Learning R packages

Cristian Villegas

2023-06-16

Sumário

Capítulo 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)
```

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).

Capítulo 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://fueleconomy.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~
## $ 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
## $ manufacturer <fct> audi, audi, audi, audi, audi, audi, audi, audi, audi, aud~
## $ model        <fct> a4, a4, a4, a4, a4, a4, a4, a4 quattro, a4 quattro, a4 qu~
## $ 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        <fct> auto(l5), manual(m5), manual(m6), auto(av), auto(l5), man~
## $ drv          <fct> f, f, f, f, f, f, f, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, r, ~
## $ 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           <fct> p, p, p, p, p, p, p, p, p, p, p, p, p, p, p, p, p, p, r, ~
## $ class        <fct> compact, compact, compact, compact, compact, compact, com~
```

2.3 Lista de funções do pacote dplyr

```
ls("package:dplyr")
```

```
##   [1] "%>%"                  "across"               "add_count"
##   [4] "add_count_"           "add_row"              "add_rownames"
##   [7] "add_tally"            "add_tally_"           "all_equal"
##  [10] "all_of"               "all_vars"             "anti_join"
##  [13] "any_of"               "any_vars"             "arrange"
##  [16] "arrange_"             "arrange_all"          "arrange_at"
##  [19] "arrange_if"           "as.tbl"               "as_data_frame"
##  [22] "as_label"             "as_tibble"            "auto_copy"
##  [25] "band_instruments"     "band_instruments2"    "band_members"
##  [28] "bench_tbls"           "between"              "bind_cols"
##  [31] "bind_rows"            "c_across"             "case_match"
##  [34] "case_when"            "changes"              "check_dbplyr"
##  [37] "coalesce"             "collapse"             "collect"
##  [40] "combine"              "common_by"            "compare_tbls"
##  [43] "compare_tbls2"        "compute"              "consecutive_id"
##  [46] "contains"             "copy_to"              "count"
##  [49] "count_"               "cross_join"           "cumall"
##  [52] "cumany"               "cume_dist"            "cummean"
```

```
##  [55] "cur_column"            "cur_data"              "cur_data_all"
##  [58] "cur_group"             "cur_group_id"          "cur_group_rows"
##  [61] "current_vars"          "data_frame"            "db_analyze"
##  [64] "db_begin"              "db_commit"             "db_create_index"
##  [67] "db_create_indexes"     "db_create_table"       "db_data_type"
##  [70] "db_desc"               "db_drop_table"         "db_explain"
##  [73] "db_has_table"          "db_insert_into"        "db_list_tables"
##  [76] "db_query_fields"       "db_query_rows"         "db_rollback"
##  [79] "db_save_query"         "db_write_table"        "dense_rank"
##  [82] "desc"                  "dim_desc"              "distinct"
##  [85] "distinct_"             "distinct_all"          "distinct_at"
##  [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"                   "symdiff"               "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 colunas

```
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(l5)      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(l5)      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(l5)      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(l5)   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(l5)   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(l5)   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(l5)   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(l5)   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(l5)   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 =
```

```
## # 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_nao_duplicados <- distinct(dados_duplicados)
dados_nao_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
```

Capítulo 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

3.2 Lista de funções do pacote ggplot2

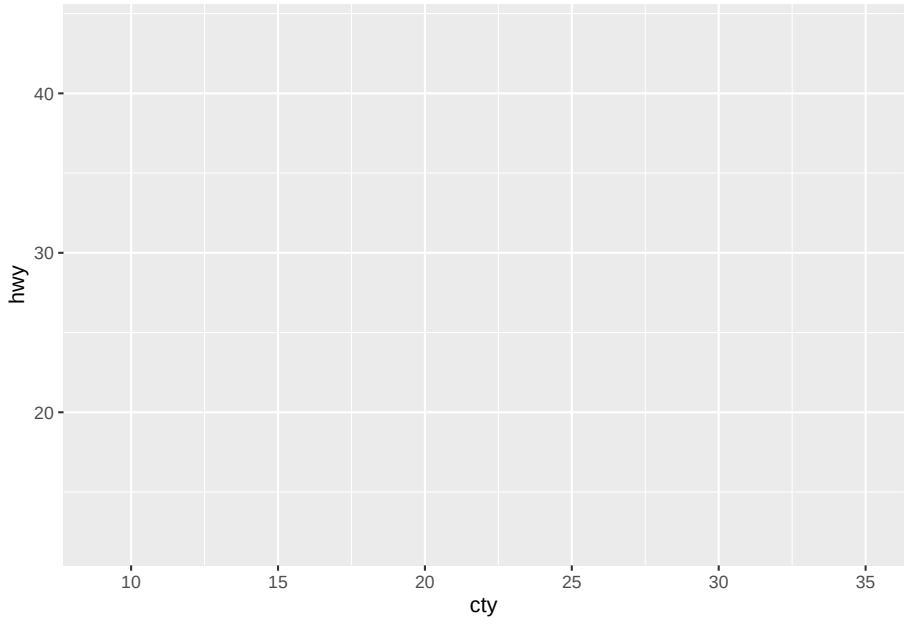
```
ls("package:ggplot2")
```

3.3 Primeiros passos usando geom_point

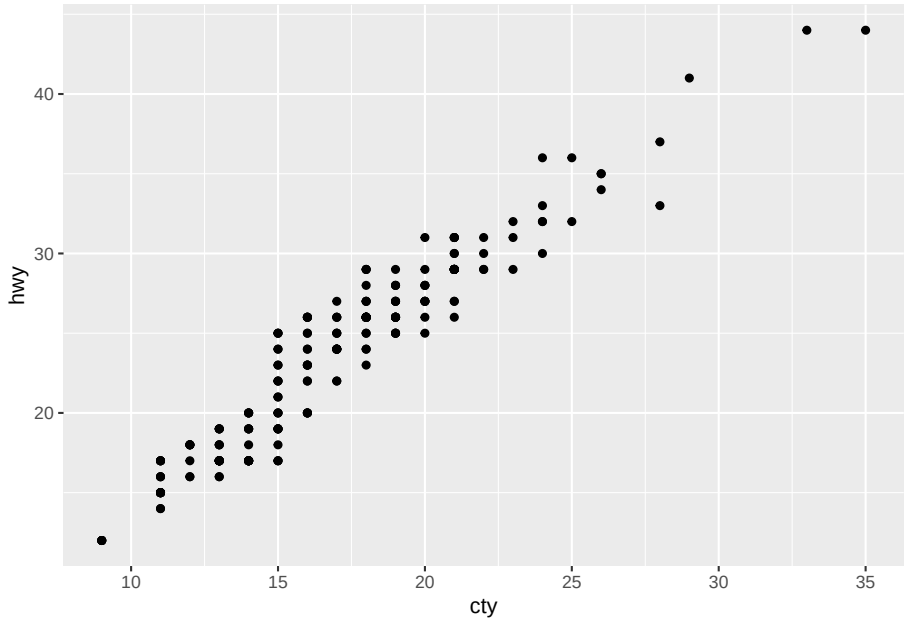
```
dados <- mpg
ggplot(dados)
```



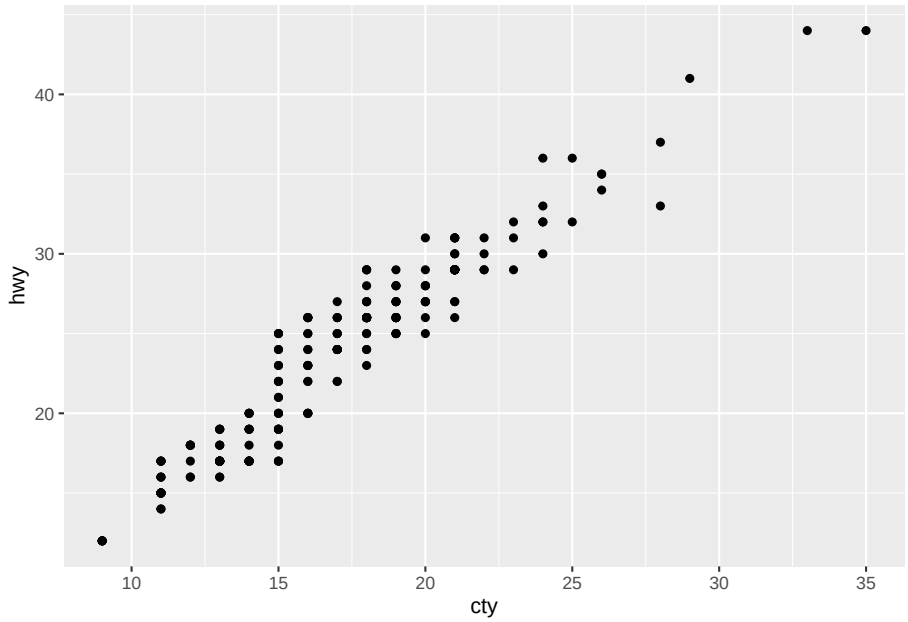
```
ggplot(dados, aes(x = cty, y = hwy))
```



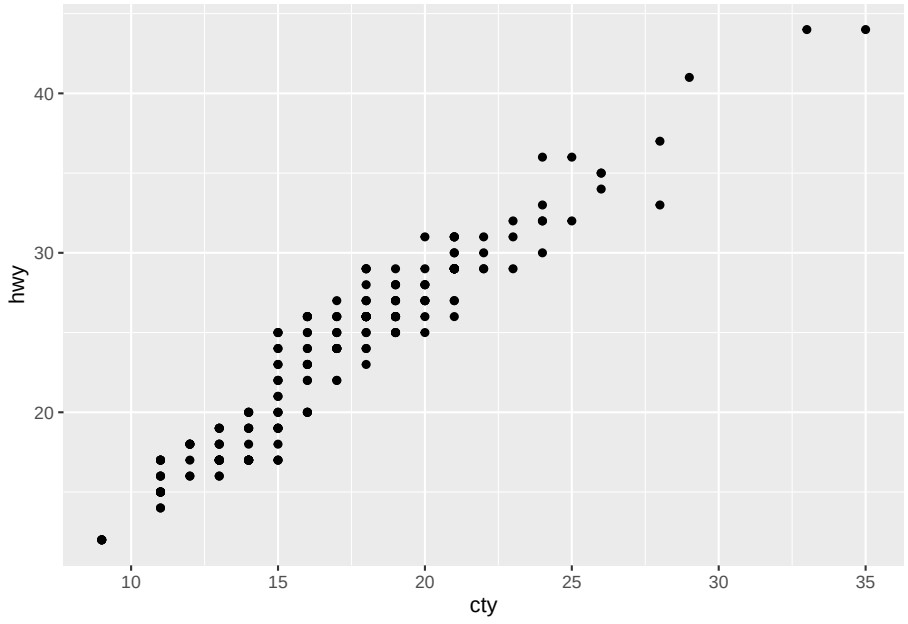

```
# Alternativas
ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point()
```



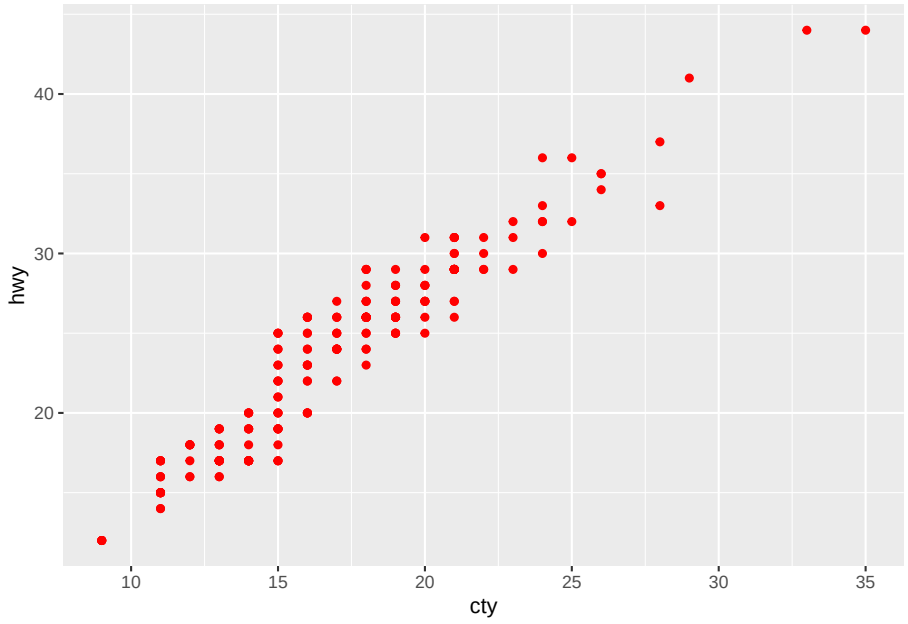
```
ggplot(dados) +
  geom_point(aes(x = cty, y = hwy))
```



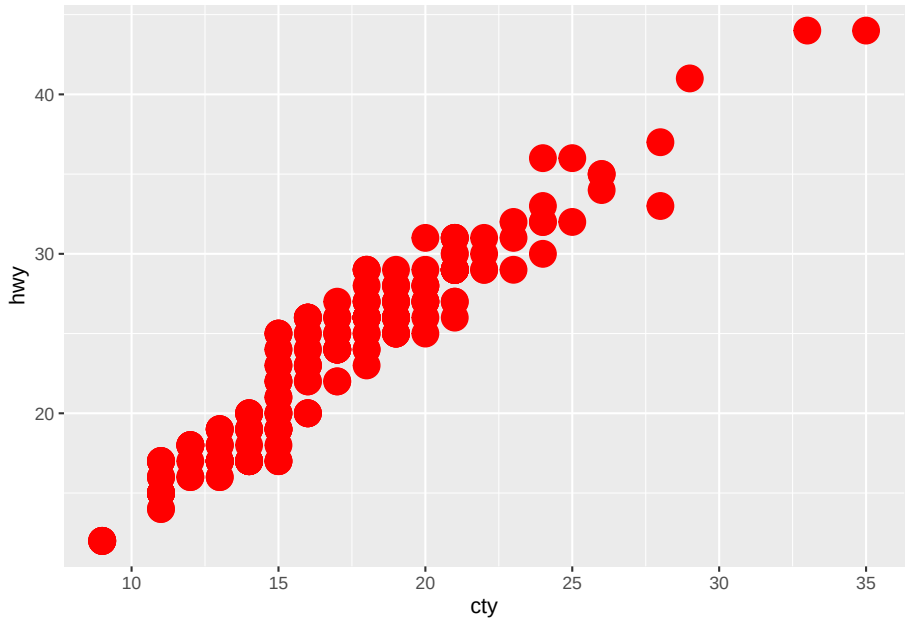

```
ggplot() +
  geom_point(data = dados, aes(x = cty, y = hwy))
```



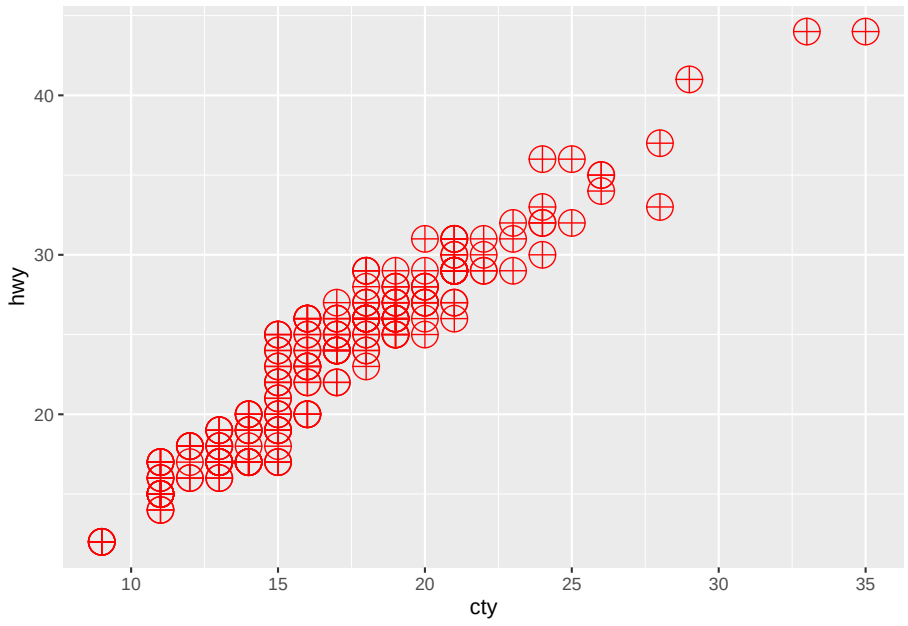

```
# Fim

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



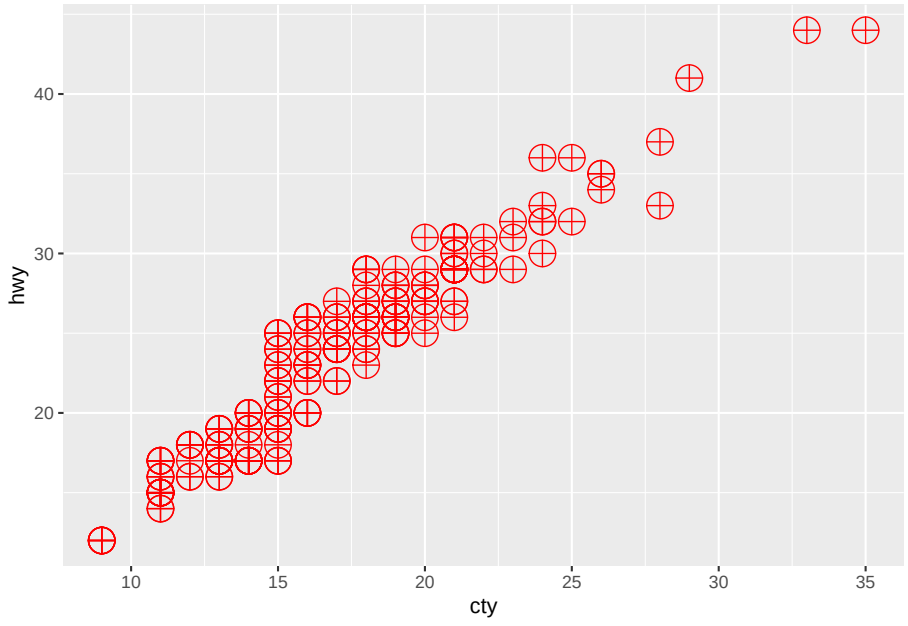
```
ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point(colour = "red", size = 6)
```



