 audi         2008 manual(m6) 4
## # ... with 224 more rows
```

```
select(dados, ends_with("y"))
```

```
## # A tibble: 234 x 2
##   cty    hwy
##   <int> <int>
```

```
## 1 18 29
## 2 21 29
## 3 20 31
## 4 21 30
## 5 16 26
## 6 18 26
## 7 18 27
## 8 18 26
## 9 16 25
## 10 20 28
## # ... with 224 more rows
```

```
select(dados, matches("[abc]"))
```

```
## # A tibble: 234 x 6
##   manufacturer year   cyl trans      cty class
##   <fct>         <int> <int> <fct>    <int> <fct>
## 1 audi         1999     4 auto(15)    18 compact
## 2 audi         1999     4 manual(m5)  21 compact
## 3 audi         2008     4 manual(m6)  20 compact
## 4 audi         2008     4 auto(av)    21 compact
## 5 audi         1999     6 auto(15)    16 compact
## 6 audi         1999     6 manual(m5)  18 compact
## 7 audi         2008     6 auto(av)    18 compact
## 8 audi         1999     4 manual(m5)  18 compact
## 9 audi         1999     4 auto(15)    16 compact
## 10 audi        2008     4 manual(m6)  20 compact
## # ... with 224 more rows
```

```
select(dados, starts_with("m"), starts_with("c"))
```

```
## # A tibble: 234 x 5
##   manufacturer model     cyl cty class
##   <fct>         <fct>   <int> <int> <fct>
## 1 audi         a4         4     18 compact
## 2 audi         a4         4     21 compact
## 3 audi         a4         4     20 compact
## 4 audi         a4         4     21 compact
## 5 audi         a4         6     16 compact
## 6 audi         a4         6     18 compact
## 7 audi         a4         6     18 compact
## 8 audi         a4 quattro  4     18 compact
## 9 audi         a4 quattro  4     16 compact
## 10 audi        a4 quattro  4     20 compact
## # ... with 224 more rows
```

```
select(dados, ends_with("l"), ends_with("s"))
```

```
## # A tibble: 234 x 6
##   model      displ  cyl fl   trans      class
##   <fct>      <dbl> <int> <fct> <fct>      <fct>
## 1 a4          1.8     4 p   auto(l5) compact
## 2 a4          1.8     4 p   manual(m5) compact
## 3 a4          2       4 p   manual(m6) compact
## 4 a4          2       4 p   auto(av) compact
## 5 a4          2.8     6 p   auto(l5) compact
## 6 a4          2.8     6 p   manual(m5) compact
## 7 a4          3.1     6 p   auto(av) compact
## 8 a4 quattro  1.8     4 p   manual(m5) compact
## 9 a4 quattro  1.8     4 p   auto(l5) compact
## 10 a4 quattro  2       4 p   manual(m6) compact
## # ... with 224 more rows
```

```
select(dados, 1:3)
```

```
## # A tibble: 234 x 3
##   manufacturer model      displ
##   <fct>      <fct>      <dbl>
## 1 audi      a4          1.8
## 2 audi      a4          1.8
## 3 audi      a4          2
## 4 audi      a4          2
## 5 audi      a4          2.8
## 6 audi      a4          2.8
## 7 audi      a4          3.1
## 8 audi      a4 quattro  1.8
## 9 audi      a4 quattro  1.8
## 10 audi     a4 quattro  2
## # ... with 224 more rows
```

```
select(dados, c(2,5,7))
```

```
## # A tibble: 234 x 3
##   model      cyl drv
##   <fct>      <int> <fct>
## 1 a4          4 f
## 2 a4          4 f
## 3 a4          4 f
## 4 a4          4 f
## 5 a4          6 f
## 6 a4          6 f
## 7 a4          6 f
## 8 a4 quattro  4 4
## 9 a4 quattro  4 4
## 10 a4 quattro  4 4
```

```
## # ... with 224 more rows
```

```
select(dados, manufacturer:cyl)
```

```
## # A tibble: 234 x 5
```

```
##   manufacturer model      displ  year  cyl
##   <fct>         <fct>    <dbl> <int> <int>
## 1 audi         a4        1.8  1999   4
## 2 audi         a4        1.8  1999   4
## 3 audi         a4         2   2008   4
## 4 audi         a4         2   2008   4
## 5 audi         a4        2.8  1999   6
## 6 audi         a4        2.8  1999   6
## 7 audi         a4        3.1  2008   6
## 8 audi         a4 quattro  1.8  1999   4
## 9 audi         a4 quattro  1.8  1999   4
## 10 audi        a4 quattro   2   2008   4
```

```
## # ... with 224 more rows
```

```
select(dados,-(manufacturer:cyl))
```

```
## # A tibble: 234 x 6
```

```
##   trans      drv   cty   hwy fl   class
##   <fct>     <fct> <int> <int> <fct> <fct>
## 1 auto(l5)  f      18    29 p    compact
## 2 manual(m5) f      21    29 p    compact
## 3 manual(m6) f      20    31 p    compact
## 4 auto(av)  f      21    30 p    compact
## 5 auto(l5)  f      16    26 p    compact
## 6 manual(m5) f      18    26 p    compact
## 7 auto(av)  f      18    27 p    compact
## 8 manual(m5) 4      18    26 p    compact
## 9 auto(l5)  4      16    25 p    compact
## 10 manual(m6) 4      20    28 p    compact
```

```
## # ... with 224 more rows
```

2.6 `rename()`

```
dados1 <- rename(dados,
                 mnfc = manufacturer,
                 mod = model)
```

```
dados1
```

```
## # A tibble: 234 x 11
```

```
##   mnfc mod      displ  year  cyl trans      drv   cty   hwy fl   class
##   <fct> <fct>    <dbl> <int> <int> <fct>    <fct> <int> <int> <fct> <fct>
```

```
## 1 audi a4 1.8 1999 4 auto(15) f 18 29 p compact
## 2 audi a4 1.8 1999 4 manual(m5) f 21 29 p compact
## 3 audi a4 2 2008 4 manual(m6) f 20 31 p compact
## 4 audi a4 2 2008 4 auto(av) f 21 30 p compact
## 5 audi a4 2.8 1999 6 auto(15) f 16 26 p compact
## 6 audi a4 2.8 1999 6 manual(m5) f 18 26 p compact
## 7 audi a4 3.1 2008 6 auto(av) f 18 27 p compact
## 8 audi a4 quattro 1.8 1999 4 manual(m5) 4 18 26 p compact
## 9 audi a4 quattro 1.8 1999 4 auto(15) 4 16 25 p compact
## 10 audi a4 quattro 2 2008 4 manual(m6) 4 20 28 p compact
## # ... with 224 more rows
```

```
select(dados,
       mnfc = manufacturer,
       mod = model)
```

```
## # A tibble: 234 x 2
##   mnfc mod
##   <fct> <fct>
## 1 audi a4
## 2 audi a4
## 3 audi a4
## 4 audi a4
## 5 audi a4
## 6 audi a4
## 7 audi a4
## 8 audi a4 quattro
## 9 audi a4 quattro
## 10 audi a4 quattro
## # ... with 224 more rows
```

```
select(dados,
       mnfc = manufacturer,
       mod = model,
       everything())
```

```
## # A tibble: 234 x 11
##   mnfc mod displ year cyl trans drv cty hwy fl class
##   <fct> <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct>
## 1 audi a4 1.8 1999 4 auto(15) f 18 29 p compact
## 2 audi a4 1.8 1999 4 manual(m5) f 21 29 p compact
## 3 audi a4 2 2008 4 manual(m6) f 20 31 p compact
## 4 audi a4 2 2008 4 auto(av) f 21 30 p compact
## 5 audi a4 2.8 1999 6 auto(15) f 16 26 p compact
## 6 audi a4 2.8 1999 6 manual(m5) f 18 26 p compact
## 7 audi a4 3.1 2008 6 auto(av) f 18 27 p compact
## 8 audi a4 quattro 1.8 1999 4 manual(m5) 4 18 26 p compact
```



```
## 9 audi a4 quattro 1.8 1999 4 auto(15) 4 16 25 p compact
## 10 audi a4 quattro 2 2008 4 manual(m6) 4 20 28 p compact
## # ... with 224 more rows
```

2.7 mutate() para colunas

```
mutate(dados, sqrt_cty = sqrt(cty))
```

```
## # A tibble: 234 x 12
##   manufac~1 model displ year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 audi      a4      1.8  1999   4 auto~ f    18    29 p    comp~  4.24
## 2 audi      a4      1.8  1999   4 manu~ f    21    29 p    comp~  4.58
## 3 audi      a4      2    2008   4 manu~ f    20    31 p    comp~  4.47
## 4 audi      a4      2    2008   4 auto~ f    21    30 p    comp~  4.58
## 5 audi      a4      2.8  1999   6 auto~ f    16    26 p    comp~  4
## 6 audi      a4      2.8  1999   6 manu~ f    18    26 p    comp~  4.24
## 7 audi      a4      3.1  2008   6 auto~ f    18    27 p    comp~  4.24
## 8 audi      a4 q~    1.8  1999   4 manu~ 4    18    26 p    comp~  4.24
## 9 audi      a4 q~    1.8  1999   4 auto~ 4    16    25 p    comp~  4
## 10 audi     a4 q~    2    2008   4 manu~ 4    20    28 p    comp~  4.47
## # ... with 224 more rows, and abbreviated variable names 1: manufacturer,
## # 2: sqrt_cty
```

```
names(dados)
```

```
## [1] "manufacturer" "model"      "displ"      "year"      "cyl"
## [6] "trans"        "drv"        "cty"        "hwy"      "fl"
## [11] "class"
```

```
dados<- mutate(dados, sqrt_cty = sqrt(cty))
names(dados)
```

```
## [1] "manufacturer" "model"      "displ"      "year"      "cyl"
## [6] "trans"        "drv"        "cty"        "hwy"      "fl"
## [11] "class"        "sqrt_cty"
```

```
dados <- mutate(dados,
`soma de variáveis` = (cty + hwy) / 2)
names(dados)
```

```
## [1] "manufacturer" "model"      "displ"
## [4] "year"         "cyl"        "trans"
## [7] "drv"          "cty"        "hwy"
## [10] "fl"           "class"      "sqrt_cty"
## [13] "soma de variáveis"
```

```
dados <- mutate(dados,
  car = paste(manufacturer, model, sep = " "),
  `cyl / trans` = paste(cyl, " cylinders", " / ", trans, " transmission", sep = " "),
  dados
```

```
## # A tibble: 234 x 15
##   manufac~1 model displ  year  cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 audi      a4      1.8  1999    4 auto~ f    18    29 p    comp~  4.24
## 2 audi      a4      1.8  1999    4 manu~ f    21    29 p    comp~  4.58
## 3 audi      a4      2    2008    4 manu~ f    20    31 p    comp~  4.47
## 4 audi      a4      2    2008    4 auto~ f    21    30 p    comp~  4.58
## 5 audi      a4      2.8  1999    6 auto~ f    16    26 p    comp~  4
## 6 audi      a4      2.8  1999    6 manu~ f    18    26 p    comp~  4.24
## 7 audi      a4      3.1  2008    6 auto~ f    18    27 p    comp~  4.24
## 8 audi      a4 q~    1.8  1999    4 manu~ 4    18    26 p    comp~  4.24
## 9 audi      a4 q~    1.8  1999    4 auto~ 4    16    25 p    comp~  4
## 10 audi     a4 q~    2    2008    4 manu~ 4    20    28 p    comp~  4.47
## # ... with 224 more rows, 3 more variables: `soma de variáveis` <dbl>,
## #   car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## #   1: manufacturer, 2: sqrt_cty
```

2.8 transmute()

```
transmute(dados,
  `avg miles per gallon` = (cty + hwy) / 2)
```

```
## # A tibble: 234 x 1
##   `avg miles per gallon`
##   <dbl>
## 1          23.5
## 2          25
## 3          25.5
## 4          25.5
## 5          21
## 6          22
## 7          22.5
## 8          22
## 9          20.5
## 10         24
## # ... with 224 more rows
```

```
transmute(dados,
  car = paste(manufacturer, model, sep = " "),
  `cyl / trans` = paste(cyl, " cylinders", " / ", trans, " transmission", sep = " "),
  dados
```

```
## # A tibble: 234 x 2
##   car                `cyl / trans`
##   <chr>              <chr>
## 1 audi a4           4 cylinders / auto(l5) transmission
## 2 audi a4           4 cylinders / manual(m5) transmission
## 3 audi a4           4 cylinders / manual(m6) transmission
## 4 audi a4           4 cylinders / auto(av) transmission
## 5 audi a4           6 cylinders / auto(l5) transmission
## 6 audi a4           6 cylinders / manual(m5) transmission
## 7 audi a4           6 cylinders / auto(av) transmission
## 8 audi a4 quattro 4 cylinders / manual(m5) transmission
## 9 audi a4 quattro 4 cylinders / auto(l5) transmission
## 10 audi a4 quattro 4 cylinders / manual(m6) transmission
## # ... with 224 more rows
```

2.9 filter() para linhas

```
filter(dados, manufacturer == "audi")
```

```
## # A tibble: 18 x 15
##   manufac~1 model displ year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 audi      a4      1.8  1999     4 auto~ f     18    29 p   comp~  4.24
## 2 audi      a4      1.8  1999     4 manu~ f     21    29 p   comp~  4.58
## 3 audi      a4      2    2008     4 manu~ f     20    31 p   comp~  4.47
## 4 audi      a4      2    2008     4 auto~ f     21    30 p   comp~  4.58
## 5 audi      a4      2.8  1999     6 auto~ f     16    26 p   comp~  4
## 6 audi      a4      2.8  1999     6 manu~ f     18    26 p   comp~  4.24
## 7 audi      a4      3.1  2008     6 auto~ f     18    27 p   comp~  4.24
## 8 audi      a4 q~    1.8  1999     4 manu~ 4     18    26 p   comp~  4.24
## 9 audi      a4 q~    1.8  1999     4 auto~ 4     16    25 p   comp~  4
## 10 audi     a4 q~    2    2008     4 manu~ 4     20    28 p   comp~  4.47
## 11 audi     a4 q~    2    2008     4 auto~ 4     19    27 p   comp~  4.36
## 12 audi     a4 q~    2.8  1999     6 auto~ 4     15    25 p   comp~  3.87
## 13 audi     a4 q~    2.8  1999     6 manu~ 4     17    25 p   comp~  4.12
## 14 audi     a4 q~    3.1  2008     6 auto~ 4     17    25 p   comp~  4.12
## 15 audi     a4 q~    3.1  2008     6 manu~ 4     15    25 p   comp~  3.87
## 16 audi     a6 q~    2.8  1999     6 auto~ 4     15    24 p   mids~  3.87
## 17 audi     a6 q~    3.1  2008     6 auto~ 4     17    25 p   mids~  4.12
## 18 audi     a6 q~    4.2  2008     8 auto~ 4     16    23 p   mids~  4
## # ... with 3 more variables: `soma de variáveis` <dbl>, car <chr>,
## #   `cyl / trans` <chr>, and abbreviated variable names 1: manufacturer,
## #   2: sqrt_cty
```

```
filter(dados, manufacturer == "audi" & year == "1999")
```

```
## # A tibble: 9 x 15
##   manufact~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct>   <dbl>
## 1 audi      a4      1.8  1999    4 auto~ f    18    29 p    comp~  4.24
## 2 audi      a4      1.8  1999    4 manu~ f    21    29 p    comp~  4.58
## 3 audi      a4      2.8  1999    6 auto~ f    16    26 p    comp~  4
## 4 audi      a4      2.8  1999    6 manu~ f    18    26 p    comp~  4.24
## 5 audi      a4 q~    1.8  1999    4 manu~ 4    18    26 p    comp~  4.24
## 6 audi      a4 q~    1.8  1999    4 auto~ 4    16    25 p    comp~  4
## 7 audi      a4 q~    2.8  1999    6 auto~ 4    15    25 p    comp~  3.87
## 8 audi      a4 q~    2.8  1999    6 manu~ 4    17    25 p    comp~  4.12
## 9 audi      a6 q~    2.8  1999    6 auto~ 4    15    24 p    mids~  3.87
## # ... with 3 more variables: `soma de variáveis` <dbl>, car <chr>,
## #   `cyl / trans` <chr>, and abbreviated variable names 1: manufacturer,
## #   2: sqrt_cty
```

```
filter(dados, manufacturer == "audi", year == 1999)
```

```
## # A tibble: 9 x 15
##   manufact~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct>   <dbl>
## 1 audi      a4      1.8  1999    4 auto~ f    18    29 p    comp~  4.24
## 2 audi      a4      1.8  1999    4 manu~ f    21    29 p    comp~  4.58
## 3 audi      a4      2.8  1999    6 auto~ f    16    26 p    comp~  4
## 4 audi      a4      2.8  1999    6 manu~ f    18    26 p    comp~  4.24
## 5 audi      a4 q~    1.8  1999    4 manu~ 4    18    26 p    comp~  4.24
## 6 audi      a4 q~    1.8  1999    4 auto~ 4    16    25 p    comp~  4
## 7 audi      a4 q~    2.8  1999    6 auto~ 4    15    25 p    comp~  3.87
## 8 audi      a4 q~    2.8  1999    6 manu~ 4    17    25 p    comp~  4.12
## 9 audi      a6 q~    2.8  1999    6 auto~ 4    15    24 p    mids~  3.87
## # ... with 3 more variables: `soma de variáveis` <dbl>, car <chr>,
## #   `cyl / trans` <chr>, and abbreviated variable names 1: manufacturer,
## #   2: sqrt_cty
```

```
filter(dados, manufacturer == "audi" | manufacturer == "dodge") %>%
  print(n = 20)
```

```
## # A tibble: 55 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct>   <dbl>
## 1 audi      a4      1.8  1999    4 auto~ f    18    29 p    comp~  4.24
## 2 audi      a4      1.8  1999    4 manu~ f    21    29 p    comp~  4.58
## 3 audi      a4      2    2008    4 manu~ f    20    31 p    comp~  4.47
## 4 audi      a4      2    2008    4 auto~ f    21    30 p    comp~  4.58
## 5 audi      a4      2.8  1999    6 auto~ f    16    26 p    comp~  4
```

```
## 6 audi      a4      2.8 1999      6 manu~ f      18      26 p      comp~ 4.24
## 7 audi      a4      3.1 2008      6 auto~ f      18      27 p      comp~ 4.24
## 8 audi      a4 q~   1.8 1999      4 manu~ 4      18      26 p      comp~ 4.24
## 9 audi      a4 q~   1.8 1999      4 auto~ 4      16      25 p      comp~ 4
## 10 audi     a4 q~   2    2008      4 manu~ 4      20      28 p      comp~ 4.47
## 11 audi     a4 q~   2    2008      4 auto~ 4      19      27 p      comp~ 4.36
## 12 audi     a4 q~   2.8 1999      6 auto~ 4      15      25 p      comp~ 3.87
## 13 audi     a4 q~   2.8 1999      6 manu~ 4      17      25 p      comp~ 4.12
## 14 audi     a4 q~   3.1 2008      6 auto~ 4      17      25 p      comp~ 4.12
## 15 audi     a4 q~   3.1 2008      6 manu~ 4      15      25 p      comp~ 3.87
## 16 audi     a6 q~   2.8 1999      6 auto~ 4      15      24 p      mids~ 3.87
## 17 audi     a6 q~   3.1 2008      6 auto~ 4      17      25 p      mids~ 4.12
## 18 audi     a6 q~   4.2 2008      8 auto~ 4      16      23 p      mids~ 4
## 19 dodge    cara~   2.4 1999      4 auto~ f      18      24 r      mini~ 4.24
## 20 dodge    cara~   3    1999      6 auto~ f      17      24 r      mini~ 4.12
## # ... with 35 more rows, 3 more variables: `soma de variáveis` <dbl>,
## #   car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## #   1: manufacturer, 2: sqrt_cty

filter(dados, manufacturer %in% c("audi", "dodge")) %>%
  print(n = 20)
```

```
## # A tibble: 55 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 audi      a4      1.8 1999     4 auto~ f     18    29 p   comp~ 4.24
## 2 audi      a4      1.8 1999     4 manu~ f     21    29 p   comp~ 4.58
## 3 audi      a4      2    2008     4 manu~ f     20    31 p   comp~ 4.47
## 4 audi      a4      2    2008     4 auto~ f     21    30 p   comp~ 4.58
## 5 audi      a4      2.8 1999     6 auto~ f     16    26 p   comp~ 4
## 6 audi      a4      2.8 1999     6 manu~ f     18    26 p   comp~ 4.24
## 7 audi      a4      3.1 2008     6 auto~ f     18    27 p   comp~ 4.24
## 8 audi      a4 q~   1.8 1999     4 manu~ 4     18    26 p   comp~ 4.24
## 9 audi      a4 q~   1.8 1999     4 auto~ 4     16    25 p   comp~ 4
## 10 audi     a4 q~   2    2008     4 manu~ 4     20    28 p   comp~ 4.47
## 11 audi     a4 q~   2    2008     4 auto~ 4     19    27 p   comp~ 4.36
## 12 audi     a4 q~   2.8 1999     6 auto~ 4     15    25 p   comp~ 3.87
## 13 audi     a4 q~   2.8 1999     6 manu~ 4     17    25 p   comp~ 4.12
## 14 audi     a4 q~   3.1 2008     6 auto~ 4     17    25 p   comp~ 4.12
## 15 audi     a4 q~   3.1 2008     6 manu~ 4     15    25 p   comp~ 3.87
## 16 audi     a6 q~   2.8 1999     6 auto~ 4     15    24 p   mids~ 3.87
## 17 audi     a6 q~   3.1 2008     6 auto~ 4     17    25 p   mids~ 4.12
## 18 audi     a6 q~   4.2 2008     8 auto~ 4     16    23 p   mids~ 4
## 19 dodge    cara~   2.4 1999     4 auto~ f     18    24 r   mini~ 4.24
## 20 dodge    cara~   3    1999     6 auto~ f     17    24 r   mini~ 4.12
## # ... with 35 more rows, 3 more variables: `soma de variáveis` <dbl>,
```

```
## #   car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## #   1: manufacturer, 2: sqrt_cty
```

```
filter(dados, hwy >= 30) %>%
  select(hwy) %>%
  print(n = 26)
```

```
## # A tibble: 26 x 1
```

```
##       hwy
```

```
##   <int>
```

```
## 1     31
```

```
## 2     30
```

```
## 3     30
```

```
## 4     33
```

```
## 5     32
```

```
## 6     32
```

```
## 7     32
```

```
## 8     34
```

```
## 9     36
```

```
## 10    36
```

```
## 11    30
```

```
## 12    31
```

```
## 13    31
```

```
## 14    32
```

```
## 15    31
```

```
## 16    31
```

```
## 17    31
```

```
## 18    31
```

```
## 19    30
```

```
## 20    33
```

```
## 21    35
```

```
## 22    37
```

```
## 23    35
```

```
## 24    44
```

```
## 25    44
```

```
## 26    41
```

```
filter(dados, year != 1999) %>%
  select(year) %>%
  print(n = 30)
```

```
## # A tibble: 117 x 1
```

```
##       year
```

```
##   <int>
```

```
## 1  2008
```

```
## 2  2008
```

```
## 3  2008
```

```
## 4 2008
## 5 2008
## 6 2008
## 7 2008
## 8 2008
## 9 2008
## 10 2008
## 11 2008
## 12 2008
## 13 2008
## 14 2008
## 15 2008
## 16 2008
## 17 2008
## 18 2008
## 19 2008
## 20 2008
## 21 2008
## 22 2008
## 23 2008
## 24 2008
## 25 2008
## 26 2008
## 27 2008
## 28 2008
## 29 2008
## 30 2008
## # ... with 87 more rows
filter(dados, between(cty, 15, 22))
```

```
## # A tibble: 143 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 audi      a4      1.8  1999     4 auto~ f    18    29 p    comp~ 4.24
## 2 audi      a4      1.8  1999     4 manu~ f    21    29 p    comp~ 4.58
## 3 audi      a4      2    2008     4 manu~ f    20    31 p    comp~ 4.47
## 4 audi      a4      2    2008     4 auto~ f    21    30 p    comp~ 4.58
## 5 audi      a4      2.8  1999     6 auto~ f    16    26 p    comp~ 4
## 6 audi      a4      2.8  1999     6 manu~ f    18    26 p    comp~ 4.24
## 7 audi      a4      3.1  2008     6 auto~ f    18    27 p    comp~ 4.24
## 8 audi      a4 q~    1.8  1999     4 manu~ 4    18    26 p    comp~ 4.24
## 9 audi      a4 q~    1.8  1999     4 auto~ 4    16    25 p    comp~ 4
## 10 audi     a4 q~    2    2008     4 manu~ 4    20    28 p    comp~ 4.47
## # ... with 133 more rows, 3 more variables: `soma de variáveis` <dbl>,
## #   car <chr>, `cyl / trans` <chr>, and abbreviated variable names
```

```
## # 1: manufacturer, 2: sqrt_cty
```

2.10 slice() para linhas

```
slice(dados, 1:5)
```

```
## # A tibble: 5 x 15
##   manufact~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 audi      a4      1.8  1999     4 auto~ f     18    29 p   comp~  4.24
## 2 audi      a4      1.8  1999     4 manu~ f     21    29 p   comp~  4.58
## 3 audi      a4      2    2008     4 manu~ f     20    31 p   comp~  4.47
## 4 audi      a4      2    2008     4 auto~ f     21    30 p   comp~  4.58
## 5 audi      a4      2.8  1999     6 auto~ f     16    26 p   comp~  4
## # ... with 3 more variables: `soma de variáveis` <dbl>, car <chr>,
## #   `cyl / trans` <chr>, and abbreviated variable names 1: manufacturer,
## #   2: sqrt_cty
```

```
# dados[1:5,]
```

```
slice(dados, 20:30)
```

```
## # A tibble: 11 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 chevrolet c150~  5.3  2008     8 auto~ r     11    15 e   suv    3.32
## 2 chevrolet c150~  5.3  2008     8 auto~ r     14    20 r   suv    3.74
## 3 chevrolet c150~  5.7  1999     8 auto~ r     13    17 r   suv    3.61
## 4 chevrolet c150~  6    2008     8 auto~ r     12    17 r   suv    3.46
## 5 chevrolet corv~  5.7  1999     8 manu~ r     16    26 p   2sea~  4
## 6 chevrolet corv~  5.7  1999     8 auto~ r     15    23 p   2sea~  3.87
## 7 chevrolet corv~  6.2  2008     8 manu~ r     16    26 p   2sea~  4
## 8 chevrolet corv~  6.2  2008     8 auto~ r     15    25 p   2sea~  3.87
## 9 chevrolet corv~  7    2008     8 manu~ r     15    24 p   2sea~  3.87
## 10 chevrolet k150~  5.3  2008     8 auto~ 4     14    19 r   suv    3.74
## 11 chevrolet k150~  5.3  2008     8 auto~ 4     11    14 e   suv    3.32
## # ... with 3 more variables: `soma de variáveis` <dbl>, car <chr>,
## #   `cyl / trans` <chr>, and abbreviated variable names 1: manufacturer,
## #   2: sqrt_cty
```

```
# dados[20:30,]
```


2.11 arrange() para linhas

```
# ordenar "displ" de menor a maior
arrange(dados, displ)
```

```
## # A tibble: 234 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 honda    civic  1.6  1999     4 manu~ f     28    33 r   subc~  5.29
## 2 honda    civic  1.6  1999     4 auto~ f     24    32 r   subc~  4.90
## 3 honda    civic  1.6  1999     4 manu~ f     25    32 r   subc~    5
## 4 honda    civic  1.6  1999     4 manu~ f     23    29 p   subc~  4.80
## 5 honda    civic  1.6  1999     4 auto~ f     24    32 r   subc~  4.90
## 6 audi     a4      1.8  1999     4 auto~ f     18    29 p   comp~  4.24
## 7 audi     a4      1.8  1999     4 manu~ f     21    29 p   comp~  4.58
## 8 audi     a4 q~   1.8  1999     4 manu~ 4     18    26 p   comp~  4.24
## 9 audi     a4 q~   1.8  1999     4 auto~ 4     16    25 p   comp~    4
## 10 honda   civic  1.8  2008     4 manu~ f     26    34 r   subc~  5.10
## # ... with 224 more rows, 3 more variables: `soma de variáveis` <dbl>,
## #   car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## #   1: manufacturer, 2: sqrt_cty

arrange(dados, displ) %>%
  print(n=20)
```

```
## # A tibble: 234 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 honda    civic  1.6  1999     4 manu~ f     28    33 r   subc~  5.29
## 2 honda    civic  1.6  1999     4 auto~ f     24    32 r   subc~  4.90
## 3 honda    civic  1.6  1999     4 manu~ f     25    32 r   subc~    5
## 4 honda    civic  1.6  1999     4 manu~ f     23    29 p   subc~  4.80
## 5 honda    civic  1.6  1999     4 auto~ f     24    32 r   subc~  4.90
## 6 audi     a4      1.8  1999     4 auto~ f     18    29 p   comp~  4.24
## 7 audi     a4      1.8  1999     4 manu~ f     21    29 p   comp~  4.58
## 8 audi     a4 q~   1.8  1999     4 manu~ 4     18    26 p   comp~  4.24
## 9 audi     a4 q~   1.8  1999     4 auto~ 4     16    25 p   comp~    4
## 10 honda   civic  1.8  2008     4 manu~ f     26    34 r   subc~  5.10
## 11 honda   civic  1.8  2008     4 auto~ f     25    36 r   subc~    5
## 12 honda   civic  1.8  2008     4 auto~ f     24    36 c   subc~  4.90
## 13 toyota  coro~   1.8  1999     4 auto~ f     24    30 r   comp~  4.90
## 14 toyota  coro~   1.8  1999     4 auto~ f     24    33 r   comp~  4.90
## 15 toyota  coro~   1.8  1999     4 manu~ f     26    35 r   comp~  5.10
## 16 toyota  coro~   1.8  2008     4 manu~ f     28    37 r   comp~  5.29
## 17 toyota  coro~   1.8  2008     4 auto~ f     26    35 r   comp~  5.10
## 18 volkswag~ pass~ 1.8  1999     4 manu~ f     21    29 p   mids~  4.58
```

```
## 19 volkswag~ pass~ 1.8 1999 4 auto~ f 18 29 p mids~ 4.24
## 20 volkswag~ jetta 1.9 1999 4 manu~ f 33 44 d comp~ 5.74
## # ... with 214 more rows, 3 more variables: `soma de variáveis` <dbl>,
## # car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## # 1: manufacturer, 2: sqrt_cty
```

```
# ordenar "displ" de maior a menor
arrange(dados, desc(displ))
```

```
## # A tibble: 234 x 15
##   manufac~1 model displ year cyl trans drv cty hwy fl class sqrt_~2
##   <fct> <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 chevrolet corv~ 7 2008 8 manu~ r 15 24 p 2sea~ 3.87
## 2 chevrolet k150~ 6.5 1999 8 auto~ 4 14 17 d suv 3.74
## 3 chevrolet corv~ 6.2 2008 8 manu~ r 16 26 p 2sea~ 4
## 4 chevrolet corv~ 6.2 2008 8 auto~ r 15 25 p 2sea~ 3.87
## 5 jeep gran~ 6.1 2008 8 auto~ 4 11 14 p suv 3.32
## 6 chevrolet c150~ 6 2008 8 auto~ r 12 17 r suv 3.46
## 7 dodge dura~ 5.9 1999 8 auto~ 4 11 15 r suv 3.32
## 8 dodge ram ~ 5.9 1999 8 auto~ 4 11 15 r pick~ 3.32
## 9 chevrolet c150~ 5.7 1999 8 auto~ r 13 17 r suv 3.61
## 10 chevrolet corv~ 5.7 1999 8 manu~ r 16 26 p 2sea~ 4
## # ... with 224 more rows, 3 more variables: `soma de variáveis` <dbl>,
## # car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## # 1: manufacturer, 2: sqrt_cty
```