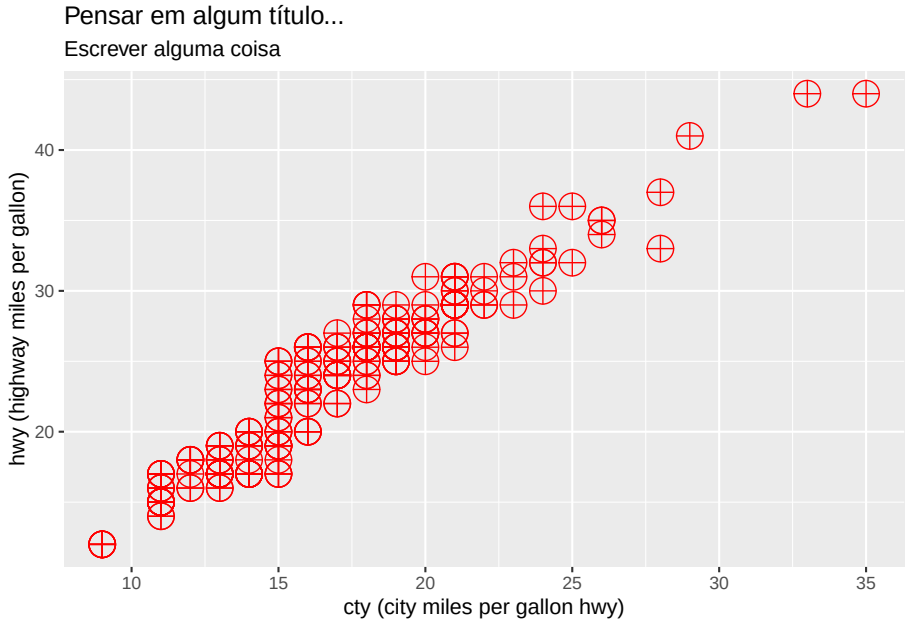

```
ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point(colour = "red", size = 6, shape = 10)
```




```
# 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

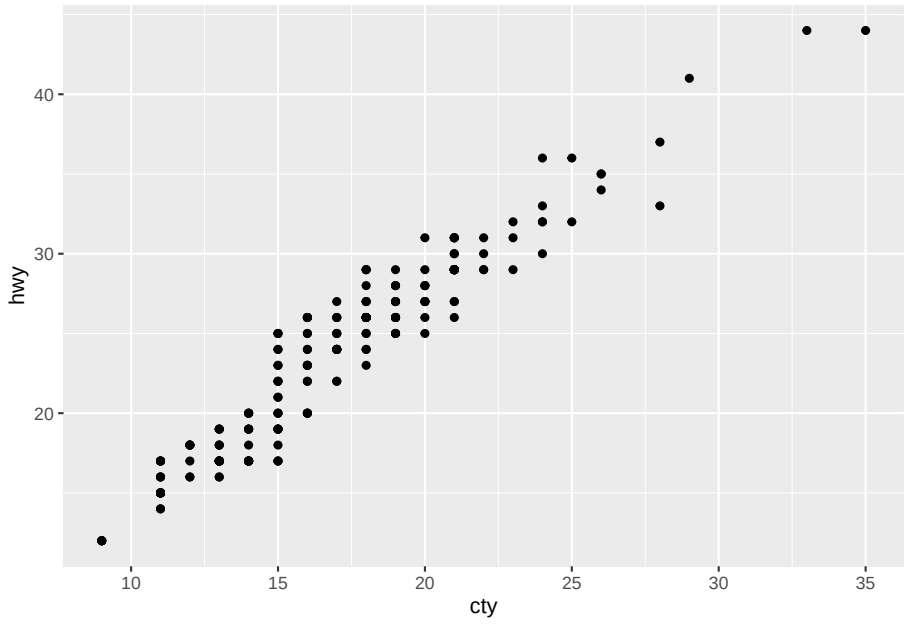
geom_point() understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- colour
- fill
- group
- shape

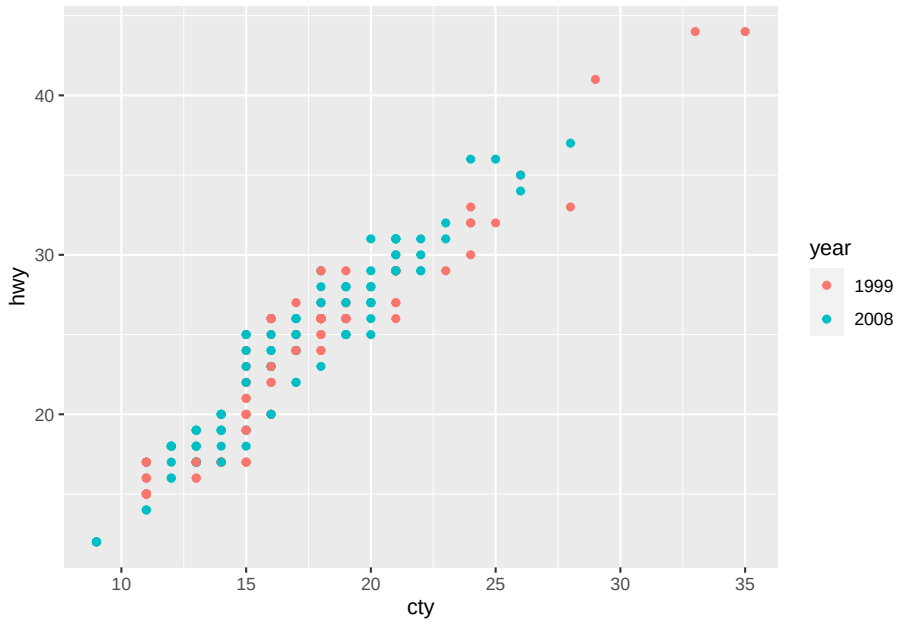
- size

- stroke

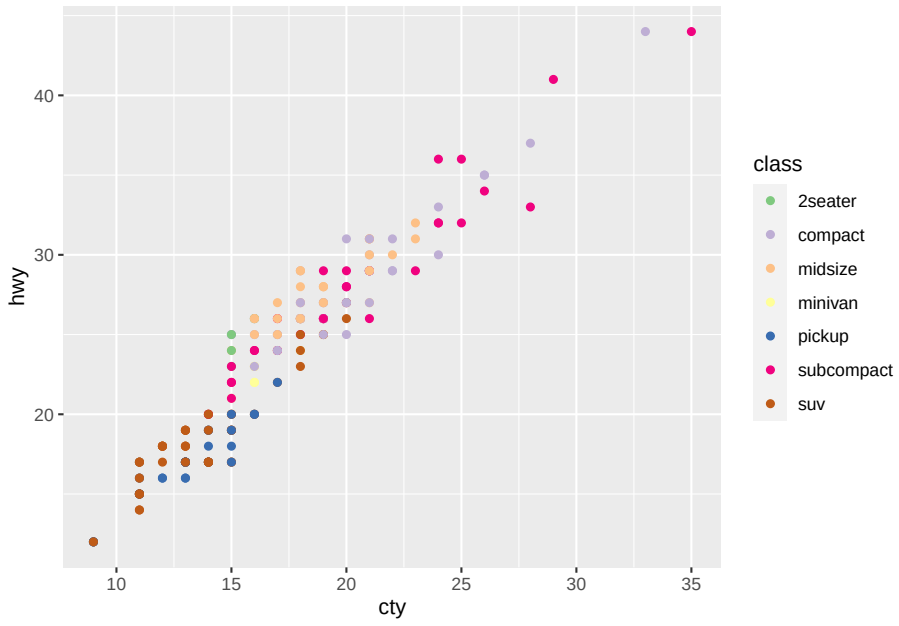
```
ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point()
```



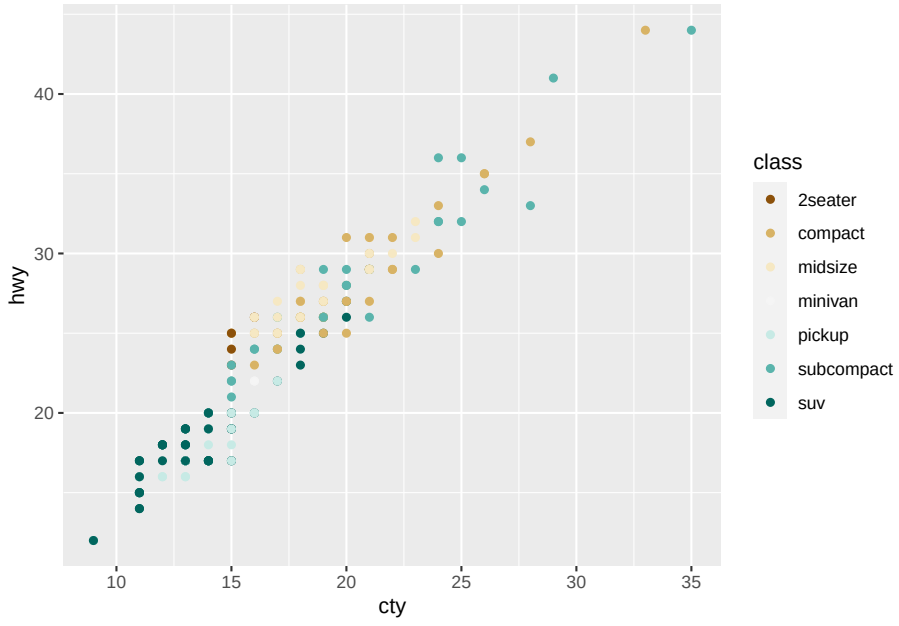

```
ggplot(dados, aes(x = cty, y = hwy, col = factor(year))) +
  geom_point() +
  labs(col = "year")
```



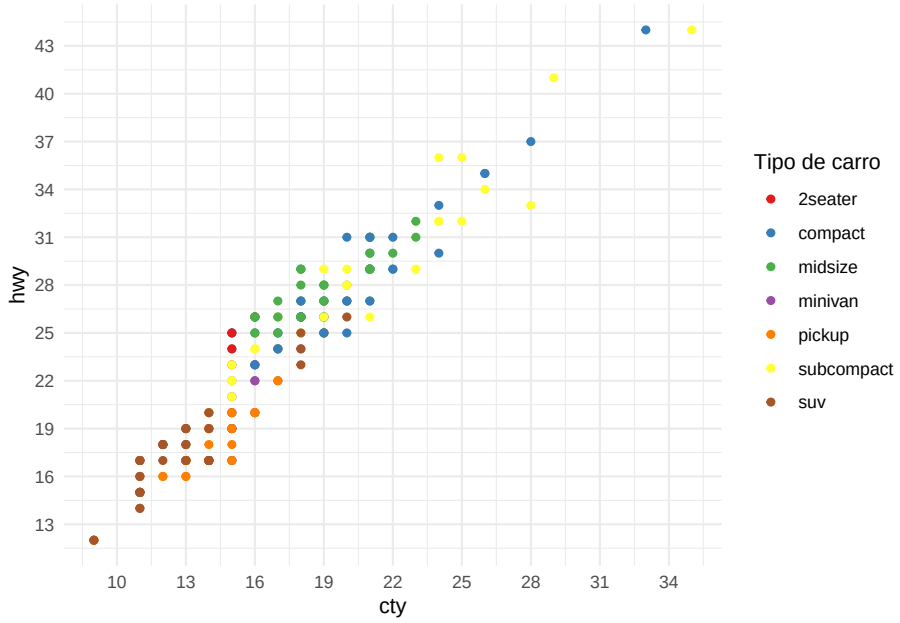

```
# Alternativa
ggplot(dados, aes(x = cty, y = hwy, col = factor(class))) +
  geom_point() +
  labs(col = "class")+
  scale_color_brewer(type = "qual")
```



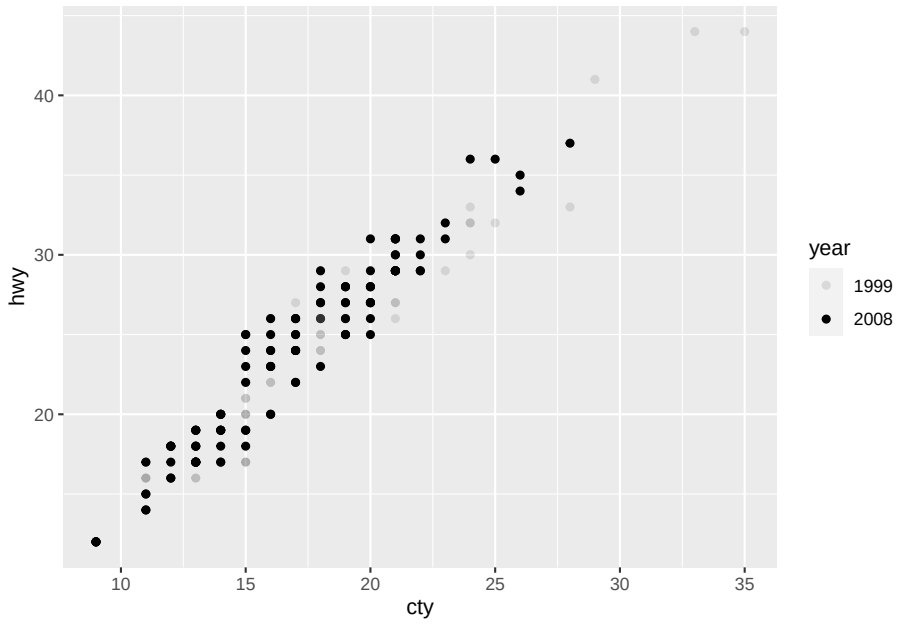

```
ggplot(dados, aes(x = cty, y = hwy, col = factor(class))) +
  geom_point() +
  labels(col = "class")+
  scale_color_brewer(type = "div")
```



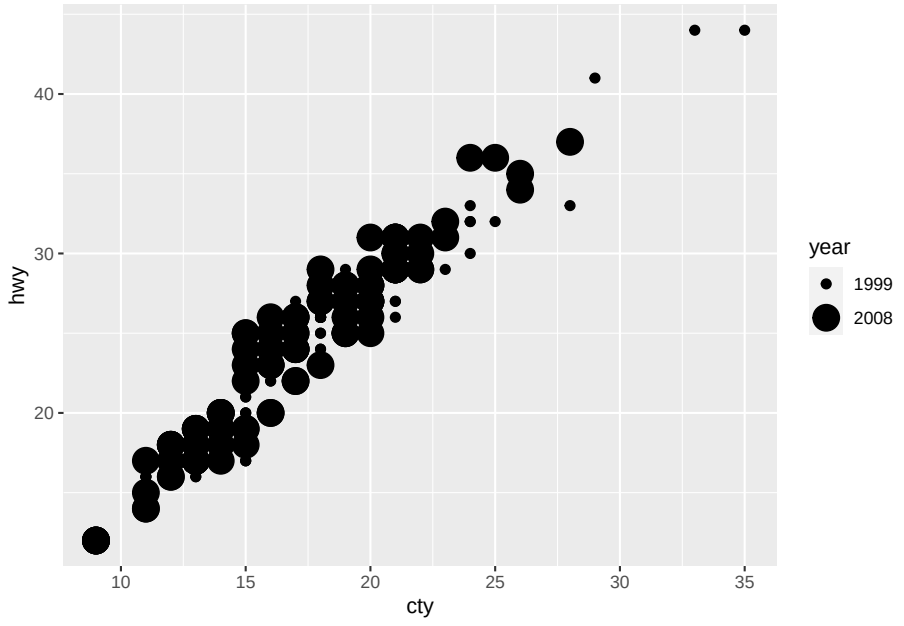
```
ggplot(dados, aes(x = cty, y = hwy, col = factor(class))) +
  geom_point() +
  labs(col = "class")+
  scale_color_brewer(palette = "Set1", name = "Tipo de carro")+
  scale_y_continuous(breaks = seq(10,60,3))+
  scale_x_continuous(breaks = seq(10,40,3))+
  theme_minimal()
```



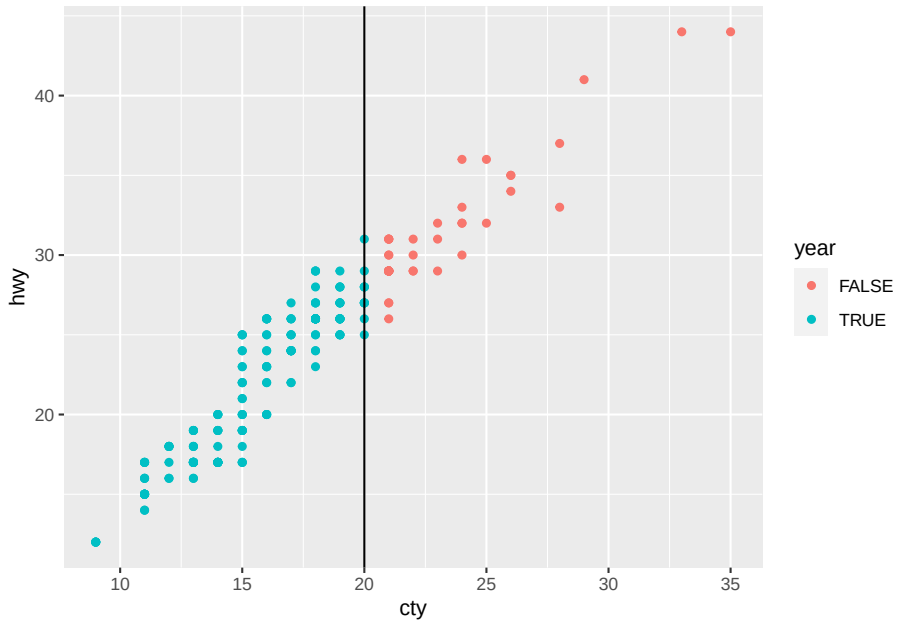

```
ggplot(dados, aes(x = cty, y = hwy, alpha = factor(year))) +
  geom_point() +
  labs(alpha = "year")
```



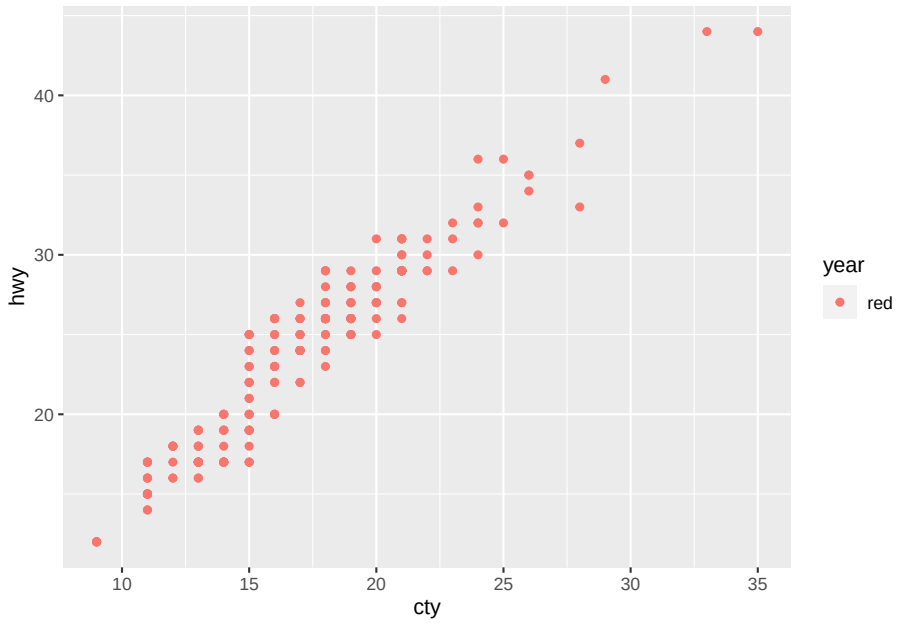

```
ggplot(dados, aes(x = cty, y = hwy, size = factor(year))) +
  geom_point() +
  labs(size = "year")
```



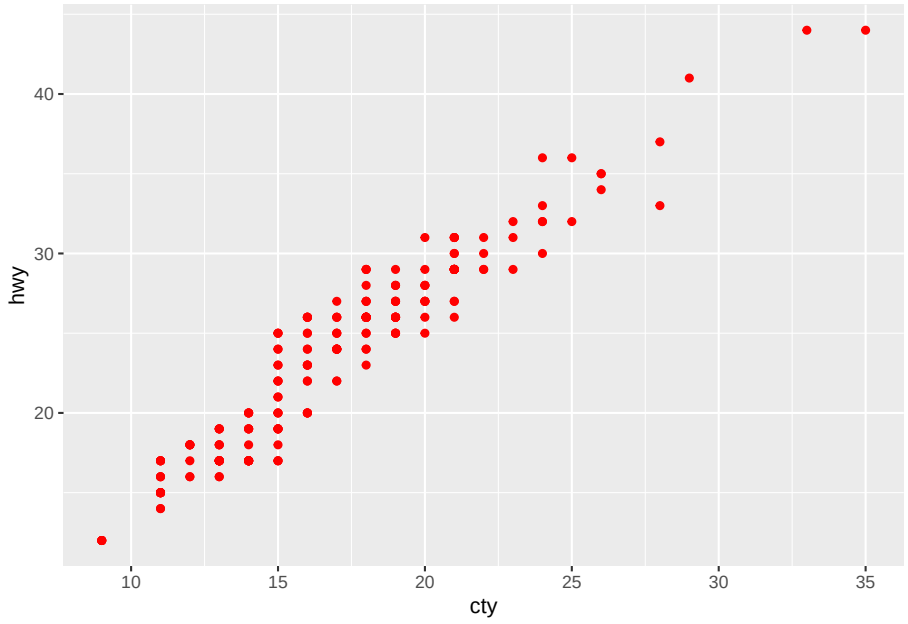
```
# Alternativa
ggplot(dados, aes(x = cty, y = hwy, col = cty <= 20)) +
  geom_point() +
  geom_vline(xintercept = 20)+
  labs(col = "year")
```



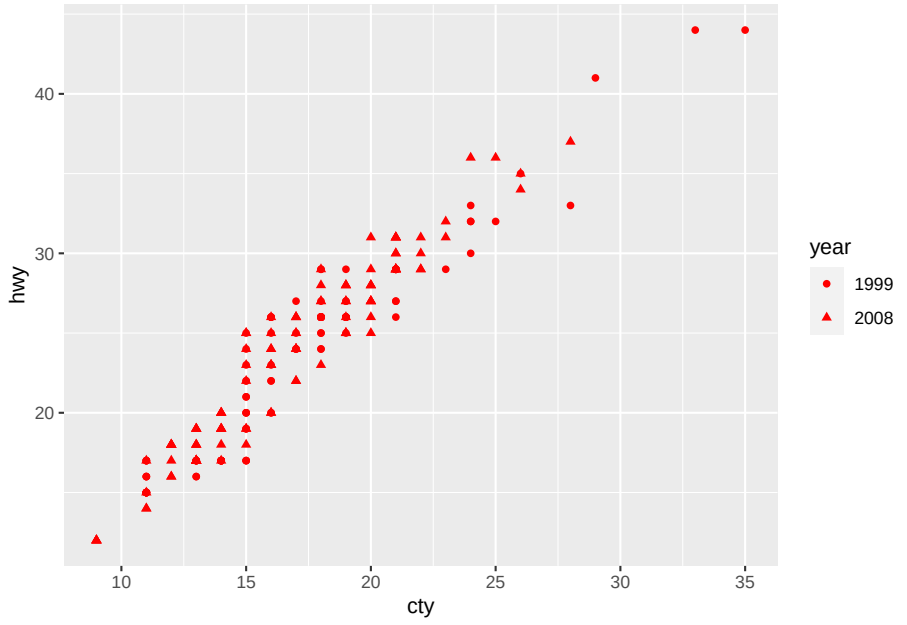

```
# Erro comum
ggplot(dados, aes(x = cty, y = hwy, col = "red")) +
  geom_point()+
  labs(col = "year")
```



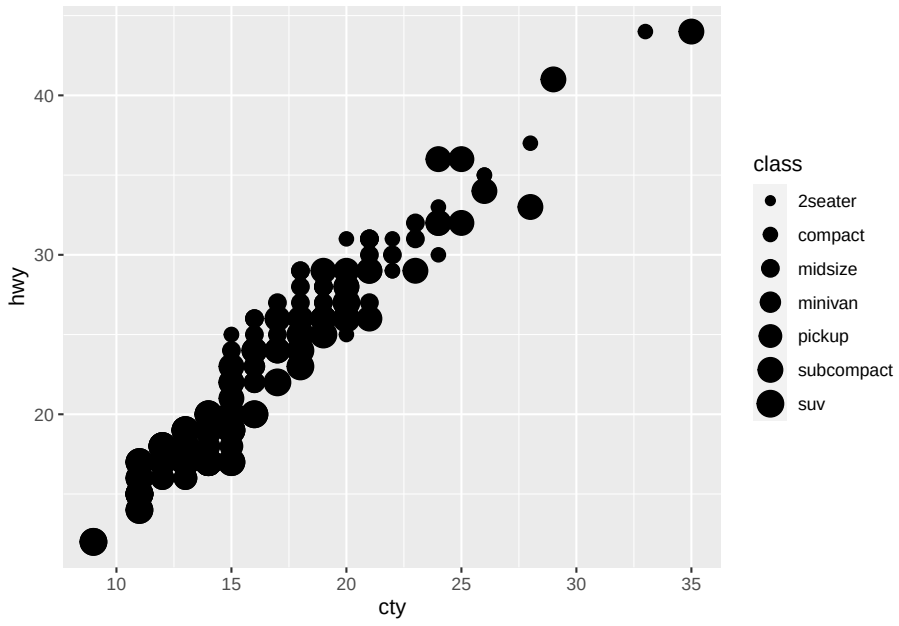

```
ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point(col = "red")+
  labs(col = "year")
```



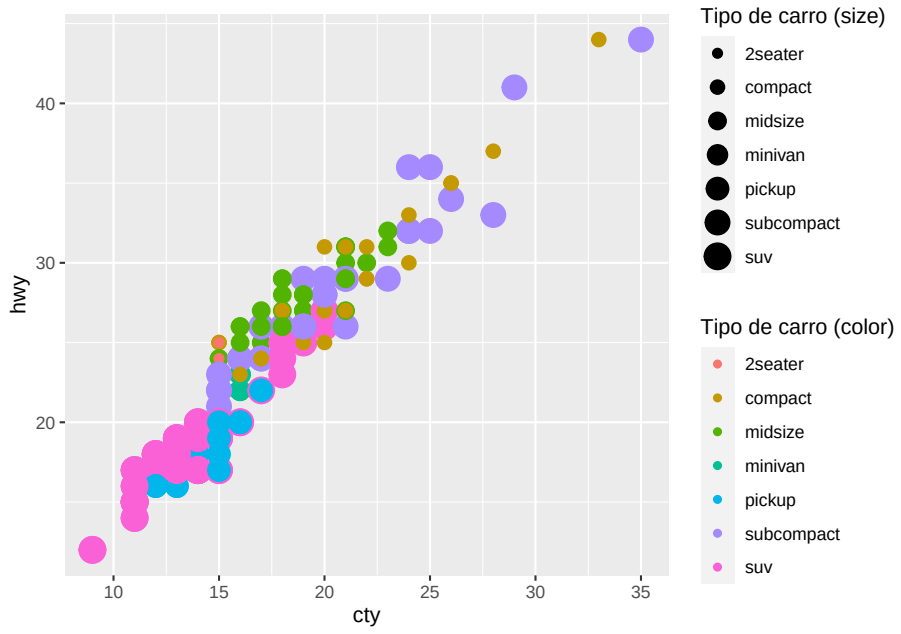
```
# Fim  Erro comum

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



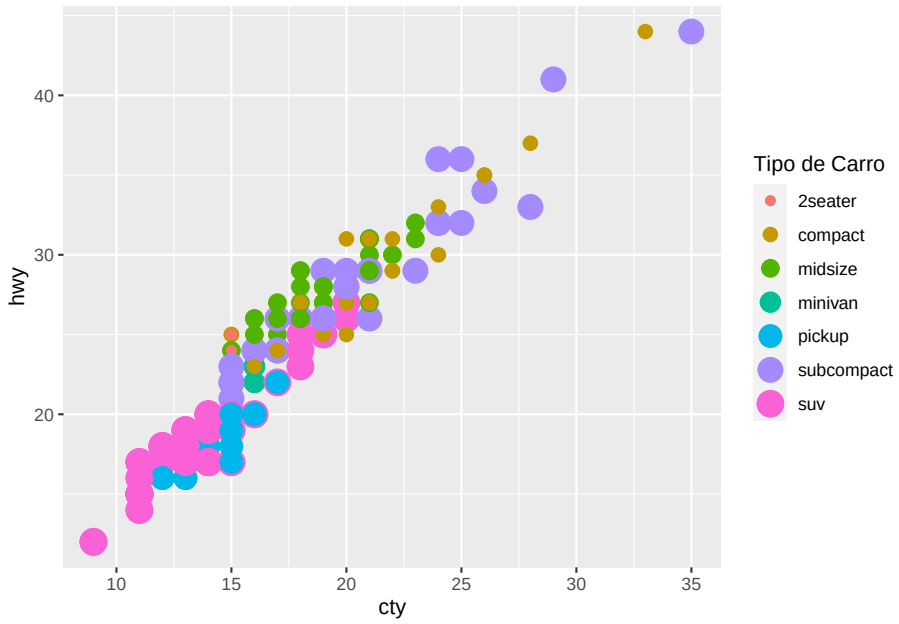

```
ggplot(dados, aes(x = cty, y = hwy, size = class)) +
  geom_point() +
  labs(size = "class")
```




```
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)"))
```

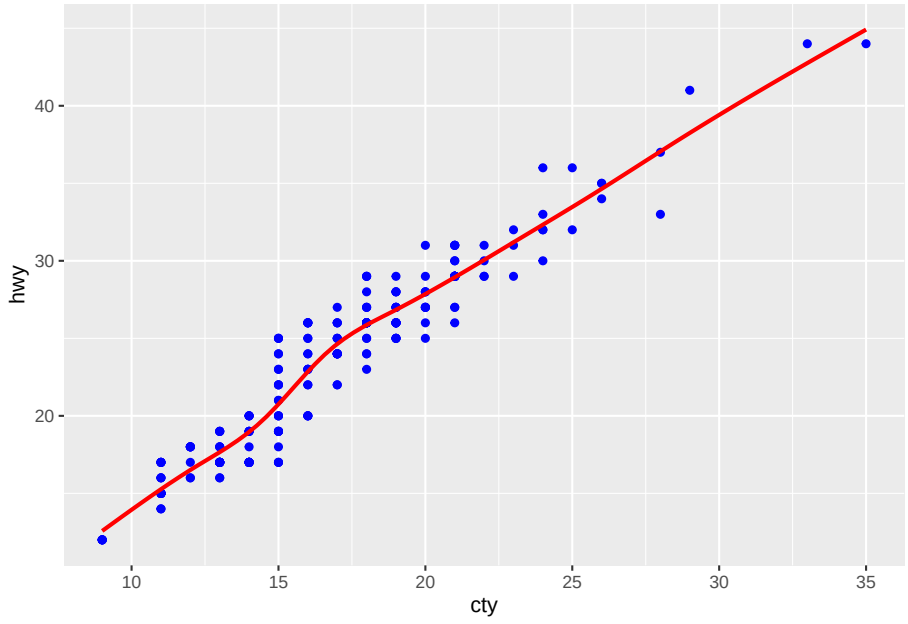



```
ggplot(dados, aes(x = cty, y = hwy,
                  size = class,
                  col = class)) +
  geom_point() +
  labs(col = "Tipo de Carro", size = "Tipo de Carro")+
  guides(col = "legend")
```

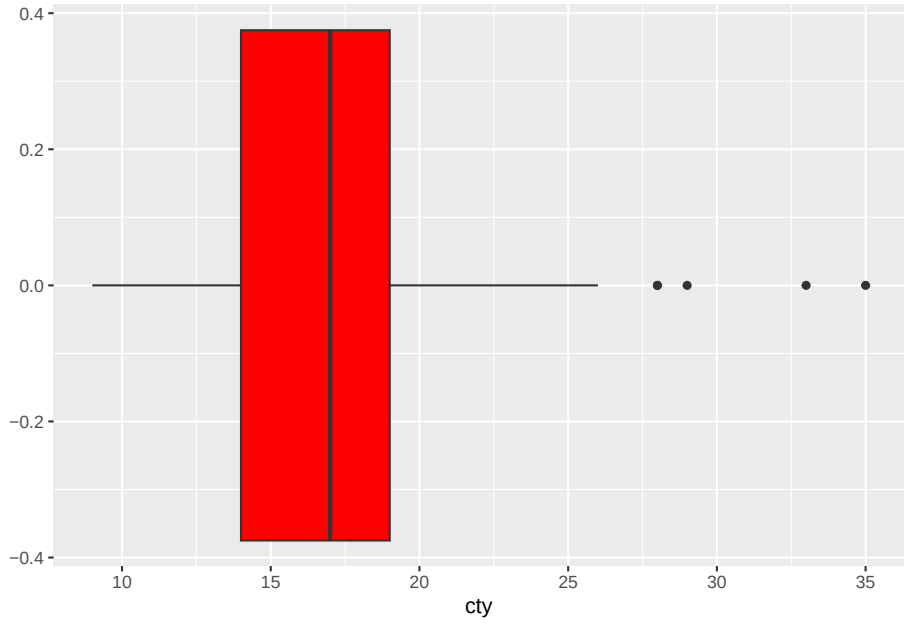


3.4 smooth, boxplot, histogram

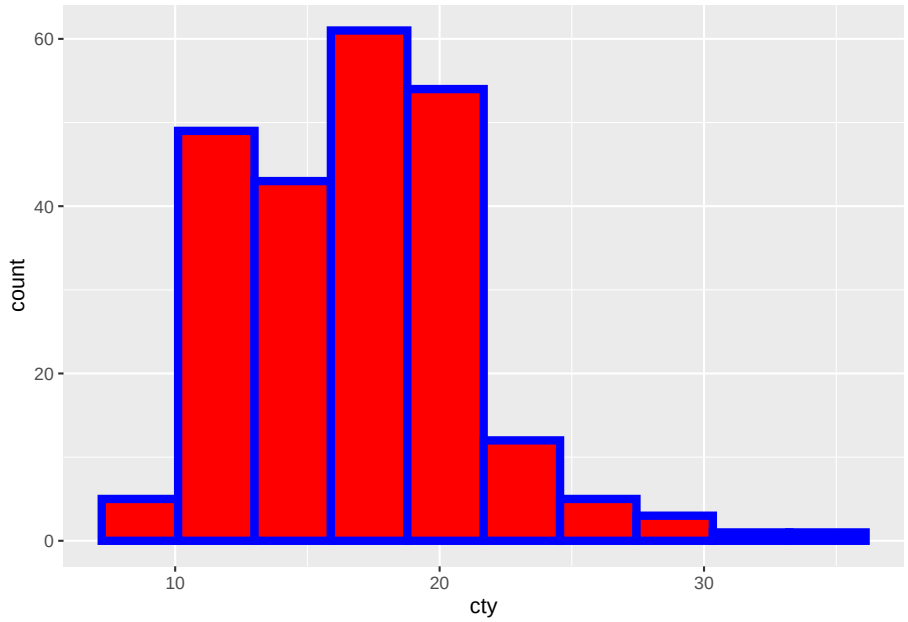
```
v1<- ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point(col = "blue")+
  geom_smooth(method = mgcv::gam,
              formula = y ~ s(x, bs = "cs") ,
              col = "red",
              se = FALSE)
v1
```



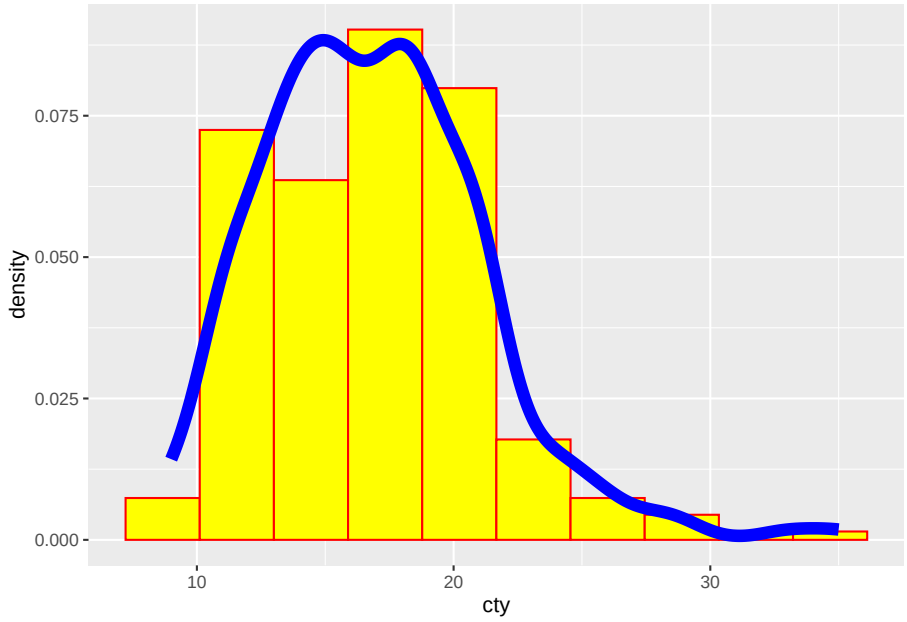

```
v2 <- ggplot(dados, aes(x = cty)) +
  geom_boxplot(fill = "red")
v2
```



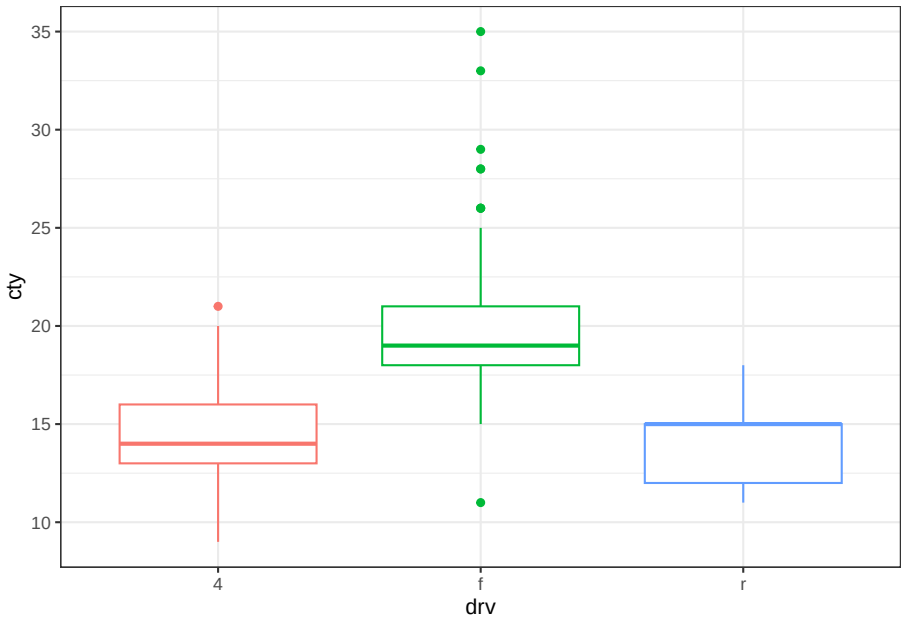
```
v3 <- ggplot(dados, aes(x = cty)) +
  geom_histogram(bins = 10, fill = "red", col = "blue", lwd=2)
v3
```




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



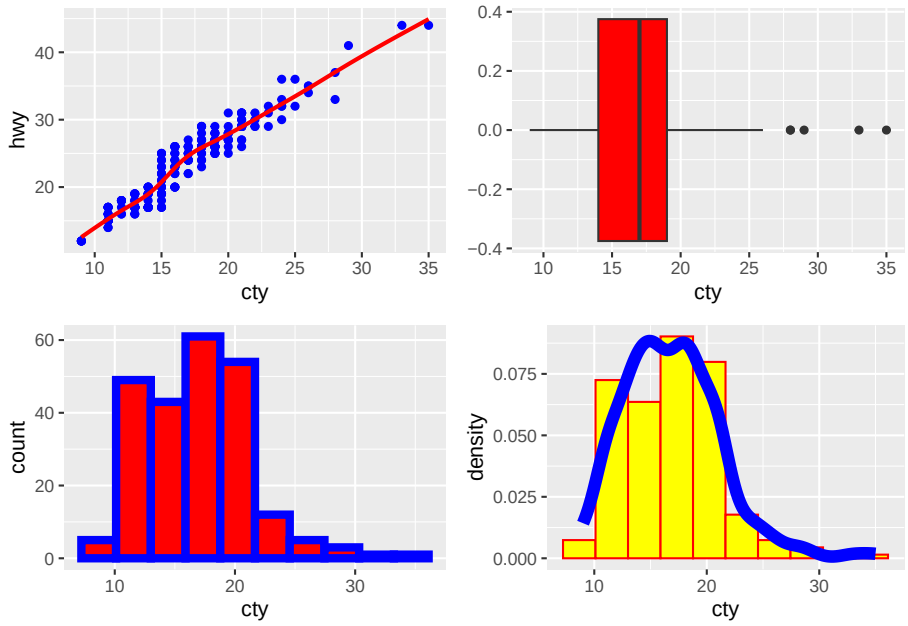
3.5 gridExtra e patchwork

Alguns links

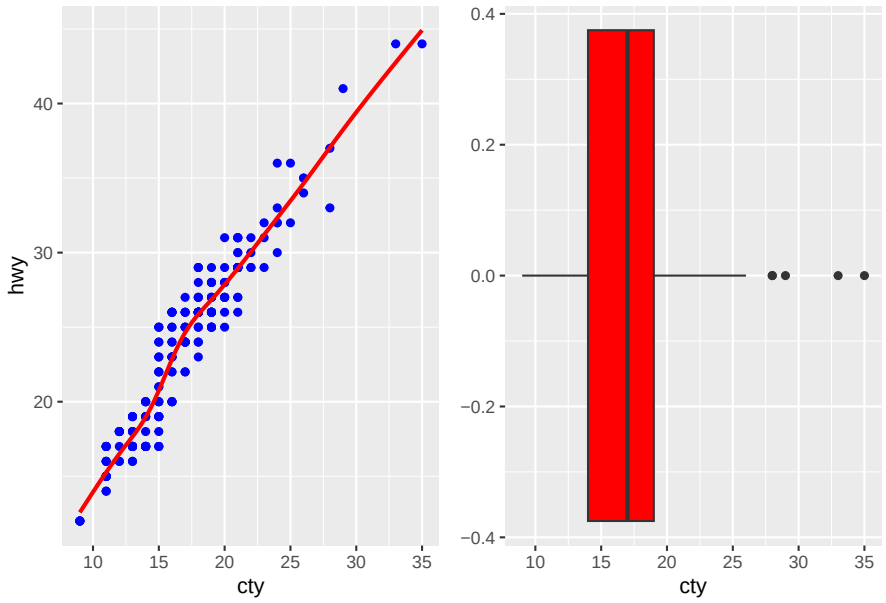
link 1: patchwork

link 2: patchwork

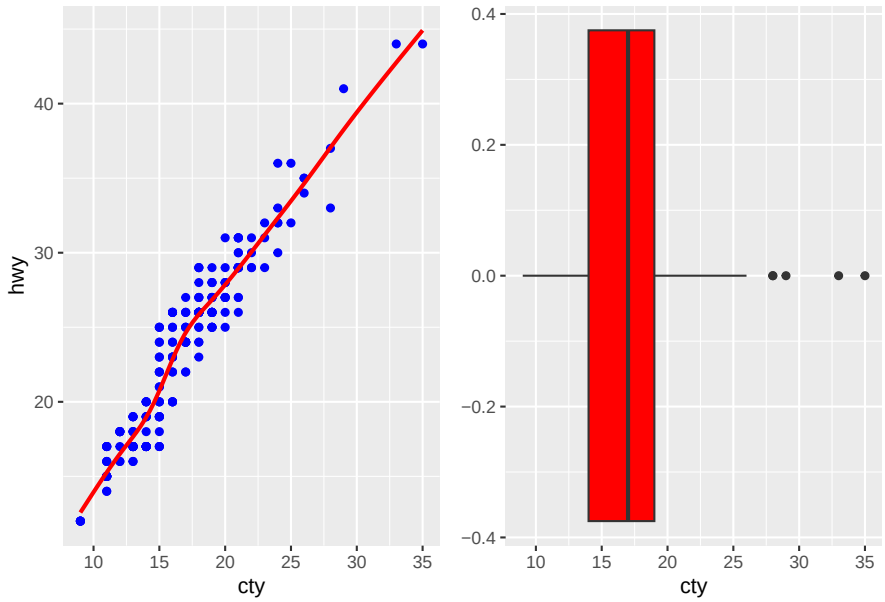
```
# gridExtra
grid.arrange(v1, v2, v3, v4)
```