```
arrange(dados, desc(displ)) %>%
  print(n=20)
```

```
## # A tibble: 234 x 15
##   manufac~1 model displ year cyl trans drv cty hwy fl class sqrt_~2
##   <fct> <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 chevrolet corv~ 7 2008 8 manu~ r 15 24 p 2sea~ 3.87
## 2 chevrolet k150~ 6.5 1999 8 auto~ 4 14 17 d suv 3.74
## 3 chevrolet corv~ 6.2 2008 8 manu~ r 16 26 p 2sea~ 4
## 4 chevrolet corv~ 6.2 2008 8 auto~ r 15 25 p 2sea~ 3.87
## 5 jeep gran~ 6.1 2008 8 auto~ 4 11 14 p suv 3.32
## 6 chevrolet c150~ 6 2008 8 auto~ r 12 17 r suv 3.46
## 7 dodge dura~ 5.9 1999 8 auto~ 4 11 15 r suv 3.32
## 8 dodge ram ~ 5.9 1999 8 auto~ 4 11 15 r pick~ 3.32
## 9 chevrolet c150~ 5.7 1999 8 auto~ r 13 17 r suv 3.61
## 10 chevrolet corv~ 5.7 1999 8 manu~ r 16 26 p 2sea~ 4
## 11 chevrolet corv~ 5.7 1999 8 auto~ r 15 23 p 2sea~ 3.87
## 12 chevrolet k150~ 5.7 1999 8 auto~ 4 11 15 r suv 3.32
## 13 dodge dura~ 5.7 2008 8 auto~ 4 13 18 r suv 3.61
## 14 dodge ram ~ 5.7 2008 8 auto~ 4 13 17 r pick~ 3.61
## 15 jeep gran~ 5.7 2008 8 auto~ 4 13 18 r suv 3.61
```

```
## 16 toyota    land~  5.7  2008      8 auto~ 4      13    18 r    suv      3.61
## 17 nissan    path~  5.6  2008      8 auto~ 4      12    18 p    suv      3.46
## 18 ford     expe~  5.4  1999      8 auto~ r     11    17 r    suv      3.32
## 19 ford     expe~  5.4  2008      8 auto~ r     12    18 r    suv      3.46
## 20 ford     f150~  5.4  1999      8 auto~ 4      11    15 r    pick~   3.32
## # ... with 214 more rows, 3 more variables: `soma de variáveis` <dbl>,
## #   car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## #   1: manufacturer, 2: sqrt_cty
```

```
select(dados, displ, cty) %>%
  arrange(displ, cty) %>%
  print(n = 20)
```

```
## # A tibble: 234 x 2
##   displ  cty
##   <dbl> <int>
## 1  1.6    23
## 2  1.6    24
## 3  1.6    24
## 4  1.6    25
## 5  1.6    28
## 6  1.8    16
## 7  1.8    18
## 8  1.8    18
## 9  1.8    18
## 10 1.8    21
## 11 1.8    21
## 12 1.8    24
## 13 1.8    24
## 14 1.8    24
## 15 1.8    25
## 16 1.8    26
## 17 1.8    26
## 18 1.8    26
## 19 1.8    28
## 20 1.9    29
## # ... with 214 more rows
```

```
select(dados, displ, cty) %>%
  arrange(displ, desc(cty)) %>%
  print(n = 20)
```

```
## # A tibble: 234 x 2
##   displ  cty
##   <dbl> <int>
## 1  1.6    28
## 2  1.6    25
```

```
## 3 1.6 24
## 4 1.6 24
## 5 1.6 23
## 6 1.8 28
## 7 1.8 26
## 8 1.8 26
## 9 1.8 26
## 10 1.8 25
## 11 1.8 24
## 12 1.8 24
## 13 1.8 24
## 14 1.8 21
## 15 1.8 21
## 16 1.8 18
## 17 1.8 18
## 18 1.8 18
## 19 1.8 16
## 20 1.9 35
## # ... with 214 more rows
```

2.12 distinct() para linhas

```
dados_exemplo <- data.frame(id = 1:3,
                             name = c("John", "Max", "Julia"))
dados_exemplo

##   id name
## 1  1 John
## 2  2  Max
## 3  3 Julia

# bind_rows == rbind()
dados_exemplo <- bind_rows(dados_exemplo, slice(dados_exemplo, 2))
dados_exemplo

##   id name
## 1  1 John
## 2  2  Max
## 3  3 Julia
## 4  2  Max

distinct(dados_exemplo)

##   id name
## 1  1 John
## 2  2  Max
```

```
## 3 3 Julia
dados_exemplo2 <- data.frame(id = c(1,1,2),
                             name = c("John", "Max", "Julia"))
dados_exemplo2
```

```
##   id name
## 1  1 John
## 2  1  Max
## 3  2 Julia
distinct(dados_exemplo2)
```

```
##   id name
## 1  1 John
## 2  1  Max
## 3  2 Julia
dados_duplicados <- select(dados, manufacturer, model)
dados_duplicados
```

```
## # A tibble: 234 x 2
##   manufacturer model
##   <fct>         <fct>
## 1 audi          a4
## 2 audi          a4
## 3 audi          a4
## 4 audi          a4
## 5 audi          a4
## 6 audi          a4
## 7 audi          a4
## 8 audi          a4 quattro
## 9 audi          a4 quattro
## 10 audi          a4 quattro
## # ... with 224 more rows
dados_ao_duplicados <- distinct(dados_duplicados)
dados_ao_duplicados
```

```
## # A tibble: 38 x 2
##   manufacturer model
##   <fct>         <fct>
## 1 audi          a4
## 2 audi          a4 quattro
## 3 audi          a6 quattro
## 4 chevrolet     c1500 suburban 2wd
## 5 chevrolet     corvette
## 6 chevrolet     k1500 tahoe 4wd
```

```
## 7 chevrolet    malibu
## 8 dodge        caravan 2wd
## 9 dodge        dakota pickup 4wd
## 10 dodge       durango 4wd
## # ... with 28 more rows
```

2.13 summarise()

```
summarise(dados, `média hwy` = mean(hwy))
```

```
## # A tibble: 1 x 1
##   `média hwy`
##         <dbl>
## 1         23.4
```

```
summarise(dados,
  `num. de dados` = n(),
  `num. modelos` = n_distinct(model))
```

```
## # A tibble: 1 x 2
##   `num. de dados` `num. modelos`
##         <int>         <int>
## 1           234             38
```

```
# levels(dados$model)
```

```
summarise(dados,
  `mín. hwy` = min(hwy, na.rm = TRUE),
  `mín. cty` = min(cty, na.rm = TRUE),
  `máx. hwy` = max(hwy, na.rm = TRUE),
  `máx. cty` = max(cty, na.rm = TRUE))
```

```
## # A tibble: 1 x 4
##   `mín. hwy` `mín. cty` `máx. hwy` `máx. cty`
##         <int>    <int>    <int>    <int>
## 1         12         9        44        35
```

```
dados %>%
```

```
  summarise_at(c("hwy", "cty"), list(min, max), na.rm = TRUE)
```

```
## # A tibble: 1 x 4
##   hwy_fn1 cty_fn1 hwy_fn2 cty_fn2
##     <int>  <int>  <int>  <int>
## 1      12     9      44     35
```

```
dados %>%
```

```
  summarise_if(is.numeric, list(min, max), na.rm = TRUE)
```

```
## # A tibble: 1 x 14
```

```
##   displ_fn1 year_fn1 cyl_fn1 cty_fn1 hwy_fn1 sqrt_cty_~1 soma ~2 displ~3 year_~4
##   <dbl>    <int>    <int>    <int>    <int>    <dbl>    <dbl>    <dbl>    <int>
## 1      1.6     1999      4      9      12      3     10.5      7     2008
## # ... with 5 more variables: cyl_fn2 <int>, cty_fn2 <int>, hwy_fn2 <int>,
## #   sqrt_cty_fn2 <dbl>, `soma de variáveis_fn2` <dbl>, and abbreviated variable
## #   names 1: sqrt_cty_fn1, 2: `soma de variáveis_fn1`, 3: displ_fn2,
## #   4: year_fn2
```

```
dados %>%
  summarise_if(is.numeric, min, na.rm = TRUE)
```

```
## # A tibble: 1 x 7
##   displ year   cyl   cty   hwy sqrt_cty `soma de variáveis`
##   <dbl> <int> <int> <int> <int>    <dbl>          <dbl>
## 1   1.6  1999     4     9    12      3          10.5
```

```
dados %>%
  summarise_if(is.numeric, max, na.rm = TRUE)
```

```
## # A tibble: 1 x 7
##   displ year   cyl   cty   hwy sqrt_cty `soma de variáveis`
##   <dbl> <int> <int> <int> <int>    <dbl>          <dbl>
## 1     7  2008     8    35    44    5.92          39.5
```

```
Tiago<- function(dados){
  sd(dados)/mean(dados)
}
```

```
dados %>%
  summarise_if(is.numeric, Tiago)
```

```
## # A tibble: 1 x 7
##   displ   year   cyl   cty   hwy sqrt_cty `soma de variáveis`
##   <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>          <dbl>
## 1 0.372 0.00225 0.274 0.252 0.254    0.125          0.251
```

2.14 group_by()

```
group_by(dados, manufacturer)
```

```
## # A tibble: 234 x 15
## # Groups:   manufacturer [15]
##   manufac~1 model displ year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>     <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct>    <dbl>
## 1 audi      a4      1.8  1999     4 auto~ f     18    29 p   comp~    4.24
## 2 audi      a4      1.8  1999     4 manu~ f     21    29 p   comp~    4.58
## 3 audi      a4      2    2008     4 manu~ f     20    31 p   comp~    4.47
```

```
## 4 audi      a4      2      2008      4 auto~ f      21      30 p      comp~      4.58
## 5 audi      a4      2.8    1999      6 auto~ f      16      26 p      comp~      4
## 6 audi      a4      2.8    1999      6 manu~ f      18      26 p      comp~      4.24
## 7 audi      a4      3.1    2008      6 auto~ f      18      27 p      comp~      4.24
## 8 audi      a4 q~    1.8    1999      4 manu~ 4      18      26 p      comp~      4.24
## 9 audi      a4 q~    1.8    1999      4 auto~ 4      16      25 p      comp~      4
## 10 audi     a4 q~    2      2008      4 manu~ 4      20      28 p      comp~      4.47
## # ... with 224 more rows, 3 more variables: `soma de variáveis` <dbl>,
## #   car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## #   1: manufacturer, 2: sqrt_cty
```

```
dados %>%
  group_by(manufacturer) %>%
  summarise(`num. carros` = n())
```

```
## # A tibble: 15 x 2
##   manufacturer `num. carros`
##   <fct>         <int>
## 1 audi             18
## 2 chevrolet        19
## 3 dodge            37
## 4 ford             25
## 5 honda             9
## 6 hyundai          14
## 7 jeep             8
## 8 land rover        4
## 9 lincoln           3
## 10 mercury          4
## 11 nissan            13
## 12 pontiac           5
## 13 subaru            14
## 14 toyota            34
## 15 volkswagen       27
```

```
dados %>%
  group_by(model) %>%
  summarise(`média hwy` = mean(hwy),
            `min. hwy` = min(hwy),
            `max. hwy` = max(hwy))
```

```
## # A tibble: 38 x 4
##   model          `média hwy` `min. hwy` `max. hwy`
##   <fct>         <dbl>     <int>     <int>
## 1 4runner 4wd      18.8         17         20
## 2 a4              28.3         26         31
## 3 a4 quattro      25.8         25         28
## 4 a6 quattro      24          23         25
```



```
## 5 altima                28.7        26        32
## 6 c1500 suburban 2wd    17.8        15        20
## 7 camry                 28.3        26        31
## 8 camry solara          28.1        26        31
## 9 caravan 2wd          22.4        17        24
## 10 civic                32.6        29        36
## # ... with 28 more rows
```

2.15 count()

```
count(dados)
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   234
```

```
dados %>%
  group_by(manufacturer) %>%
  count()
```

```
## # A tibble: 15 x 2
## # Groups:   manufacturer [15]
##   manufacturer      n
##   <fct>          <int>
## 1 audi             18
## 2 chevrolet        19
## 3 dodge            37
## 4 ford             25
## 5 honda             9
## 6 hyundai          14
## 7 jeep              8
## 8 land rover        4
## 9 lincoln           3
## 10 mercury          4
## 11 nissan            13
## 12 pontiac           5
## 13 subaru            14
## 14 toyota            34
## 15 volkswagen        27
```

```
# Equivalente com o código anterior
```

```
dados %>%
  group_by(manufacturer) %>%
  summarise(cars = n())
```

```
## # A tibble: 15 x 2
```

```
##      manufacturer cars
##      <fct>         <int>
##  1 audi           18
##  2 chevrolet      19
##  3 dodge          37
##  4 ford           25
##  5 honda           9
##  6 hyundai        14
##  7 jeep            8
##  8 land rover      4
##  9 lincoln         3
## 10 mercury         4
## 11 nissan          13
## 12 pontiac         5
## 13 subaru          14
## 14 toyota          34
## 15 volkswagen      27
```

2.16 sample_n()

```
set.seed(567)
sample_n(dados, size = 10, replace = F)
```

```
## # A tibble: 10 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
##  1 mercury  moun~    5   1999    8 auto~ 4    13    17 r    suv    3.61
##  2 chevrolet corv~    7   2008    8 manu~ r    15    24 p    2sea~  3.87
##  3 dodge    ram ~   4.7  2008    8 manu~ 4    12    16 r    pick~  3.46
##  4 toyota   land~   4.7  1999    8 auto~ 4    11    15 r    suv    3.32
##  5 volkswag~ jetta  2    1999    4 auto~ f    19    26 r    comp~  4.36
##  6 dodge    cara~   3.8  1999    6 auto~ f    15    21 r    mini~  3.87
##  7 honda    civic  1.8  2008    4 auto~ f    25    36 r    subc~  5
##  8 ford     must~   4.6  1999    8 auto~ r    15    21 r    subc~  3.87
##  9 chevrolet c150~   5.3  2008    8 auto~ r    14    20 r    suv    3.74
## 10 ford     expe~   5.4  1999    8 auto~ r    11    17 r    suv    3.32
## # ... with 3 more variables: `soma de variáveis` <dbl>, car <chr>,
## #   `cyl / trans` <chr>, and abbreviated variable names 1: manufacturer,
## #   2: sqrt_cty
```

```
sample_n(dados, size = 10, replace = T)
```

```
## # A tibble: 10 x 15
##   manufac~1 model displ  year   cyl trans drv   cty   hwy fl   class sqrt_~2
##   <fct>      <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
```

```
## 1 chevrolet c150~ 5.3 2008 8 auto~ r 11 15 e suv 3.32
## 2 volkswag~ gti 2 2008 4 auto~ f 22 29 p comp~ 4.69
## 3 dodge dako~ 4.7 2008 8 auto~ 4 14 19 r pick~ 3.74
## 4 ford expl~ 4.6 2008 8 auto~ 4 13 19 r suv 3.61
## 5 dodge cara~ 3.8 2008 6 auto~ f 16 23 r mini~ 4
## 6 chevrolet k150~ 5.3 2008 8 auto~ 4 14 19 r suv 3.74
## 7 dodge dura~ 5.2 1999 8 auto~ 4 11 16 r suv 3.32
## 8 toyota camry 2.4 2008 4 manu~ f 21 31 r mids~ 4.58
## 9 toyota camry 3 1999 6 manu~ f 18 26 r mids~ 4.24
## 10 subaru impr~ 2.2 1999 4 auto~ 4 21 26 r subc~ 4.58
## # ... with 3 more variables: `soma de variáveis` <dbl>, car <chr>,
## # `cyl / trans` <chr>, and abbreviated variable names 1: manufacturer,
## # 2: sqrt_cty
```

2.17 sample_frac()

```
sample_frac(dados, size = 0.1, replace = F)
```

```
## # A tibble: 23 x 15
##   manufac~1 model displ year cyl trans drv cty hwy fl class sqrt_~2
##   <fct> <fct> <dbl> <int> <int> <fct> <fct> <int> <int> <fct> <fct> <dbl>
## 1 toyota coro~ 1.8 2008 4 manu~ f 28 37 r comp~ 5.29
## 2 lincoln navi~ 5.4 1999 8 auto~ r 11 17 r suv 3.32
## 3 honda civic 1.6 1999 4 auto~ f 24 32 r subc~ 4.90
## 4 audi a6 q~ 2.8 1999 6 auto~ 4 15 24 p mids~ 3.87
## 5 nissan path~ 4 2008 6 auto~ 4 14 20 p suv 3.74
## 6 toyota camry 3.5 2008 6 auto~ f 19 28 r mids~ 4.36
## 7 subaru impr~ 2.5 2008 4 auto~ 4 20 25 p comp~ 4.47
## 8 toyota toyo~ 3.4 1999 6 auto~ 4 15 19 r pick~ 3.87
## 9 audi a4 q~ 3.1 2008 6 manu~ 4 15 25 p comp~ 3.87
## 10 toyota coro~ 1.8 1999 4 manu~ f 26 35 r comp~ 5.10
## # ... with 13 more rows, 3 more variables: `soma de variáveis` <dbl>,
## # car <chr>, `cyl / trans` <chr>, and abbreviated variable names
## # 1: manufacturer, 2: sqrt_cty
```


Chapter 3

ggplot2 (60 minutos)

3.1 Carrega pacotes a serem usados

```
#install.packages("tidyverse")
#install.packages("dplyr")
#install.packages("tidyr")
#install.packages("ggplot2")

library(tidyverse)
# Manipulação de dados
#library(dplyr)

# Visualização de gráficos
library(ggplot2)
library(gridExtra)
library(patchwork)
library(plotly)
library(esquisse)

# Para dados gráfico de perfis
library(nlme)
```

Alguns links

The R Graph Gallery

120 registered extensions available to explore

link 1: patchwork

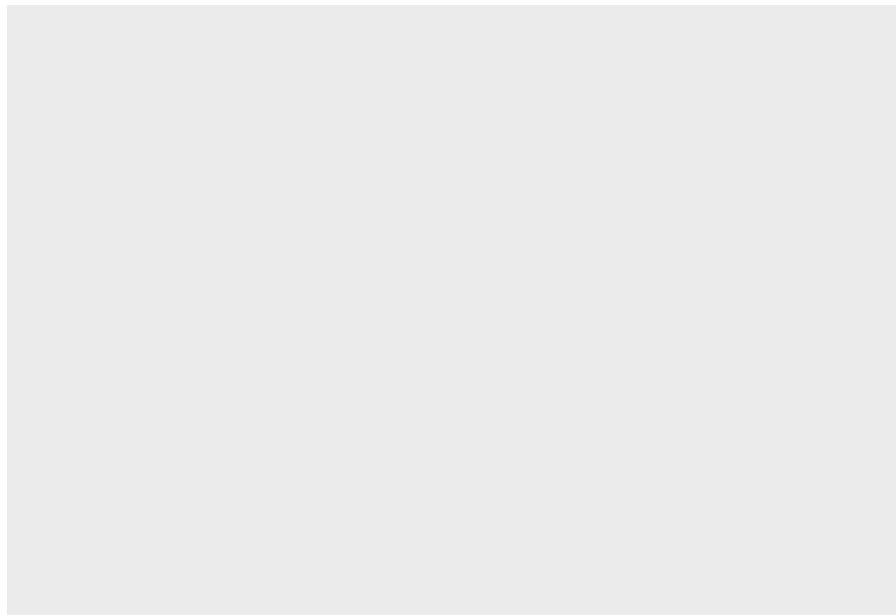
link 2: patchwork

```
ls("package:ggplot2")
```

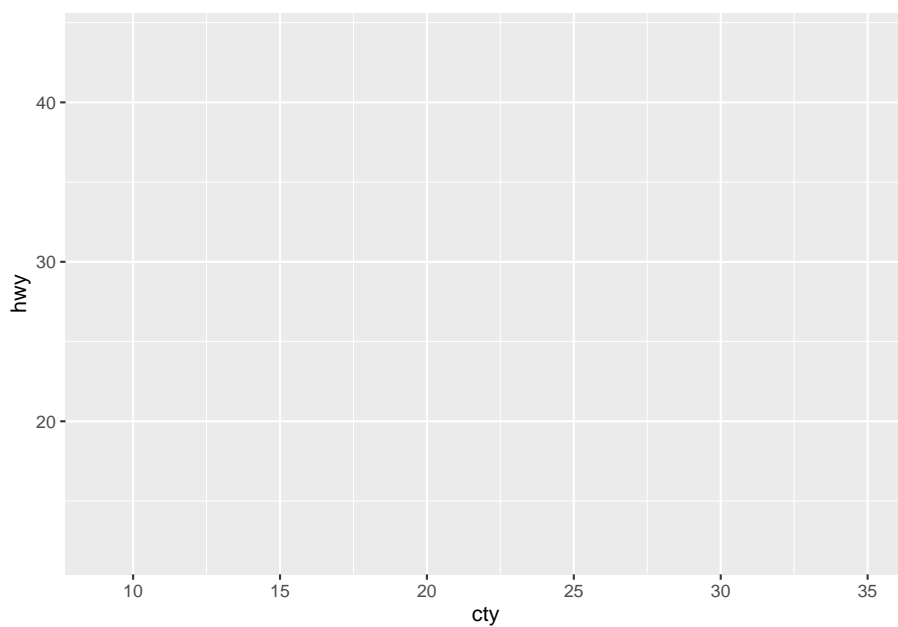
3.2 Lista de funções do pacote ggplot2

```
dados <- mpg  
ggplot(dados)
```

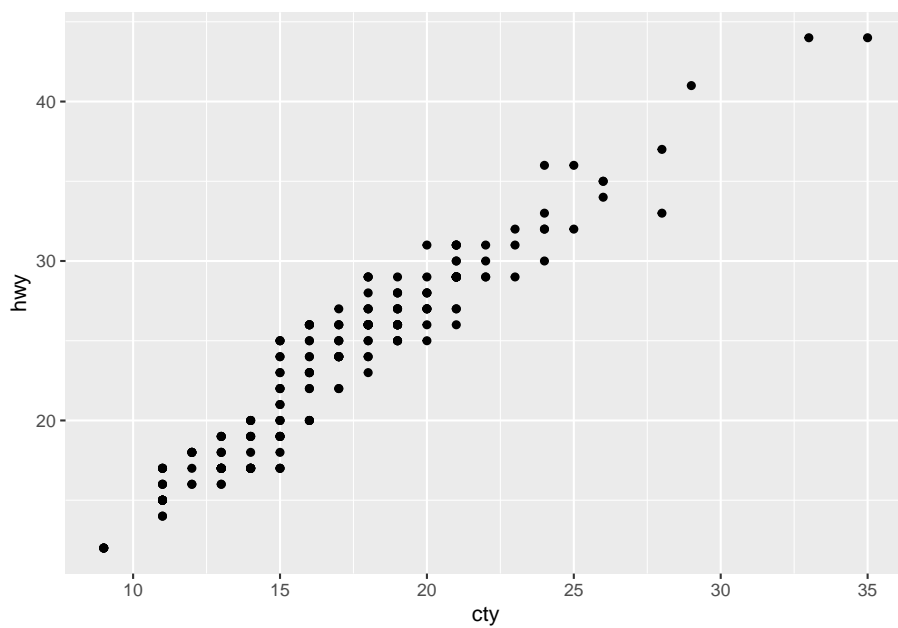
3.3 Primeiros passos usando geom_point



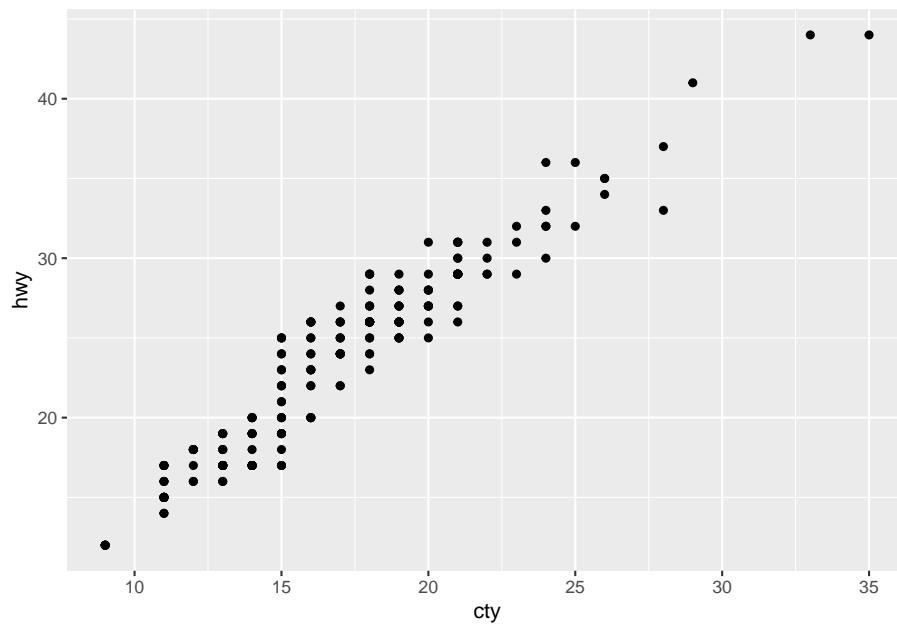
```
ggplot(dados, aes(x = cty, y = hwy))
```



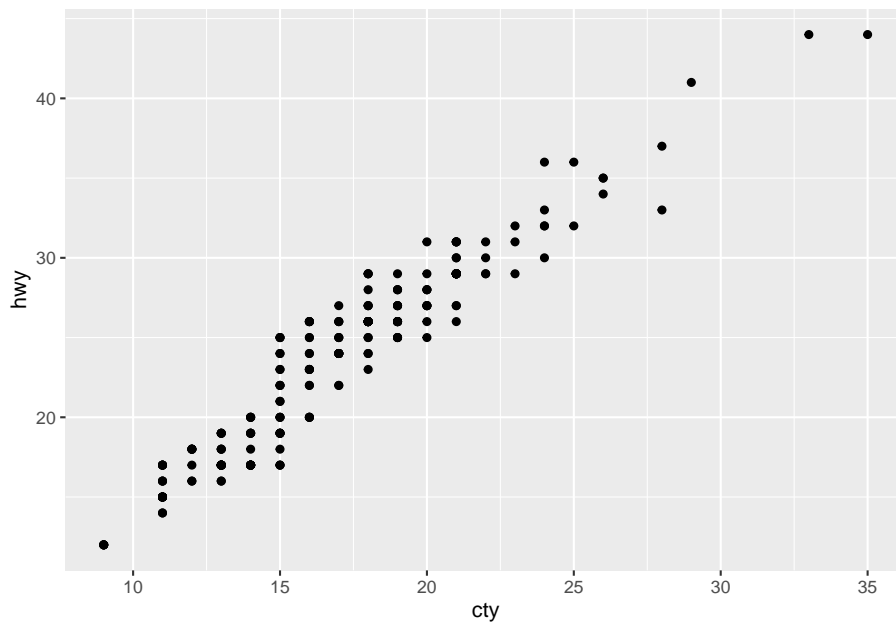
```
# Alternativas  
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point()
```



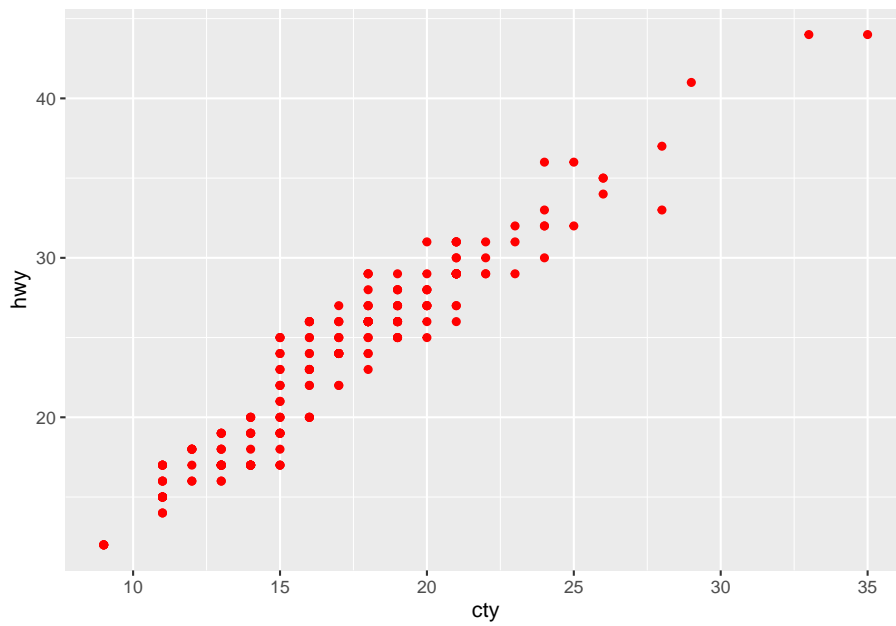
```
ggplot(dados) +  
  geom_point(aes(x = cty, y = hwy))
```



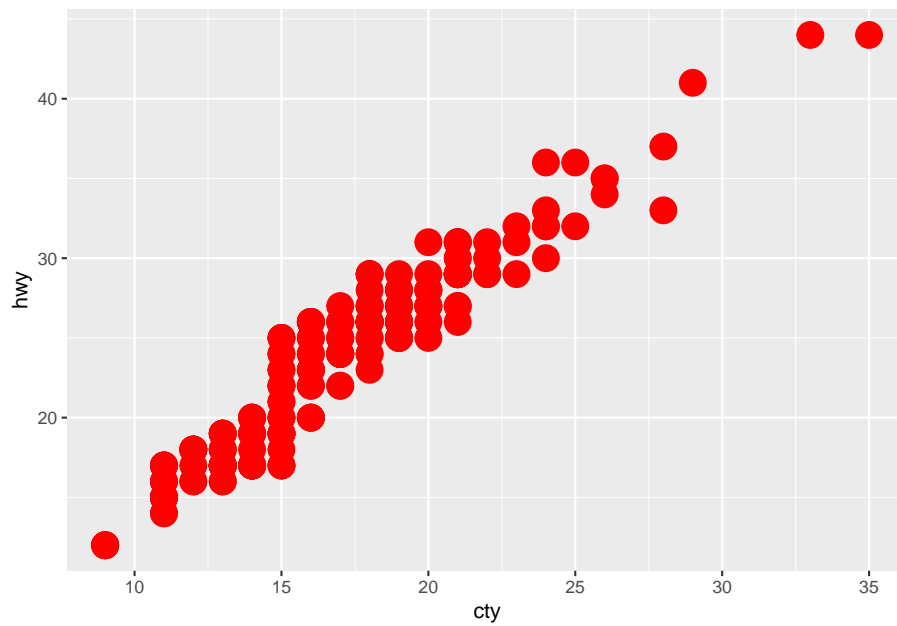
```
ggplot() +  
  geom_point(data = dados, aes(x = cty, y = hwy))
```

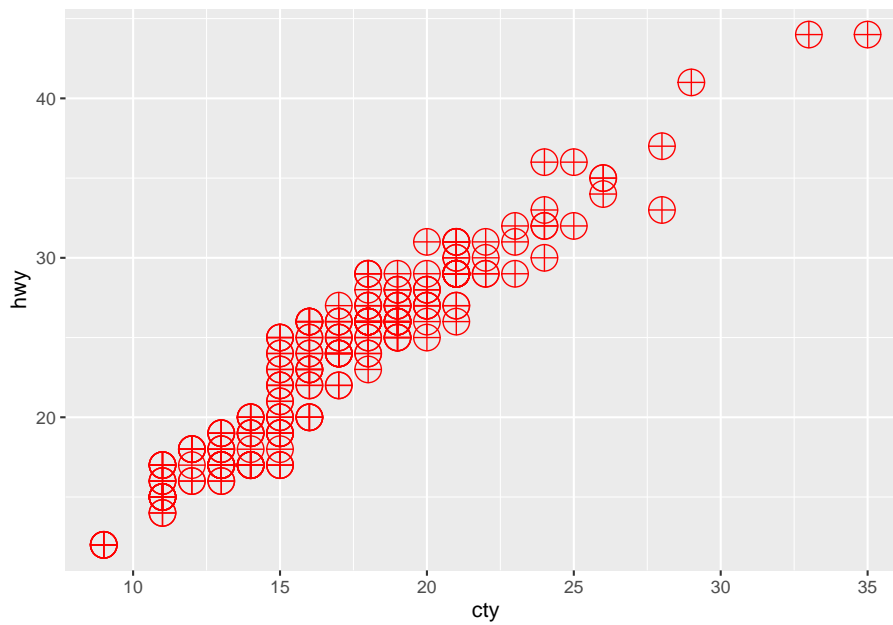
```
# Fim  
  