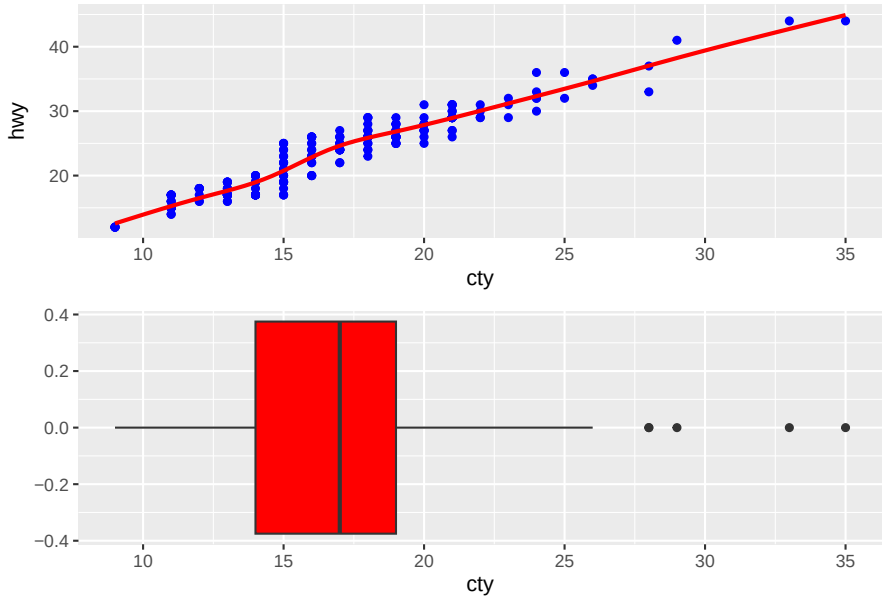

```
# patchwork
v1 + v2
```



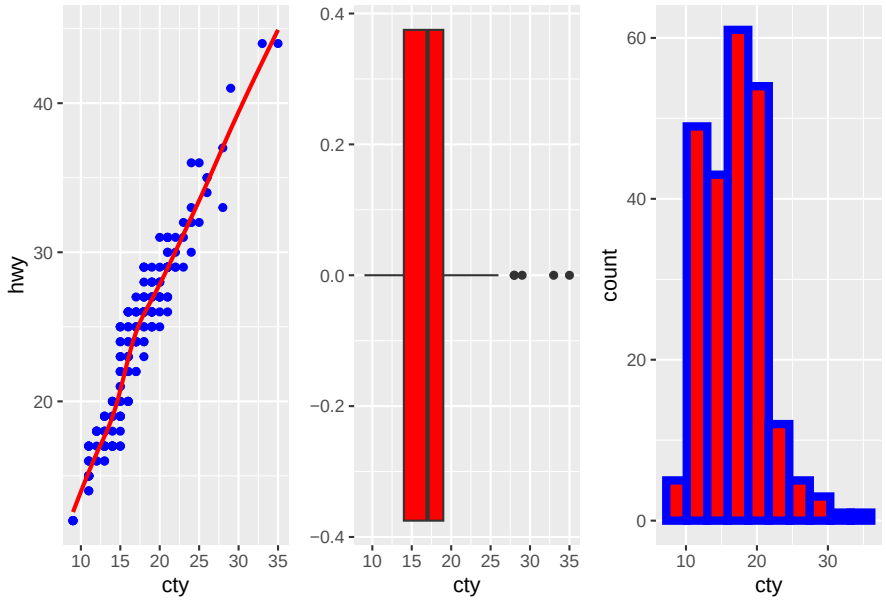
```
v1 | v2
```


hwy
cty
40
30
20
10
15
20
25
30
35
0.4
0.2
0.0
−0.2
−0.4

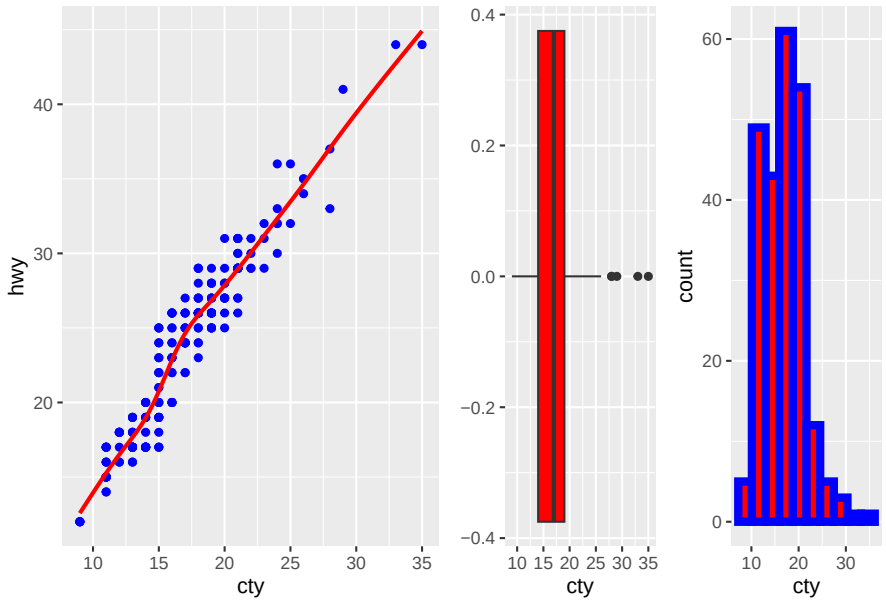
```
v1 / v2
```


hwy
cty
40
30
20
10
15
20
25
30
35
0.4
0.2
0.0
−0.2
−0.4

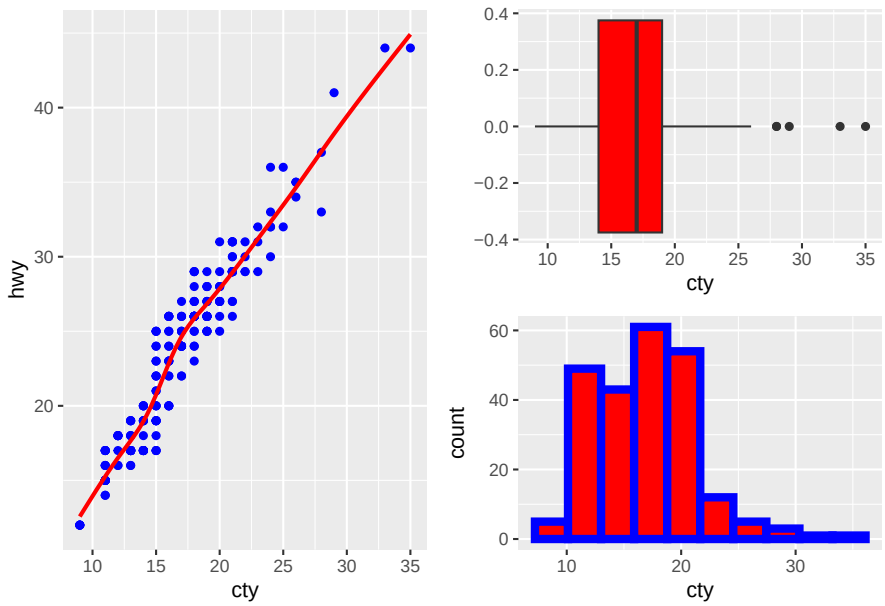
```
v1 + v2 + v3
```



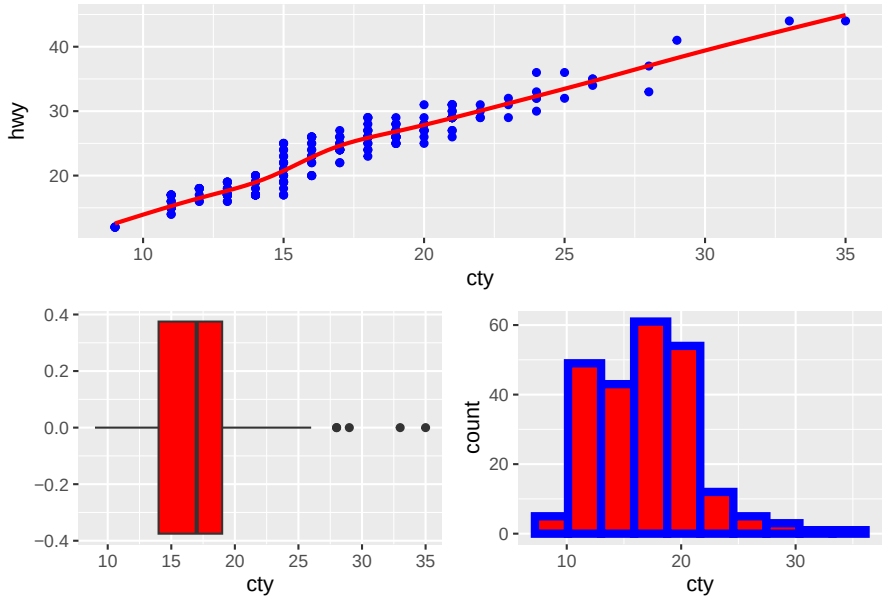

```
v1 + (v2 + v3)
```



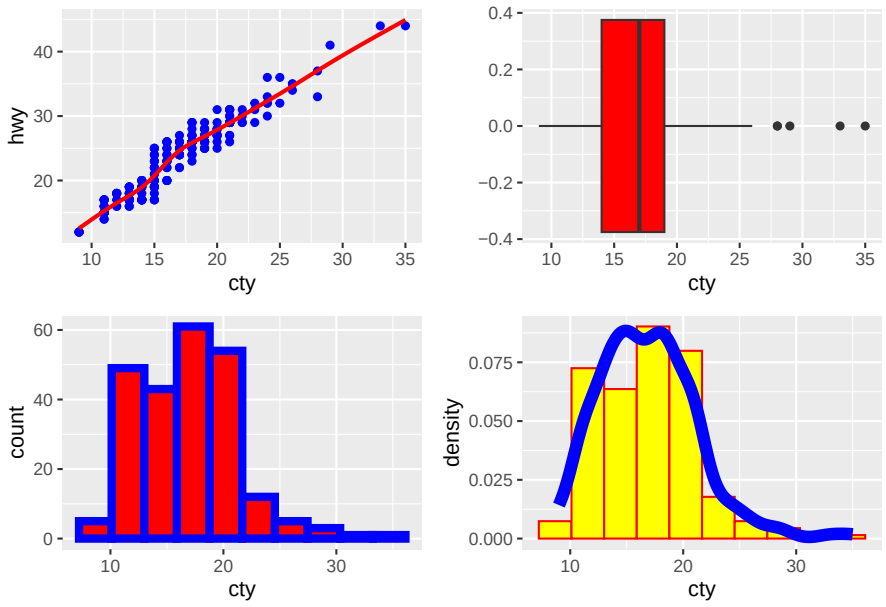
```
v1 | (v2 / v3)
```



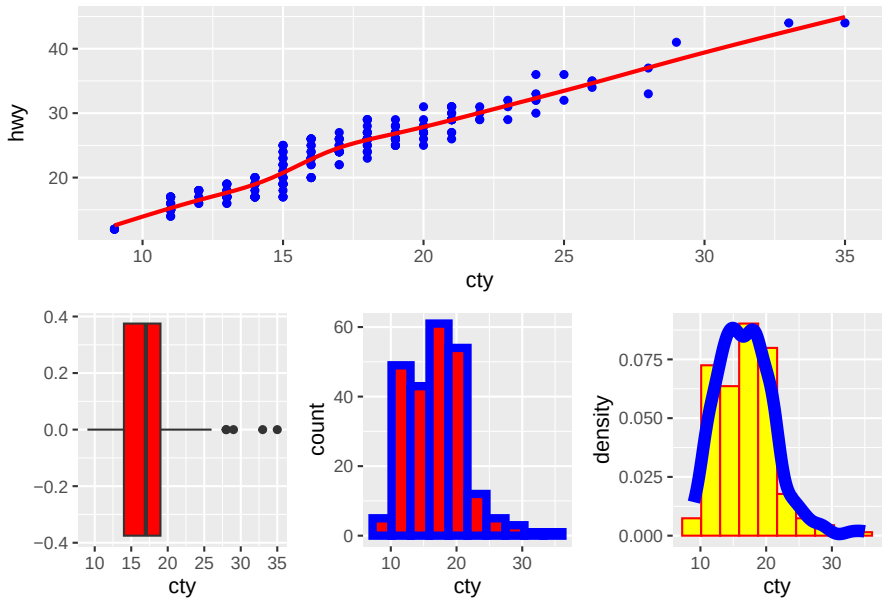

```
v1 / (v2 + v3)
```



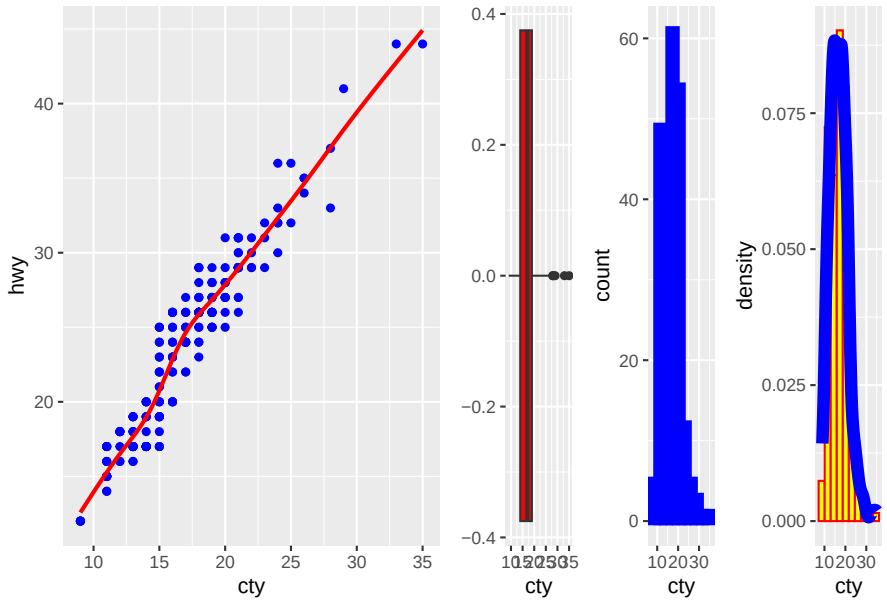
```
v1 + v2 + v3 + v4
```



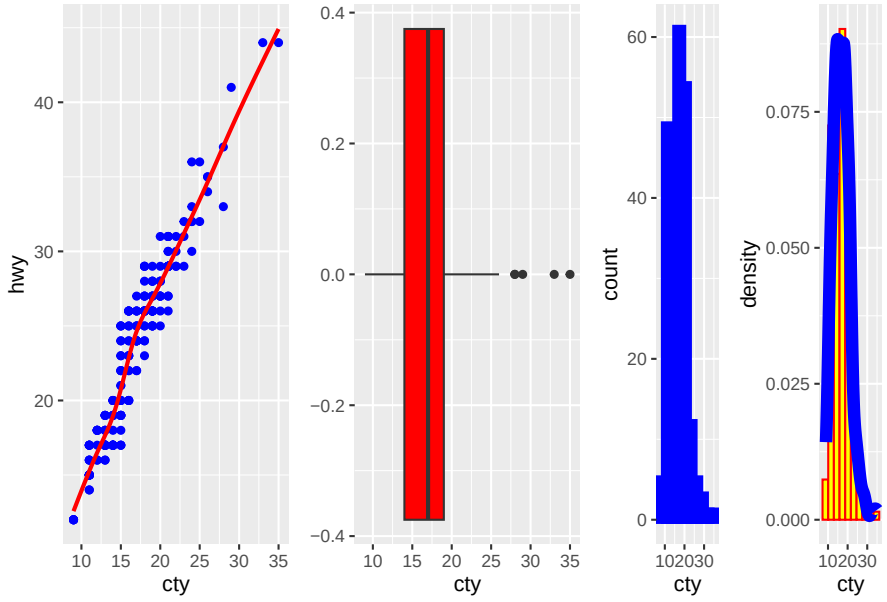

```
v1/(v2+v3+v4)
```



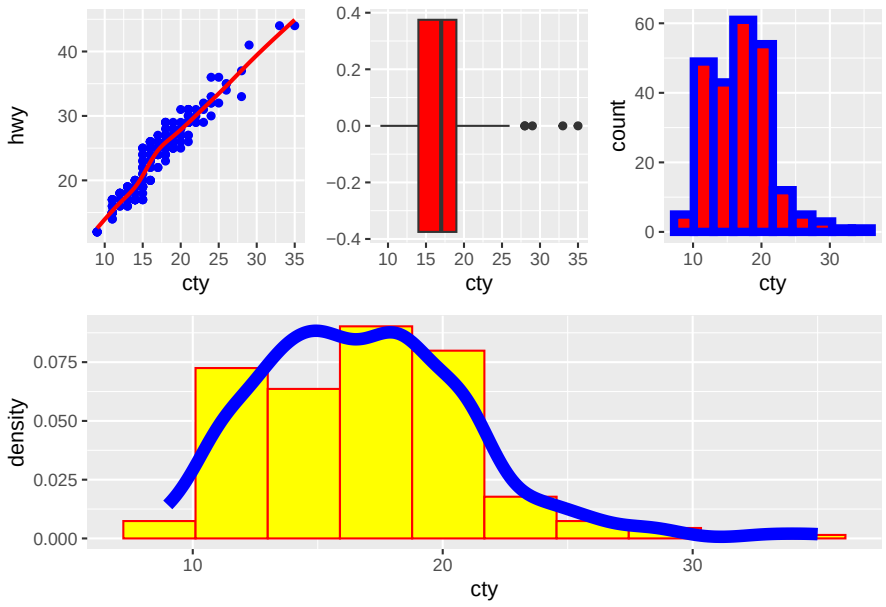
```
v1  + (v2 + v3 + v4)
```




```
v1  + v2 + (v3 + v4)
```

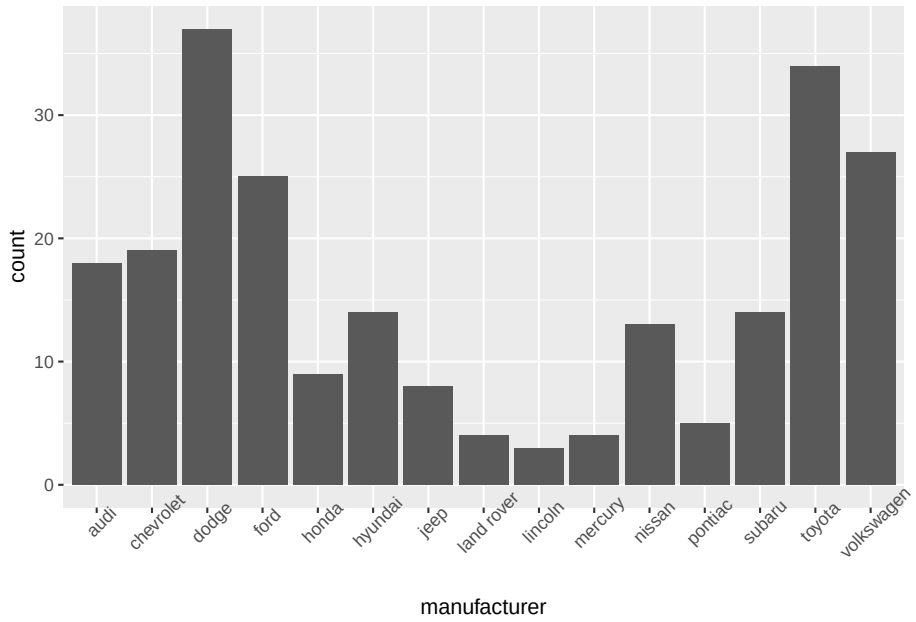


```
(v1 | v2 | v3) / v4
```

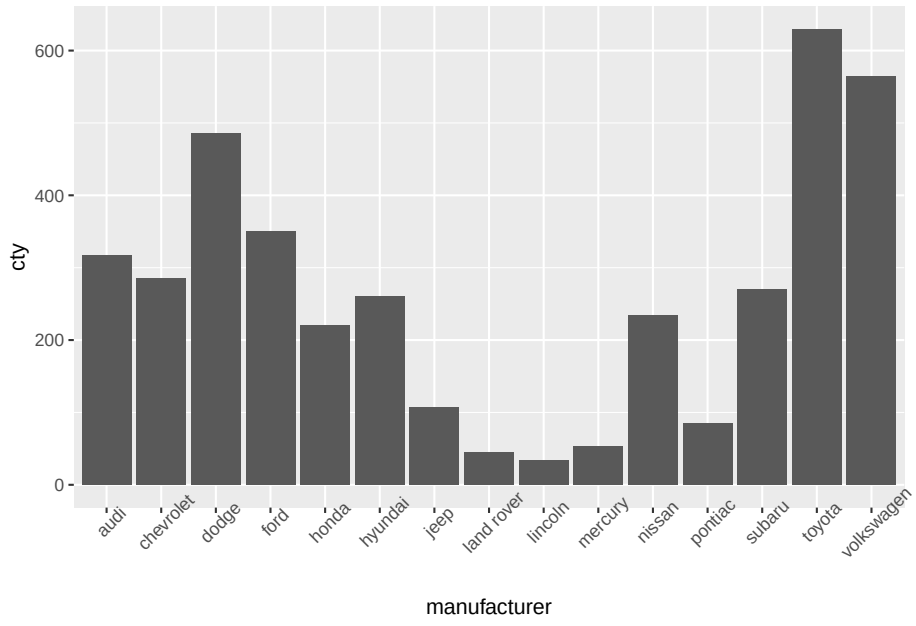


3.6 bar, col, density, density2d

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



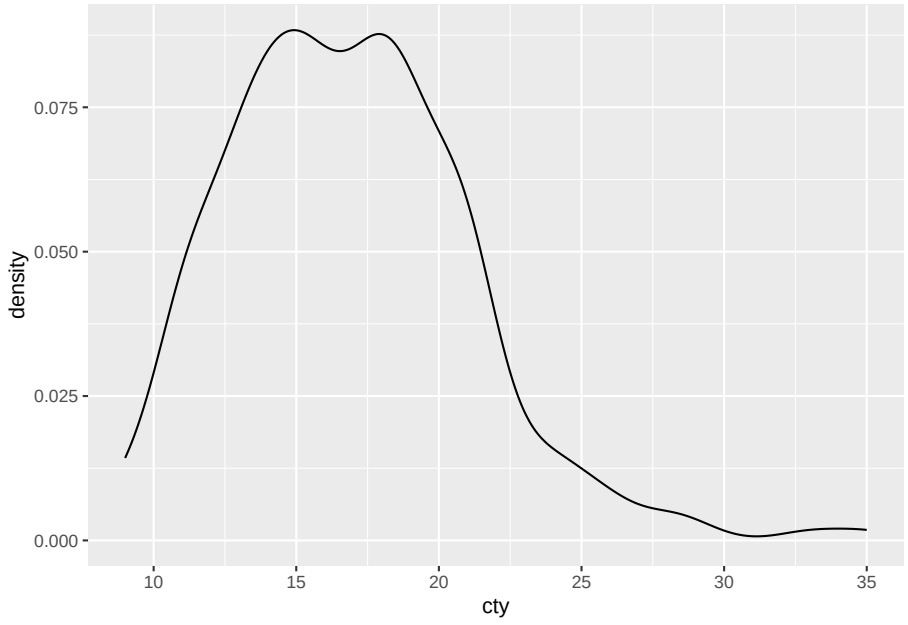

```
# Dúvidas no geom_col
v6 <- ggplot(dados , aes(x = manufacturer, y = cty)) +
  geom_col()+
  theme(axis.text.x = element_text(angle = 45))
v6
```



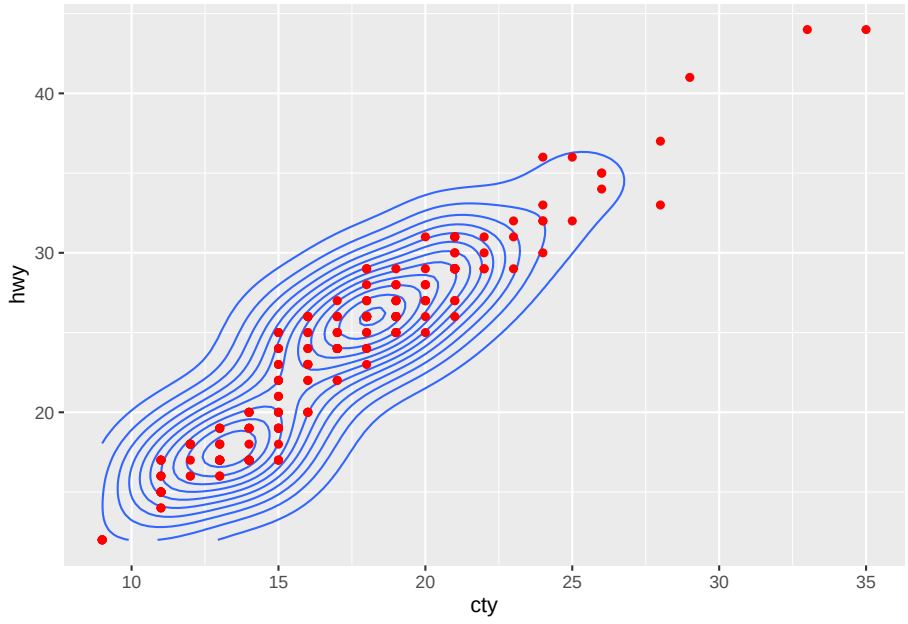

```
dados %>%
  select(manufacturer, cty) %>%
  group_by(manufacturer) %>%
  summarise(soma_total_cty = sum(cty),
            n = n())
```

```
## # 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
```