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point(colour = "red")
```



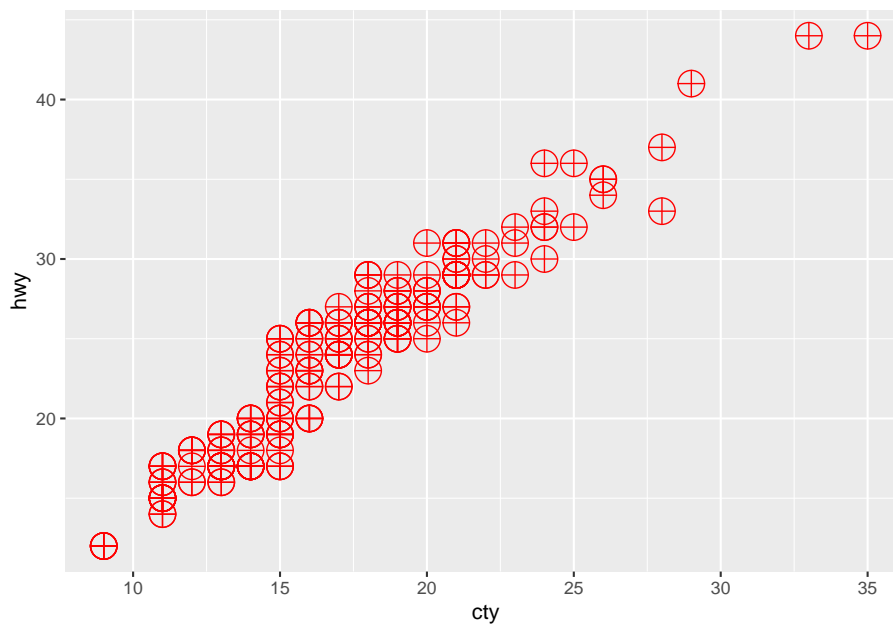
```
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point(colour = "red", size = 6)
```



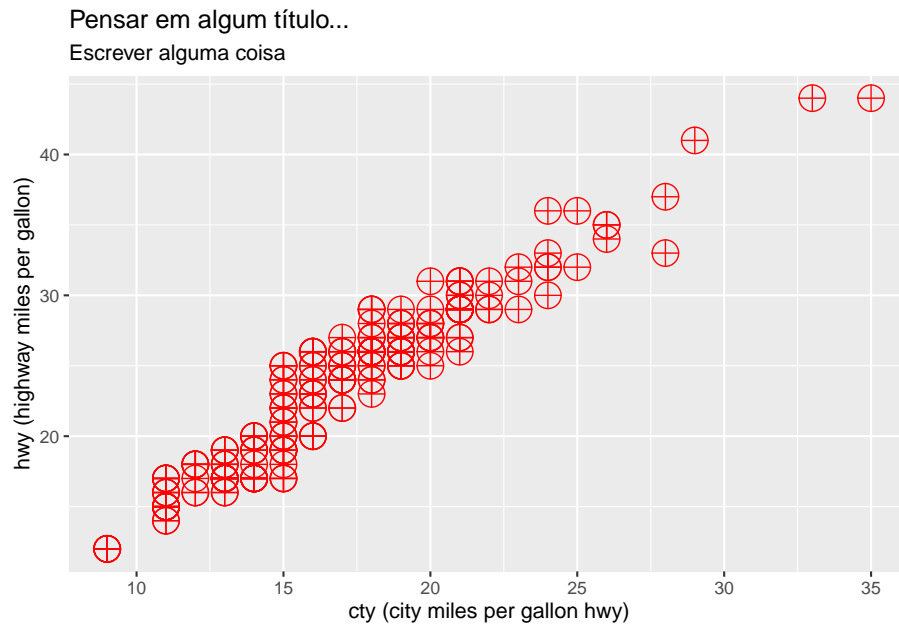
```
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point(colour = "red", size = 6, shape = 10)
```



```
# Alternativa  
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point(colour = "red", size = 6, shape = "circle plus")
```



```
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point(colour = "red", size = 6, shape = 10)+  
  labs(x = "cty (city miles per gallon hwy)",  
       y = "hwy (highway miles per gallon)",  
       title = "Pensar em algum título...",  
       subtitle = "Escrever alguma coisa")
```

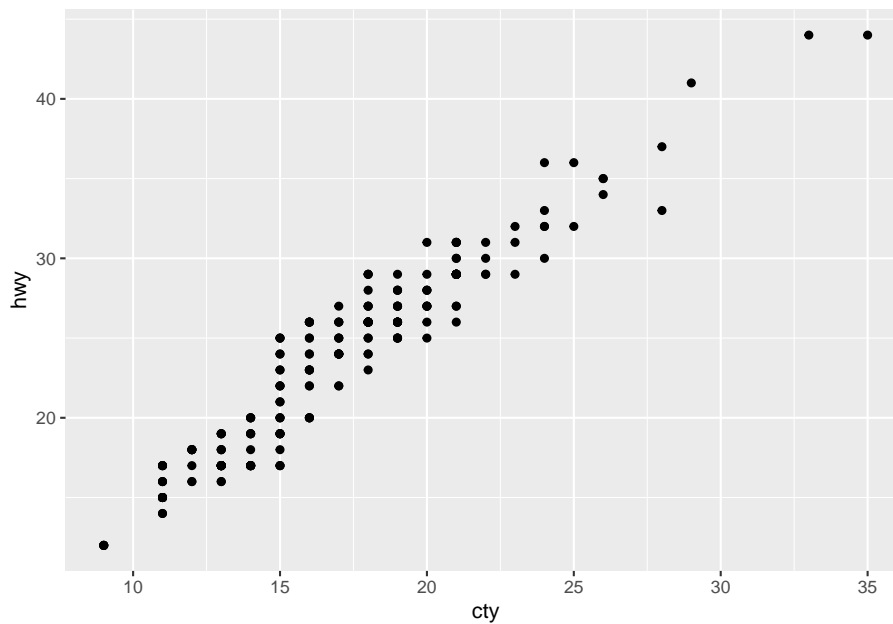


3.3.1 Mais detalhes sobre `geom_point`

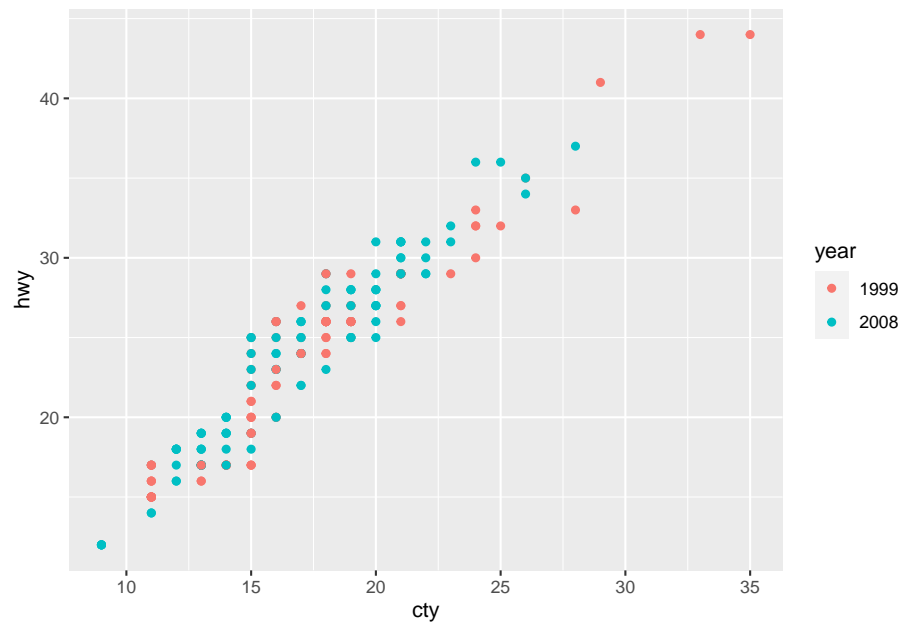
`geom_point()` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke

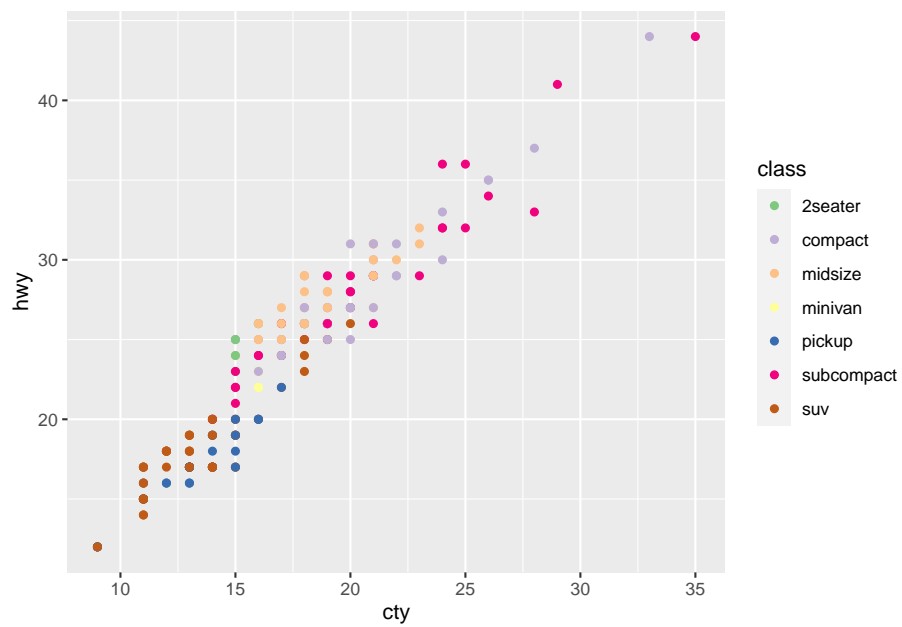
```
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point()
```



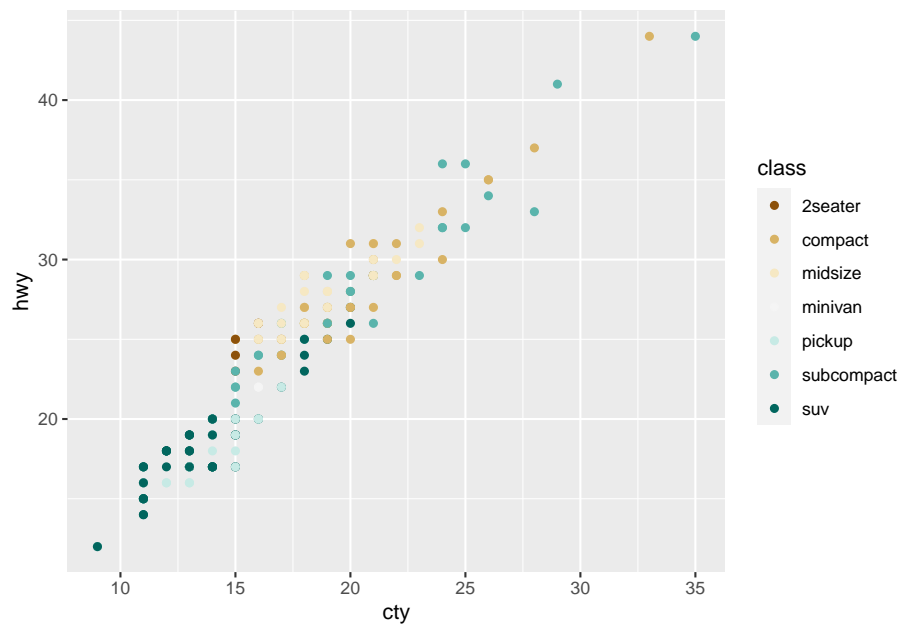
```
ggplot(dados, aes(x = cty, y = hwy, col = factor(year))) +  
  geom_point() +  
  labs(col = "year")
```



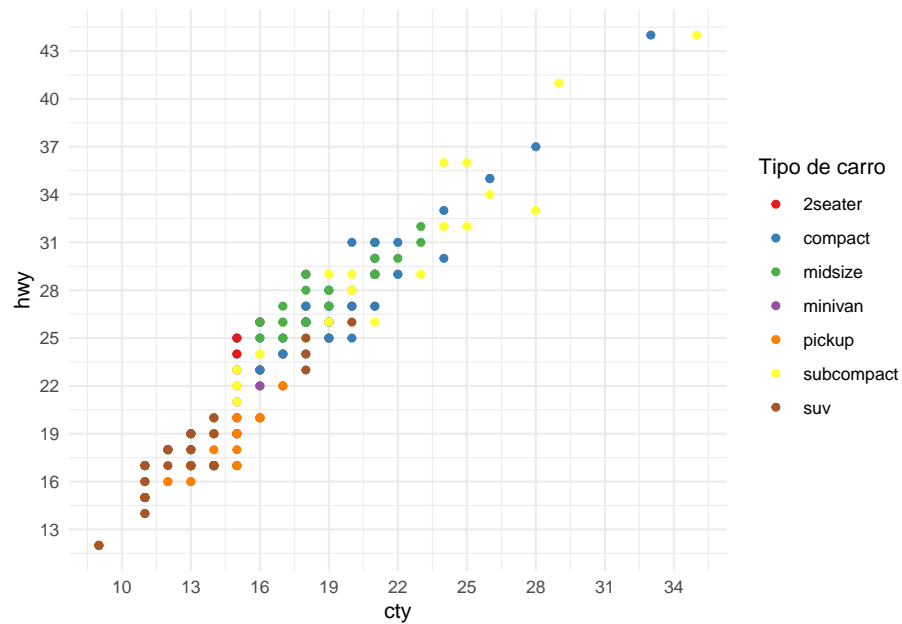
```
# Alternativa
ggplot(dados, aes(x = cty, y = hwy, col = factor(class))) +
  geom_point() +
  labs(col = "class")+
  scale_color_brewer(type = "qual")
```



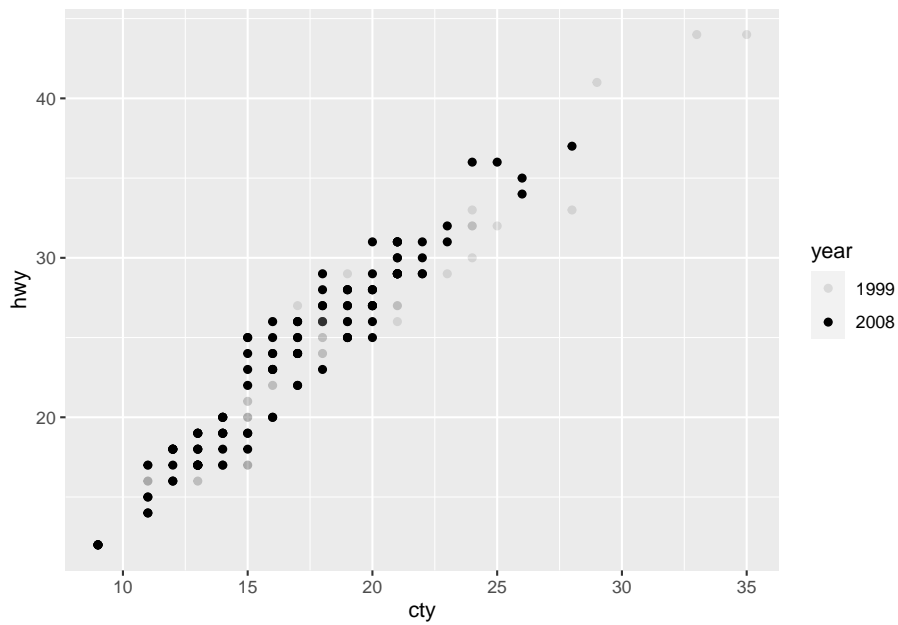
```
ggplot(dados, aes(x = cty, y = hwy, col = factor(class))) +  
  geom_point() +  
  labs(col = "class") +  
  scale_color_brewer(type = "div")
```



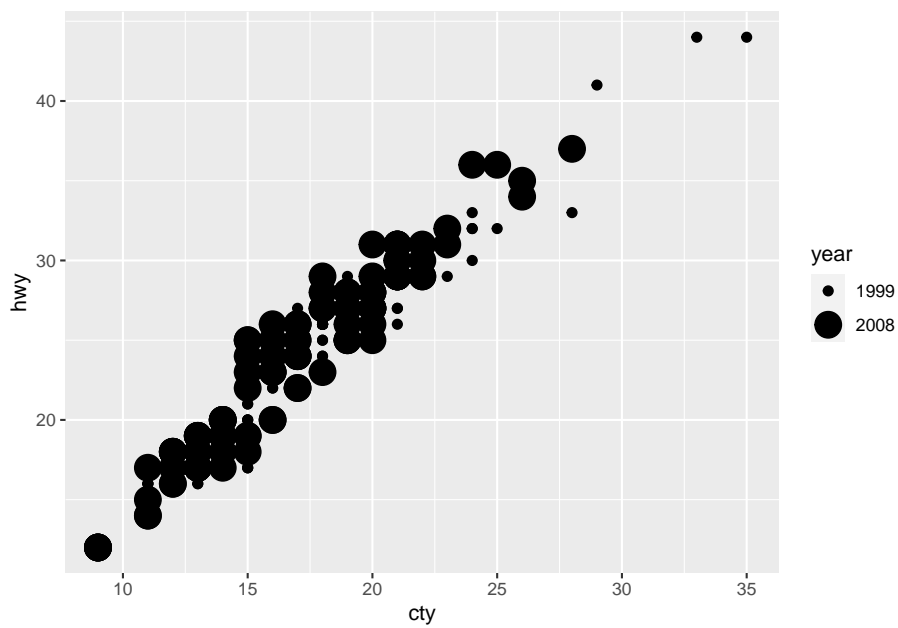
```
ggplot(dados, aes(x = cty, y = hwy, col = factor(class))) +
  geom_point() +
  labs(col = "class")+
  scale_color_brewer(palette = "Set1", name = "Tipo de carro")+
  scale_y_continuous(breaks = seq(10,60,3))+
  scale_x_continuous(breaks = seq(10,40,3))+
  theme_minimal()
```



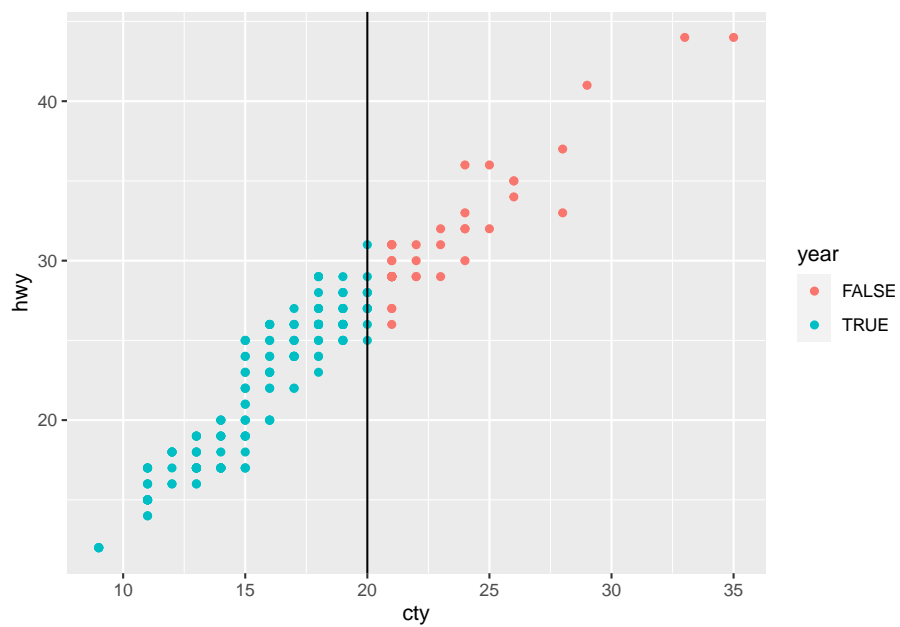
```
ggplot(dados, aes(x = cty, y = hwy, alpha = factor(year))) +
  geom_point() +
  labs(alpha = "year")
```

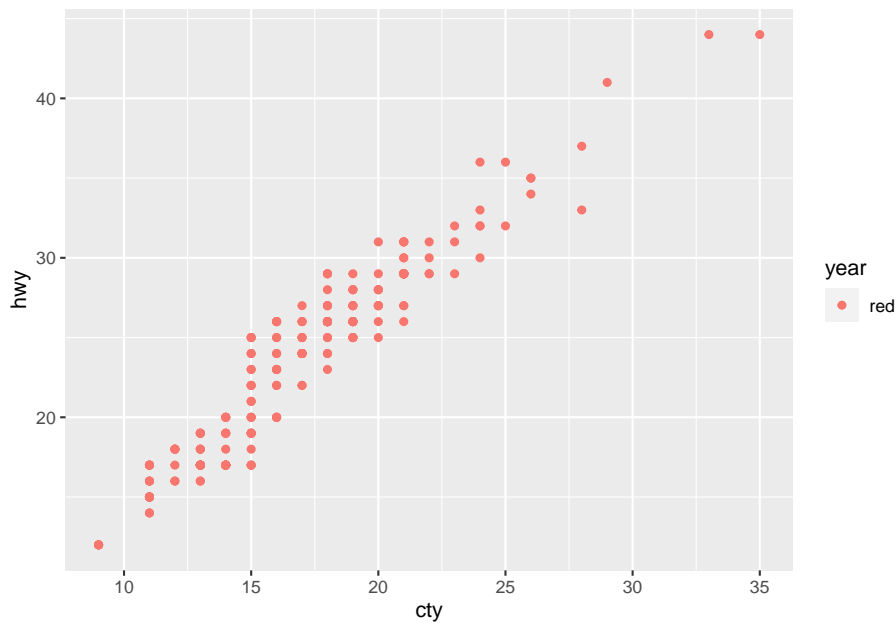
```
ggplot(dados, aes(x = cty, y = hwy, size = factor(year))) +  
  geom_point() +  
  labs(size = "year")
```



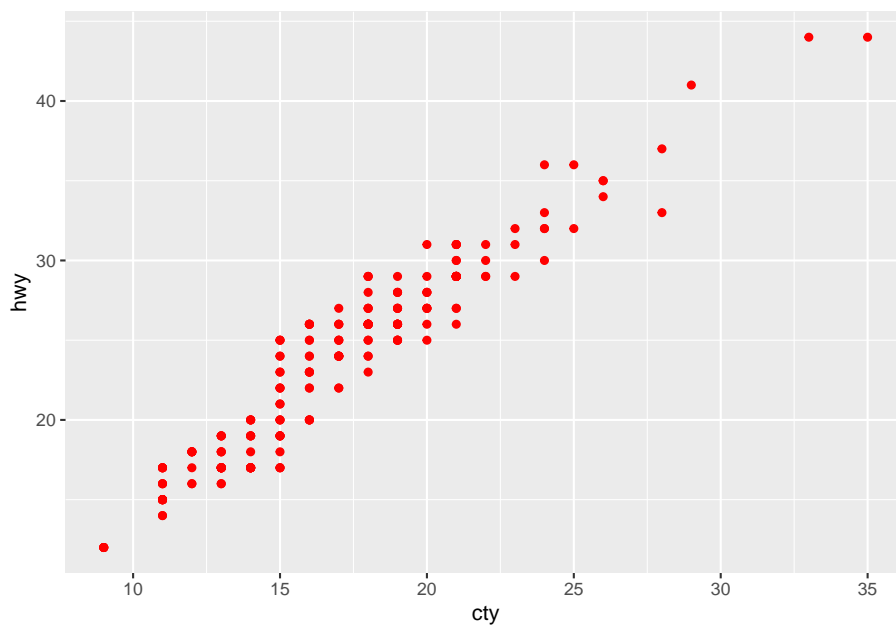
```
# Alternativa
ggplot(dados, aes(x = cty, y = hwy, col = cty <= 20)) +
  geom_point() +
  geom_vline(xintercept = 20)+
  labs(col = "year")
```



```
# Erro comum
ggplot(dados, aes(x = cty, y = hwy, col = "red")) +
  geom_point()+
  labs(col = "year")
```

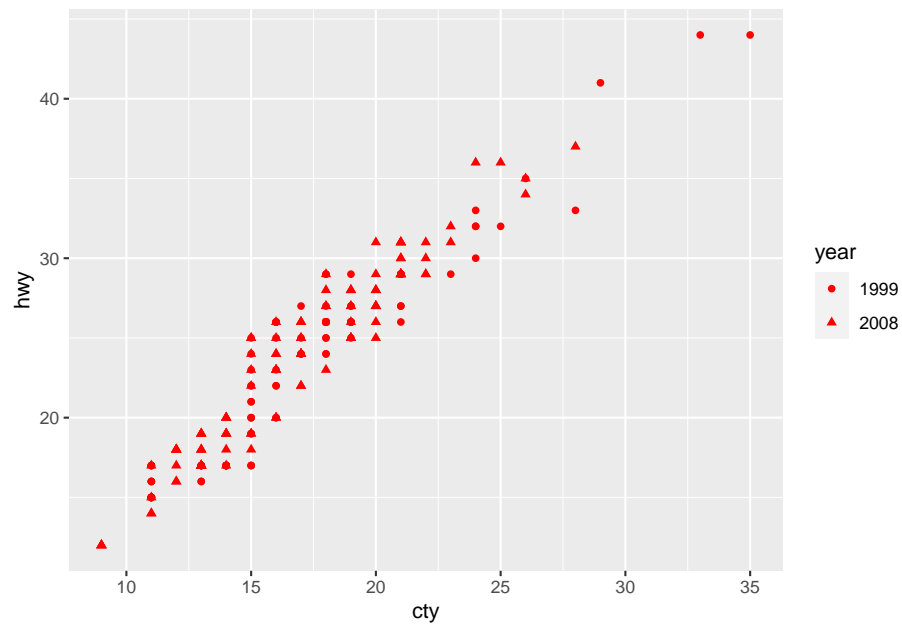


```
ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point(col = "red")+  
  labs(col = "year")
```

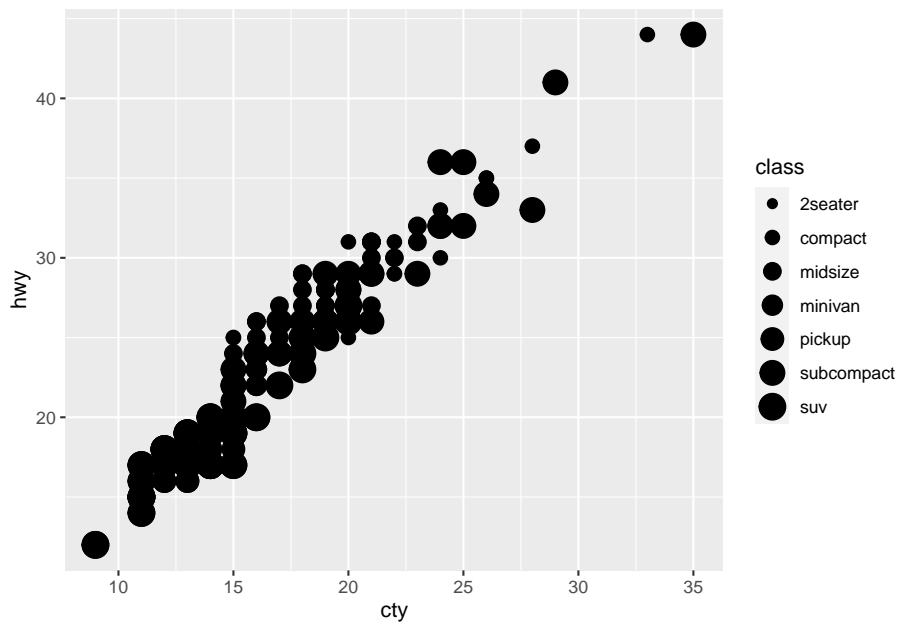


```
# Fim Erro comum
```

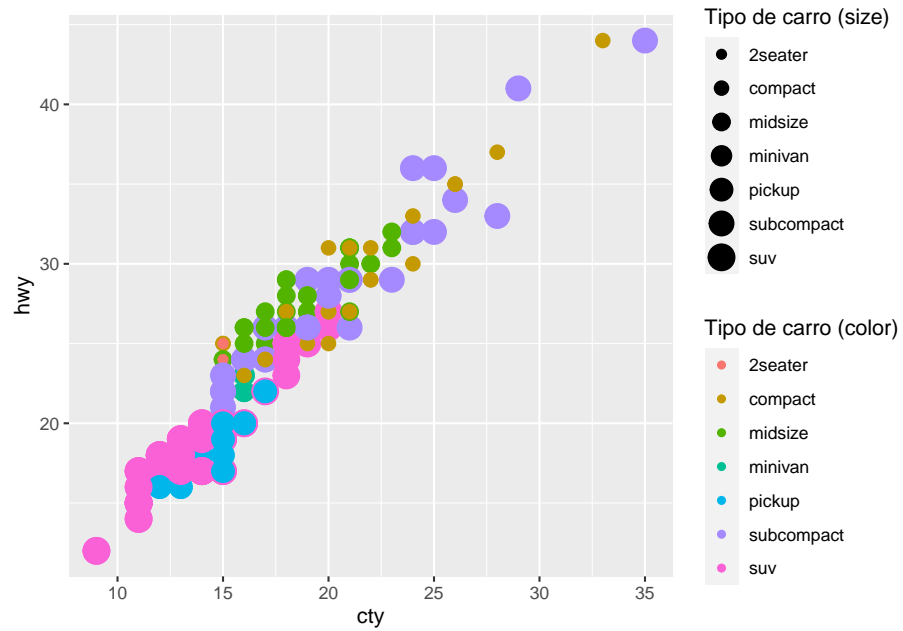
```
ggplot(dados, aes(x = cty, y = hwy, shape = factor(year))) +  
  geom_point(col = "red") +  
  labs(shape = "year")
```



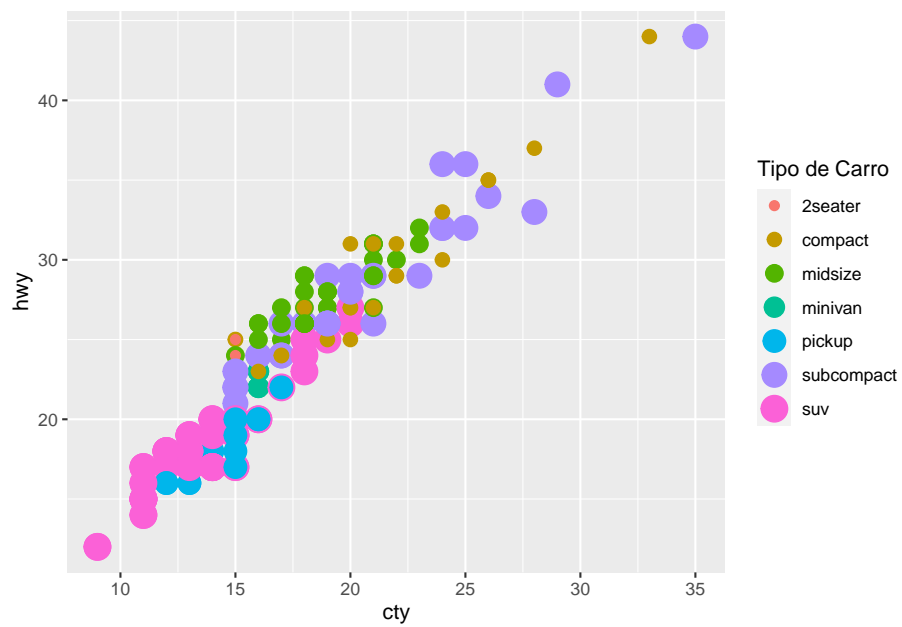
```
ggplot(dados, aes(x = cty, y = hwy, size = class)) +  
  geom_point() +  
  labs(size = "class")
```