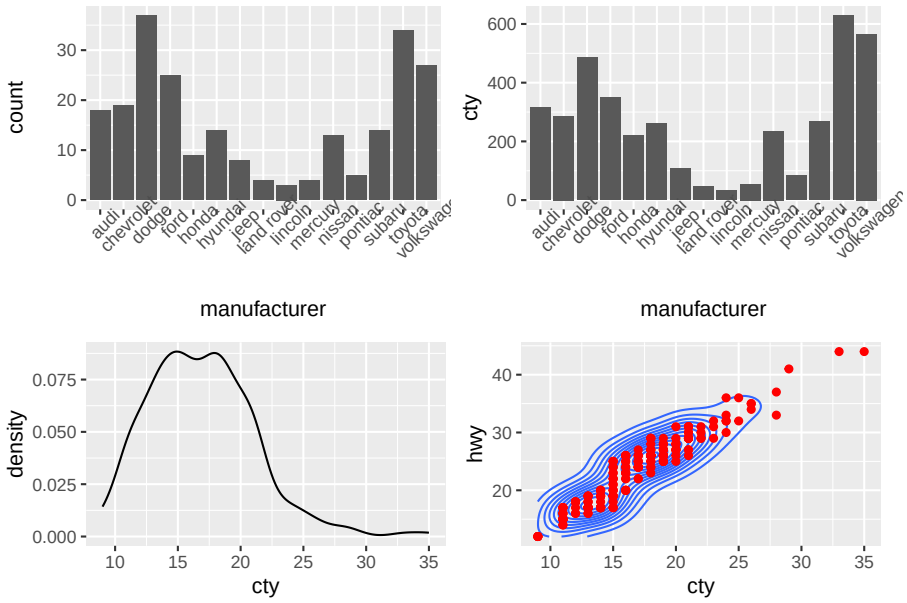
```
# dados %>%
#   filter(manufacturer == "audi") %>%
#   select(cty) %>%
#   sum()
v7 <- ggplot(dados , aes(x = cty)) +
  geom_density()
v7
```




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


hwy
cty
40
30
20
10
15
20
25
30
35

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


count
cty
density
hwy
manufacturer
manufacturer
cty
cty
audi
chevrolet
dodge
ford
honda
hyundai
jeep
land rover
lincoln
mercury
nissan
pontiac
subaru
toyota
volkswagen

```
# Deixar pra depois...
 dados %>%
    select(manufacturer, hwy, year) %>%
    filter(manufacturer == "audi", year == "1999") %>%
    summarise(media = max(hwy))
```

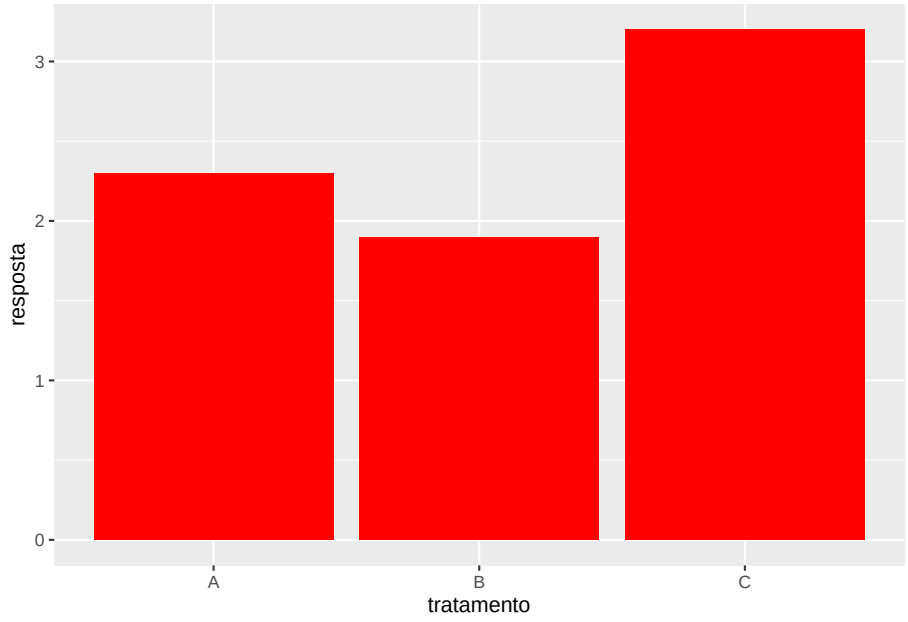
```
## # A tibble: 1 x 1
##   media
##   <int>
## 1    29
```

```
# 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))
```

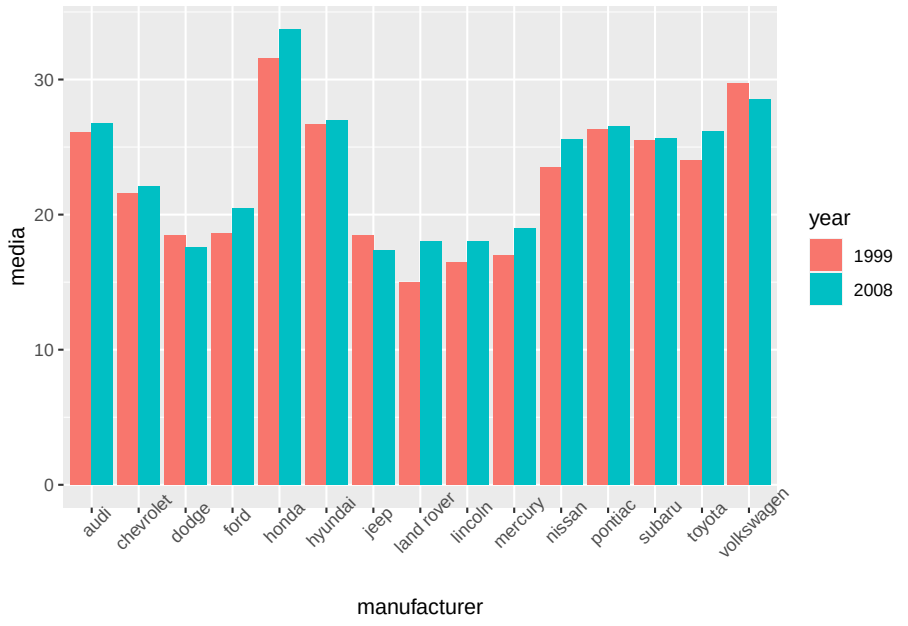
```
# Para pensar

(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")
```

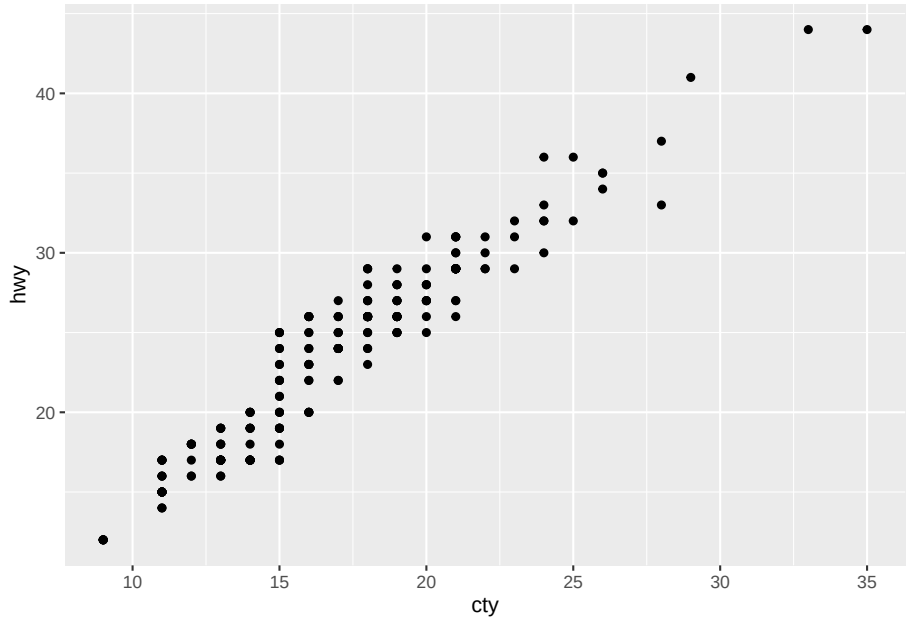



```
# Mais detalhes...
dados %>% select(manufacturer, hwy, year) %>%
  group_by(manufacturer, year) %>%
  summarise(media = mean(hwy), .groups = "drop") %>%
  ggplot(aes(x = manufacturer, y = media, fill = factor(year)))+
  geom_col(position = "dodge")+
  labs(fill = "year") +
  theme(axis.text.x = element_text(angle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

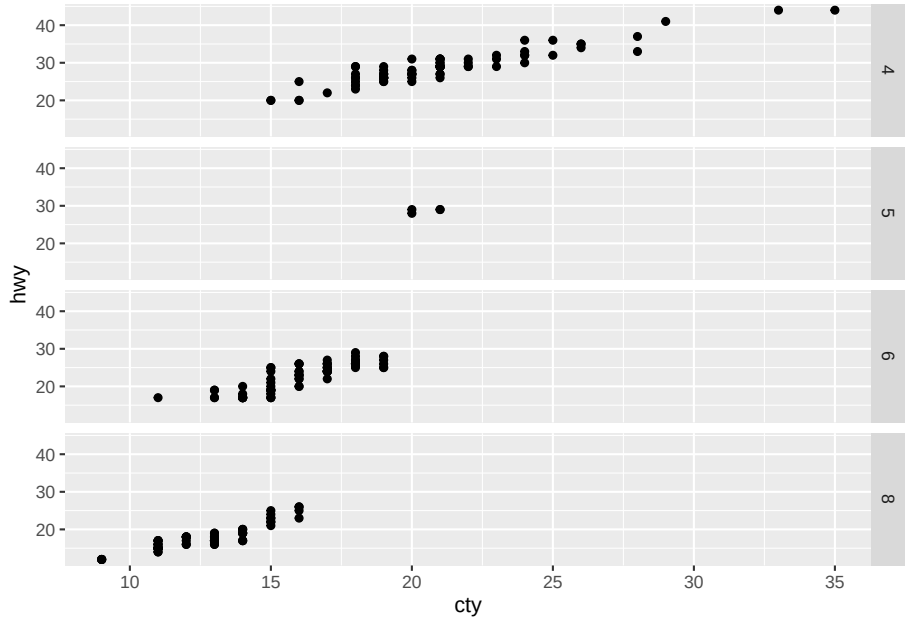


3.7 facet_grid, facet_wrap

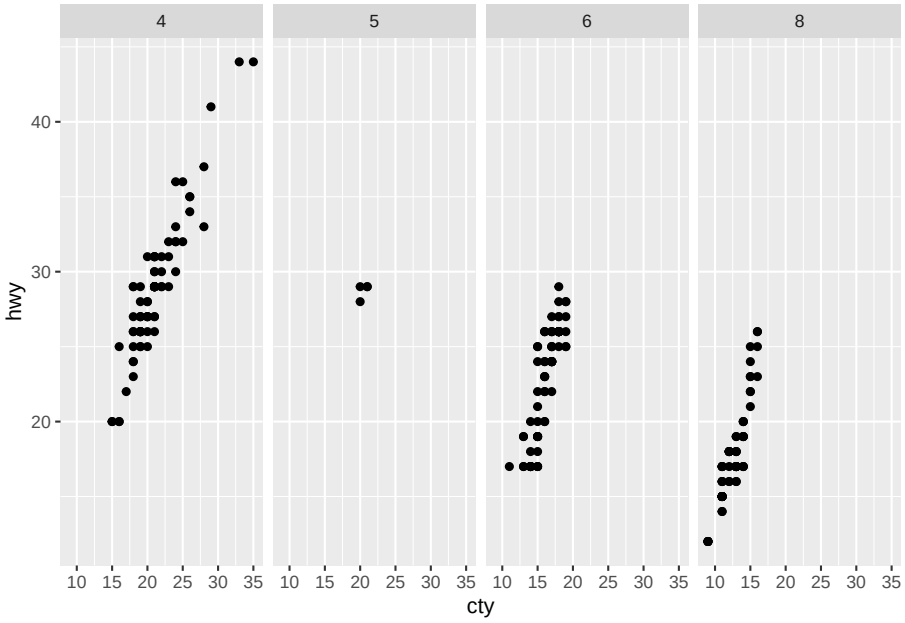
```
p1<- ggplot(dados, aes(x = cty, y = hwy)) +
  geom_point()
p1
```



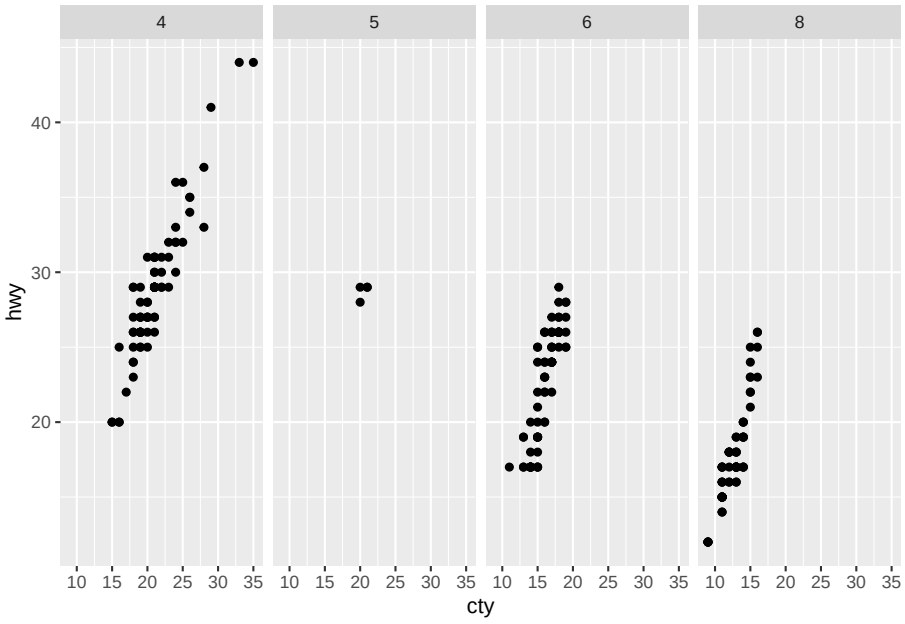

```
p1 + facet_grid(rows = vars(cyl))
```



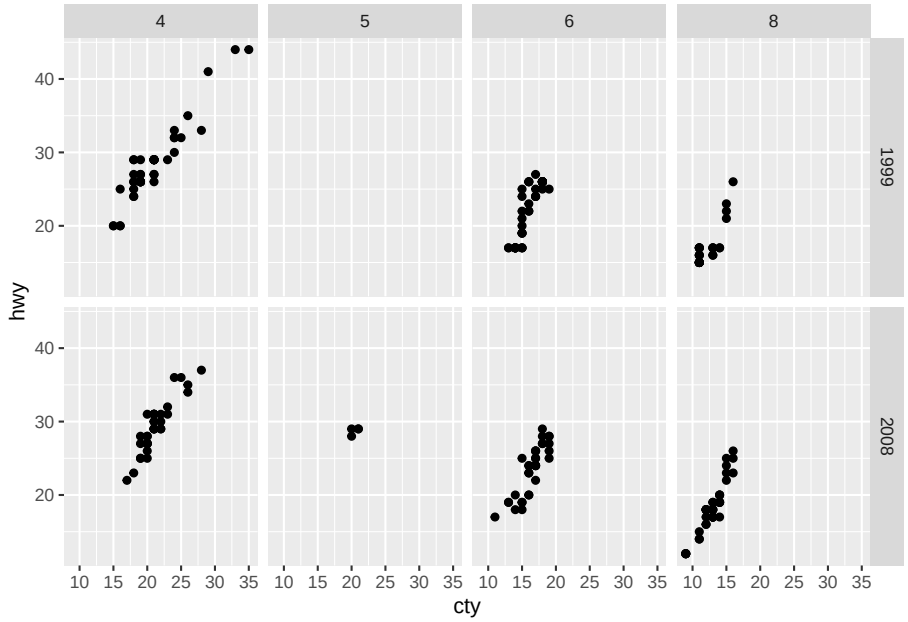
```
p1 + facet_grid(cols = vars(cyl))
```



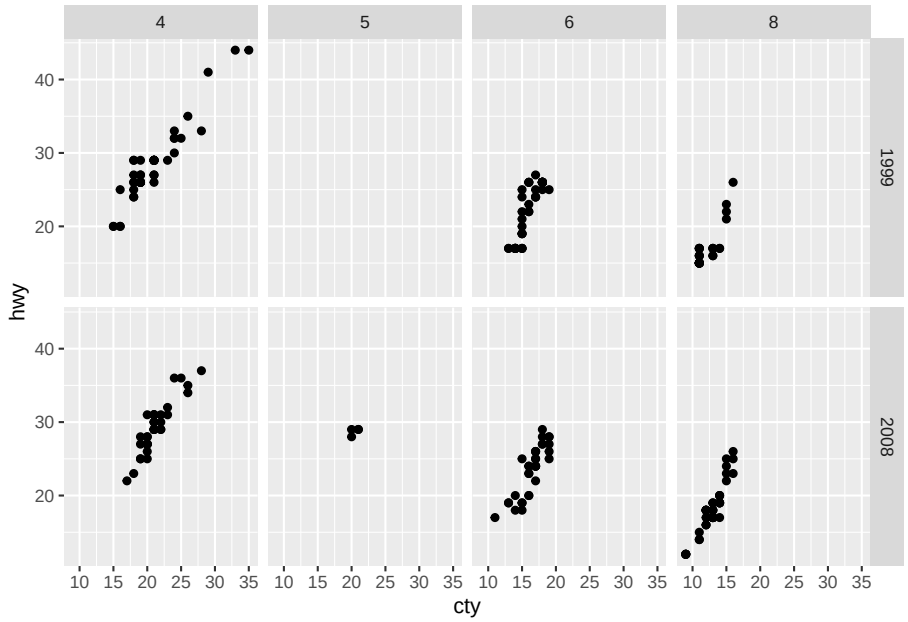

```
p1 + facet_grid(~cyl)
```



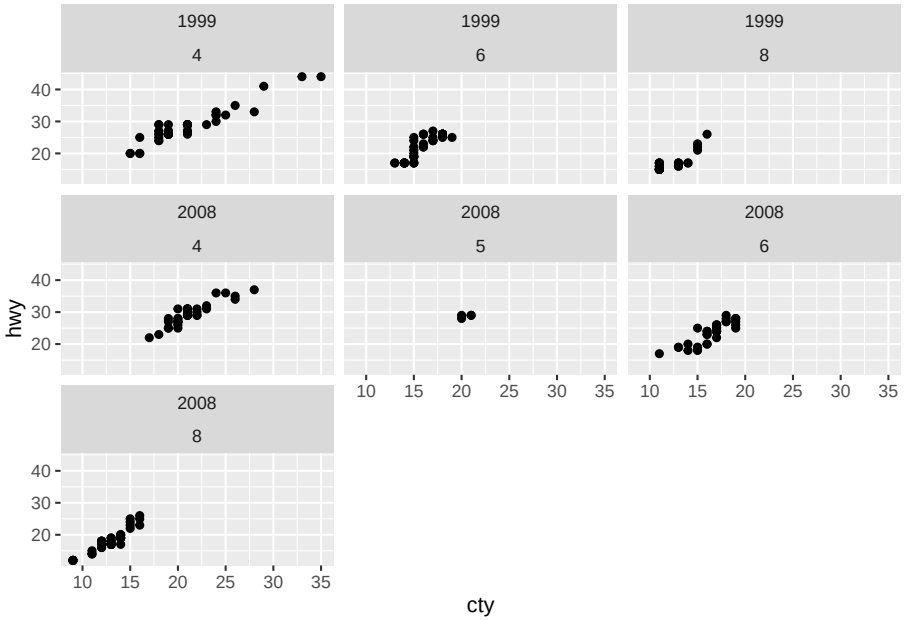
```
p1 + facet_grid(rows = vars(year), cols =vars(cyl))
```



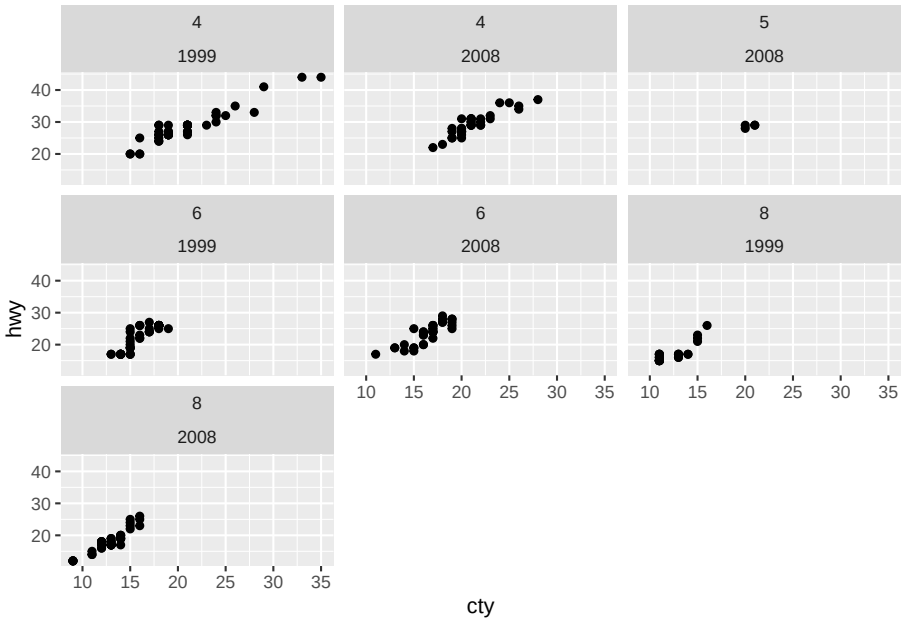