```
ggplot(dados, aes(x = cty, y = hwy,  
                  size = class,  
                  col = class)) +  
  geom_point() +  
  guides(colour = guide_legend("Tipo de carro (color)"),  
         size = guide_legend("Tipo de carro (size)"))
```

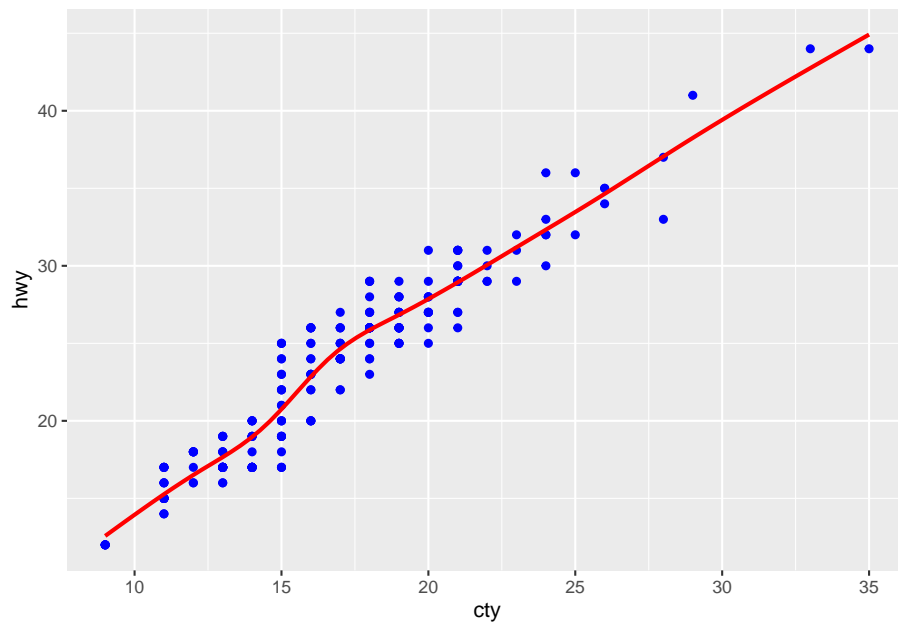


```
ggplot(dados, aes(x = cty, y = hwy,
                  size = class,
                  col = class)) +
  geom_point() +
  labs(col = "Tipo de Carro", size = "Tipo de Carro")+
  guides(col = "legend")
```

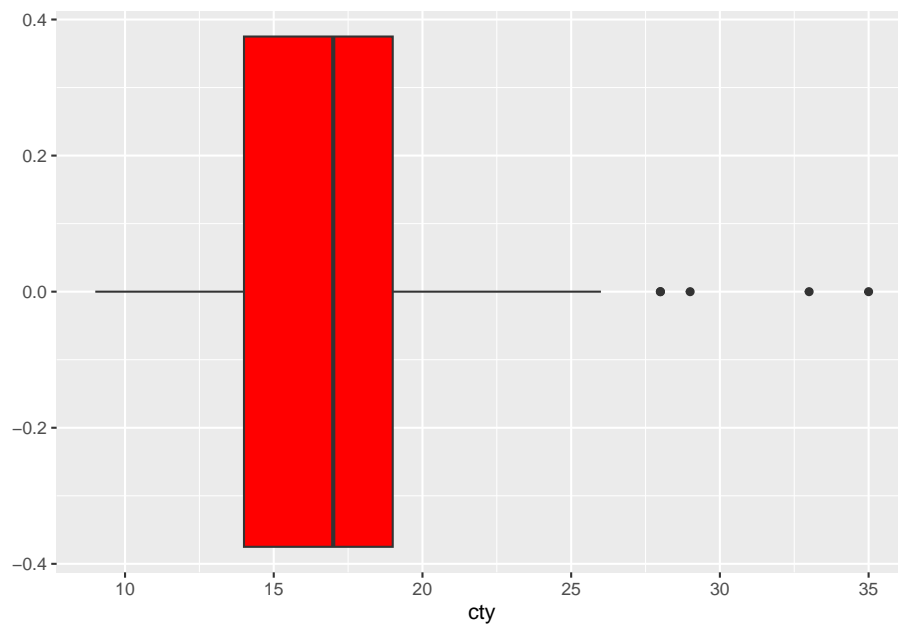


```
v1<- ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point(col = "blue")+
  geom_smooth(method = mgcv::gam,
              formula = y ~ s(x, bs = "cs") ,
              col = "red",
              se = FALSE)

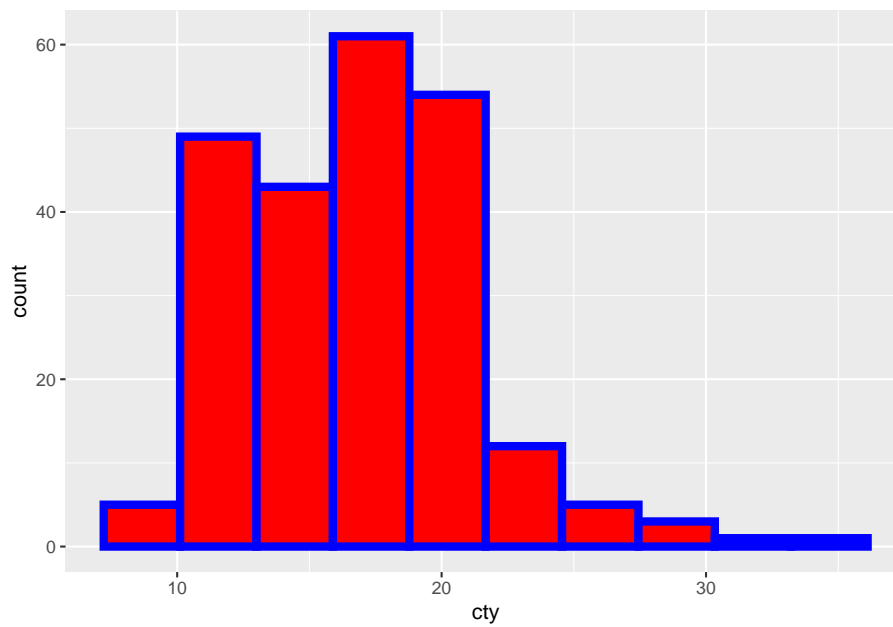
v1
```



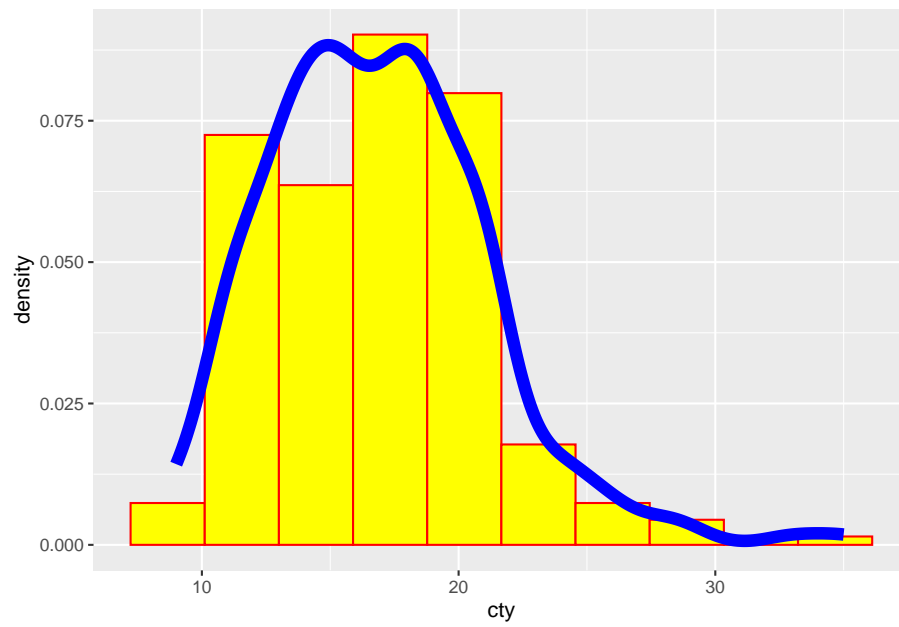
```
v2 <- ggplot(dados, aes(x = cty)) +  
  geom_boxplot(fill = "red")  
v2
```



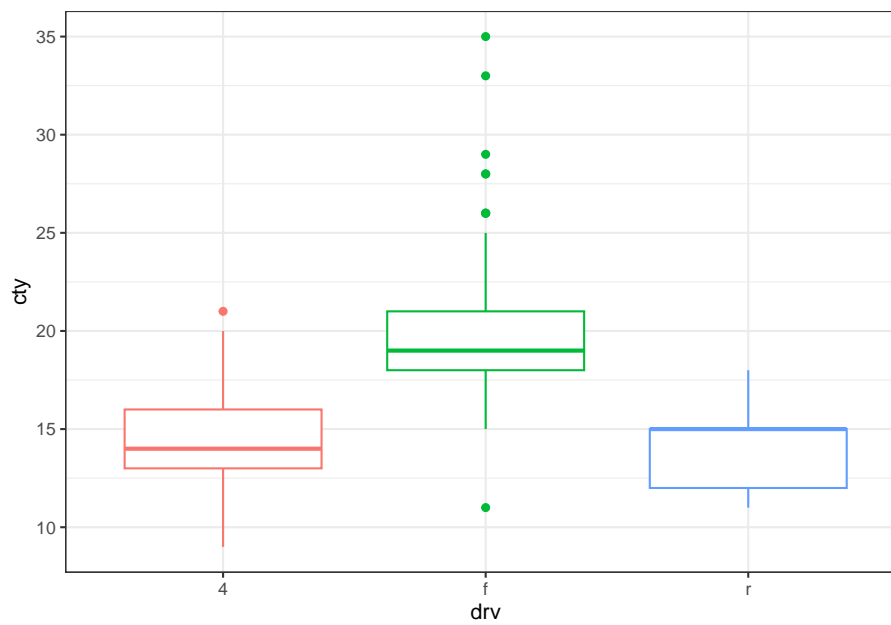

```
v3 <- ggplot(dados, aes(x = cty)) +  
  geom_histogram(bins = 10, fill = "red", col = "blue", lwd=2)  
v3
```



```
v4<- ggplot(dados, aes(x = cty)) +  
  geom_histogram(aes(y = after_stat(density)),  
    bins = 10, fill = "yellow", col = "red") +  
  geom_density(col = "blue", lwd =3)  
v4
```



```
# Adicional (estadística experimental)
ggplot(dados, aes(x = drv, y = cty, col = drv)) +
  geom_boxplot()+
  theme_bw()+
  theme(legend.position = "none")
```



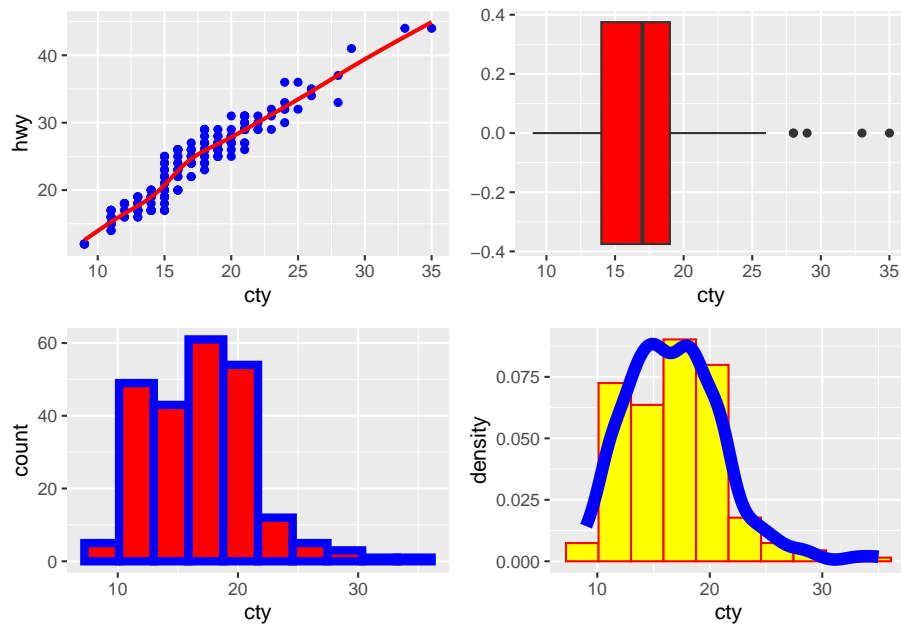
3.5 gridExtra e patchwork

Alguns links

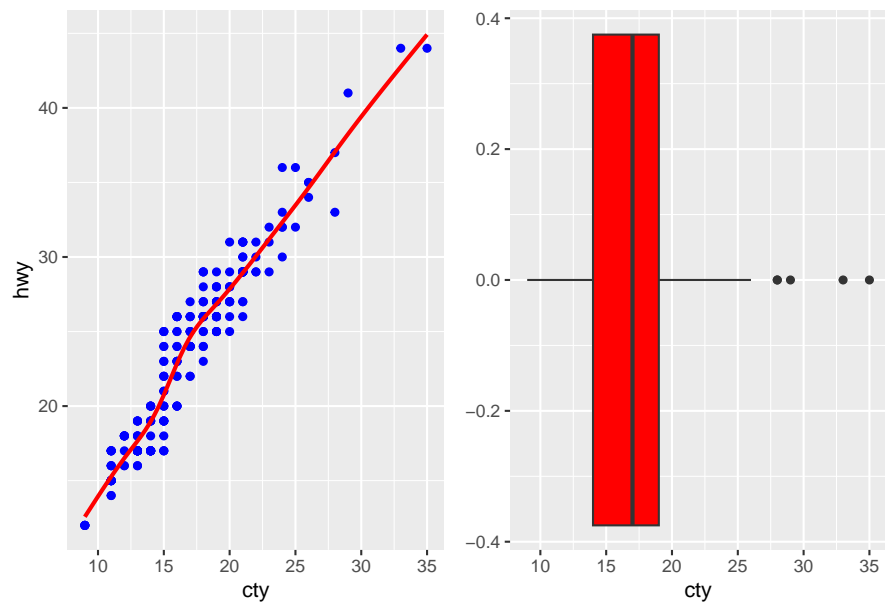
link 1: patchwork

link 2: patchwork

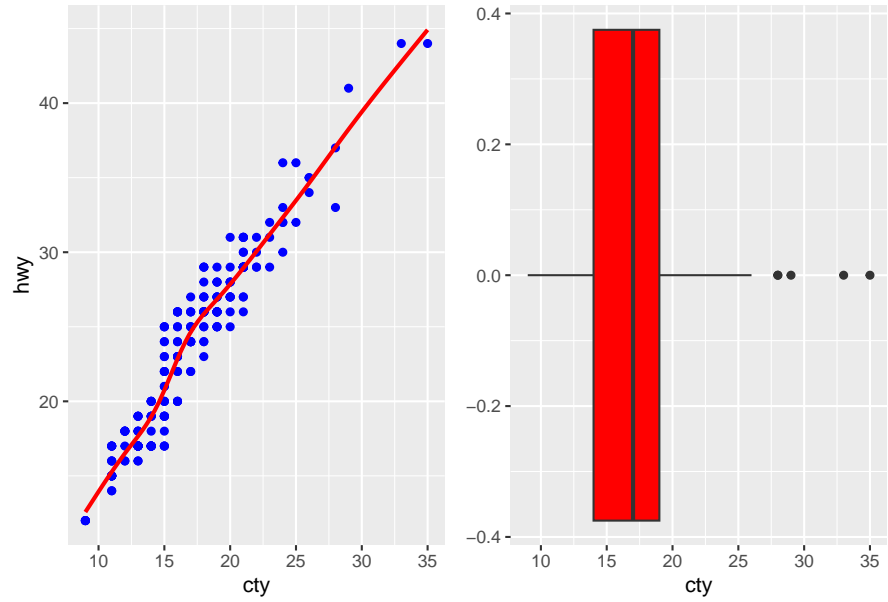
```
# gridExtra
grid.arrange(v1, v2, v3, v4)
```



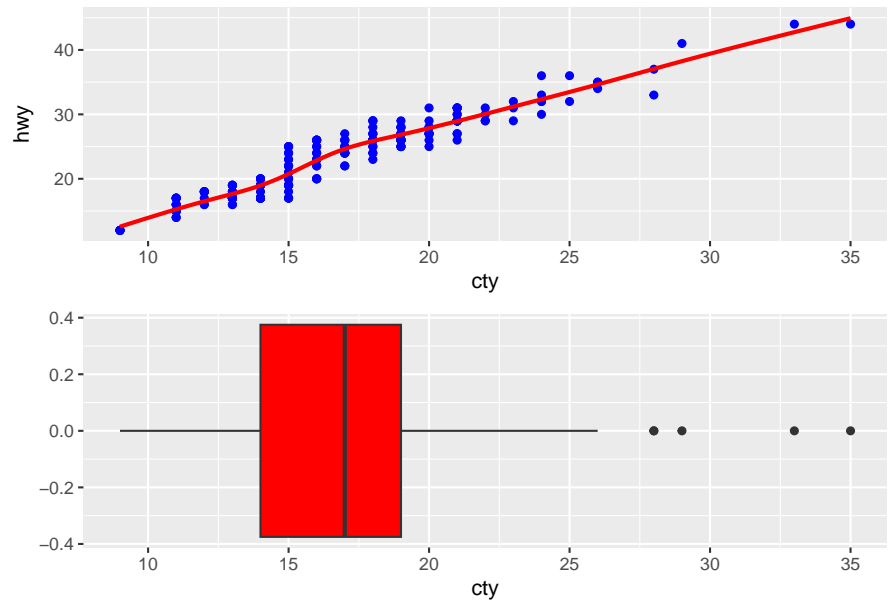
```
# patchwork  
v1 + v2
```



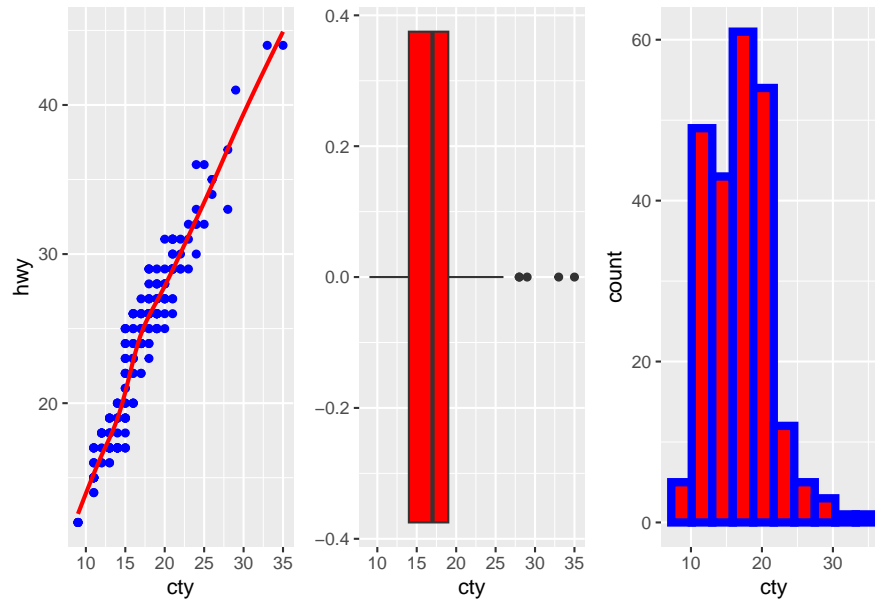
v1 | v2



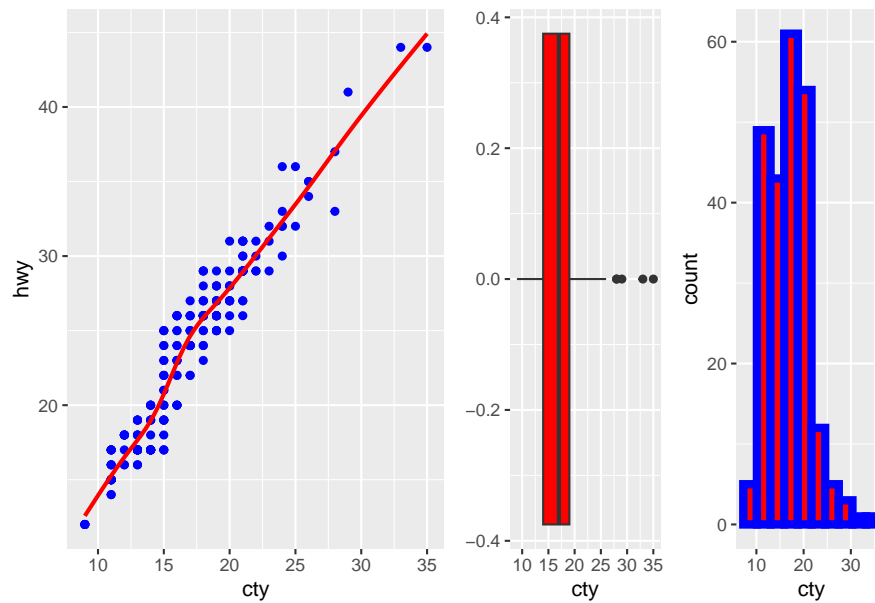
v1 / v2



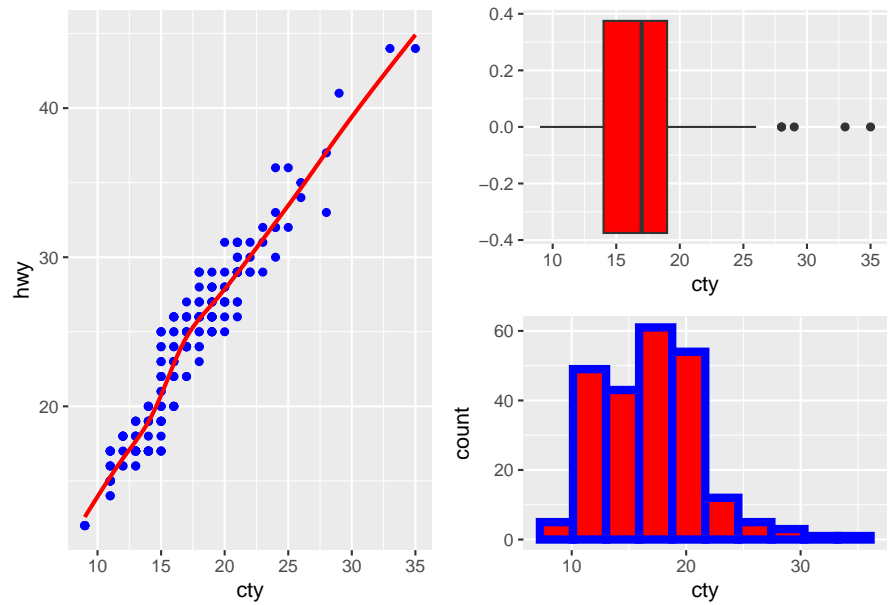
```
v1 + v2 + v3
```



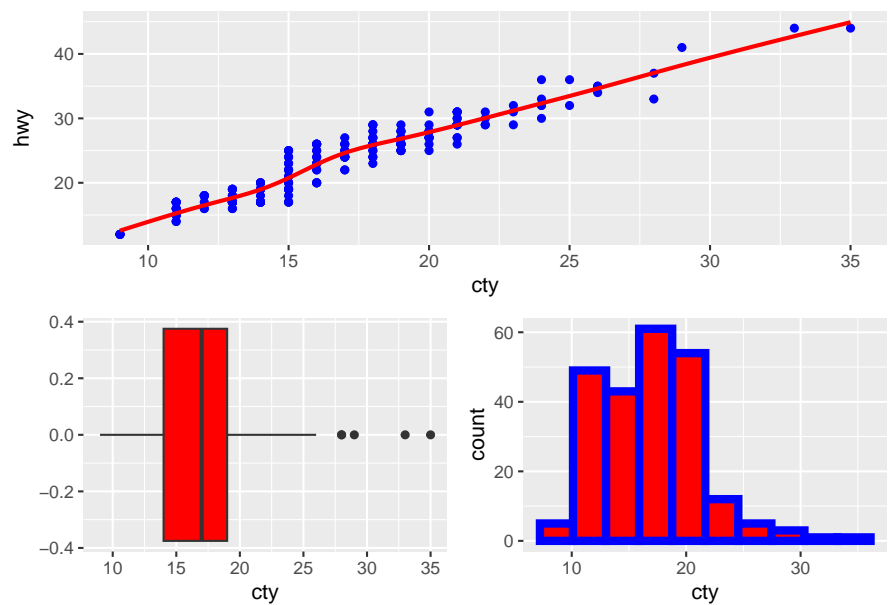
```
v1 + (v2 + v3)
```



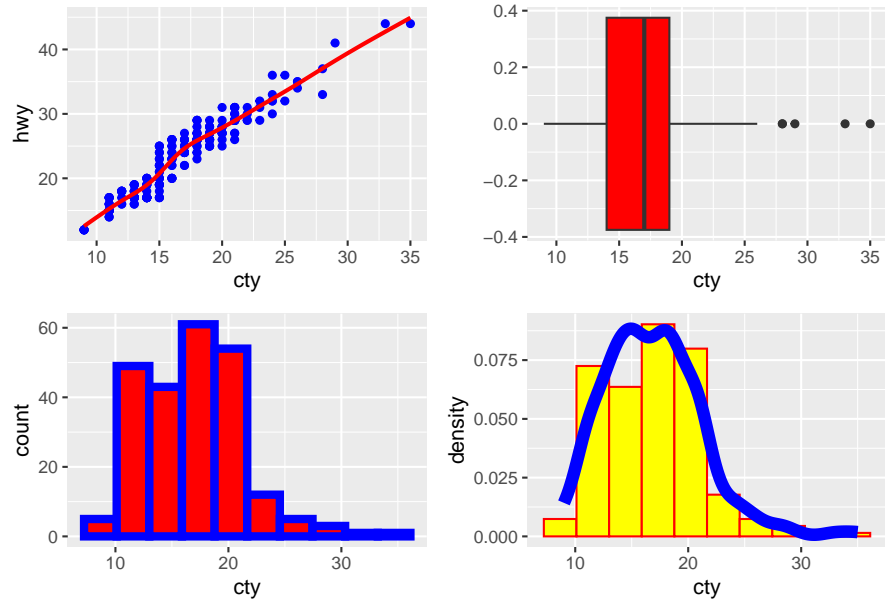
v1 | (v2 / v3)



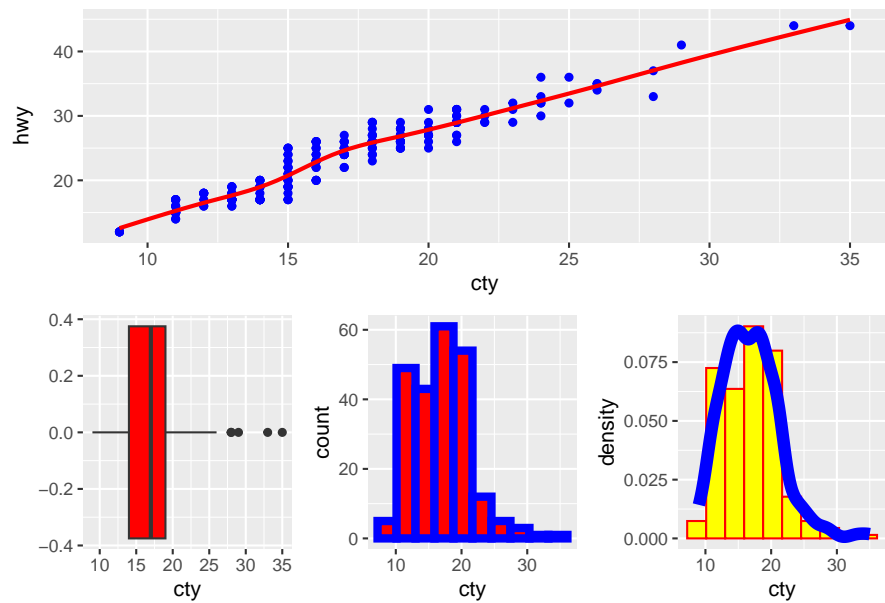
v1 / (v2 + v3)



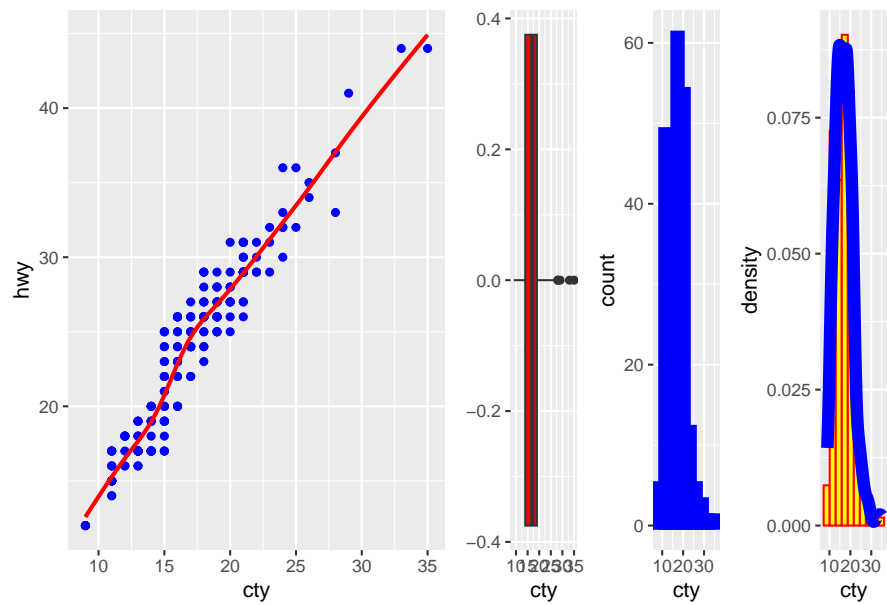
`v1 + v2 + v3 + v4`



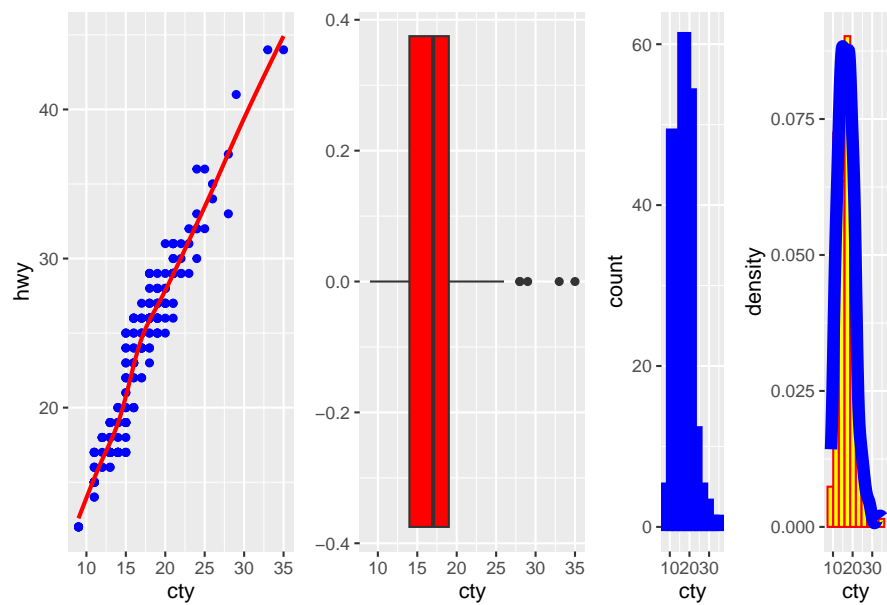
`v1/(v2+v3+v4)`



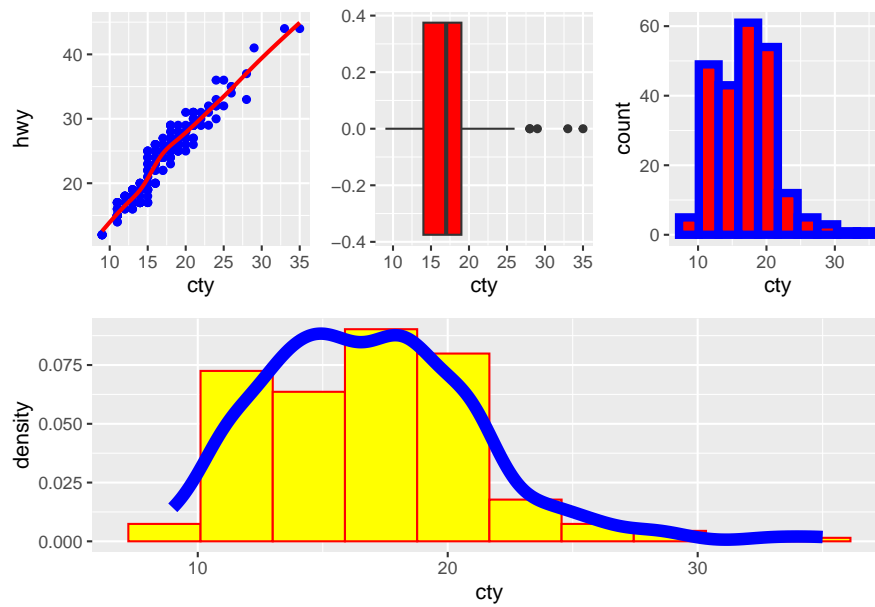
v1 + (v2 + v3 + v4)



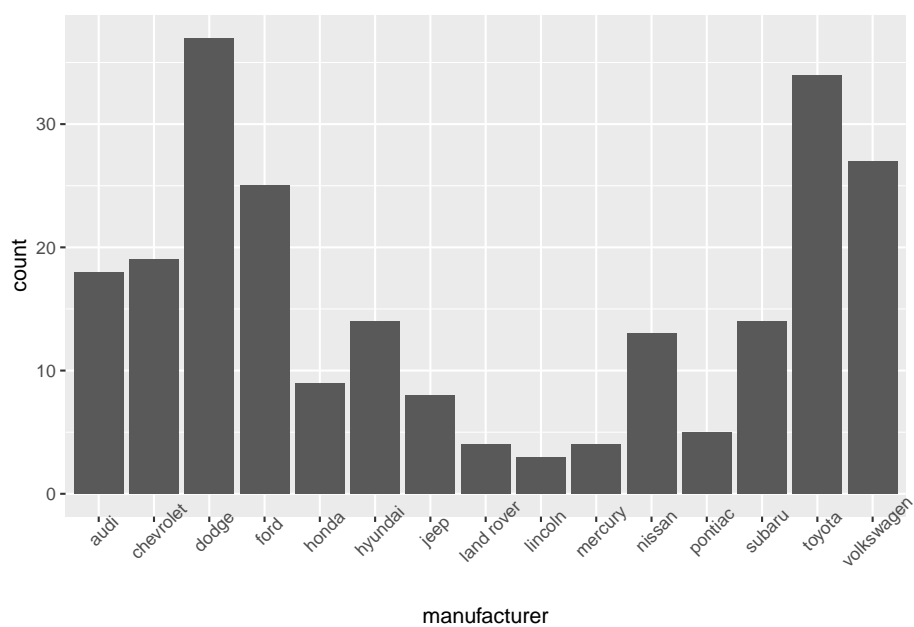
v1 + v2 + (v3 + v4)



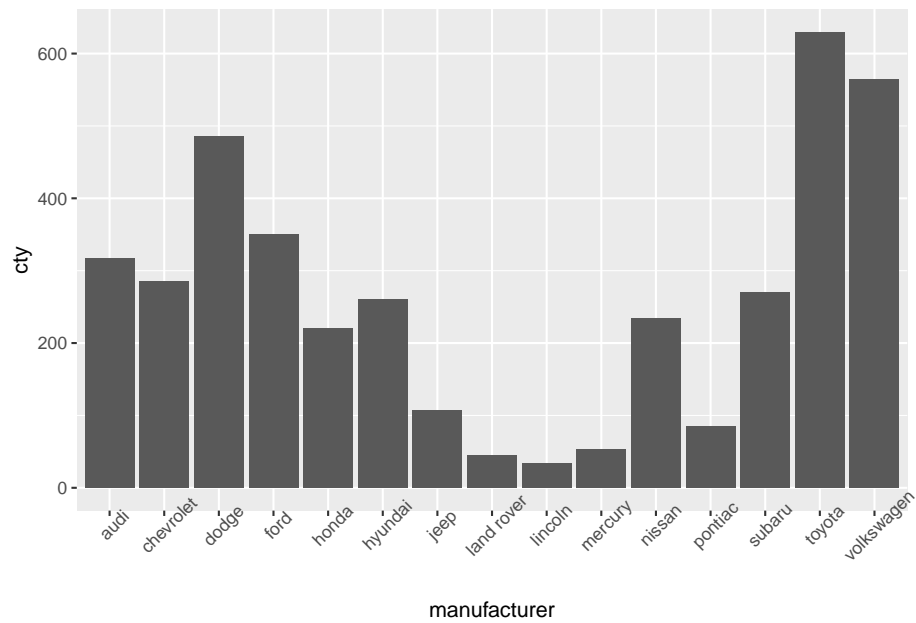
```
(v1 | v2 | v3) / v4
```



```
v5 <- ggplot(dados , aes(x = manufacturer)) +  
  geom_bar()+  
  theme(axis.text.x = element_text(angle = 45))  
v5
```



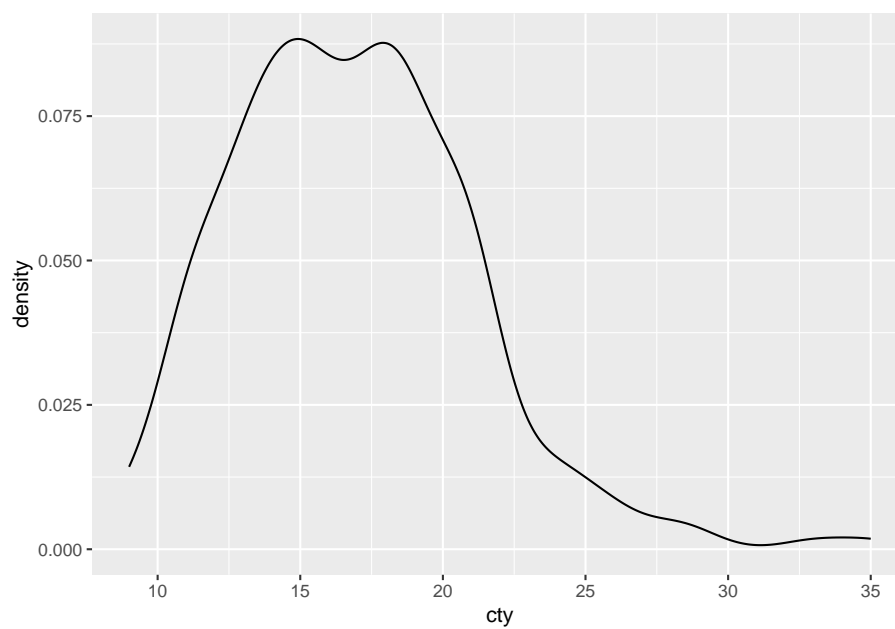
```
# Dúvidas no geom_col
v6 <- ggplot(dados , aes(x = manufacturer, y = cty)) +
  geom_col()+
  theme(axis.text.x = element_text(angle = 45))
v6
```



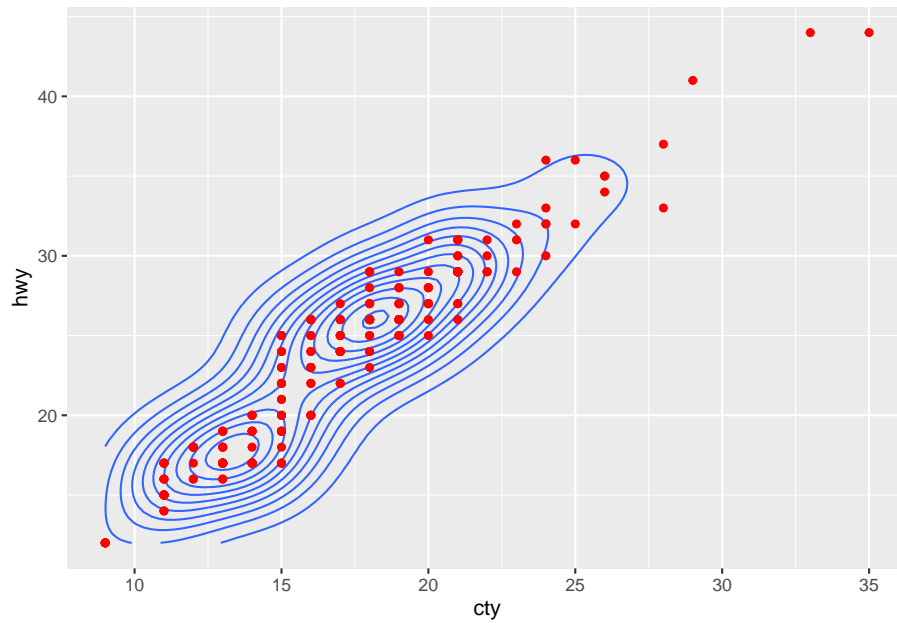
```
dados %>%
  select(manufacturer, cty) %>%
  group_by(manufacturer) %>%
  summarise(soma_total_cty = sum(cty),
            n = n())
```

```
## # A tibble: 15 x 3
##   manufacturer soma_total_cty     n
##   <chr>          <int> <int>
## 1 audi           317     18
## 2 chevrolet      285     19
## 3 dodge          486     37
## 4 ford           350     25
## 5 honda          220      9
## 6 hyundai        261     14
## 7 jeep           108      8
## 8 land rover      46      4
## 9 lincoln         34      3
## 10 mercury        53      4
## 11 nissan          235     13
## 12 pontiac         85      5
## 13 subaru          270     14
## 14 toyota          630     34
## 15 volkswagen      565     27
```

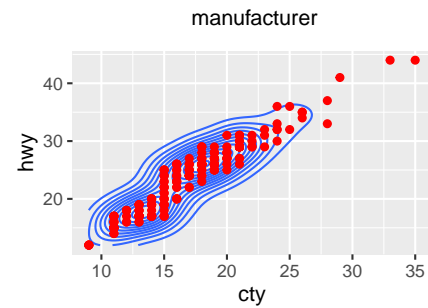
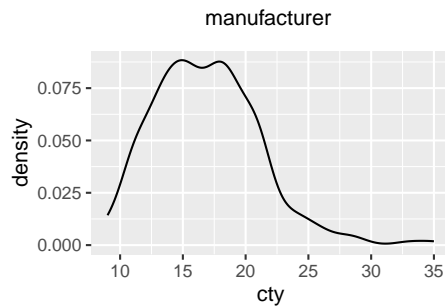
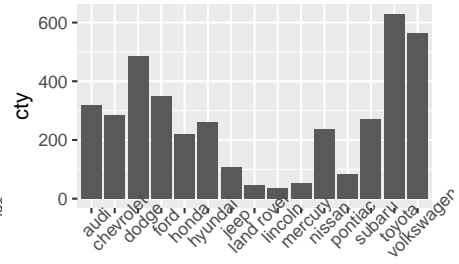
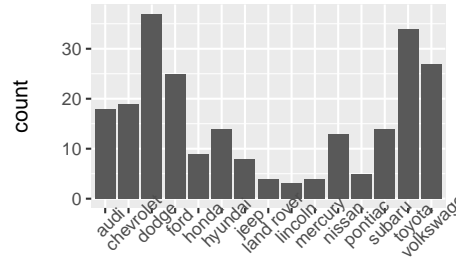
```
# dados %>%  
#   filter(manufacturer == "audi") %>%  
#   select(cty) %>%  
#   sum()  
v7 <- ggplot(dados , aes(x = cty)) +  
  geom_density()  
v7
```



```
v8 <- ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_density2d()+  
  geom_point(colour = "red")  
v8
```



```
(v5+v6)/ (v7 + v8)
```



```
# Deixar pra depois...
```

```
dados %>%
```

```
  select(manufacturer, hwy, year) %>%
```

```
filter(manufacturer == "audi", year == "1999") %>%
summarise(media = max(hwy))
```

```
## # A tibble: 1 x 1
##   media
##   <int>
## 1     29
```

```
# plotly
ggplotly(
ggplot(dados, aes(x = manufacturer, y = hwy, fill = factor(year))) +
  geom_col(position = "dodge") +
  labs(fill = "year") +
  theme(axis.text.x = element_text(angle = 45)))
```

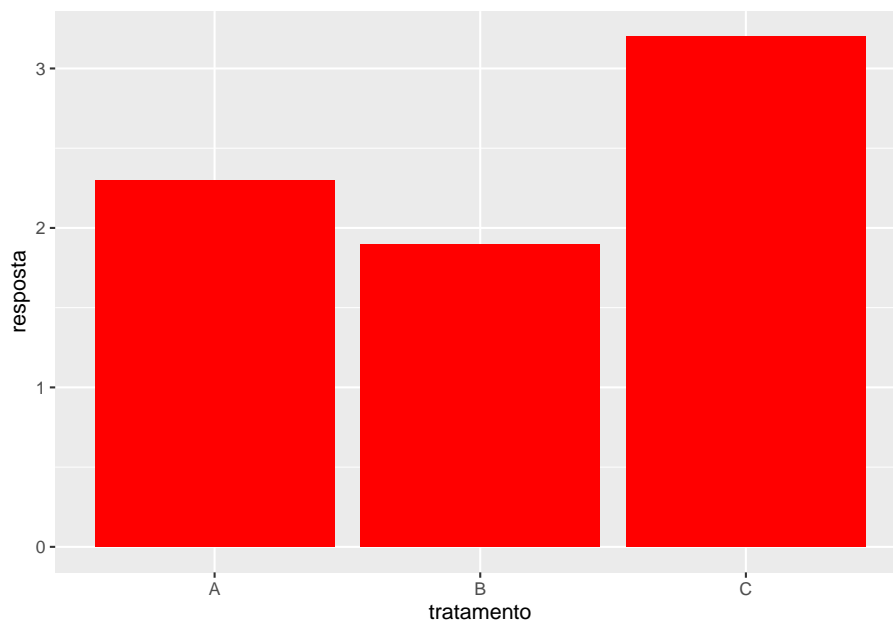
```
dados %>% select(manufacturer, hwy, year) %>%
  group_by(manufacturer, year) %>%
  summarise(media = mean(hwy))
```

```
# Para pensar
```

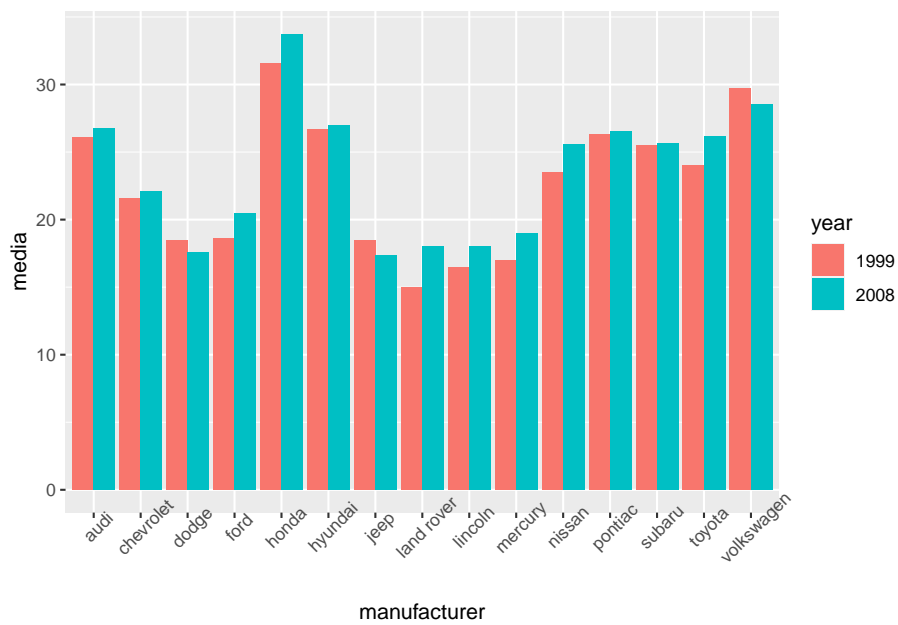
```
(dados_trat <- data.frame(tratamento = LETTERS[1:3],
  resposta = c(2.3, 1.9, 3.2)))
```

```
##   tratamento resposta
## 1          A        2.3
## 2          B        1.9
## 3          C        3.2
```

```
ggplot(dados_trat, aes(tratamento, resposta)) +
  geom_col(fill = "red")
```

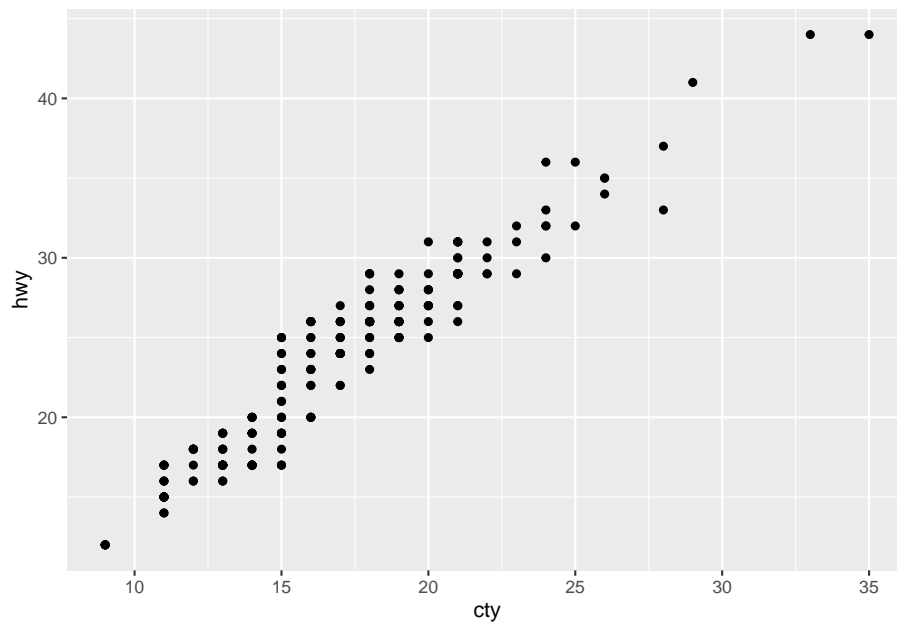


```
# Mais detalhes...
dados %>% select(manufacturer, hwy, year) %>%
  group_by(manufacturer, year) %>%
  summarise(media = mean(hwy), .groups = "drop") %>%
  ggplot(aes(x = manufacturer, y = media, fill = factor(year)))+
  geom_col(position = "dodge")+
  labs(fill = "year") +
  theme(axis.text.x = element_text(angle = 45))
```

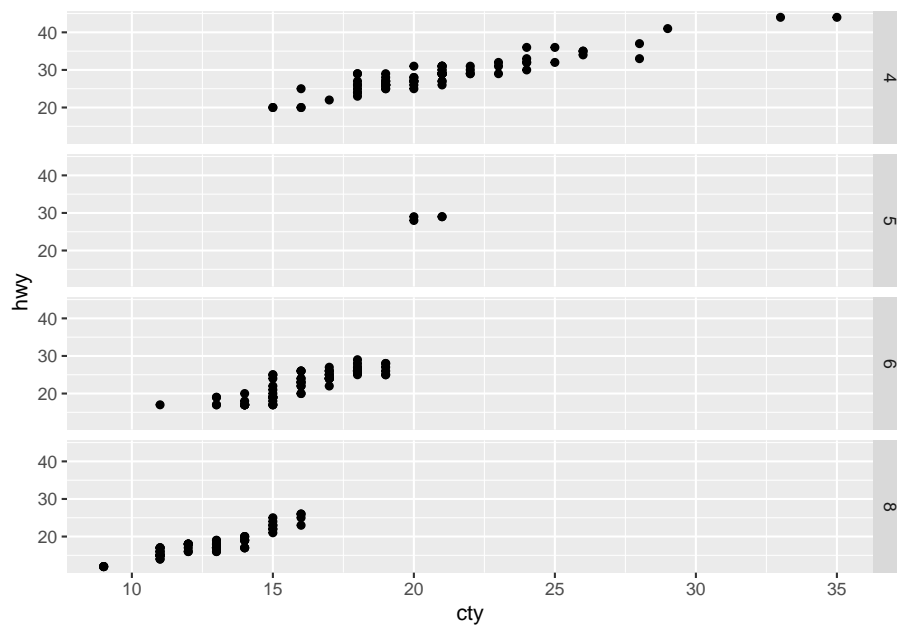



```
p1<- ggplot(dados, aes(x = cty, y = hwy)) +  
  geom_point()  
p1
```

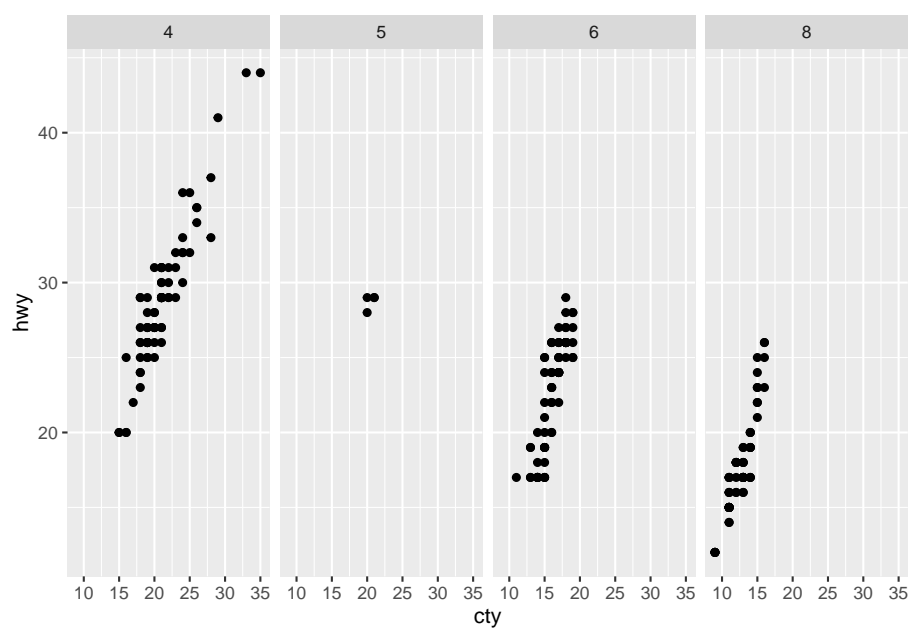
3.7 facet_grid, facet_wrap



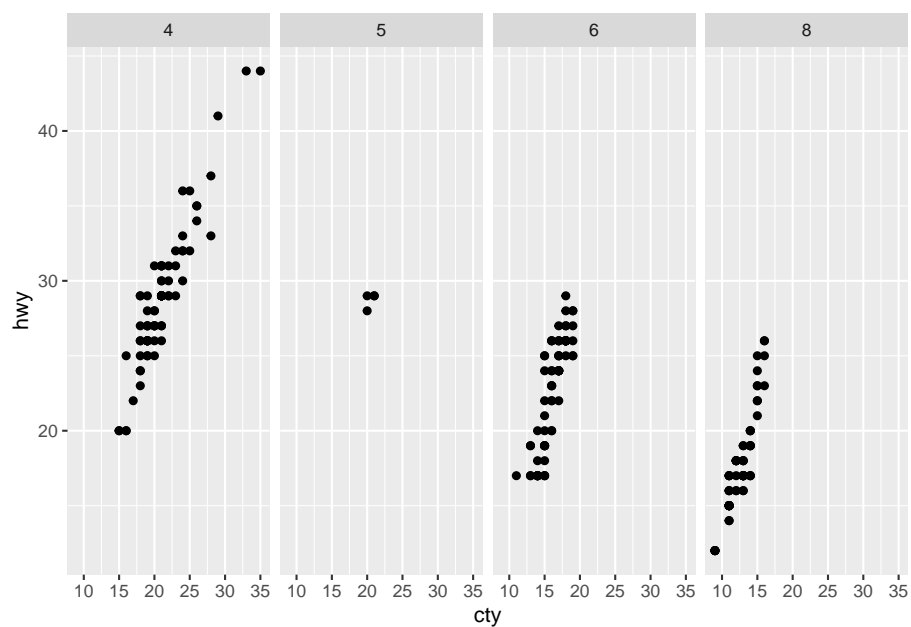
```
p1 + facet_grid(rows = vars(cyl))
```



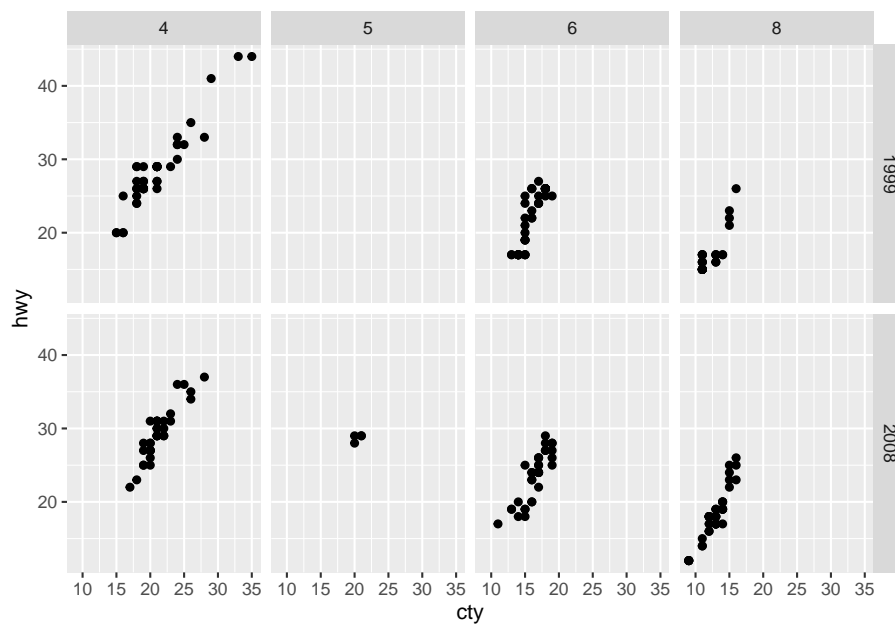
```
p1 + facet_grid(cols = vars(cyl))
```



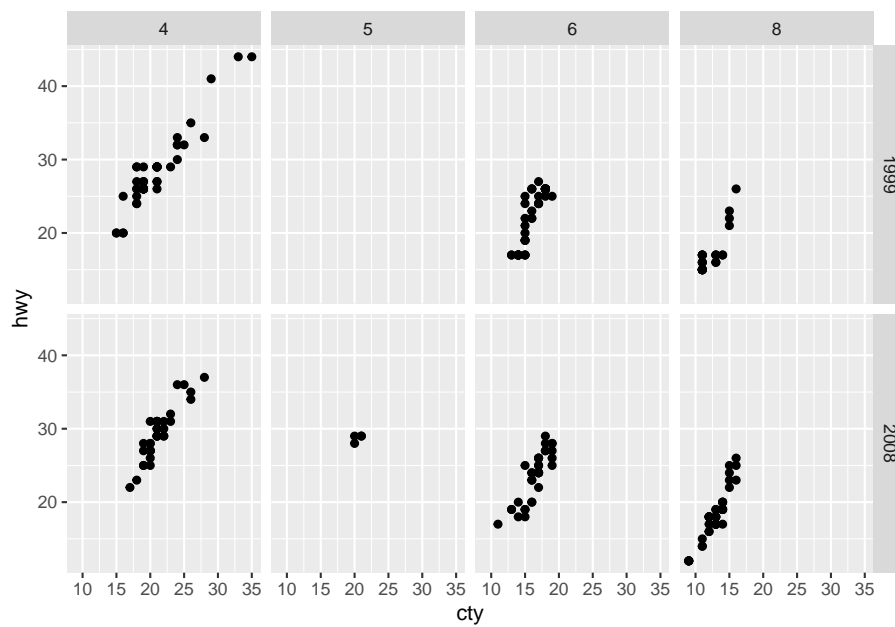
```
p1 + facet_grid(~cyl)
```



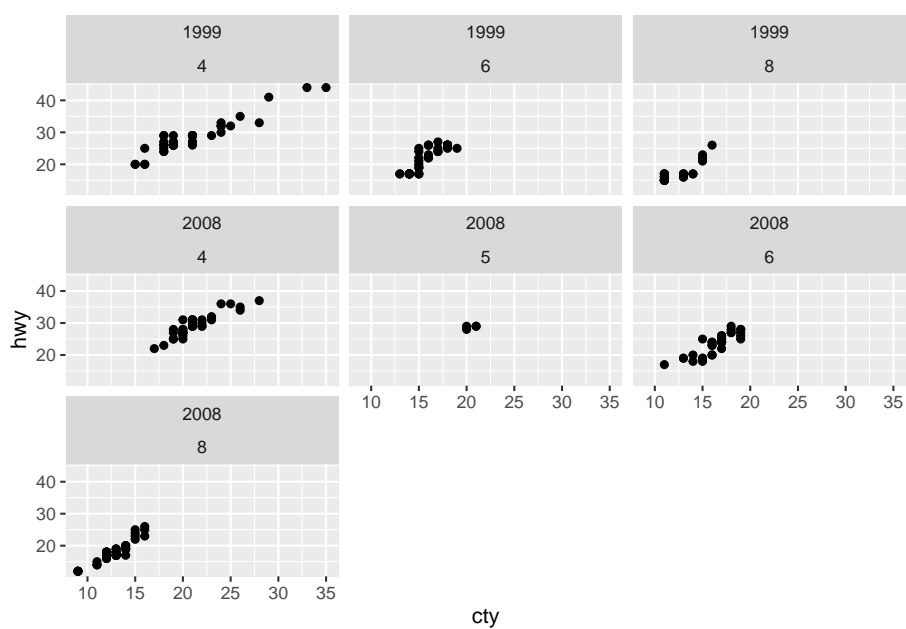
```
p1 + facet_grid(rows = vars(year), cols =vars(cyl))
```



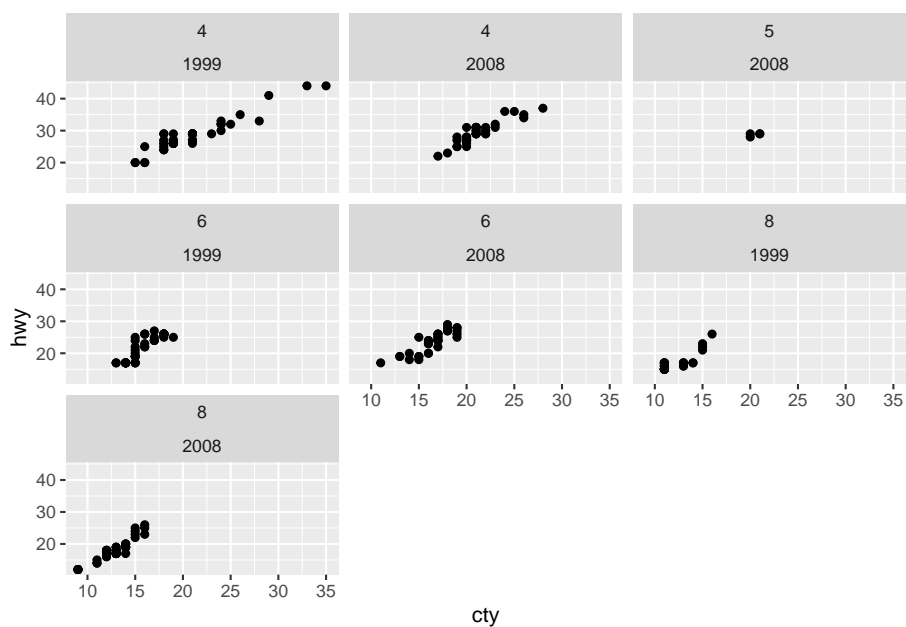
```
p1 + facet_grid(year~cyl)
```



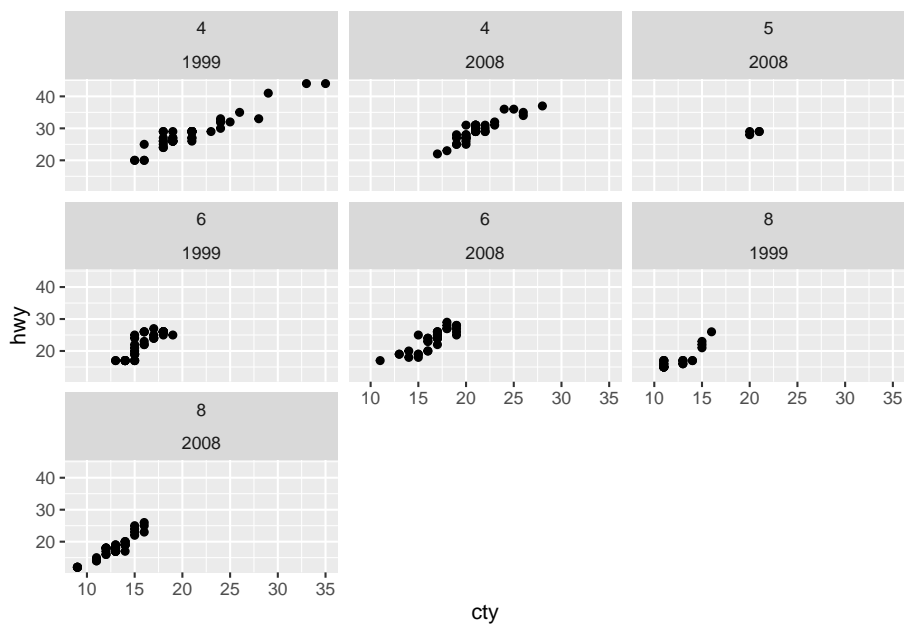
```
p1 + facet_wrap(year ~ cyl)
```



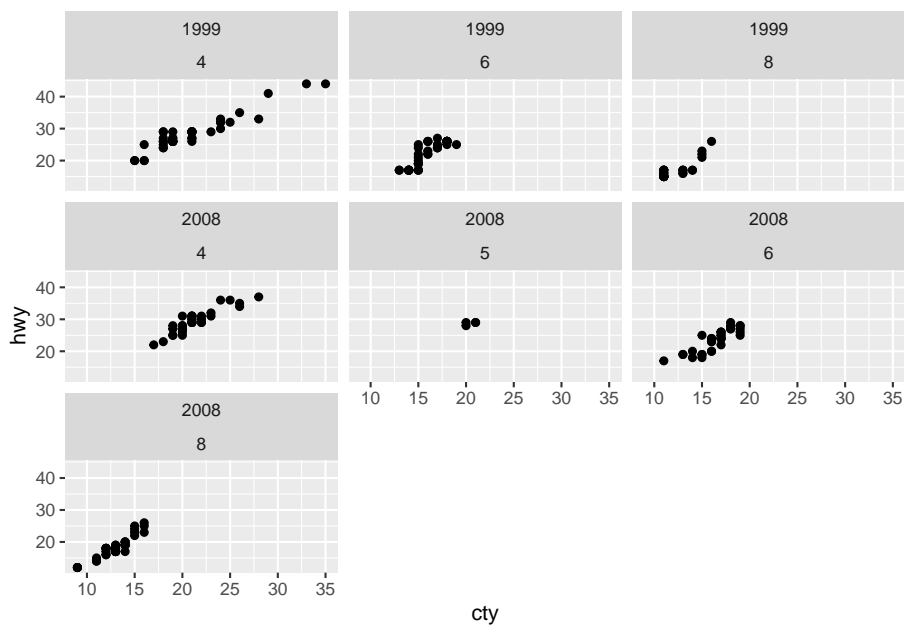
```
p1 + facet_wrap(cyl ~ year)
```



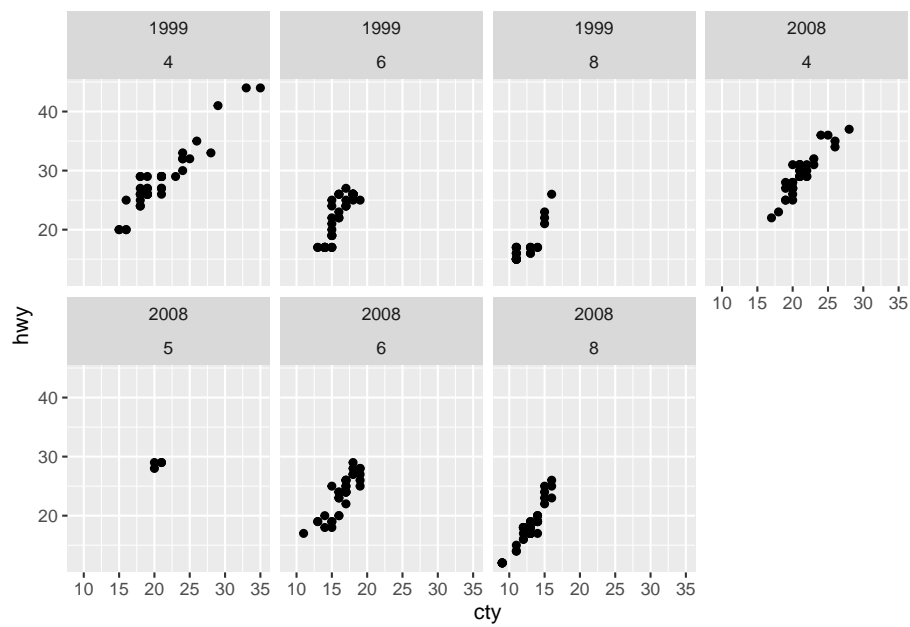
```
p1 + facet_wrap(~cyl + year)
```



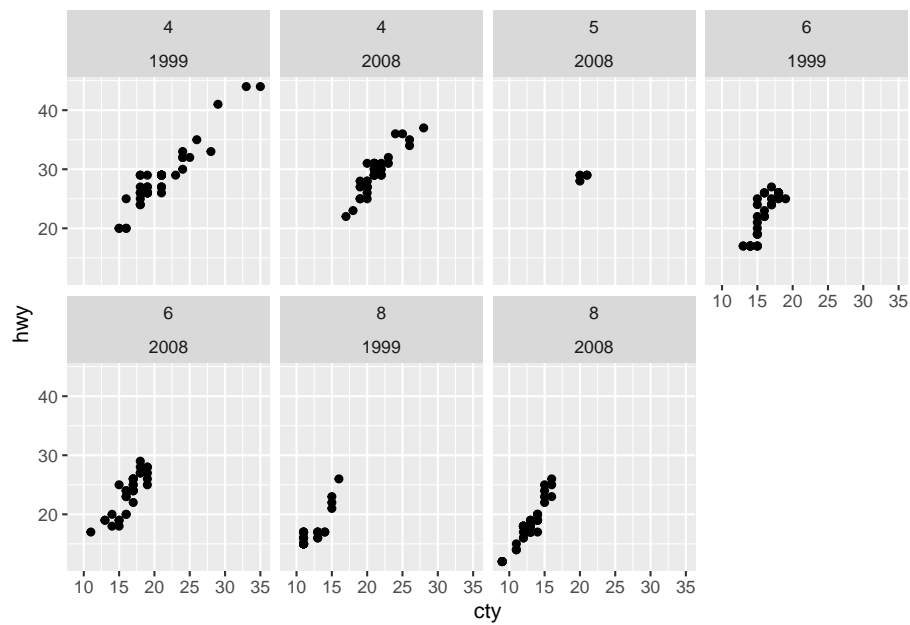
```
p1 + facet_wrap(~year + cyl)
```