```
p1 + facet_grid(year~cyl)
```



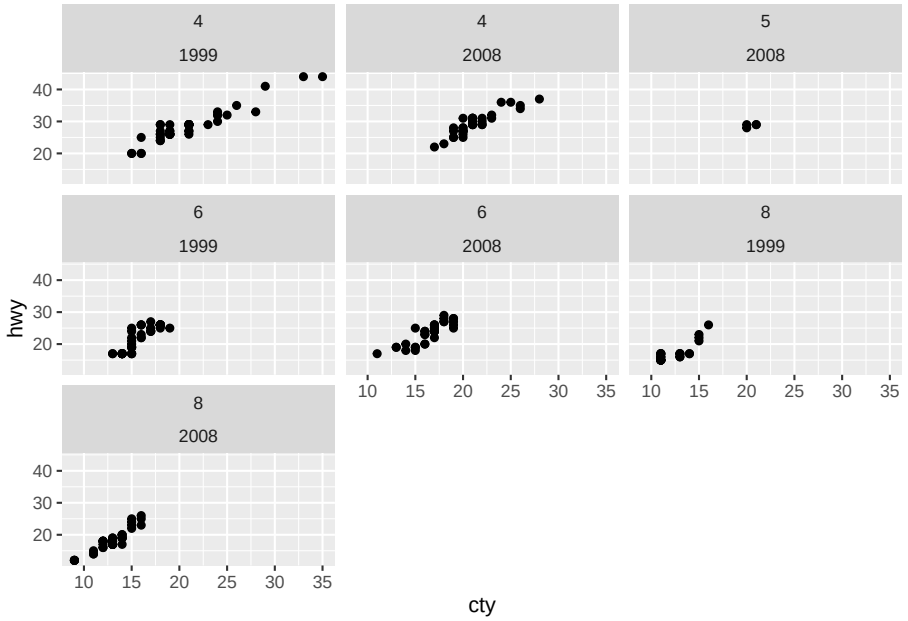
```
p1 + facet_wrap(year ~ cyl)
```



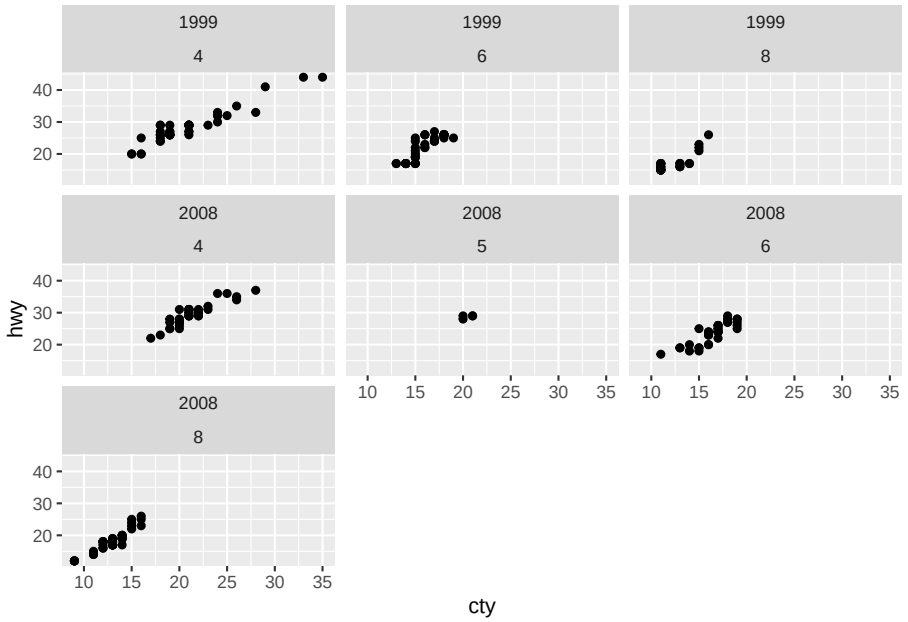

```
p1 + facet_wrap(cyl ~ year)
```



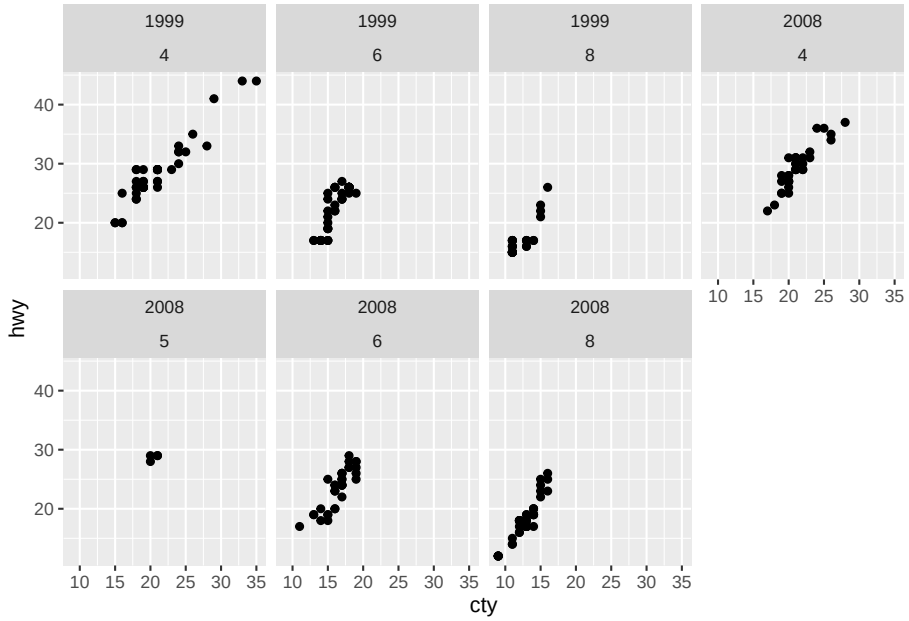
```
p1 + facet_wrap(~cyl + year)
```



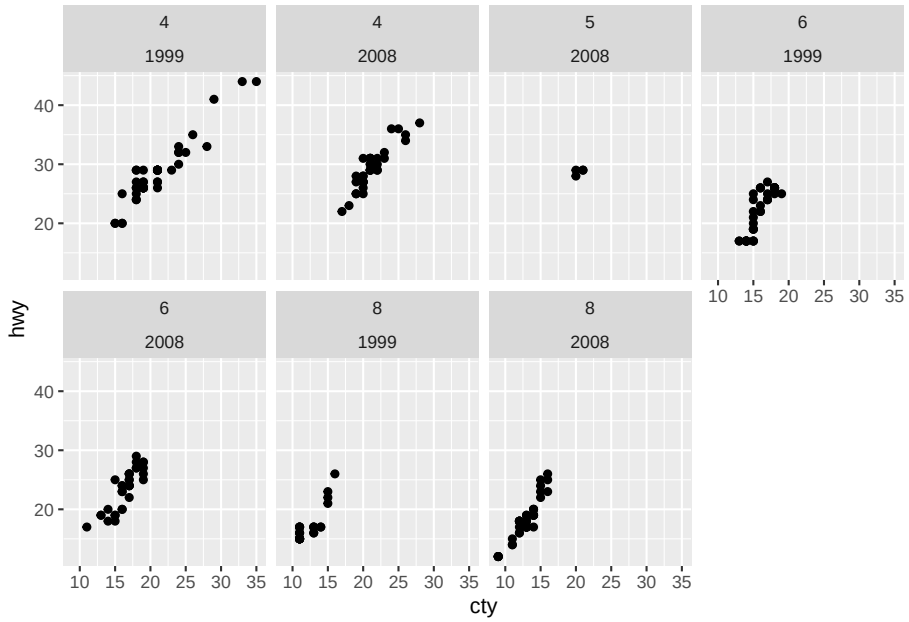

```
p1 + facet_wrap(~year + cyl)
```



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

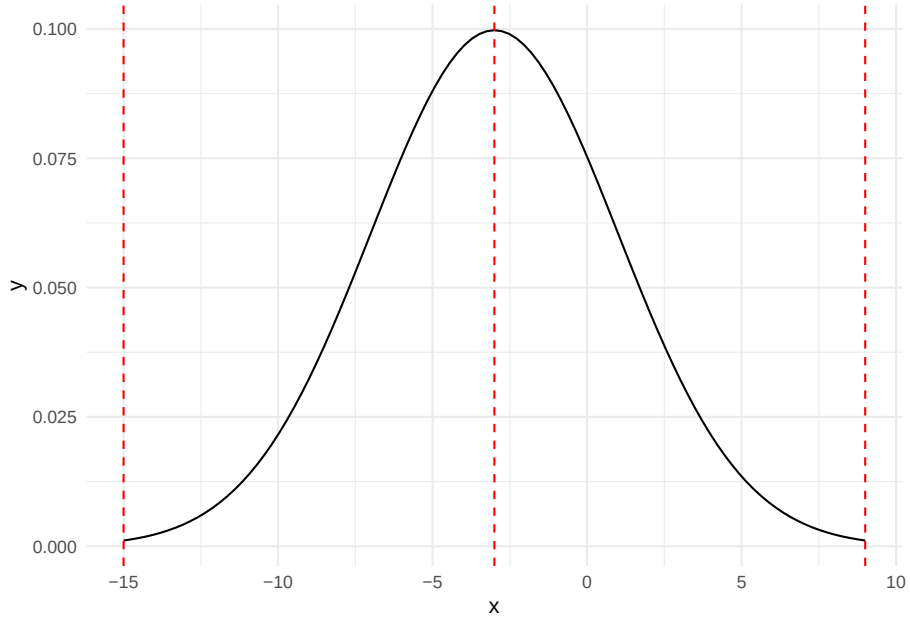



```
p1 + facet_wrap(cyl ~ year, ncol = 4)
```



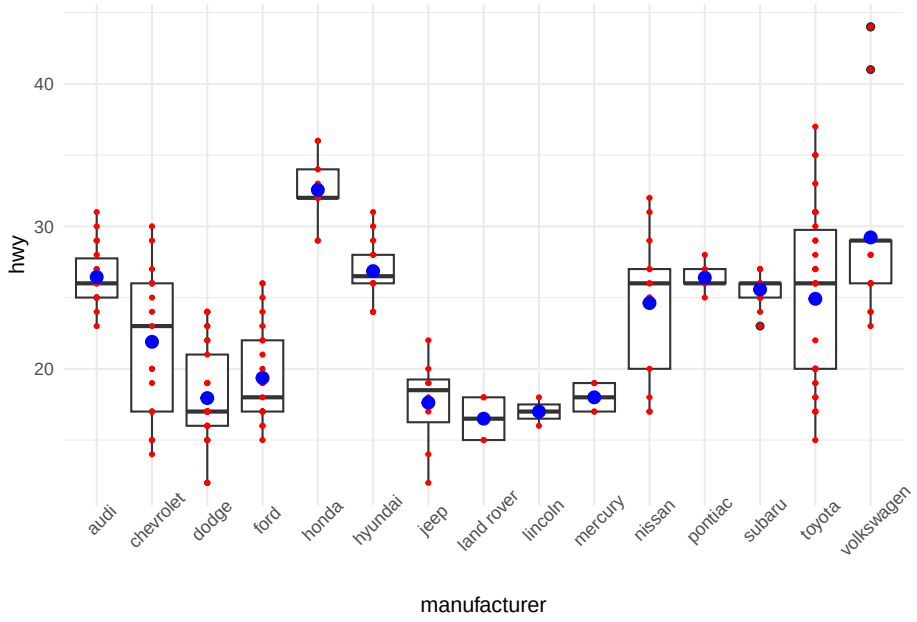
3.8 stat_function

```
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()
```



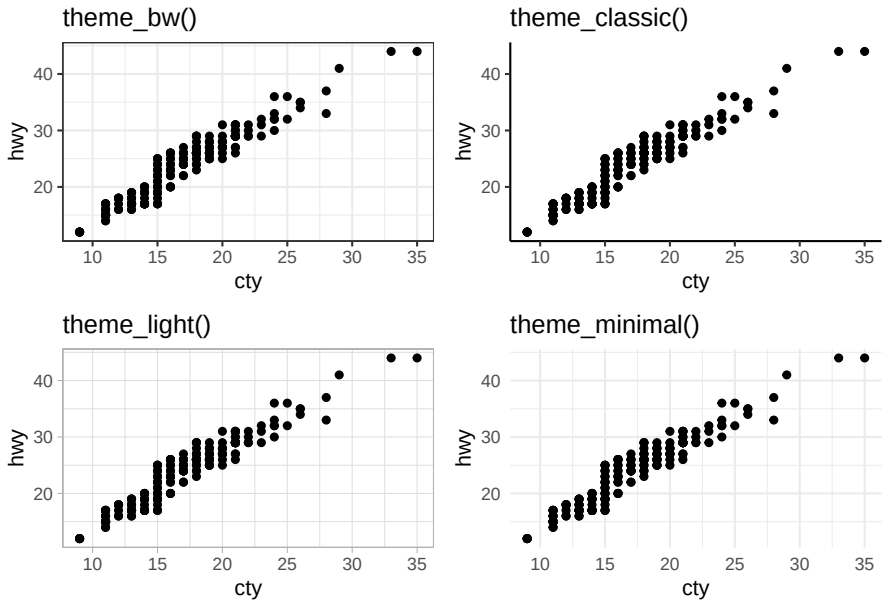
3.9 stat_summary

```
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))
```



3.10 theme_*()

```
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
```

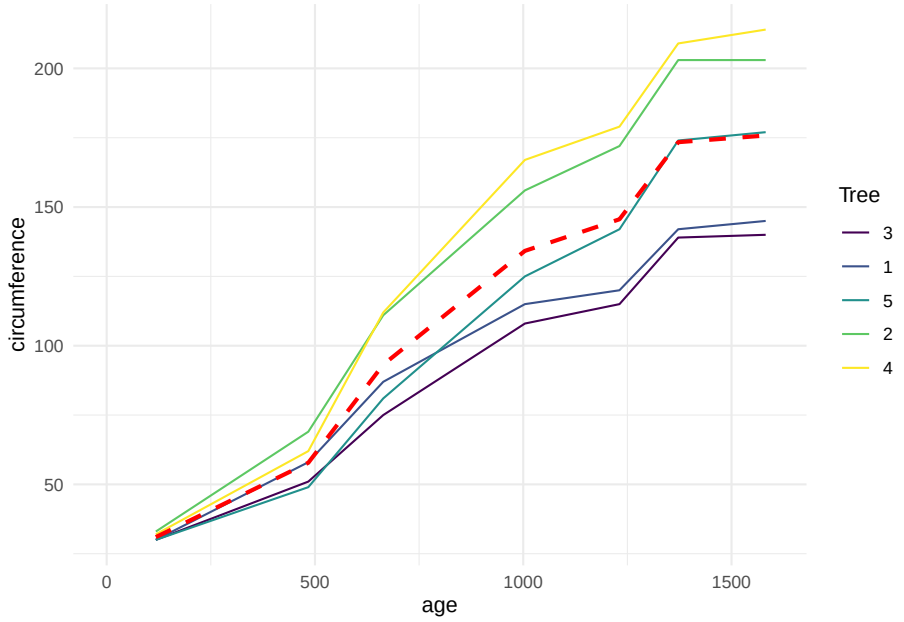


3.11 Gráfico de perfis (Spaguetti plot)

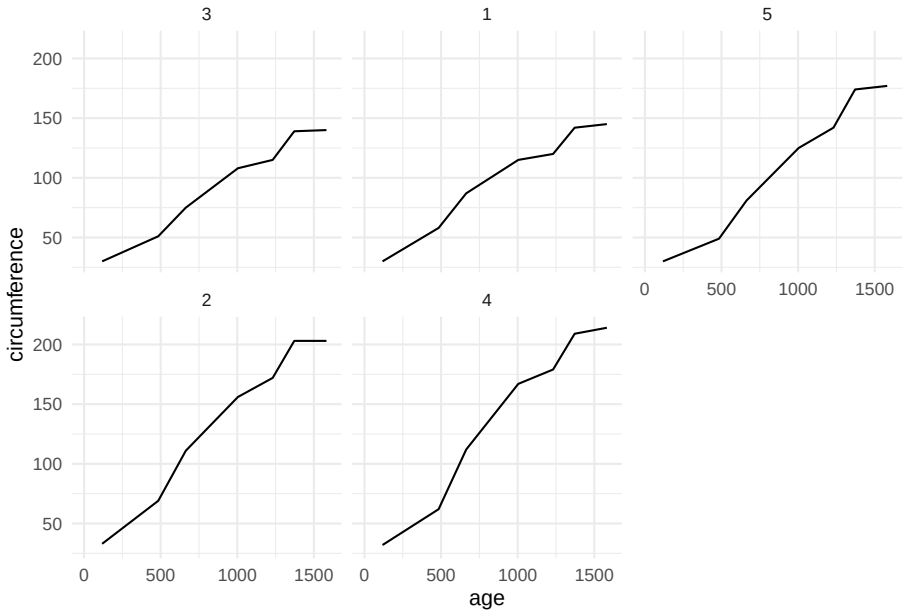
```
glimpse(Orange)
```

```
## 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

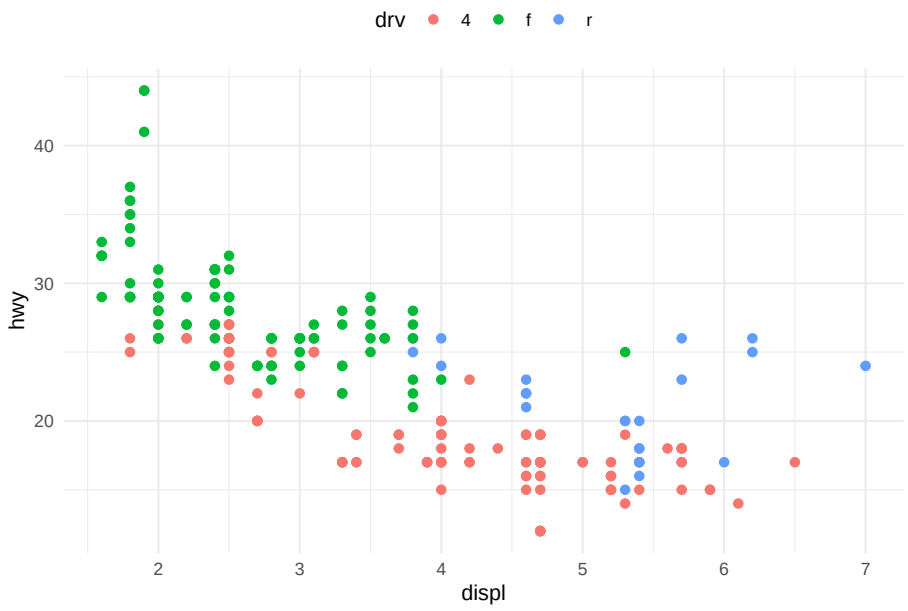
esquisse

esquisse + shiny

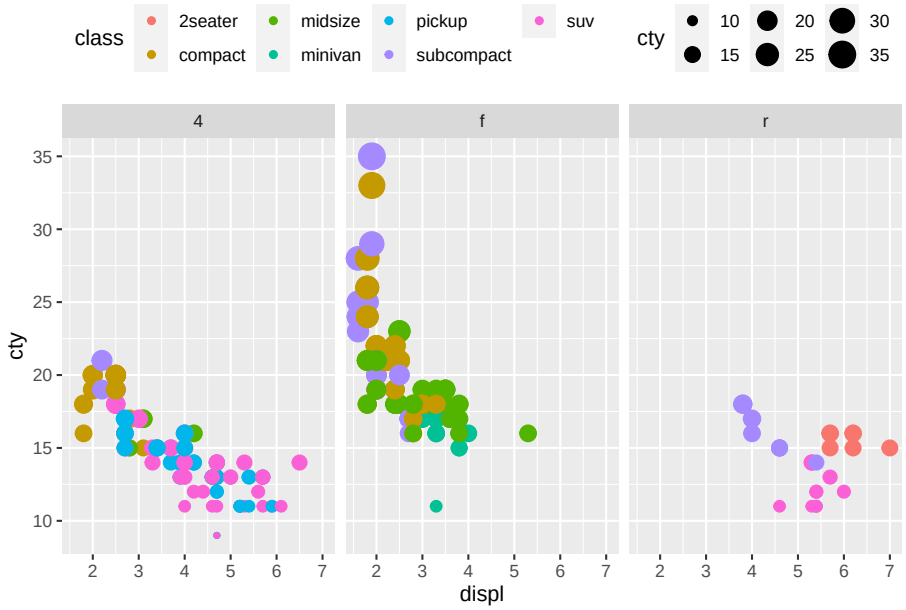
```
esquisser(dados)
```

3.14 Exemplo esquisse

```
ggplot(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")
```




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


class
2seater
compact
midsize
minivan
pickup
subcompact
suv
cty
10
15
20
25
30
35
4
f
r
cty
displ