```
p1 + facet_wrap(year ~ cyl, ncol = 4)
```



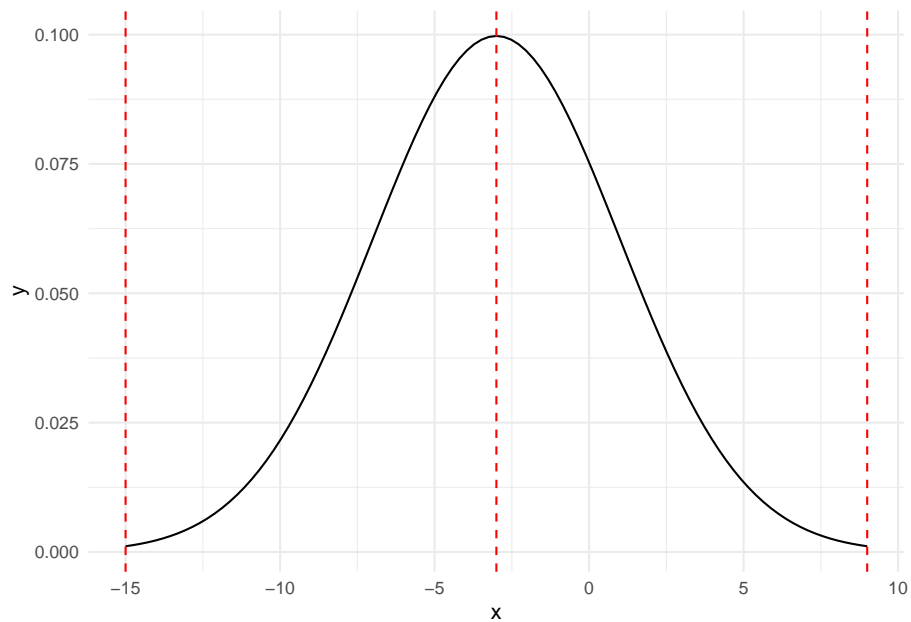
```
p1 + facet_wrap(cyl ~ year, ncol = 4)
```



```

a<- -3 # média
b<- 4 # desv. padrão
ggplot(data.frame(x = c(a - 3*b, a + 3*b)), aes(x)) +
  stat_function(fun = dnorm, args = list(mean = a, sd = b))+
  geom_vline(xintercept = c(a - 3*b, a, a + 3*b), col = "red", lty = 2)+
  theme_minimal()

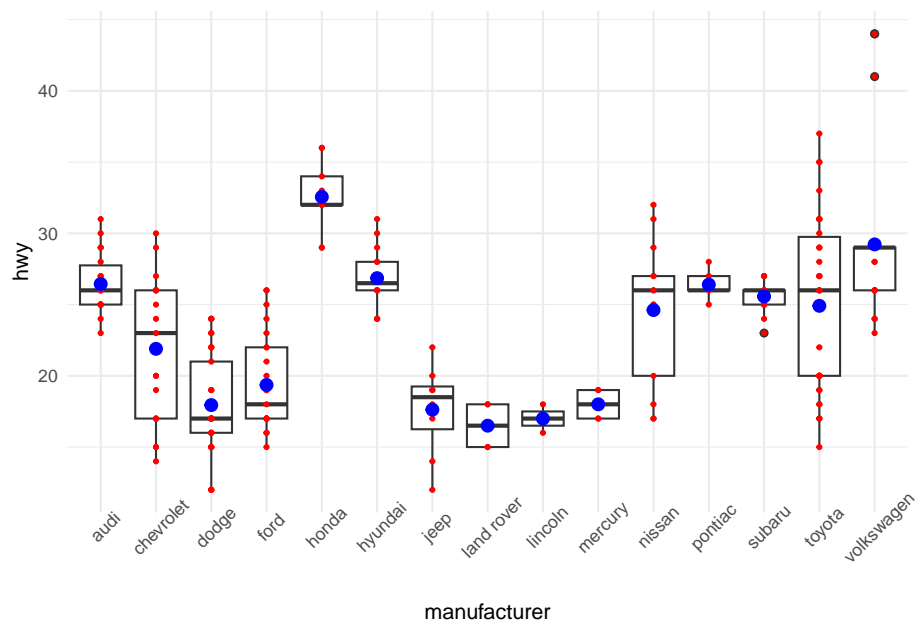
```



```

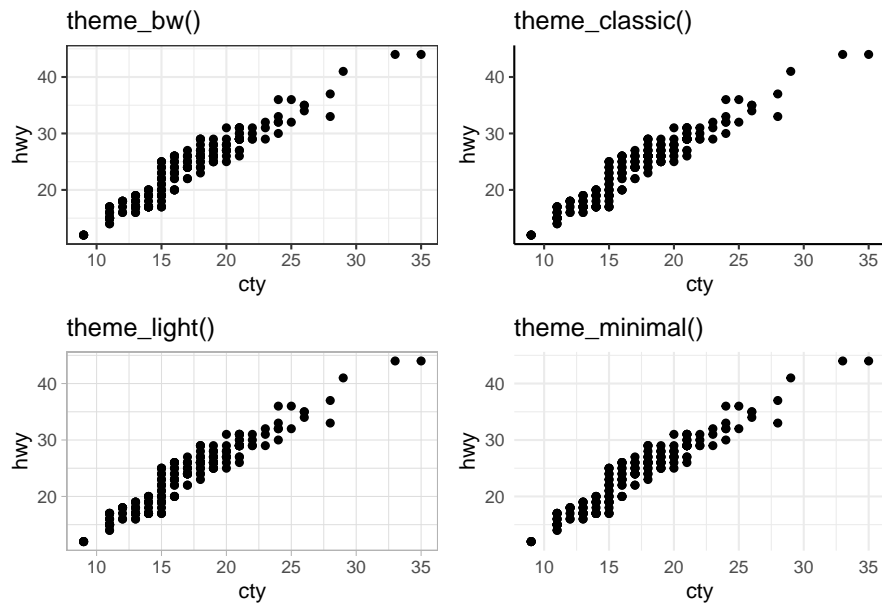
ggplot(dados, aes(x = manufacturer, y = hwy)) +
  geom_boxplot()+
  geom_point(col = "red", size=0.8)+
  stat_summary(fun = mean, col = "blue")+
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45))

```

```
a1<- p1 + theme_bw() + labs(title = "theme_bw()")
a2<- p1 + theme_classic() + labs(title = "theme_classic()")
a3<- p1 + theme_light() + labs(title = "theme_light()")
a4<- p1 + theme_minimal() + labs(title = "theme_minimal()")

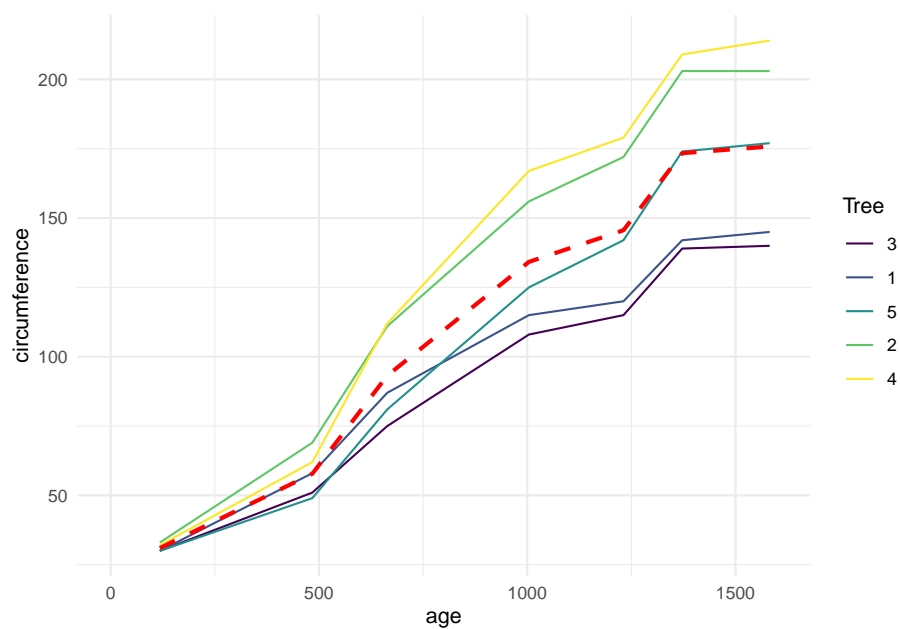
a1 + a2 + a3 + a4
```



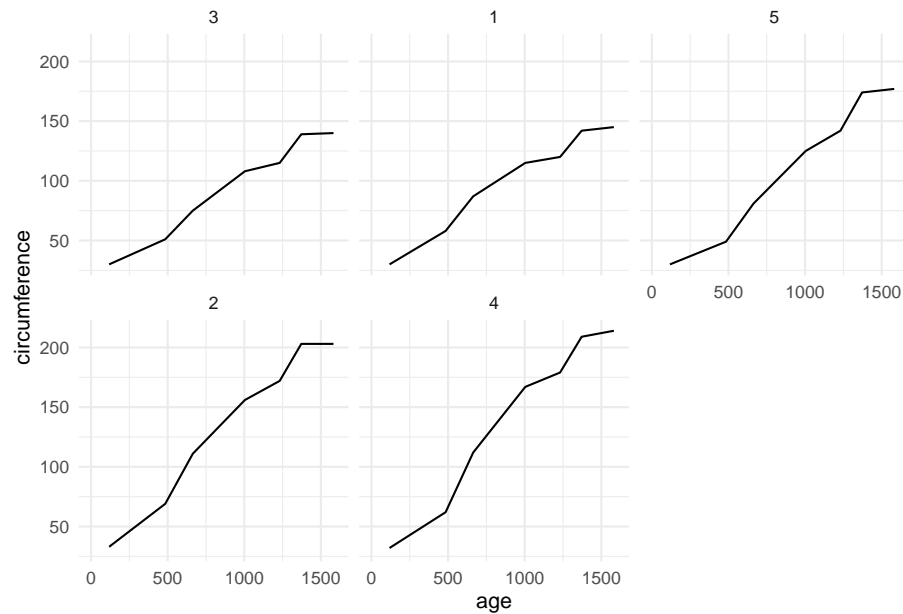
```
## 11. Orange: 1. Orange (a dataset with 14 columns)
glimpse(Orange)
```

```
## Rows: 35
## Columns: 3
## $ Tree      <ord> 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, ~
## $ age       <dbl> 118, 484, 664, 1004, 1231, 1372, 1582, 118, 484, 664, 10~
## $ circumference <dbl> 30, 58, 87, 115, 120, 142, 145, 33, 69, 111, 156, 172, 2~

ggplot(Orange, aes(x = age, y = circumference, group = Tree,
                    col = Tree)) +
  geom_line()+
  stat_summary(aes(group = 1), fun = mean, col = "red",
               geom = "line", size = 1, show.legend = FALSE,
               linetype = 2)+
  xlim(0, 1600)+
  theme_minimal()
```



```
ggplot(Orange, aes(x = age, y = circumference, group = Tree)) +  
  geom_line()+  
  xlim(0, 1600)+  
  facet_wrap(~Tree)+  
  theme_minimal()+  
  theme(legend.position = "none")
```



3.12 plotly

plotly cran

Interactive web-based data visualization with R, plotly, and shiny

Plotly R Open Source Graphing Library

```
ggplotly(v1)
ggplotly(v2)
ggplotly(v4)
ggplotly(v5)
```

3.13 esquisse

Alguns links de interesse

esquisse

esquisse + shiny

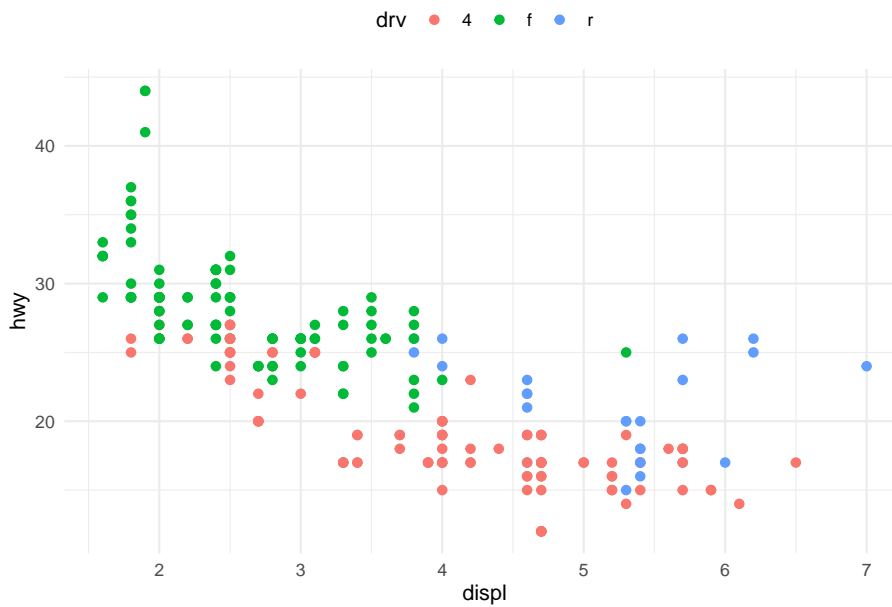
```
esquisser(dados)
```

```

aes(x = displ, y = hwy, colour = drv) +
geom_point(shape = "circle", size = 1.85) +
scale_color_hue(direction = 1) +
theme_minimal() +
theme(legend.position = "top")

```

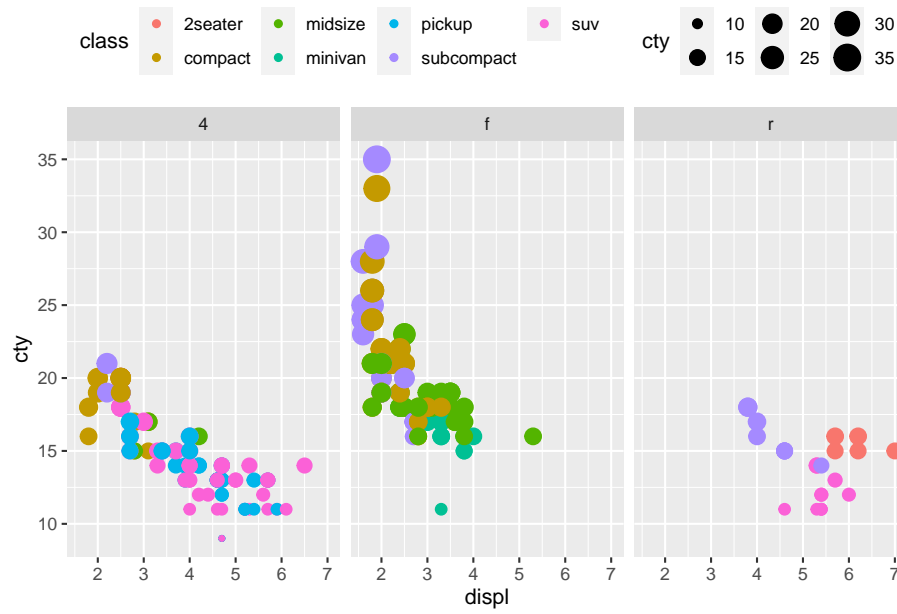
3.14 Exemplo esquisse



```

ggplot(dados) +
  aes(x = displ, y = hwy, colour = drv, size = displ) +
  geom_point(shape = "circle") +
  scale_color_hue(direction = 1) +
  theme(legend.position = "top") +
  facet_wrap(vars(drv))

```



Chapter 4

purrr

```
library(tidyverse)
ls("package:purrr")
```

##	[1]	"%@"	"% %"	"%>%"
##	[4]	"accumulate"	"accumulate_right"	"accumulate2"
##	[7]	"array_branch"	"array_tree"	"as_mapper"
##	[10]	"as_vector"	"assign_in"	"at_depth"
##	[13]	"attr_getter"	"auto_browse"	"chuck"
##	[16]	"compact"	"compose"	"cross"
##	[19]	"cross_d"	"cross_df"	"cross_n"
##	[22]	"cross2"	"cross3"	"detect"
##	[25]	"detect_index"	"discard"	"discard_at"
##	[28]	"done"	"every"	"exec"
##	[31]	"flatten"	"flatten_chr"	"flatten_dbl"
##	[34]	"flatten_df"	"flatten_dfc"	"flatten_dfr"
##	[37]	"flatten_int"	"flatten_lgl"	"flatten_raw"
##	[40]	"has_element"	"head_while"	"imap"
##	[43]	"imap_chr"	"imap_dbl"	"imap_dfc"
##	[46]	"imap_dfr"	"imap_int"	"imap_lgl"
##	[49]	"imap_raw"	"imodify"	"insistently"
##	[52]	"invoke"	"invoke_map"	"invoke_map_chr"
##	[55]	"invoke_map_dbl"	"invoke_map_df"	"invoke_map_dfc"
##	[58]	"invoke_map_dfr"	"invoke_map_int"	"invoke_map_lgl"
##	[61]	"invoke_map_raw"	"is_atomic"	"is_bare_atomic"
##	[64]	"is_bare_character"	"is_bare_double"	"is_bare_integer"
##	[67]	"is_bare_list"	"is_bare_logical"	"is_bare_numeric"
##	[70]	"is_bare_vector"	"is_character"	"is_double"
##	[73]	"is_empty"	"is_formula"	"is_function"
##	[76]	"is_integer"	"is_list"	"is_logical"

## [79]	"is_null"	"is_rate"	"is_scalar_atomic"
## [82]	"is_scalar_character"	"is_scalar_double"	"is_scalar_integer"
## [85]	"is_scalar_list"	"is_scalar_logical"	"is_scalar_vector"
## [88]	"is_vector"	"iwalk"	"keep"
## [91]	"keep_at"	"lift"	"lift_dl"
## [94]	"lift_dv"	"lift_ld"	"lift_lv"
## [97]	"lift_vd"	"lift_vl"	"list_along"
## [100]	"list_assign"	"list_c"	"list_cbind"
## [103]	"list_flatten"	"list_merge"	"list_modify"
## [106]	"list_rbind"	"list_simplify"	"list_transpose"
## [109]	"lmap"	"lmap_at"	"lmap_if"
## [112]	"map"	"map_at"	"map_chr"
## [115]	"map_dbl"	"map_depth"	"map_df"
## [118]	"map_dfc"	"map_dfr"	"map_if"
## [121]	"map_int"	"map_lgl"	"map_raw"
## [124]	"map_vec"	"map2"	"map2_chr"
## [127]	"map2_dbl"	"map2_df"	"map2_dfc"
## [130]	"map2_dfr"	"map2_int"	"map2_lgl"
## [133]	"map2_raw"	"map2_vec"	"modify"
## [136]	"modify_at"	"modify_depth"	"modify_if"
## [139]	"modify_in"	"modify_tree"	"modify2"
## [142]	"negate"	"none"	"partial"
## [145]	"pluck"	"pluck_depth"	"pluck_exists"
## [148]	"pluck<-"	"pmap"	"pmap_chr"
## [151]	"pmap_dbl"	"pmap_df"	"pmap_dfc"
## [154]	"pmap_dfr"	"pmap_int"	"pmap_lgl"
## [157]	"pmap_raw"	"pmap_vec"	"possibly"
## [160]	"prepend"	"pwalk"	"quietly"
## [163]	"rate_backoff"	"rate_delay"	"rate_reset"
## [166]	"rate_sleep"	"rbernoulli"	"rdunif"
## [169]	"reduce"	"reduce_right"	"reduce2"
## [172]	"reduce2_right"	"rep_along"	"rerun"
## [175]	"safely"	"set_names"	"simplify"
## [178]	"simplify_all"	"slowly"	"some"
## [181]	"splice"	"tail_while"	"transpose"
## [184]	"update_list"	"vec_depth"	"walk"
## [187]	"walk2"	"when"	"zap"

4.1 Apply a function to each element of a list or atomic vector

The map functions transform their input by applying a function to each element of a list or atomic vector and returning an object of the same length as the input.

- `map()` always returns a list. See the `modify()` family for versions that return an object of the same type as the input.
- `map_lgl()`, `map_int()`, `map_dbl()` and `map_chr()` return an atomic vector of the indicated type (or die trying).
- `map_dfr()` and `map_dfc()` return a data frame created by row-binding and column-binding respectively. They require `dplyr` to be installed.
- The returned values of `.f` must be of length one for each element of `.x`. If `.f` uses an extractor function shortcut, `.default` can be specified to handle values that are absent or empty. See `as_mapper()` for more on `.default`.
- `walk()` calls `.f` for its side-effect and returns the input `.x`.

4.1.1 Usage

- `map(.x, .f, ...)`
- `map_lgl(.x, .f, ...)`
- `map_chr(.x, .f, ...)`
- `map_int(.x, .f, ...)`
- `map_dbl(.x, .f, ...)`
- `map_raw(.x, .f, ...)`
- `map_dfr(.x, .f, ..., .id = NULL)`
- `map_dfc(.x, .f, ...)`
- `walk(.x, .f, ...)`

4.1.2 Arguments

- `.x` A list or atomic vector.
- `.f` A function, formula, or vector (not necessarily atomic).

If a function, it is used as is.

If a formula, e.g. `~ .x + 2`, it is converted to a function. There are three ways to refer to the arguments:

- For a single argument function, use `.`
- For a two argument function, use `.x` and `.y`
- For more arguments, use `..1`, `..2`, `..3` etc

This syntax allows you to create very compact anonymous functions.

If character vector, numeric vector, or list, it is converted to an extractor function. Character vectors index by name and numeric vectors index by position;

use a list to index by position and name at different levels. If a component is not present, the value of `.default` will be returned.

- ... Additional arguments passed on to the mapped function.
- `.id` Either a string or `NULL`. If a string, the output will contain a variable with that name, storing either the name (if `.x` is named) or the index (if `.x` is unnamed) of the input. If `NULL`, the default, no variable will be created.

Only applies to `_dfr` variant.

4.1.3 Value

- `map()` Returns a list the same length as `.x`.
- `map_lgl()` returns a logical vector, `map_int()` an integer vector, `map_dbl()` a double vector, and `map_chr()` a character vector.
- `map_df()`, `map_dfc()`, `map_dfr()` all return a data frame.
- If `.x` has `names()`, the return value preserves those names.
- The output of `.f` will be automatically typed upwards, e.g. logical -> integer -> double -> character.
- `walk()` returns the input `.x` (invisibly). This makes it easy to use in pipe.

4.1.4 See Also

`map_if()` for applying a function to only those elements of `.x` that meet a specified condition.

Other map variants: `imap()`, `invoke()`, `lmap()`, `map2()`, `map_if()`, `modify()`

4.2 Examples

```
# Compute normal distributions from an atomic vector
```

```
1:10 %>%
```

```
  map(rnorm, n = 10)
```

```
## [[1]]
```

```
## [1] 0.6339724 2.7880373 -0.1984222 1.1442054 0.7124724 1.2276084
```

```
## [7] 1.6474717 1.0929991 -0.6985577 1.4222392
```

```
##
```

```
## [[2]]
```

```
## [1] 2.133486 2.629332 1.065583 1.809737 2.560628 2.689161 1.176797 1.436152
```

```
## [9] 1.966085 2.364917
```

```
##
```

```
## [[3]]
```

```
## [1] 1.100445 2.781537 2.299967 2.580975 2.761169 2.844217 2.702150 3.707112
## [9] 2.153652 3.114340
##
## [[4]]
## [1] 3.198181 5.492960 3.899454 5.824106 4.974750 4.306030 4.521728 4.913784
## [9] 6.094295 2.242841
##
## [[5]]
## [1] 5.930144 5.108468 6.096394 5.757209 4.764078 3.975482 7.795530 4.206314
## [9] 4.938812 4.420335
##
## [[6]]
## [1] 6.463582 6.880319 6.300040 6.451217 8.143553 5.230962 5.460726 6.625071
## [9] 7.612566 6.368246
##
## [[7]]
## [1] 8.293561 5.829829 6.865768 5.770135 7.291877 6.921683 5.799535 6.156567
## [9] 6.752780 8.158770
##
## [[8]]
## [1] 7.038720 8.869194 8.518592 7.817315 6.673090 7.987795 8.056667 6.852389
## [9] 9.615797 6.401303
##
## [[9]]
## [1] 9.620093 7.623280 9.325927 9.210802 9.204712 8.556545 8.931994 9.970865
## [9] 7.602060 7.525370
##
## [[10]]
## [1] 10.142422 9.976939 9.624969 9.196717 11.881371 9.223565 7.831989
## [8] 10.022924 10.202374 9.375127