Capítulo 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]  0.9606835  0.2558870  1.3668572  1.1732371  0.2350102  2.6521804
##  [7]  0.8214040  1.9771913 -0.6155543 -0.9988018
##
## [[2]]
##  [1] 0.9377754 0.9214704 2.8330264 3.0352480 0.5238653 2.5443251 2.2924893
##  [8] 2.7829582 2.4419409 0.3743757
##
## [[3]]
##  [1] 1.879637 3.818392 2.890525 3.872515 3.140802 2.768388 2.584703 2.922238
##  [9] 2.616681 4.430572
##
## [[4]]
##  [1] 3.567559 4.349109 3.615555 5.362684 2.652931 5.404148 3.194384 2.184058
##  [9] 3.673871 5.500680
##
## [[5]]
##  [1] 5.036916 4.557861 3.973224 6.193965 4.280406 5.230274 5.726284 5.856957
##  [9] 5.082852 3.858142
##
## [[6]]
##  [1] 4.656112 7.127457 7.521381 6.674265 6.095799 6.279487 6.779528 6.858636
##  [9] 6.285432 8.188434
##
## [[7]]
##  [1] 7.006924 7.218360 9.174065 7.025654 5.990612 7.551149 8.043741 7.910483
##  [9] 8.426142 6.625891
##
## [[8]]
##  [1]  8.002099  8.256513  6.616554  9.059241  7.288628  7.784038 10.333387
##  [8]  7.588237  8.447337  7.238239
##
## [[9]]
##  [1]  7.935737  8.246600 10.050744  7.661163  8.686803  6.817224 10.071338
##  [8]  9.372334  7.385286  9.223673
##
## [[10]]
##  [1]  9.330173  9.315890 11.722364 10.076962  8.569037 10.816171  9.890369
##  [8] 10.161575 12.456419  9.847980
```

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

```
## [[1]]
##  [1]  2.4355148  1.3161904 -0.3844751  0.2743863  0.6748907  1.7813773
##  [7]  2.4774534  0.9030152  2.0938987  0.7016563
##
## [[2]]
##  [1] 4.134756 2.155219 2.612060 2.072941 2.212364 1.161664 3.075493 2.425679
##  [9] 2.495402 1.971725
##
## [[3]]
##  [1] 3.486822 2.451827 1.297666 2.291035 4.398815 3.260149 3.528562 1.261258
##  [9] 4.190041 4.215717
##
## [[4]]
##  [1] 5.009319 4.773983 4.106116 3.835846 3.576150 2.326984 5.184813 3.810730
##  [9] 2.117968 3.568571
##
## [[5]]
##  [1] 3.414407 4.265428 5.562993 5.277723 5.809059 4.492726 6.804638 3.683302
##  [9] 6.856712 4.673499
##
## [[6]]
##  [1] 6.501189 4.649399 3.941000 7.249335 6.460261 7.380625 6.432399 7.486122
##  [9] 5.982924 7.892294
##
## [[7]]
##  [1] 5.963699 7.444964 6.695306 6.825283 7.318234 4.847514 7.119791 7.097203
##  [9] 7.174268 6.752557
##
## [[8]]
##  [1] 7.624474 7.737118 7.432223 8.107198 6.294585 7.416914 8.153201 8.554924
##  [9] 7.239460 7.551029
##
## [[9]]
##  [1]  9.714895  8.727867 10.330333 10.059361  7.865472  8.657559 11.955671
##  [8]  8.627070  9.512751  9.812425
##
## [[10]]
##  [1]  9.475433 10.877835  9.728579 11.468436 11.286864  9.036773  9.359317
##  [8]  8.569081 11.450863  9.169504
```

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

```
## [[1]]
##  [1]  3.0076519  1.3518697  0.7273344  0.3630838  0.5742323  0.1116815
##  [7] -0.3309606  2.6579127 -0.6714084 -0.5835447
##
## [[2]]
##  [1] 1.8177842 3.0601065 1.4946339 3.3512698 4.0715896 1.9266118 0.1991496
##  [8] 1.2843011 3.1614962 2.6488691
##
## [[3]]
##  [1] 2.0150944 3.3520639 4.3315523 0.8447573 3.3162547 4.2627852 2.7075133
##  [8] 2.9769775 2.1460154 1.7398364
##
## [[4]]
##  [1] 2.581916 4.808275 3.368369 4.329507 3.458508 4.789342 2.740603 3.141206
##  [9] 3.398164 5.694223
##
## [[5]]
##  [1] 7.134640 4.254732 5.143223 4.848392 6.329070 4.984556 6.355305 4.787746
##  [9] 4.854475 3.621560
##
## [[6]]
##  [1] 6.668563 6.849455 5.349847 6.371768 5.304878 8.179186 5.507334 4.445319
##  [9] 7.270777 7.253198
##
## [[7]]
##  [1] 6.663330 9.986253 6.115810 6.244033 5.867123 7.941277 7.945039 8.765729
##  [9] 7.763603 7.744577
##
## [[8]]
##  [1] 6.909167 7.448366 5.721610 8.387959 7.137138 7.407341 9.087489 8.648508
##  [9] 6.143680 8.053039
##
## [[9]]
##  [1] 5.729458 9.210299 7.224728 9.911272 8.154753 9.608185 8.063629 7.206674
##  [9] 8.543258 9.196022
##
## [[10]]
##  [1] 10.242230  9.523368  9.630883  9.686558  9.136119 11.507728  8.930809
##  [8]  9.987174 10.816272  9.131987
```

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

```
##  [1] 0.6709021 2.2725584 3.0476521 3.9502926 4.3599626 6.3029332 7.4379788
##  [8] 8.0346655 8.6689626 9.8911214
```

```
# 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", Eliott = "pancakes", Karina = "choco
favorite_desserts
```

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

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

```
##                  Sophia                  Eliott                  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
l2 %>%
  map(list("num", 3))
```

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

```
l2 %>%
  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")
```

Capítulo 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

Capítulo 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:

```
#ls("package:tidyr")
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 useful later on when we start working with list-columns.

```
data(Orange)

Orange <- as_tibble(Orange)
Orange
```

```
## # A tibble: 35 x 3
```

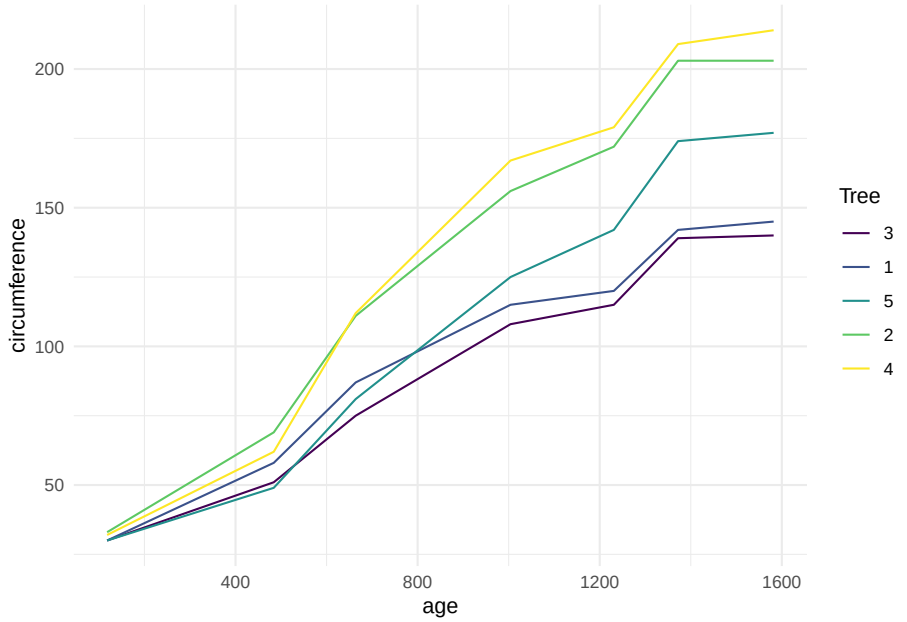
```
##    Tree    age circumference
##    <ord> <dbl>         <dbl>
##  1 1       118            30
##  2 1       484            58
##  3 1       664            87
##  4 1      1004           115
##  5 1      1231           120
##  6 1      1372           142
##  7 1      1582           145
##  8 2       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 `nest`ing 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.

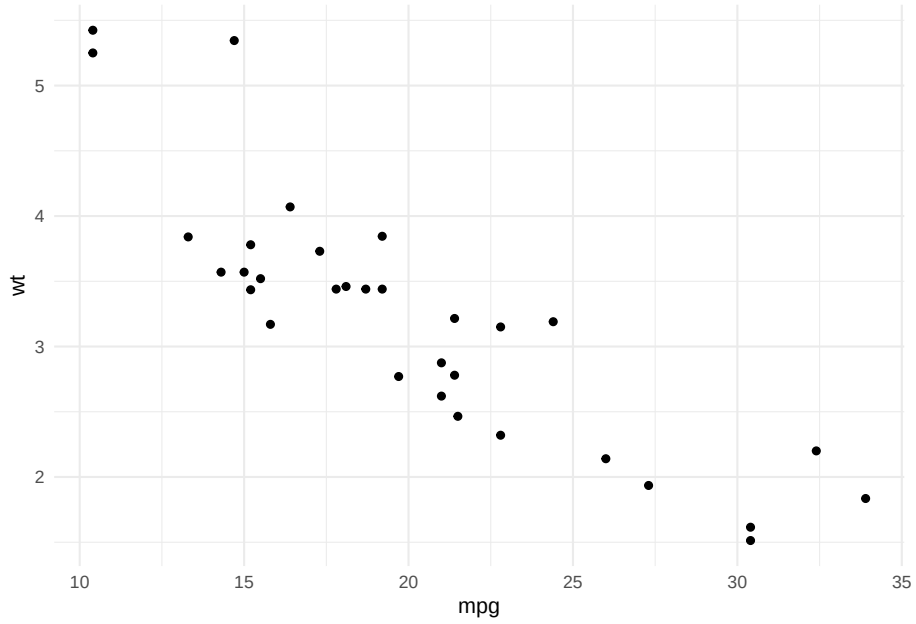
6.1 Tidy bootstrapping

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()
```



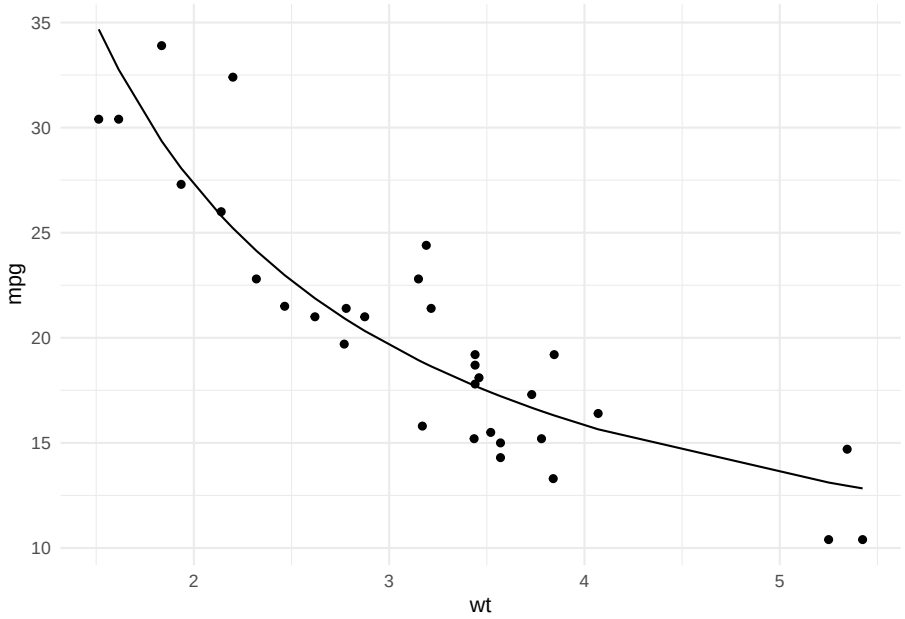
We might use the method of nonlinear least squares (via the `nls` function) to fit a model.

```
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
```

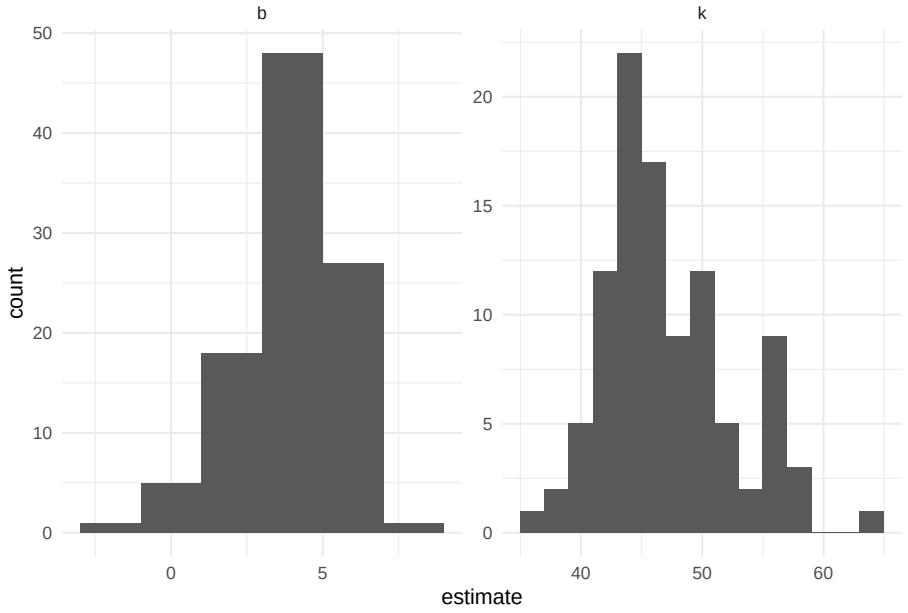
We can then calculate confidence intervals (using what is called the percentile method):

```
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")
```



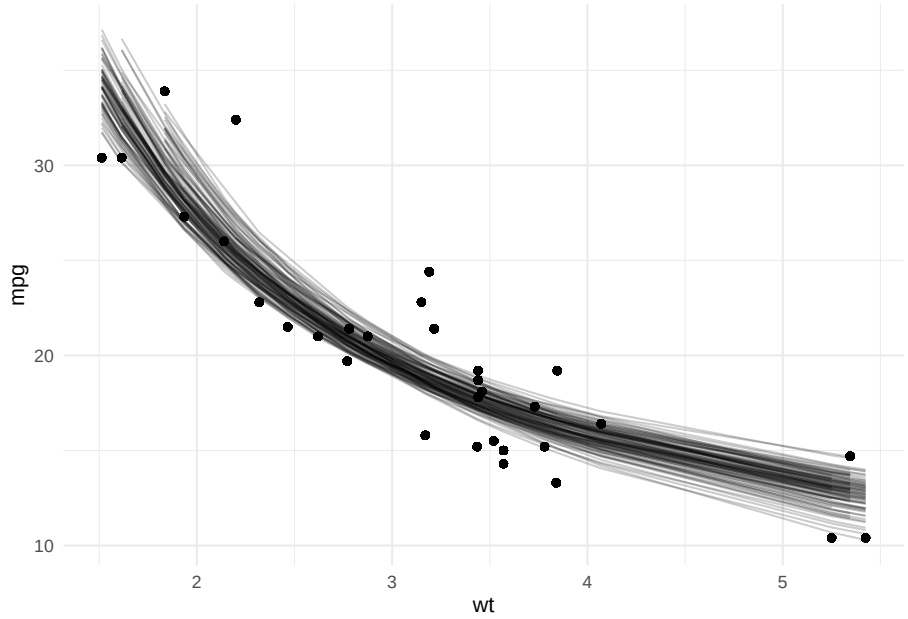
Or we can use `augment` to visualize the uncertainty in the curve:

```
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
```

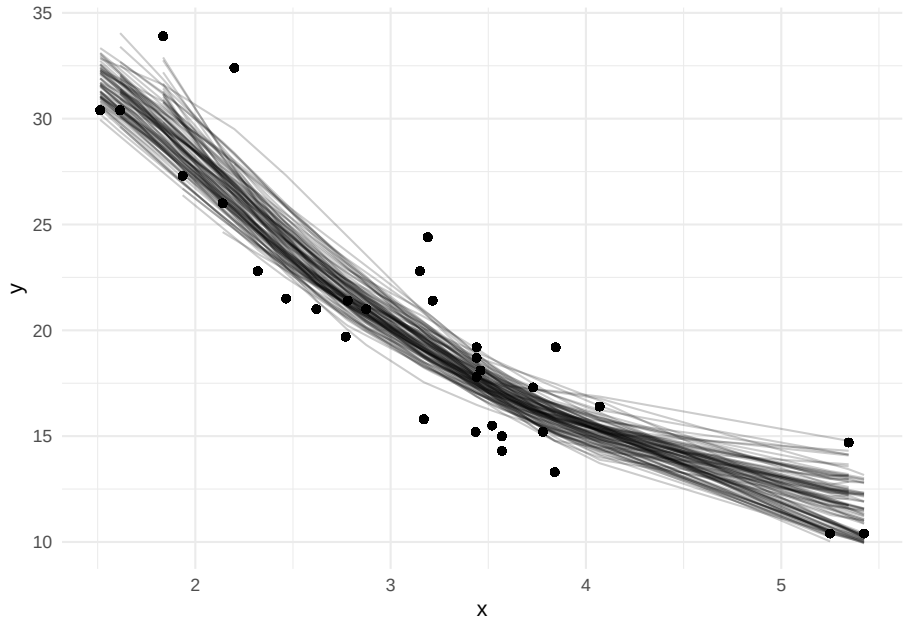
```
ggplot(boot_aug, aes(wt, mpg)) +
    geom_point() +
    geom_line(aes(y = .fitted, group = id), alpha=.2)
```



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 works 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 Mais detalhes (glance)

6.2.1 lm()

```
fit <- lm(Volume ~ Girth + Height, trees)
tidy(fit)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  -58.0       8.64      -6.71 2.75e- 7
## 2 Girth          4.71      0.264     17.8  8.22e-17
## 3 Height         0.339     0.130      2.61 1.45e- 2
```

```
glance(fit)
```

```
## # A tibble: 1 x 12
```

```
##   r.squared adj.r.squa~1 sigma stati~2  p.value    df logLik   AIC   BIC devia~3
##       <dbl>        <dbl> <dbl>   <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.948        0.944  3.88    255. 1.07e-18     2  -84.5  177.  183.    422.
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

```
augment(fit, data = trees)
```

```
## # A tibble: 31 x 9
##    Girth Height Volume .fitted .resid   .hat .sigma   .cooksd .std.resid
##    <dbl>  <dbl>  <dbl>   <dbl>  <dbl>  <dbl>  <dbl>     <dbl>      <dbl>
##  1   8.3     70   10.3    4.84  5.46  0.116    3.79 0.0978         1.50
##  2   8.6     65   10.3    4.55  5.75  0.147    3.77 0.148          1.60
##  3   8.8     63   10.2    4.82  5.38  0.177    3.78 0.167          1.53
##  4  10.5     72   16.4   15.9   0.526 0.0592   3.95 0.000409       0.140
##  5  10.7     81   18.8   19.9  -1.07  0.121    3.95 0.00394       -0.294
##  6  10.8     83   19.7   21.0  -1.32  0.156    3.94 0.00840       -0.370
##  7  11       66   15.6   16.2  -0.593 0.115    3.95 0.00114       -0.162
##  8  11       75   18.2   19.2  -1.05  0.0515   3.95 0.00138       -0.277
##  9  11.1     80   22.6   21.4   1.19  0.0920   3.95 0.00348        0.321
## 10  11.2     75   19.9   20.2  -0.288 0.0480   3.95 0.0000968     -0.0759
## # ... with 21 more rows
```

```
methods("tidy")
```

```
##   [1] tidy.aareg*                  tidy.acf*
##   [3] tidy.anova*                  tidy.aov*
##   [5] tidy.aovlist*                tidy.Arima*
##   [7] tidy.betamfx*                tidy.betareg*
##   [9] tidy.biglm*                  tidy.binDesign*
##  [11] tidy.binWidth*               tidy.boot*
##  [13] tidy.btergm*                 tidy.cch*
##  [15] tidy.character*              tidy.cld*
##  [17] tidy.clm*                    tidy.clmm*
##  [19] tidy.coeftest*               tidy.confint.glht*
##  [21] tidy.confusionMatrix*        tidy.coxph*
##  [23] tidy.crr*                    tidy.cv.glmnet*
##  [25] tidy.data.frame*             tidy.default*
##  [27] tidy.density*                tidy.dist*
##  [29] tidy.drc*                    tidy.durbinWatsonTest*
##  [31] tidy.emmGrid*                tidy.epi.2by2*
##  [33] tidy.ergm*                   tidy.factanal*
##  [35] tidy.felm*                   tidy.fitdistr*
##  [37] tidy.fixest*                 tidy.ftable*
```

```
##  [39] tidy.gam*                      tidy.Gam*
##  [41] tidy.garch*                    tidy.geeglm*
##  [43] tidy.glht*                     tidy.glm*
##  [45] tidy.glmnet*                   tidy.glmrob*
##  [47] tidy.glmRob*                   tidy.gmm*
##  [49] tidy.htest*                    tidy.ivreg*
##  [51] tidy.kappa*                    tidy.kde*
##  [53] tidy.Kendall*                  tidy.kmeans*
##  [55] tidy.lavaan*                   tidy.leveneTest*
##  [57] tidy.Line*                     tidy.Lines*
##  [59] tidy.list*                     tidy.lm*
##  [61] tidy.lm.beta*                  tidy.lmodel2*
##  [63] tidy.lmrob*                    tidy.lmRob*
##  [65] tidy.logical*                  tidy.logitmfx*
##  [67] tidy.lsmobj*                   tidy.manova*
##  [69] tidy.map*                      tidy.margins*
##  [71] tidy.Mclust*                   tidy.mediate*
##  [73] tidy.mfx*                      tidy.mjoint*
##  [75] tidy.mle2*                     tidy.mlm*
##  [77] tidy.mlogit*                   tidy.muhaz*
##  [79] tidy.multinom*                 tidy.negbin*
##  [81] tidy.negbinmfx*                tidy.nested_cv*
##  [83] tidy.nlrq*                     tidy.nls*
##  [85] tidy.NULL*                     tidy.numeric*
##  [87] tidy.orcutt*                   tidy.pairwise.htest*
##  [89] tidy.pam*                      tidy.plm*
##  [91] tidy.poissonmfx*               tidy.poLCA*
##  [93] tidy.polr*                     tidy.Polygon*
##  [95] tidy.Polygons*                 tidy.power.htest*
##  [97] tidy.prcomp*                   tidy.probitmfx*
##  [99] tidy.pyears*                   tidy.rcorr*
## [101] tidy.ref.grid*                 tidy.regsubsets*
## [103] tidy.ridgelm*                  tidy.rlm*
## [105] tidy.rma*                      tidy.roc*
## [107] tidy.rq*                       tidy.rqs*
## [109] tidy.rset*                     tidy.rsplit*
## [111] tidy.sarlm*                    tidy.Sarlm*
## [113] tidy.SpatialLinesDataFrame*    tidy.SpatialPolygons*
## [115] tidy.SpatialPolygonsDataFrame* tidy.spec*
## [117] tidy.speedglm*                 tidy.speedlm*
## [119] tidy.summary.glht*             tidy.summary.lm*
## [121] tidy.summary.plm*              tidy.summary_emm*
## [123] tidy.summaryDefault*           tidy.survdiff*
## [125] tidy.survexp*                  tidy.survfit*
## [127] tidy.survreg*                  tidy.svyglm*
## [129] tidy.svyolr*                   tidy.systemfit*
```