# You can also use an anonymous function
1:10 %>%
  map(function(x) rnorm(10, x))

## [[1]]
## [1] 1.4242364 1.5871954 1.9318785 2.8434393 1.3602100 0.2791915 0.7598485
## [8] 2.3902681 1.1827565 1.0402025
##
## [[2]]
## [1] 1.9331236 0.9889410 1.3633294 2.7567648 2.1605803 -0.1928810
## [7] 1.5173740 4.1824461 0.3918547 2.6427651
##
## [[3]]
## [1] 1.833696 3.731865 2.035081 2.670794 2.587494 3.623561 3.397302 1.174053
## [9] 2.521653 3.231178
```

```
##
## [[4]]
## [1] 3.322055 3.300941 2.778499 2.397965 3.612766 3.487014 4.929250 5.088878
## [9] 3.720418 4.326392
##
## [[5]]
## [1] 6.992499 4.047125 4.791362 4.609602 6.009338 5.774938 5.733584 4.357322
## [9] 4.755341 5.405129
##
## [[6]]
## [1] 4.442480 6.916796 6.436412 7.281615 6.459011 5.833078 5.170422 5.808270
## [9] 6.636652 6.574892
##
## [[7]]
## [1] 4.631180 6.212651 5.914558 7.782220 6.864881 8.121306 7.271194 5.879798
## [9] 6.159553 7.419076
##
## [[8]]
## [1] 6.124578 8.142177 7.150341 7.655491 7.855576 8.156402 6.106497 7.450578
## [9] 6.997167 7.921414
##
## [[9]]
## [1] 8.456928 8.259565 8.654143 11.275767 8.942627 11.686104 10.876411
## [8] 8.816997 7.241138 7.681535
##
## [[10]]
## [1] 10.420855 9.895795 10.549959 11.329046 10.400282 10.658443 12.107371
## [8] 10.350309 11.412465 10.551717

# Or a formula
1:10 %>%
  map(~ rnorm(10, .x))

## [[1]]
## [1] 0.4318584 2.2339985 2.0797803 2.1416742 0.8599021 1.4803502 1.1686782
## [8] 0.5791437 0.9945326 2.4618327
##
## [[2]]
## [1] 1.9993582 1.8140886 1.8983625 2.6912501 2.1727339 1.1397381 2.6211254
## [8] 1.7293576 1.7228316 0.8387017
##
## [[3]]
## [1] 2.844852 4.423551 3.035988 2.884852 3.534187 3.276245 1.676339 4.923968
## [9] 2.110770 2.774593
##
## [[4]]
```

```
## [1] 2.688604 3.988290 4.259162 3.272460 1.780160 2.152761 4.989109 3.847146
## [9] 3.600067 4.622678
##
## [[5]]
## [1] 5.811755 4.345995 4.607296 3.982432 3.988317 4.977129 6.448051 5.059798
## [9] 6.990571 5.790329
##
## [[6]]
## [1] 8.205088 4.223011 5.445546 5.696194 6.525025 3.447123 5.533415 5.879608
## [9] 6.289842 5.726105
##
## [[7]]
## [1] 6.179094 8.639896 5.773333 6.809194 6.992966 6.422275 7.185122 4.389173
## [9] 5.105871 6.047818
##
## [[8]]
## [1] 7.901826 7.428609 8.357170 8.413080 8.874393 6.759053 9.345543 8.335065
## [9] 7.779112 7.638771
##
## [[9]]
## [1] 9.843409 7.628733 9.984434 9.465567 9.496891 9.675503 9.910865 8.918554
## [9] 7.914487 8.901029
##
## [[10]]
## [1] 8.417983 9.905794 10.553245 10.411573 8.897385 8.332058 11.554748
## [8] 11.098174 10.898944 8.815265

# Simplify output to a vector instead of a list by computing the mean of the distributions
1:10 %>%
  map(rnorm, n = 10) %>% # output a list
  map_dbl(mean)          # output an atomic vector

## [1] 0.7318374 2.4697774 3.0166276 4.0095843 5.3207942 5.5948163 6.8471041
## [8] 7.5384458 9.0545152 9.8331927

# Using set_names() with character vectors is handy to keep track
# of the original inputs:
set_names(c("foo", "bar")) %>%
  map_chr(paste0, ":suffix")

##           foo           bar
## "foo:suffix" "bar:suffix"

# Working with lists
favorite_desserts <- list(Sophia = "banana bread", Elliott = "pancakes", Karina = "chocolate cake")
favorite_desserts

## $Sophia
```

```
## [1] "banana bread"
##
## $Eliott
## [1] "pancakes"
##
## $Karina
## [1] "chocolate cake"
```

```
favorite_desserts %>%
  map_chr(~ paste(.x, "rocks!"))
```

```
##                Sophia                Elliott                Karina
## "banana bread rocks!" "pancakes rocks!" "chocolate cake rocks!"
```

```
# Extract by name or position
# .default specifies value for elements that are missing or NULL
l1 <- list(list(a = 1L),
           list(a = NULL, b = 2L),
           list(b = 3L))
l1
```

```
## [[1]]
## [[1]]$a
## [1] 1
##
##
## [[2]]
## [[2]]$a
## NULL
##
## [[2]]$b
## [1] 2
##
##
## [[3]]
## [[3]]$b
## [1] 3
```

```
l1 %>%
  map("a", .default = "???)")
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "???"
##
##
## [[3]]
```

```
## [1] "???"
l1 %>%
  map_int("b", .default = NA)

## [1] NA  2  3
l1 %>%
  map_int(2, .default = NA)

## [1] NA  2 NA
# Supply multiple values to index deeply into a list
l2 <- list(
  list(num = 1:3,      letters[1:3]),
  list(num = 101:103, letters[4:6]),
  list())
l2

## [[1]]
## [[1]]$num
## [1] 1 2 3
##
## [[1]][[2]]
## [1] "a" "b" "c"
##
##
## [[2]]
## [[2]]$num
## [1] 101 102 103
##
## [[2]][[2]]
## [1] "d" "e" "f"
##
##
## [[3]]
## list()

l2 %>%
  map(c(2, 2))

## [[1]]
## [1] "b"
##
## [[2]]
## [1] "e"
##
## [[3]]
## NULL
```

```
# Use a list to build an extractor that mixes numeric indices and names,
# and .default to provide a default value if the element does not exist
```

```
12 %>%
  map(list("num", 3))
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 103
##
## [[3]]
## NULL
```

```
12 %>%
  map_int(list("num", 3), .default = NA)
```

```
## [1] 3 103 NA
```

```
# Working with data frames
```

```
# Use map_lgl(), map_dbl(), etc to return a vector instead of a list:
```

```
mtcars %>%
  map_dbl(sum)
```

```
##      mpg      cyl    disp      hp      drat      wt      qsec      vs
## 642.900 198.000 7383.100 4694.000 115.090 102.952 571.160 14.000
##      am      gear    carb
## 13.000 118.000 90.000
```

```
# A more realistic example: split a data frame into pieces, fit a
# model to each piece, summarise and extract R^2
```

```
mtcars %>%
  split(.$cyl)
```

```
## $`4`
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710  22.8  4 108.0  93 3.85 2.320 18.61 1  1   4   1
## 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
## 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
## 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
## Volvo 142E  21.4  4 121.0 109 4.11 2.780 18.60 1  1   4   2
##
```



```
## $`6`
##           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
## Hornet 4 Drive  21.4   6  258.0 110 3.08 3.215 19.44 1  0    3    1
## Valiant        18.1   6  225.0 105 2.76 3.460 20.22 1  0    3    1
## 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
## Ferrari Dino   19.7   6  145.0 175 3.62 2.770 15.50 0  1    5    6
##
## $`8`
##           mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## Hornet Sportabout 18.7   8  360.0 175 3.15 3.440 17.02 0  0    3    2
## Duster 360       14.3   8  360.0 245 3.21 3.570 15.84 0  0    3    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
## 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
## Ford Pantera L   15.8   8  351.0 264 4.22 3.170 14.50 0  1    5    4
## Maserati Bora     15.0   8  301.0 335 3.54 3.570 14.60 0  1    5    8

mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .x))

## $`4`
##
## Call:
## lm(formula = mpg ~ wt, data = .x)
##
## Coefficients:
## (Intercept)          wt
##      39.571       -5.647
##
##
## $`6`
##
## Call:
## lm(formula = mpg ~ wt, data = .x)
##
```

```
## Coefficients:
## (Intercept)          wt
##      28.41      -2.78
##
##
## $`8`
##
## Call:
## lm(formula = mpg ~ wt, data = .x)
##
## Coefficients:
## (Intercept)          wt
##      23.868      -2.192

mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .x)) %>%
  map(summary)

## $`4`
##
## Call:
## lm(formula = mpg ~ wt, data = .x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.1513 -1.9795 -0.6272  1.9299  5.2523
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   39.571      4.347   9.104 7.77e-06 ***
## wt           -5.647      1.850  -3.052  0.0137 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.332 on 9 degrees of freedom
## Multiple R-squared:  0.5086, Adjusted R-squared:  0.454
## F-statistic: 9.316 on 1 and 9 DF,  p-value: 0.01374
##
##
## $`6`
##
## Call:
## lm(formula = mpg ~ wt, data = .x)
##
## Residuals:
```

```
##      Mazda RX4  Mazda RX4 Wag  Hornet 4 Drive      Valiant      Merc 280
##      -0.1250      0.5840      1.9292      -0.6897      0.3547
##      Merc 280C   Ferrari Dino
##      -1.0453      -1.0080
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  28.409      4.184    6.789 0.00105 **
## wt          -2.780      1.335   -2.083 0.09176 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.165 on 5 degrees of freedom
## Multiple R-squared:  0.4645, Adjusted R-squared:  0.3574
## F-statistic: 4.337 on 1 and 5 DF,  p-value: 0.09176
##
## $`8`
##
## Call:
## lm(formula = mpg ~ wt, data = .x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.1491 -1.4664 -0.8458  1.5711  3.7619
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23.8680      3.0055    7.942 4.05e-06 ***
## wt          -2.1924      0.7392   -2.966 0.0118 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.024 on 12 degrees of freedom
## Multiple R-squared:  0.423, Adjusted R-squared:  0.3749
## F-statistic: 8.796 on 1 and 12 DF,  p-value: 0.01179
# original
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .x)) %>%
  map(summary) %>%
  map_dbl("r.squared")

##           4           6           8
## 0.5086326 0.4645102 0.4229655
```

```

# If each element of the output is a data frame, use
# map_dfr to row-bind them together:
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .x)) %>%
  map_dfr(~ as.data.frame(t(as.matrix(coef(.))))))

##      (Intercept)          wt
## 1      39.57120 -5.647025
## 2      28.40884 -2.780106
## 3      23.86803 -2.192438

# (if you also want to preserve the variable names see
# the broom package)

#nest, unest() estudar!

mtcars %>%
  group_by(cyl) %>%
  nest()

## # A tibble: 3 x 2
## # Groups:   cyl [3]
##       cyl data
##   <dbl> <list>
## 1     6 <tibble [7 x 10]>
## 2     4 <tibble [11 x 10]>
## 3     8 <tibble [14 x 10]>

#mtcars %>%
#  group_by(cyl) %>%
#  nest() %>%
#  map(~ lm(mpg ~ wt, data = .x))

```

4.3 map functions

```

example("map")
example("map_at")
example("map_chr")
example("map_dbl")
example("map_df")
example("map_dfc")
example("map_dfr")

```

```
example("map_int")  
example("map_lgl")  
example("map_vec")
```

4.4 map2 functions

```
example("map2")  
example("map2_chr")  
example("map2_dbl")  
example("map2_df")  
example("map2_dfc")  
example("map2_dfr")  
example("map2_int")  
example("map2_lgl")  
example("map2_raw")  
example("map2_vec")
```


Chapter 5

tidyr

More details in <https://tidyr.tidyverse.org/articles/nest.html>

5.1 nest()

```
library(tidyverse)

mtcars %>%
  group_by(cyl) %>%
  nest()

## # A tibble: 3 x 2
## # Groups:   cyl [3]
##   cyl data
##   <dbl> <list>
## 1     6 <tibble [7 x 10]>
## 2     4 <tibble [11 x 10]>
## 3     8 <tibble [14 x 10]>

um<- mtcars %>%
  group_by(cyl) %>%
  nest() %>%
  mutate(
    linMod = map(data, ~lm(mpg ~ wt, data = .)),
    coeffs = map(linMod, coefficients),
    slope = map_dbl(coeffs, 2))
um

## # A tibble: 3 x 5
## # Groups:   cyl [3]
```

```
##      cyl data      linMod coeffs      slope
##      <dbl> <list>      <list> <list>      <dbl>
## 1      6 <tibble [7 x 10]> <lm>      <dbl [2]> -2.78
## 2      4 <tibble [11 x 10]> <lm>      <dbl [2]> -5.65
## 3      8 <tibble [14 x 10]> <lm>      <dbl [2]> -2.19
```

```
um$linMod
```

```
## [[1]]
##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Coefficients:
## (Intercept)      wt
##      28.41      -2.78
##
##
## [[2]]
##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Coefficients:
## (Intercept)      wt
##      39.571      -5.647
##
##
## [[3]]
##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Coefficients:
## (Intercept)      wt
##      23.868      -2.192
```

```
um$coeffs
```

```
## [[1]]
## (Intercept)      wt
##  28.408845  -2.780106
##
## [[2]]
## (Intercept)      wt
##  39.571196  -5.647025
##
```



```
## [[3]]
## (Intercept)          wt
## 23.868029    -2.192438

um$slope

## [1] -2.780106 -5.647025 -2.192438

um$linMod[[1]]

##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Coefficients:
## (Intercept)          wt
##      28.41         -2.78

dois<- mtcars %>%
  group_by(cyl) %>%
  nest() %>%
  mutate(model = map(data, function(df) lm(mpg ~ wt, data = df)))
dois

## # A tibble: 3 x 3
## # Groups:   cyl [3]
##   cyl data          model
##   <dbl> <list>        <list>
## 1     6 <tibble [7 x 10]> <lm>
## 2     4 <tibble [11 x 10]> <lm>
## 3     8 <tibble [14 x 10]> <lm>

dois$cyl

## [1] 6 4 8

dois$data

## [[1]]
## # A tibble: 7 x 10
##   mpg  disp  hp  drat   wt  qsec   vs  am  gear  carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  21    160   110  3.9   2.62  16.5    0    1     4     4
## 2  21    160   110  3.9   2.88  17.0    0    1     4     4
## 3  21.4  258   110  3.08  3.22  19.4    1    0     3     1
## 4  18.1  225   105  2.76  3.46  20.2    1    0     3     1
## 5  19.2  168.   123  3.92  3.44  18.3    1    0     4     4
## 6  17.8  168.   123  3.92  3.44  18.9    1    0     4     4
## 7  19.7  145   175  3.62  2.77  15.5    0    1     5     6
##
```

```
## [[2]]
## # A tibble: 11 x 10
##   mpg   disp  hp  drat    wt  qsec    vs  am  gear carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  22.8  108    93  3.85  2.32  18.6    1    1    4    1
## 2  24.4  147    62  3.69  3.19  20      1    0    4    2
## 3  22.8  141    95  3.92  3.15  22.9    1    0    4    2
## 4  32.4   78.7   66  4.08  2.2   19.5    1    1    4    1
## 5  30.4   75.7   52  4.93  1.62  18.5    1    1    4    2
## 6  33.9   71.1   65  4.22  1.84  19.9    1    1    4    1
## 7  21.5  120    97  3.7   2.46  20.0    1    0    3    1
## 8  27.3   79    66  4.08  1.94  18.9    1    1    4    1
## 9   26   120    91  4.43  2.14  16.7    0    1    5    2
## 10  30.4  95.1   113  3.77  1.51  16.9    1    1    5    2
## 11  21.4  121   109  4.11  2.78  18.6    1    1    4    2
##
## [[3]]
## # A tibble: 14 x 10
##   mpg   disp  hp  drat    wt  qsec    vs  am  gear carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  18.7  360   175  3.15  3.44  17.0    0    0    3    2
## 2  14.3  360   245  3.21  3.57  15.8    0    0    3    4
## 3  16.4  276   180  3.07  4.07  17.4    0    0    3    3
## 4  17.3  276   180  3.07  3.73  17.6    0    0    3    3
## 5  15.2  276   180  3.07  3.78  18      0    0    3    3
## 6  10.4  472   205  2.93  5.25  18.0    0    0    3    4
## 7  10.4  460   215  3     5.42  17.8    0    0    3    4
## 8  14.7  440   230  3.23  5.34  17.4    0    0    3    4
## 9  15.5  318   150  2.76  3.52  16.9    0    0    3    2
## 10  15.2  304   150  3.15  3.44  17.3    0    0    3    2
## 11  13.3  350   245  3.73  3.84  15.4    0    0    3    4
## 12  19.2  400   175  3.08  3.84  17.0    0    0    3    2
## 13  15.8  351   264  4.22  3.17  14.5    0    1    5    4
## 14  15    301   335  3.54  3.57  14.6    0    1    5    8
##
dois$model

## [[1]]
##
## Call:
## lm(formula = mpg ~ wt, data = df)
##
## Coefficients:
## (Intercept)          wt
##      28.41       -2.78
##
```

```
##
## [[2]]
##
## Call:
## lm(formula = mpg ~ wt, data = df)
##
## Coefficients:
## (Intercept)          wt
##      39.571      -5.647
##
## [[3]]
##
## Call:
## lm(formula = mpg ~ wt, data = df)
##
## Coefficients:
## (Intercept)          wt
##      23.868      -2.192
dois$model[[3]]

##
## Call:
## lm(formula = mpg ~ wt, data = df)
##
## Coefficients:
## (Intercept)          wt
##      23.868      -2.192
tres<- dois %>%
  mutate(model = map(model, predict))
tres

## # A tibble: 3 x 3
## # Groups:   cyl [3]
##   cyl data          model
##   <dbl> <list>        <list>
## 1     6 <tibble [7 x 10]> <dbl [7]>
## 2     4 <tibble [11 x 10]> <dbl [11]>
## 3     8 <tibble [14 x 10]> <dbl [14]>
tres$model

## [[1]]
##      1      2      3      4      5      6      7
## 21.12497 20.41604 19.47080 18.78968 18.84528 18.84528 20.70795
```

```
##
## [[2]]
##      1      2      3      4      5      6      7      8
## 26.47010 21.55719 21.78307 27.14774 30.45125 29.20890 25.65128 28.64420
##      9     10     11
## 27.48656 31.02725 23.87247
##
## [[3]]
##      1      2      3      4      5      6      7      8
## 16.32604 16.04103 14.94481 15.69024 15.58061 12.35773 11.97625 12.14945
##      9     10     11     12     13     14
## 16.15065 16.33700 15.44907 15.43811 16.91800 16.04103
tres$model[[3]]

##      1      2      3      4      5      6      7      8
## 16.32604 16.04103 14.94481 15.69024 15.58061 12.35773 11.97625 12.14945
##      9     10     11     12     13     14
## 16.15065 16.33700 15.44907 15.43811 16.91800 16.04103
```

5.2 unnest()

```
um %>%
  unnest(data)
```

```
## # A tibble: 32 x 14
## # Groups:   cyl [3]
##      cyl  mpg  disp   hp  drat   wt  qsec   vs  am  gear  carb  linMod
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <list>
## 1     6   21   160   110  3.9   2.62  16.5    0    1    4    4 <lm>
## 2     6   21   160   110  3.9   2.88  17.0    0    1    4    4 <lm>
## 3     6  21.4  258   110  3.08  3.22  19.4    1    0    3    1 <lm>
## 4     6  18.1  225   105  2.76  3.46  20.2    1    0    3    1 <lm>
## 5     6  19.2  168.   123  3.92  3.44  18.3    1    0    4    4 <lm>
## 6     6  17.8  168.   123  3.92  3.44  18.9    1    0    4    4 <lm>
## 7     6  19.7  145   175  3.62  2.77  15.5    0    1    5    6 <lm>
## 8     4  22.8  108    93  3.85  2.32  18.6    1    1    4    1 <lm>
## 9     4  24.4  147.    62  3.69  3.19  20      1    0    4    2 <lm>
## 10    4  22.8  141.    95  3.92  3.15  22.9    1    0    4    2 <lm>
## # ... with 22 more rows, and 2 more variables: coeffs <list>, slope <dbl>
```

5.3 Exemplos da ajuda do R

```
df <- tibble(x = c(1, 1, 1, 2, 2, 3),
             y = 1:6,
```

```

      z = 6:1)
df

## # A tibble: 6 x 3
##       x     y     z
##   <dbl> <int> <int>
## 1     1     1     6
## 2     1     2     5
## 3     1     3     4
## 4     2     4     3
## 5     2     5     2
## 6     3     6     1

# Note that we get one row of output for each unique combination of
# non-nested variables
df %>%
  nest(data = c(y, z))

## # A tibble: 3 x 2
##       x data
##   <dbl> <list>
## 1     1 <tibble [3 x 2]>
## 2     2 <tibble [2 x 2]>
## 3     3 <tibble [1 x 2]>

# chop does something similar, but retains individual columns
df %>%
  chop(c(y, z))

## # A tibble: 3 x 3
##       x     y     z
##   <dbl> <list<int>> <list<int>>
## 1     1         [3]         [3]
## 2     2         [2]         [2]
## 3     3         [1]         [1]

# use tidyselect syntax and helpers, just like in dplyr::select()
df %>%
  nest(data = any_of(c("y", "z")))

## # A tibble: 3 x 2
##       x data
##   <dbl> <list>
## 1     1 <tibble [3 x 2]>
## 2     2 <tibble [2 x 2]>
## 3     3 <tibble [1 x 2]>

```

```

iris %>%
  nest(data = !Species)

## # A tibble: 3 x 2
##   Species    data
##   <fct>      <list>
## 1 setosa     <tibble [50 x 4]>
## 2 versicolor <tibble [50 x 4]>
## 3 virginica <tibble [50 x 4]>

nest_vars <- names(iris)[1:4]
iris %>%
  nest(data = any_of(nest_vars))

## # A tibble: 3 x 2
##   Species    data
##   <fct>      <list>
## 1 setosa     <tibble [50 x 4]>
## 2 versicolor <tibble [50 x 4]>
## 3 virginica <tibble [50 x 4]>

iris %>%
  nest(petal = starts_with("Petal"), sepal = starts_with("Sepal"))

## # A tibble: 3 x 3
##   Species    petal          sepal
##   <fct>      <list>        <list>
## 1 setosa     <tibble [50 x 2]> <tibble [50 x 2]>
## 2 versicolor <tibble [50 x 2]> <tibble [50 x 2]>
## 3 virginica <tibble [50 x 2]> <tibble [50 x 2]>

iris %>%
  nest(width = contains("Width"), length = contains("Length"))

## # A tibble: 3 x 3
##   Species    width          length
##   <fct>      <list>        <list>
## 1 setosa     <tibble [50 x 2]> <tibble [50 x 2]>
## 2 versicolor <tibble [50 x 2]> <tibble [50 x 2]>
## 3 virginica <tibble [50 x 2]> <tibble [50 x 2]>

# Nesting a grouped data frame nests all variables apart from the group vars
fish_encounters %>%
  group_by(fish) %>%
  nest()

## # A tibble: 19 x 2
## # Groups:   fish [19]

```

```
##      fish  data
##      <fct> <list>
## 1 4842 <tibble [11 x 2]>
## 2 4843 <tibble [11 x 2]>
## 3 4844 <tibble [11 x 2]>
## 4 4845 <tibble [5 x 2]>
## 5 4847 <tibble [3 x 2]>
## 6 4848 <tibble [4 x 2]>
## 7 4849 <tibble [2 x 2]>
## 8 4850 <tibble [6 x 2]>
## 9 4851 <tibble [2 x 2]>
## 10 4854 <tibble [2 x 2]>
## 11 4855 <tibble [5 x 2]>
## 12 4857 <tibble [9 x 2]>
## 13 4858 <tibble [11 x 2]>
## 14 4859 <tibble [5 x 2]>
## 15 4861 <tibble [11 x 2]>
## 16 4862 <tibble [9 x 2]>
## 17 4863 <tibble [2 x 2]>
## 18 4864 <tibble [2 x 2]>
## 19 4865 <tibble [3 x 2]>

# Nesting is often useful for creating per group models
mtcars %>%
  group_by(cyl) %>%
  nest() %>%
  mutate(models = lapply(data, function(df) lm(mpg ~ wt, data = df)))

## # A tibble: 3 x 3
## # Groups:   cyl [3]
##      cyl data          models
##   <dbl> <list>         <list>
## 1     6 <tibble [7 x 10]> <lm>
## 2     4 <tibble [11 x 10]> <lm>
## 3     8 <tibble [14 x 10]> <lm>

# unnest() is primarily designed to work with lists of data frames
df <- tibble(
  x = 1:3,
  y = list(
    NULL,
    tibble(a = 1, b = 2),
    tibble(a = 1:3, b = 3:1)
  )
)

df %>%
```

```

unnest(y)

## # A tibble: 4 x 3
##       x     a     b
##   <int> <dbl> <dbl>
## 1     2     1     2
## 2     3     1     3
## 3     3     2     2
## 4     3     3     1

df %>%
  unnest(y, keep_empty = TRUE)

## # A tibble: 5 x 3
##       x     a     b
##   <int> <dbl> <dbl>
## 1     1    NA    NA
## 2     2     1     2
## 3     3     1     3
## 4     3     2     2
## 5     3     3     1

# If you have lists of lists, or lists of atomic vectors, instead
# see hoist(), unnest_wider(), and unnest_longer()

#' # You can unnest multiple columns simultaneously
df <- tibble(
  a = list(c("a", "b"), "c"),
  b = list(1:2, 3),
  c = c(11, 22)
)
df

## # A tibble: 2 x 3
##       a           b           c
##   <list>   <list>   <dbl>
## 1 <chr [2]> <int [2]>    11
## 2 <chr [1]> <dbl [1]>    22

df %>%
  unnest(c(a, b))

## # A tibble: 3 x 3
##       a           b           c
##   <chr> <dbl> <dbl>
## 1 a           1     11
## 2 b           2     11
## 3 c           3     22

```



```
# Compare with unnesting one column at a time, which generates  
# the Cartesian product
```

```
df %>%  
  unnest(a) %>%  
  unnest(b)
```

```
## # A tibble: 5 x 3  
##   a      b      c  
##   <chr> <dbl> <dbl>  
## 1 a      1     11  
## 2 a      2     11  
## 3 b      1     11  
## 4 b      2     11  
## 5 c      3     22
```

5.4 Mais detalhes

https://github.com/tidymodels/broom/blob/main/vignettes/broom_and_dplyr.Rmd

Chapter 6

broom and dplyr

While broom is useful for summarizing the result of a single analysis in a consistent format, it is really designed for high-throughput applications, where you must combine results from multiple analyses. These could be subgroups of data, analyses using different models, bootstrap replicates, permutations, and so on. In particular, it plays well with the **nest/unnest** functions in **tidyr** and the **map** function in **purrr**. First, loading necessary packages and setting some defaults:

```
library(broom)
library(tibble)
library(ggplot2)
library(dplyr)
library(tidyr)
library(purrr)

theme_set(theme_minimal())
```

Let's try this on a simple dataset, the built-in **Orange**. We start by coercing **Orange** to a **tibble**. This gives a nicer print method that will especially be useful later on when we start working with list-columns.

```
data(Orange)

Orange <- as_tibble(Orange)
Orange

## # A tibble: 35 x 3
##   Tree   age circumference
##   <ord> <dbl>         <dbl>
## 1 1     118           30
## 2 1     484           58
## 3 1     664           87
```

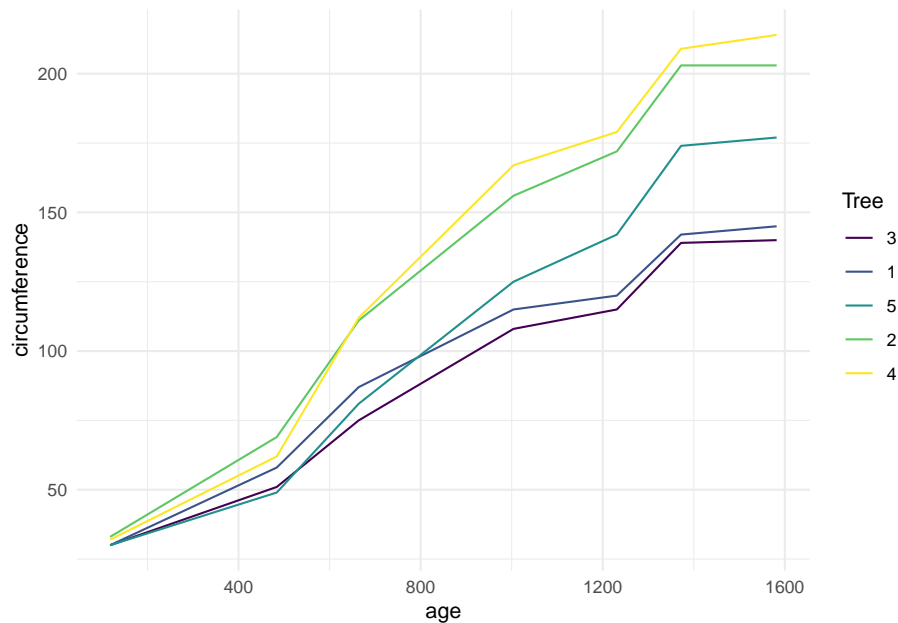
```
## 4 1      1004      115
## 5 1      1231      120
## 6 1      1372      142
## 7 1      1582      145
## 8 2       118       33
## 9 2       484       69
## 10 2      664      111
## # ... with 25 more rows
```

This contains 35 observations of three variables: `Tree`, `age`, and `circumference`. `Tree` is a factor with five levels describing five trees. As might be expected, age and circumference are correlated:

```
cor(Orange$age, Orange$circumference)
```

```
## [1] 0.9135189
```

```
ggplot(Orange, aes(age, circumference, color = Tree)) +
  geom_line()
```



Suppose you want to test for correlations individually *within* each tree. You can do this with dplyr's `group_by`:

```
Orange %>%
  group_by(Tree) %>%
  summarize(correlation = cor(age, circumference))
```

```
## # A tibble: 5 x 2
```

```
## Tree correlation
## <ord>      <dbl>
## 1 3      0.988
## 2 1      0.985
## 3 5      0.988
## 4 2      0.987
## 5 4      0.984
```

(Note that the correlations are much higher than the aggregated one, and furthermore we can now see it is similar across trees).

Suppose that instead of simply estimating a correlation, we want to perform a hypothesis test with `cor.test`:

```
ct <- cor.test(Orange$age, Orange$circumference)
ct

##
## Pearson's product-moment correlation
##
## data: Orange$age and Orange$circumference
## t = 12.9, df = 33, p-value = 1.931e-14
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8342364 0.9557955
## sample estimates:
##      cor
## 0.9135189
```

This contains multiple values we could want in our output. Some are vectors of length 1, such as the p-value and the estimate, and some are longer, such as the confidence interval. We can get this into a nicely organized tibble using the `tidy` function:

```
tidy(ct)

## # A tibble: 1 x 8
##   estimate statistic p.value parameter conf.low conf.high method      alter~1
##   <dbl>      <dbl>   <dbl>      <int>    <dbl>    <dbl> <chr>      <chr>
## 1    0.914      12.9 1.93e-14        33    0.834    0.956 Pearson's pr~ two.si~
## # ... with abbreviated variable name 1: alternative
```

Often, we want to perform multiple tests or fit multiple models, each on a different part of the data. In this case, we recommend a `nest-map-unnest` workflow. For example, suppose we want to perform correlation tests for each different tree. We start by `nesting` our data based on the group of interest:

```
nested <- Orange %>%
  nest(data = ~Tree)
```

Then we run a correlation test for each nested tibble using `purrr::map`:

```
nested %>%
  mutate(test = map(data, ~ cor.test(.x$age, .x$circumference)))

## # A tibble: 5 x 3
##   Tree data          test
##   <ord> <list>        <list>
## 1 1     <tibble [7 x 2]> <htest>
## 2 2     <tibble [7 x 2]> <htest>
## 3 3     <tibble [7 x 2]> <htest>
## 4 4     <tibble [7 x 2]> <htest>
## 5 5     <tibble [7 x 2]> <htest>
```

This results in a list-column of S3 objects. We want to tidy each of the objects, which we can also do with `map`.

```
nested %>%
  mutate(
    test = map(data, ~ cor.test(.x$age, .x$circumference)), # S3 list-col
    tidied = map(test, tidy)
  )
```

```
## # A tibble: 5 x 4
##   Tree data          test  tidied
##   <ord> <list>        <list> <list>
## 1 1     <tibble [7 x 2]> <htest> <tibble [1 x 8]>
## 2 2     <tibble [7 x 2]> <htest> <tibble [1 x 8]>
## 3 3     <tibble [7 x 2]> <htest> <tibble [1 x 8]>
## 4 4     <tibble [7 x 2]> <htest> <tibble [1 x 8]>
## 5 5     <tibble [7 x 2]> <htest> <tibble [1 x 8]>
```

Finally, we want to unnest the tidied data frames so we can see the results in a flat tibble. All together, this looks like:

```
Orange %>%
  nest(data = -Tree) %>%
  mutate(
    test = map(data, ~ cor.test(.x$age, .x$circumference)), # S3 list-col
    tidied = map(test, tidy)
  ) %>%
  unnest(tidied)
```

```
## # A tibble: 5 x 11
##   Tree data    test estimate stati~1 p.value param~2 conf.~3 conf.~4 method
##   <ord> <list>  <list>   <dbl>   <dbl>   <dbl>   <int>   <dbl>   <dbl> <chr>
## 1 1     <tibble> <htest>  0.985    13.0 4.85e-5     5    0.901    0.998 Pears~
## 2 2     <tibble> <htest>  0.987    13.9 3.43e-5     5    0.914    0.998 Pears~
## 3 3     <tibble> <htest>  0.988    14.4 2.90e-5     5    0.919    0.998 Pears~
```

```
## 4 4      <tibble> <htest>      0.984      12.5 5.73e-5      5      0.895      0.998 Pears~
## 5 5      <tibble> <htest>      0.988      14.1 3.18e-5      5      0.916      0.998 Pears~
## # ... with 1 more variable: alternative <chr>, and abbreviated variable names
## #   1: statistic, 2: parameter, 3: conf.low, 4: conf.high
```

This workflow becomes even more useful when applied to regressions. Untidy output for a regression looks like:

```
lm_fit <- lm(age ~ circumference, data = Orange)
summary(lm_fit)

##
## Call:
## lm(formula = age ~ circumference, data = Orange)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -317.88 -140.90  -17.20   96.54  471.16
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   16.6036    78.1406   0.212   0.833
## circumference    7.8160     0.6059  12.900 1.93e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 203.1 on 33 degrees of freedom
## Multiple R-squared:  0.8345, Adjusted R-squared:  0.8295
## F-statistic: 166.4 on 1 and 33 DF,  p-value: 1.931e-14
```

where we tidy these results, we get multiple rows of output for each model:

```
tidy(lm_fit)

## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    16.6      78.1      0.212 8.33e- 1
## 2 circumference    7.82     0.606    12.9  1.93e-14
```

Now we can handle multiple regressions at once using exactly the same workflow as before:

```
Orange %>%
  nest(data = -Tree) %>%
  mutate(
    fit = map(data, ~ lm(age ~ circumference, data = .x)),
    tidied = map(fit, tidy)
  ) %>%
```

```
unnest(tidied)
```

```
## # A tibble: 10 x 8
##   Tree data          fit term          estimate std.er~1 stati~2 p.value
##   <ord> <list>         <list> <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 1      <tibble [7 x 2]> <lm> (Intercept)    -265.    98.6     -2.68  4.36e-2
## 2 1      <tibble [7 x 2]> <lm> circumference    11.9     0.919   13.0   4.85e-5
## 3 2      <tibble [7 x 2]> <lm> (Intercept)   -132.    83.1     -1.59  1.72e-1
## 4 2      <tibble [7 x 2]> <lm> circumference     7.80     0.560   13.9   3.43e-5
## 5 3      <tibble [7 x 2]> <lm> (Intercept)   -210.    85.3     -2.46  5.74e-2
## 6 3      <tibble [7 x 2]> <lm> circumference    12.0     0.835   14.4   2.90e-5
## 7 4      <tibble [7 x 2]> <lm> (Intercept)   -76.5    88.3     -0.867 4.26e-1
## 8 4      <tibble [7 x 2]> <lm> circumference     7.17     0.572   12.5   5.73e-5
## 9 5      <tibble [7 x 2]> <lm> (Intercept)   -54.5    76.9     -0.709 5.10e-1
## 10 5     <tibble [7 x 2]> <lm> circumference     8.79     0.621   14.1   3.18e-5
## # ... with abbreviated variable names 1: std.error, 2: statistic
```

You can just as easily use multiple predictors in the regressions, as shown here on the `mtcars` dataset. We nest the data into automatic and manual cars (the `am` column), then perform the regression within each nested tibble.

```
data(mtcars)
```

```
mtcars <- as_tibble(mtcars) # to play nicely with list-cols
mtcars
```

```
## # A tibble: 32 x 11
##   mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  21      6  160   110  3.9   2.62  16.5     0    1    4     4
## 2  21      6  160   110  3.9   2.88  17.0     0    1    4     4
## 3  22.8    4  108    93  3.85  2.32  18.6     1    1    4     1
## 4  21.4    6  258   110  3.08  3.22  19.4     1    0    3     1
## 5  18.7    8  360   175  3.15  3.44  17.0     0    0    3     2
## 6  18.1    6  225   105  2.76  3.46  20.2     1    0    3     1
## 7  14.3    8  360   245  3.21  3.57  15.8     0    0    3     4
## 8  24.4    4  147.    62  3.69  3.19  20      1    0    4     2
## 9  22.8    4  141.    95  3.92  3.15  22.9     1    0    4     2
## 10 19.2    6  168.   123  3.92  3.44  18.3     1    0    4     4
## # ... with 22 more rows
```

```
mtcars %>%
```

```
  nest(data = -am) %>%
```

```
  mutate(
```

```
    fit = map(data, ~ lm(wt ~ mpg + qsec + gear, data = .x)), # S3 list-col
```

```
    tidied = map(fit, tidy)
```

```
  ) %>%
```

```
  unnest(tidied)
```



```
## # A tibble: 8 x 8
##   am data          fit term      estimate std.error stati~1 p.value
##   <dbl> <list>      <list> <chr>      <dbl>      <dbl>    <dbl>    <dbl>
## 1     1 <tibble [13 x 10]> <lm> (Intercept)  4.28        3.46      1.24    2.47e-1
## 2     1 <tibble [13 x 10]> <lm> mpg      -0.101      0.0294   -3.43    7.50e-3
## 3     1 <tibble [13 x 10]> <lm> qsec       0.0398     0.151    0.264    7.98e-1
## 4     1 <tibble [13 x 10]> <lm> gear      -0.0229     0.349   -0.0656   9.49e-1
## 5     0 <tibble [19 x 10]> <lm> (Intercept)  4.92        1.40      3.52    3.09e-3
## 6     0 <tibble [19 x 10]> <lm> mpg      -0.192     0.0443   -4.33    5.91e-4
## 7     0 <tibble [19 x 10]> <lm> qsec       0.0919     0.0983    0.935    3.65e-1
## 8     0 <tibble [19 x 10]> <lm> gear       0.147      0.368    0.398    6.96e-1
## # ... with abbreviated variable name 1: statistic
```

What if you want not just the `tidy` output, but the `augment` and `glance` outputs as well, while still performing each regression only once? Since we're using list-columns, we can just fit the model once and use multiple list-columns to store the tidied, glanced and augmented outputs.

```
regressions <- mtcars %>%
  nest(data = -am) %>%
  mutate(
    fit = map(data, ~ lm(wt ~ mpg + qsec + gear, data = .x)),
    tidied = map(fit, tidy),
    glanced = map(fit, glance),
    augmented = map(fit, augment)
  )

regressions %>%
  unnest(tidied)
```

```
## # A tibble: 8 x 10
##   am data          fit term estim~1 std.e~2 stati~3 p.value glanced augmen~4
##   <dbl> <list>      <list> <chr>    <dbl>    <dbl>    <dbl>    <dbl> <list> <list>
## 1     1 <tibble> <lm> (Inte~  4.28     3.46     1.24    2.47e-1 <tibble> <tibble>
## 2     1 <tibble> <lm> mpg     -0.101    0.0294   -3.43    7.50e-3 <tibble> <tibble>
## 3     1 <tibble> <lm> qsec      0.0398    0.151    0.264    7.98e-1 <tibble> <tibble>
## 4     1 <tibble> <lm> gear     -0.0229    0.349   -0.0656   9.49e-1 <tibble> <tibble>
## 5     0 <tibble> <lm> (Inte~  4.92     1.40     3.52    3.09e-3 <tibble> <tibble>
## 6     0 <tibble> <lm> mpg     -0.192    0.0443   -4.33    5.91e-4 <tibble> <tibble>
## 7     0 <tibble> <lm> qsec      0.0919    0.0983    0.935    3.65e-1 <tibble> <tibble>
## 8     0 <tibble> <lm> gear      0.147     0.368    0.398    6.96e-1 <tibble> <tibble>
## # ... with abbreviated variable names 1: estimate, 2: std.error, 3: statistic,
## # 4: augmented
```

```
regressions %>%
  unnest(glanced)
```

```
## # A tibble: 2 x 17
```

```
##      am data      fit      tidied      r.squared adj.r.s~1 sigma stati~2 p.value      df
##      <dbl> <list>   <list> <list>         <dbl>      <dbl> <dbl>   <dbl>   <dbl> <dbl>
## 1      1 <tibble> <lm>   <tibble>         0.833      0.778 0.291   15.0  7.59e-4    3
## 2      0 <tibble> <lm>   <tibble>         0.625      0.550 0.522   8.32  1.70e-3    3
## # ... with 7 more variables: logLik <dbl>, AIC <dbl>, BIC <dbl>,
## #   deviance <dbl>, df.residual <int>, nobs <int>, augmented <list>, and
## #   abbreviated variable names 1: adj.r.squared, 2: statistic
```

```
regressions %>%
  unnest(augmented)
```

```
## # A tibble: 32 x 15
##      am data      fit      tidied      glanced      wt      mpg      qsec      gear .fitted
##      <dbl> <list>   <list> <list>         <list>      <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1      1 <tibble> <lm>   <tibble> <tibble>      2.62  21    16.5    4    2.73
## 2      1 <tibble> <lm>   <tibble> <tibble>      2.88  21    17.0    4    2.75
## 3      1 <tibble> <lm>   <tibble> <tibble>      2.32  22.8  18.6    4    2.63
## 4      1 <tibble> <lm>   <tibble> <tibble>      2.2   32.4  19.5    4    1.70
## 5      1 <tibble> <lm>   <tibble> <tibble>      1.62  30.4  18.5    4    1.86
## 6      1 <tibble> <lm>   <tibble> <tibble>      1.84  33.9  19.9    4    1.56
## 7      1 <tibble> <lm>   <tibble> <tibble>      1.94  27.3  18.9    4    2.19
## 8      1 <tibble> <lm>   <tibble> <tibble>      2.14  26    16.7    5    2.21
## 9      1 <tibble> <lm>   <tibble> <tibble>      1.51  30.4  16.9    5    1.77
## 10     1 <tibble> <lm>   <tibble> <tibble>      3.17  15.8  14.5    5    3.15
## # ... with 22 more rows, and 5 more variables: .resid <dbl>, .hat <dbl>,
## #   .sigma <dbl>, .cooksd <dbl>, .std.resid <dbl>
```

By combining the estimates and p-values across all groups into the same tidy data frame (instead of a list of output model objects), a new class of analyses and visualizations becomes straightforward. This includes

- Sorting by p-value or estimate to find the most significant terms across all tests
- P-value histograms
- Volcano plots comparing p-values to effect size estimates

In each of these cases, we can easily filter, facet, or distinguish based on the `term` column. In short, this makes the tools of tidy data analysis available for the *results* of data analysis and models, not just the inputs.

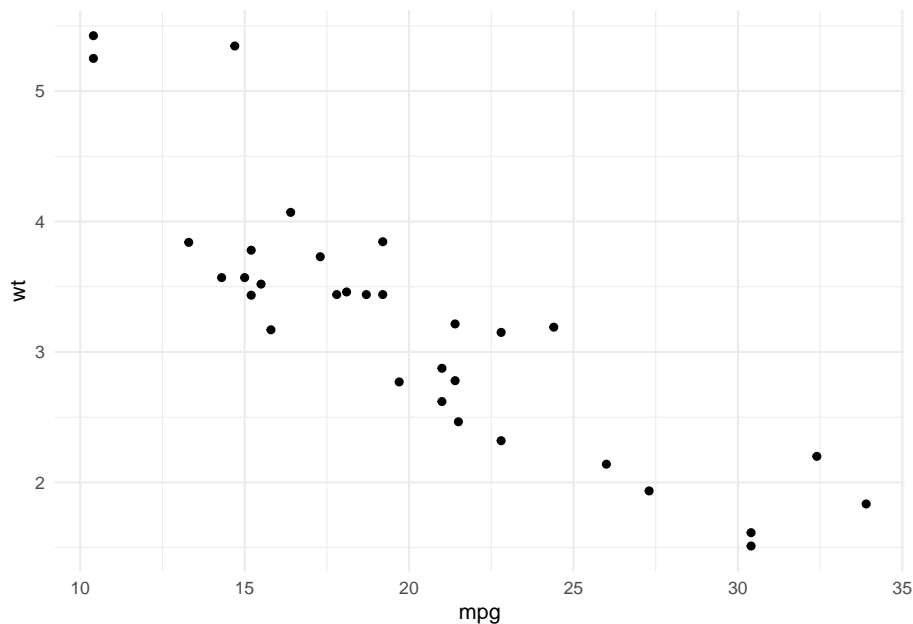
6.1 Tidy bootstrapping

Another place where combining model fits in a tidy way becomes useful is when performing bootstrapping or permutation tests. These approaches have been explored before, for instance by Andrew MacDonald here, and Hadley has explored efficient support for bootstrapping as a potential enhancement to `dplyr`. `broom` fits naturally with `dplyr` in performing these analyses.

Bootstrapping consists of randomly sampling a dataset with replacement, then performing the analysis individually on each bootstrapped replicate. The variation in the resulting estimate is then a reasonable approximation of the variance in our estimate.

Let's say we want to fit a nonlinear model to the weight/mileage relationship in the `mtcars` dataset.

```
library(ggplot2)
theme_set(theme_minimal())
ggplot(mtcars, aes(mpg, wt)) +
  geom_point()
```



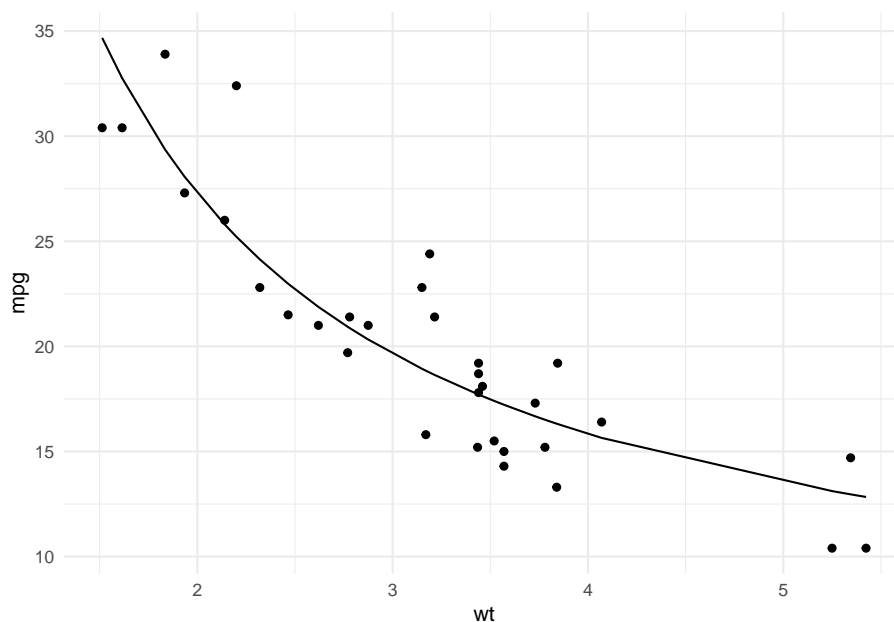
We might use the method of nonlinear least squares (via the `nls` function) to fit a model.

```
nlsfit <- nls(mpg ~ k / wt + b, mtcars, start = list(k = 1, b = 0))
summary(nlsfit)
```

```
##
## Formula: mpg ~ k/wt + b
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## k   45.829     4.249  10.786 7.64e-12 ***
## b    4.386     1.536   2.855 0.00774 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.774 on 30 degrees of freedom
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 2.877e-08

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = predict(nlsfit)))
```



While this does provide a p-value and confidence intervals for the parameters, these are based on model assumptions that may not hold in real data. Bootstrapping is a popular method for providing confidence intervals and predictions that are more robust to the nature of the data.

We can use the `bootstraps` function in the `rsample` package to sample bootstrap replications. First, we construct 100 bootstrap replications of the data, each of which has been randomly sampled with replacement. The resulting object is an `rset`, which is a dataframe with a column of `rsplit` objects.

An `rsplit` object has two main components: an analysis dataset and an assessment dataset, accessible via `analysis(rsplit)` and `assessment(rsplit)` respectively. For bootstrap samples, the analysis dataset is the bootstrap sample itself, and the assessment dataset consists of all the out of bag samples.

```
library(dplyr)
library(rsample)
library(broom)
library(purrr)
library(tidyr)
set.seed(27)
boots <- bootstraps(mtcars, times = 100)
boots
```

```
## # Bootstrap sampling
## # A tibble: 100 x 2
##   splits      id
##   <list>      <chr>
## 1 <split [32/13]> Bootstrap001
## 2 <split [32/10]> Bootstrap002
## 3 <split [32/13]> Bootstrap003
## 4 <split [32/11]> Bootstrap004
## 5 <split [32/9]>  Bootstrap005
## 6 <split [32/10]> Bootstrap006
## 7 <split [32/11]> Bootstrap007
## 8 <split [32/13]> Bootstrap008
## 9 <split [32/11]> Bootstrap009
## 10 <split [32/11]> Bootstrap010
## # ... with 90 more rows
```

We create a helper function to fit an `nls` model on each bootstrap sample, and then use `purrr::map` to apply this function to all the bootstrap samples at once. Similarly, we create a column of tidy coefficient information by unnesting.

```
fit_nls_on_bootstrap <- function(split) {
  nls(mpg ~ k / wt + b, analysis(split), start = list(k = 1, b = 0))
}
boot_models <- boots %>%
  mutate(model = map(splits, fit_nls_on_bootstrap),
         coef_info = map(model, tidy))
boot_models
```

```
## # Bootstrap sampling
## # A tibble: 100 x 4
##   splits      id      model  coef_info
##   <list>      <chr>    <list> <list>
## 1 <split [32/13]> Bootstrap001 <nls>  <tibble [2 x 5]>
## 2 <split [32/10]> Bootstrap002 <nls>  <tibble [2 x 5]>
## 3 <split [32/13]> Bootstrap003 <nls>  <tibble [2 x 5]>
## 4 <split [32/11]> Bootstrap004 <nls>  <tibble [2 x 5]>
## 5 <split [32/9]>  Bootstrap005 <nls>  <tibble [2 x 5]>
## 6 <split [32/10]> Bootstrap006 <nls>  <tibble [2 x 5]>
```

```
## 7 <split [32/11]> Bootstrap007 <nls> <tibble [2 x 5]>
## 8 <split [32/13]> Bootstrap008 <nls> <tibble [2 x 5]>
## 9 <split [32/11]> Bootstrap009 <nls> <tibble [2 x 5]>
## 10 <split [32/11]> Bootstrap010 <nls> <tibble [2 x 5]>
## # ... with 90 more rows

boot_coefs <- boot_models %>%
  unnest(coef_info)
```

The unnested coefficient information contains a summary of each replication combined in a single data frame:

```
boot_coefs
```

```
## # A tibble: 200 x 8
##   splits      id      model term estimate std.error stati-1 p.value
##   <list>      <chr>    <list> <chr>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 <split [32/13]> Bootstrap001 <nls> k      42.1      4.05    10.4  1.91e-11
## 2 <split [32/13]> Bootstrap001 <nls> b       5.39     1.43     3.78  6.93e- 4
## 3 <split [32/10]> Bootstrap002 <nls> k      49.9     5.66     8.82  7.82e-10
## 4 <split [32/10]> Bootstrap002 <nls> b       3.73     1.92     1.94  6.13e- 2
## 5 <split [32/13]> Bootstrap003 <nls> k      37.8     2.68    14.1  9.01e-15
## 6 <split [32/13]> Bootstrap003 <nls> b       6.73     1.17     5.75  2.78e- 6
## 7 <split [32/11]> Bootstrap004 <nls> k      45.6     4.45    10.2  2.70e-11
## 8 <split [32/11]> Bootstrap004 <nls> b       4.75     1.62     2.93  6.38e- 3
## 9 <split [32/9]> Bootstrap005 <nls> k      43.6     4.63     9.41  1.85e-10
## 10 <split [32/9]> Bootstrap005 <nls> b       5.89     1.68     3.51  1.44e- 3
## # ... with 190 more rows, and abbreviated variable name 1: statistic
```

We can then calculate confidence intervals (using what is called the percentile method):

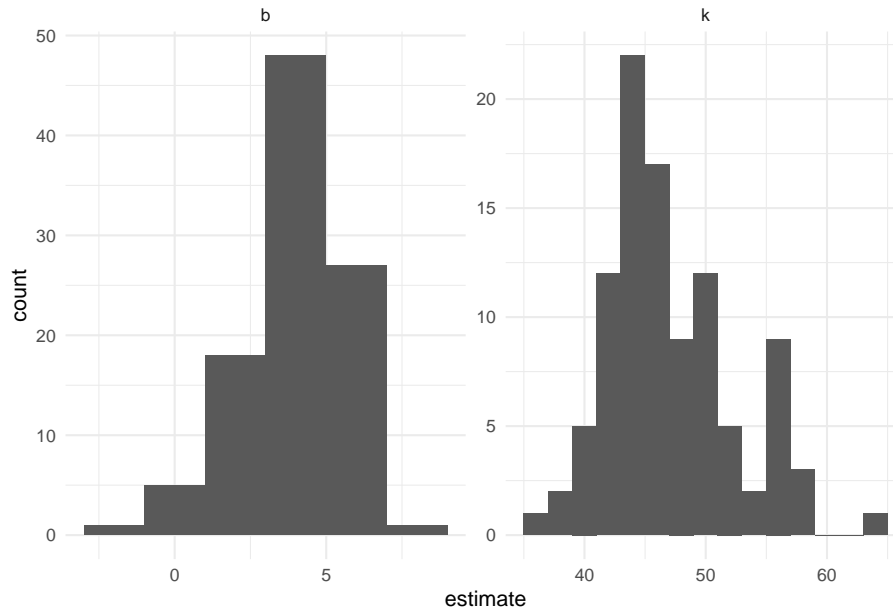
```
alpha <- .05
boot_coefs %>%
  group_by(term) %>%
  summarize(low = quantile(estimate, alpha / 2),
            high = quantile(estimate, 1 - alpha / 2))

## # A tibble: 2 x 3
##   term      low high
##   <chr>    <dbl> <dbl>
## 1 b       0.283  6.74
## 2 k      38.5  57.6
```

Or we can use histograms to get a more detailed idea of the uncertainty in each estimate:

```
ggplot(boot_coefs, aes(estimate)) +
  geom_histogram(binwidth = 2) +
```

```
facet_wrap(~ term, scales = "free")
```



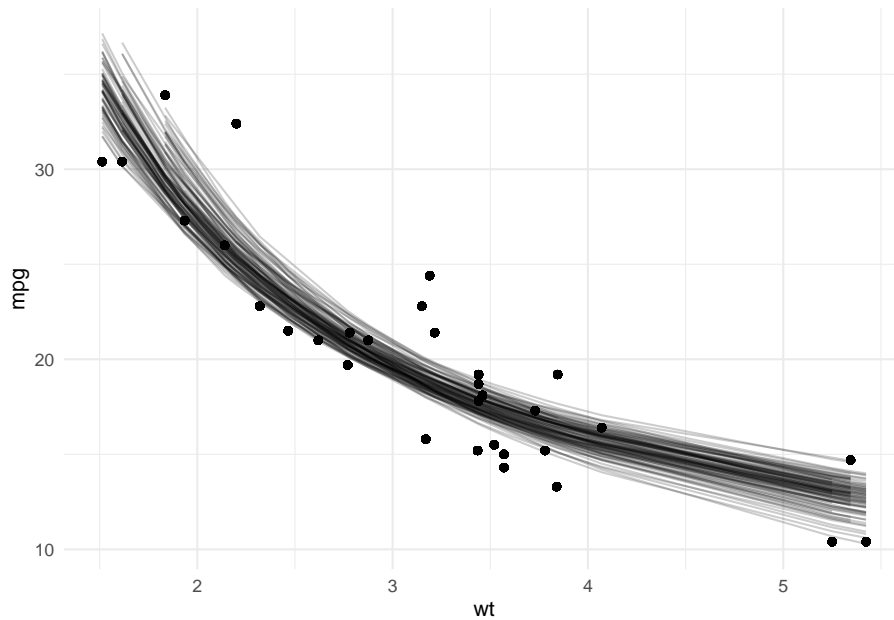
Or we can use `augment` to visualize the uncertainty in the curve:

```
boot_aug <- boot_models %>%
  mutate(augmented = map(model, augment)) %>%
  unnest(augmented)
boot_aug
```

```
## # A tibble: 3,200 x 8
##   splits      id      model coef_info  mpg  wt .fitted .resid
##   <list>      <chr>    <list> <list>    <dbl> <dbl> <dbl> <dbl>
## 1 <split [32/13]> Bootstrap001 <nls> <tibble>  18.7  3.44  17.6  1.08
## 2 <split [32/13]> Bootstrap001 <nls> <tibble>  32.4  2.2   24.5  7.89
## 3 <split [32/13]> Bootstrap001 <nls> <tibble>  15.5  3.52  17.3 -1.84
## 4 <split [32/13]> Bootstrap001 <nls> <tibble>  22.8  3.15  18.7  4.05
## 5 <split [32/13]> Bootstrap001 <nls> <tibble>  24.4  3.19  18.6  5.82
## 6 <split [32/13]> Bootstrap001 <nls> <tibble>  30.4  1.62  31.4 -1.04
## 7 <split [32/13]> Bootstrap001 <nls> <tibble>  10.4  5.42  13.1 -2.75
## 8 <split [32/13]> Bootstrap001 <nls> <tibble>  21    2.62  21.4 -0.448
## 9 <split [32/13]> Bootstrap001 <nls> <tibble>  19.2  3.84  16.3  2.87
## 10 <split [32/13]> Bootstrap001 <nls> <tibble>  21    2.62  21.4 -0.448
## # ... with 3,190 more rows
```

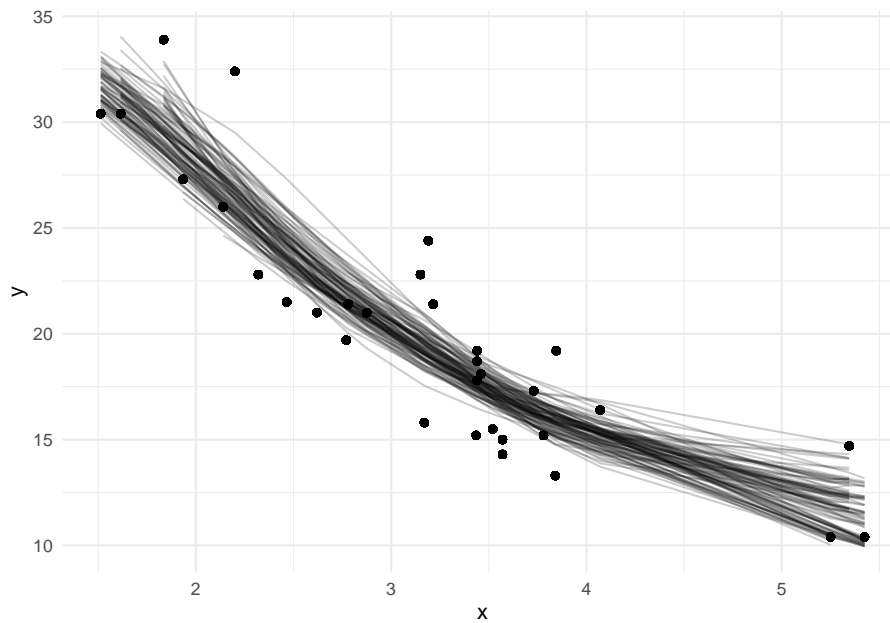
```
ggplot(boot_aug, aes(wt, mpg)) +
  geom_point() +
```

```
geom_line(aes(y = .fitted, group = id), alpha=.2)
```



With only a few small changes, we could easily perform bootstrapping with other kinds of predictive or hypothesis testing models, since the `tidy` and `augment` functions work for many statistical outputs. As another example, we could use `smooth.spline`, which fits a cubic smoothing spline to data:

```
fit_spline_on_bootstrap <- function(split) {
  data <- analysis(split)
  smooth.spline(data$wt, data$mpg, df = 4)
}
boot_splines <- boots %>%
  mutate(spline = map(splits, fit_spline_on_bootstrap),
         aug_train = map(spline, augment))
splines_aug <- boot_splines %>%
  unnest(aug_train)
ggplot(splines_aug, aes(x, y)) +
  geom_point() +
  geom_line(aes(y = .fitted, group = id), alpha = 0.2)
```

6.2 links

https://www.youtube.com/watch?v=1bnhT8t1CJQ&list=PLBnFxG6owe1F-3y0_aphRZ5YHH06Qr1Kj

https://bookdown.org/bruno_lucian_costa/CursoIntermediarioR/tidyr.html

<https://bookdown.org/Maxine/r4ds/nesting.html>

<https://livro.curso-r.com/7-3-tidyr.html>

<http://leg.ufpr.br/~walmes/cursor/data-vis/slides/04-tidyr.pdf>

Bibliography

- Hadley Wickham. *tidyverse: Easily Install and Load the Tidyverse*, 2023. URL <https://CRAN.R-project.org/package=tidyverse>. R package version 2.0.0.
- Hadley Wickham and Lionel Henry. *purrr: Functional Programming Tools*, 2023. URL <https://CRAN.R-project.org/package=purrr>. R package version 1.0.1.
- Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Golemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686.
- Hadley Wickham, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2023a. URL <https://CRAN.R-project.org/package=ggplot2>. R package version 3.4.1.
- Hadley Wickham, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. *dplyr: A Grammar of Data Manipulation*, 2023b. URL <https://CRAN.R-project.org/package=dplyr>. R package version 1.1.1.
- Hadley Wickham, Davis Vaughan, and Maximilian Girlich. *tidyr: Tidy Messy Data*, 2023c. URL <https://CRAN.R-project.org/package=tidyr>. R package version 1.3.0.
- Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*, 2023a. URL <https://CRAN.R-project.org/package=bookdown>. R package version 0.33.
- Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2023b. URL <https://yihui.org/knitr/>. R package version 1.42.