```
## [131] tidy.table*                    tidy.tobit*
## [133] tidy.ts*                       tidy.TukeyHSD*
## [135] tidy.varest*                   tidy.vfold_cv*
## [137] tidy.zoo*
## see '?methods' for accessing help and source code
```

```
methods("glance")
```

```
##  [1] glance.aareg*            glance.anova*            glance.aov*
##  [4] glance.Arima*            glance.betamfx*          glance.betareg*
##  [7] glance.biglm*            glance.binDesign*        glance.cch*
## [10] glance.clm*              glance.clmm*             glance.coeftest*
## [13] glance.coxph*            glance.crr*              glance.cv.glmnet*
## [16] glance.data.frame*       glance.default*          glance.drc*
## [19] glance.durbinWatsonTest* glance.ergm*             glance.factanal*
## [22] glance.felm*             glance.fitdistr*         glance.fixest*
## [25] glance.gam*              glance.Gam*              glance.garch*
## [28] glance.geeglm*           glance.glm*              glance.glmnet*
## [31] glance.glmRob*           glance.gmm*              glance.htest*
## [34] glance.ivreg*            glance.kmeans*           glance.lavaan*
## [37] glance.list*             glance.lm*               glance.lmodel2*
## [40] glance.lmrob*            glance.lmRob*            glance.logitmfx*
## [43] glance.margins*          glance.Mclust*           glance.mfx*
## [46] glance.mjoint*           glance.mlogit*           glance.muhaz*
## [49] glance.multinom*         glance.negbin*           glance.negbinmfx*
## [52] glance.nlrq*             glance.nls*              glance.NULL*
## [55] glance.orcutt*           glance.pam*              glance.plm*
## [58] glance.poissonmfx*       glance.poLCA*            glance.polr*
## [61] glance.probitmfx*        glance.pyears*           glance.ridgelm*
## [64] glance.rlm*              glance.rma*              glance.rq*
## [67] glance.rqs*              glance.sarlm*            glance.Sarlm*
## [70] glance.smooth.spline*    glance.speedglm*         glance.speedlm*
## [73] glance.summary.lm*       glance.summaryDefault*   glance.survdiff*
## [76] glance.survexp*          glance.survfit*          glance.survreg*
## [79] glance.svyglm*           glance.svyolr*           glance.tbl_df*
## [82] glance.varest*
## see '?methods' for accessing help and source code
```

```
methods("augment")
```

```
##  [1] augment.betamfx*       augment.betareg*       augment.clm*
##  [4] augment.coxph*         augment.data.frame*    augment.decomposed.ts*
##  [7] augment.default*       augment.drc*           augment.factanal*
## [10] augment.felm*          augment.fixest*        augment.gam*
```

```
## [13] augment.glm*           augment.glmrob*        augment.glmRob*
## [16] augment.htest*         augment.ivreg*         augment.kmeans*
## [19] augment.lm*            augment.lmrob*         augment.lmRob*
## [22] augment.loess*         augment.logitmfx*      augment.margins*
## [25] augment.Mclust*        augment.mfx*           augment.mjoint*
## [28] augment.mlogit*        augment.negbinmfx*     augment.nlrq*
## [31] augment.nls*           augment.NULL*          augment.pam*
## [34] augment.plm*           augment.poissonmfx*    augment.poLCA*
## [37] augment.polr*          augment.prcomp*        augment.probitmfx*
## [40] augment.rlm*           augment.rma*           augment.rq*
## [43] augment.rqs*           augment.sarlm*         augment.Sarlm*
## [46] augment.smooth.spline* augment.speedglm*      augment.speedlm*
## [49] augment.stl*           augment.survreg*
## see '?methods' for accessing help and source code
```

```
#tirei da ajuda do R!
# fit models
a <- lm(mpg ~ wt + qsec + disp, mtcars)
b <- lm(mpg ~ wt + qsec, mtcars)

mod <- anova(a, b)

# summarize model fit with tidiers
tidy(mod)
```

```
## # A tibble: 2 x 7
##   term                   df.residual   rss    df    sumsq statistic p.value
##   <chr>                        <dbl> <dbl> <dbl>    <dbl>     <dbl>   <dbl>
## 1 mpg ~ wt + qsec + disp          28  195.    NA NA       NA         NA
## 2 mpg ~ wt + qsec                 29  195.    -1 -0.00102  0.000147   0.990
```

```
glance(mod)
```

```
## # A tibble: 1 x 2
##   deviance df.residual
##      <dbl>       <dbl>
## 1     195.          29
```

```
# car::linearHypothesis() example
library(car)
mod_lht <- linearHypothesis(a, "wt - disp")
tidy(mod_lht)
```

```
## # A tibble: 1 x 10
```

```
##   term      null.value estim~1 std.e~2 stati~3 p.value df.re~4   rss    df sumsq
##   <chr>          <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 wt - disp          0   -5.03    1.23    16.6 3.39e-4      28  195.     1  116.
## # ... with abbreviated variable names 1: estimate, 2: std.error, 3: statistic,
## #   4: df.residual
```

```
glance(mod_lht)
```

```
## # A tibble: 1 x 2
##   deviance df.residual
##      <dbl>       <dbl>
## 1     195.          28
```

6.2.2 nls()

```
# fit model
nn <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))

# summarize model fit with tidiers + visualization
tidy(nn)
```

```
## # A tibble: 2 x 5
##   term  estimate std.error statistic  p.value
##   <chr>    <dbl>     <dbl>     <dbl>    <dbl>
## 1 k       49.7      3.79        13.1 5.96e-14
## 2 e        0.746    0.0199      37.5 8.86e-27
```

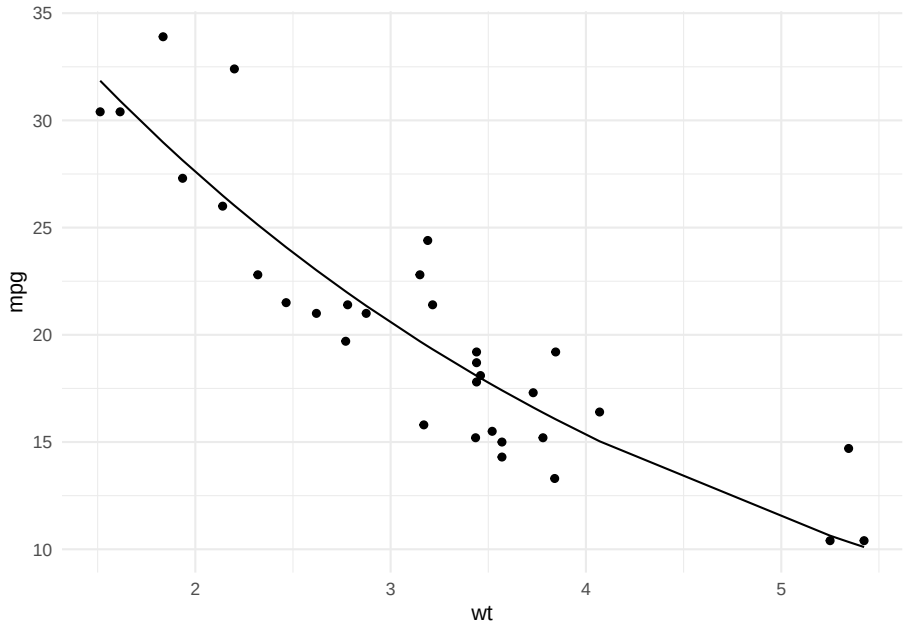
```
augment(nn)
```

```
## # A tibble: 32 x 4
##      mpg    wt .fitted .resid
##    <dbl> <dbl>   <dbl>  <dbl>
##  1  21    2.62    23.0 -2.01
##  2  21    2.88    21.4 -0.352
##  3  22.8  2.32    25.1 -2.33
##  4  21.4  3.22    19.3  2.08
##  5  18.7  3.44    18.1  0.611
##  6  18.1  3.46    18.0  0.117
##  7  14.3  3.57    17.4 -3.11
##  8  24.4  3.19    19.5  4.93
##  9  22.8  3.15    19.7  3.10
## 10  19.2  3.44    18.1  1.11
## # ... with 22 more rows
```

```
glance(nn)
```

```
## # A tibble: 1 x 9
##   sigma isConv     finTol logLik   AIC   BIC deviance df.residual  nobs
##   <dbl> <lgl>       <dbl>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
## 1  2.67 TRUE   0.00000204  -75.8  158.  162.     214.          30    32
```

```
ggplot(augment(nn), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))
```




```
newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1

augment(nn, newdata = newdata)
```

```
## # A tibble: 6 x 12
##     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb .fitted
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1  21       6   160   110  3.9   3.62  16.5     0     1     4     4    17.2
## 2  21       6   160   110  3.9   3.88  17.0     0     1     4     4    15.9
## 3  22.8     4   108    93  3.85  3.32  18.6     1     1     4     1    18.7
```

```
## 4  21.4     6   258   110  3.08  4.22  19.4     1     0     3     1    14.4
## 5  18.7     8   360   175  3.15  4.44  17.0     0     0     3     2    13.5
## 6  18.1     6   225   105  2.76  4.46  20.2     1     0     3     1    13.4
```

6.2.3 glm()

```
g <- glm(am ~ mpg, mtcars, family = "binomial")
g
```

```
##
## Call:  glm(formula = am ~ mpg, family = "binomial", data = mtcars)
##
## Coefficients:
## (Intercept)          mpg
##      -6.604        0.307
##
## Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
## Null Deviance:       43.23
## Residual Deviance: 29.68     AIC: 33.68
```

```
glance(g)
```

```
## # A tibble: 1 x 8
##   null.deviance df.null logLik   AIC   BIC deviance df.residual  nobs
##           <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
## 1          43.2      31  -14.8  33.7  36.6     29.7          30    32
```

6.2.4 gam()

```
# load libraries for models and data
library(mgcv)

# fit model
g <- gam(mpg ~ s(hp) + am + qsec, data = mtcars)
g
```

```
##
## Family: gaussian
## Link function: identity
##
```

```
## Formula:
## mpg ~ s(hp) + am + qsec
##
## Estimated degrees of freedom:
## 2.36  total = 5.36
##
## GCV score: 8.837538
```

```
# summarize model fit with tidiers
tidy(g)
```

```
## # A tibble: 1 x 5
##   term    edf ref.df statistic p.value
##   <chr> <dbl>  <dbl>     <dbl>   <dbl>
## 1 s(hp)  2.36   3.02      6.34 0.00218
```

```
tidy(g, parametric = TRUE)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)  16.7        9.83      1.70  0.101
## 2 am            4.37       1.56      2.81  0.00918
## 3 qsec          0.0904     0.525     0.172 0.865
```

```
glance(g)
```

```
## # A tibble: 1 x 7
##      df logLik   AIC   BIC deviance df.residual  nobs
##   <dbl>  <dbl> <dbl> <dbl>    <dbl>       <dbl> <int>
## 1  5.36  -74.4  162.  171.     196.        26.6    32
```

```
augment(g)
```

```
## # A tibble: 32 x 10
##      mpg    am  qsec    hp .fitted .se.fit .resid   .hat .sigma  .cooksd
##    <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>  <dbl>  <dbl> <lgl>     <dbl>
##  1  21       1  16.5   110    24.3   1.03  -3.25  0.145  NA     0.0529
##  2  21       1  17.0   110    24.3   0.925 -3.30  0.116  NA     0.0411
##  3  22.8     1  18.6    93    26.0   0.894 -3.22  0.109  NA     0.0359
##  4  21.4     0  19.4   110    20.2   0.827  1.25  0.0930 NA     0.00448
##  5  18.7     0  17.0   175    15.7   0.815  3.02  0.0902 NA     0.0251
```

```
##  6  18.1     0  20.2   105    20.7   0.914 -2.56  0.113  NA     0.0240
##  7  14.3     0  15.8   245    12.7   1.11   1.63  0.167  NA     0.0162
##  8  24.4     0  20      62    25.0   1.45  -0.618 0.287  NA     0.00545
##  9  22.8     0  22.9    95    21.8   1.81   0.959 0.446  NA     0.0340
## 10  19.2     0  18.3   123    19.0   0.864  0.211 0.102  NA     0.000142
## # ... with 22 more rows
```

6.3 links

https://www.youtube.com/watch?v=1bnhT8tlCJQ&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

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]

Capítulo 7

Extra

7.1 Introduction

The overall goal is to give you a very quick introduction to conducting correlation and regression analyses in R.

7.2 Correlation

The Pearson product moment correlation seeks to measure the linear association between two variables, x and y on a standardized scale ranging from $r = -1 - -1$.

The correlation of x and y is a covariance that has been standardized by the standard deviations of x and y. This yields a scale-insensitive measure of the linear association of x and y. For much more conceptual detail, see this: https://psu-psychology.github.io/psy-597-SEM/01_correlation_regression/01_Correlation_and_Regression.html.

$$r_{XY} = \frac{s_{XY}}{s_X s_Y}$$

7.3 Correlation matrix

```
to_correlate <- mtcars %>% dplyr::select(qsec, cyl, disp, hp)
cor(to_correlate)
```

```
##            qsec        cyl       disp         hp
## qsec  1.0000000 -0.5912421 -0.4336979 -0.7082234
## cyl  -0.5912421  1.0000000  0.9020329  0.8324475
## disp -0.4336979  0.9020329  1.0000000  0.7909486
## hp   -0.7082234  0.8324475  0.7909486  1.0000000
```

7.4 Testing a bivariate association

Recall that the significance of correlations are computed on $n - 2$ degrees of freedom.

The t-test is:

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

```
cor.test(mtcars$qsec, mtcars$cyl)
```

```
##
##  Pearson's product-moment correlation
##
## data:  mtcars$qsec and mtcars$cyl
## t = -4.0154, df = 30, p-value = 0.0003661
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.7792781 -0.3055388
## sample estimates:
##       cor
## -0.5912421
```

Note that we can use the `conf.int` argument to `cor.test` to get different levels of confidence.

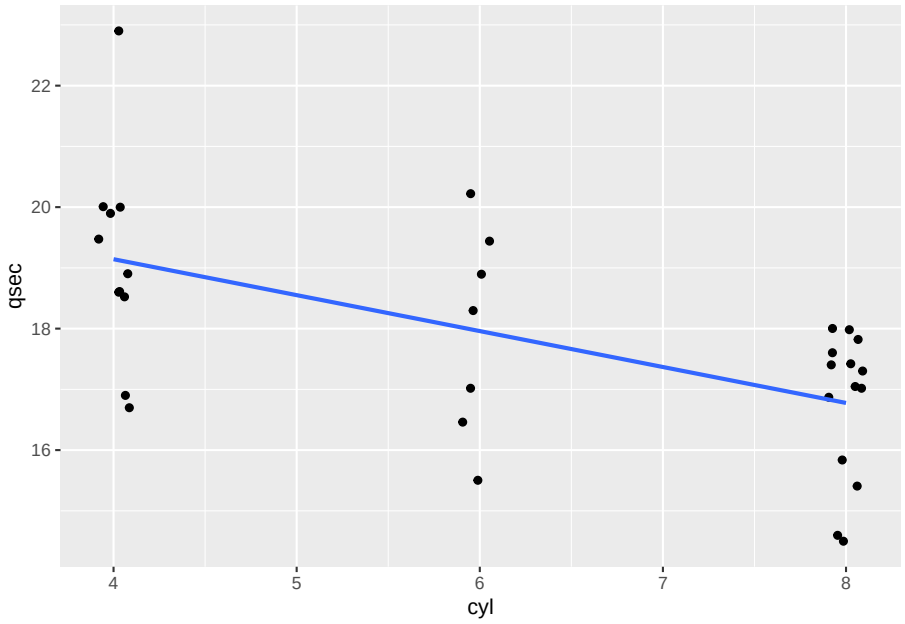
```
cor.test(mtcars$qsec, mtcars$cyl, conf.level = 0.9)
```

```
##
##  Pearson's product-moment correlation
##
## data:  mtcars$qsec and mtcars$cyl
## t = -4.0154, df = 30, p-value = 0.0003661
## alternative hypothesis: true correlation is not equal to 0
## 90 percent confidence interval:
##  -0.7552287 -0.3576005
```

```
## sample estimates:
##        cor
## -0.5912421
```

7.5 Visualizing the association

```
ggplot(to_correlate, aes(x=cyl, y=qsec)) + geom_jitter(width=0.1) + stat_smooth(method="lm", se=F
```



7.6 Testing the significance of all correlations in the matrix

```
Hmisc::rcorr(to_correlate %>% as.matrix())
```

```
##       qsec   cyl  disp    hp
## qsec  1.00 -0.59 -0.43 -0.71
## cyl  -0.59  1.00  0.90  0.83
## disp -0.43  0.90  1.00  0.79
## hp   -0.71  0.83  0.79  1.00
```

```
##
## n= 32
##
##
## P
##      qsec   cyl    disp   hp
## qsec        0.0004 0.0131 0.0000
## cyl  0.0004        0.0000 0.0000
## disp 0.0131 0.0000        0.0000
## hp   0.0000 0.0000 0.0000
```

Notice that we now get a matrix of p-values, too...

7.7 Pretty output

```
stargazer(cor(to_correlate), type = "html")
```

qsec

cyl

disp

hp

qsec

1

-0.591

-0.434

-0.708

cyl

-0.591

1

0.902

0.832

disp

-0.434

0.902

1

0.791

hp

-0.708

0.832

0.791

1

```
#you can use the filename argument to write out the table as a Word doc!
apaTables::apa.cor.table(to_correlate)
```

```
##
##
## Means, standard deviations, and correlations with confidence intervals
##
##
##   Variable M      SD     1            2          3
##   1. qsec  17.85  1.79
##
##   2. cyl   6.19   1.79   -.59**
##                          [-.78, -.31]
##
##   3. disp  230.72 123.94 -.43*        .90**
##                          [-.68, -.10] [.81, .95]
##
##   4. hp    146.69 68.56  -.71**       .83**      .79**
##                          [-.85, -.48] [.68, .92] [.61, .89]
##
##
## Note. M and SD are used to represent mean and standard deviation, respectively.
## Values in square brackets indicate the 95% confidence interval.
## The confidence interval is a plausible range of population correlations
## that could have caused the sample correlation (Cumming, 2014).
##  * indicates p < .05. ** indicates p < .01.
##
```

7.8 Keeping the correlations for further analysis

Here, we store all details of the bivariate correlation test as an R object `ctest`.

```
ctest <- cor.test(mtcars$qsec, mtcars$cyl)
```

Let's poke under the hood:

```
str(ctest)
```

```
## List of 9
##  $ statistic  : Named num -4.02
##   ..- attr(*, "names")= chr "t"
##  $ parameter  : Named int 30
##   ..- attr(*, "names")= chr "df"
##  $ p.value    : num 0.000366
##  $ estimate   : Named num -0.591
##   ..- attr(*, "names")= chr "cor"
##  $ null.value : Named num 0
##   ..- attr(*, "names")= chr "correlation"
##  $ alternative: chr "two.sided"
##  $ method     : chr "Pearson's product-moment correlation"
##  $ data.name  : chr "mtcars$qsec and mtcars$cyl"
##  $ conf.int   : num [1:2] -0.779 -0.306
##   ..- attr(*, "conf.level")= num 0.95
##  - attr(*, "class")= chr "htest"
```

So, we can poke around and grab specific things:

```
ctest$p.value
```

```
## [1] 0.0003660533
```

```
ctest$estimate
```

```
##        cor
## -0.5912421
```

And there are useful helper packages, especially `broom`, that will help you work with statistics objects as `data.frame` objects.

```
broom::glance(ctest)
```

```
## # 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   -0.591     -4.02 0.000366        30   -0.779    -0.306 Pearson's pr~ two.si~
## # ... with abbreviated variable name 1: alternative
```

7.9 Correlation method

You can use a different correlation method (e.g., Spearman) using the `method` argument:

```
cor.test(mtcars$qsec, mtcars$cyl, method = "spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  mtcars$qsec and mtcars$cyl
## S = 8578.7, p-value = 0.0006196
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## -0.5723509
```

```
cor(to_correlate, method = "spearman")
```

```
##            qsec        cyl       disp         hp
## qsec  1.0000000 -0.5723509 -0.4597818 -0.6666060
## cyl  -0.5723509  1.0000000  0.9276516  0.9017909
## disp -0.4597818  0.9276516  1.0000000  0.8510426
## hp   -0.6666060  0.9017909  0.8510426  1.0000000
```

7.10 Missing data

By default, `cor` will return an `NA` (missing) for every pair in which at least one observation is missing. We can ask for correlations to be estimated on the complete cases for each pair. This is `use="pairwise.complete.obs"`.

Here's the difference (I introduced some missing data to make the point):

First, with 'everything' as the `use` argument (any missing on a variable drops it from the correlation table).

```
to_correlate_miss <- to_correlate
to_correlate_miss$qsec[c(1, 5)] <- NA
cor(to_correlate_miss) #implicitly use="everything"
```

```
##      qsec       cyl      disp        hp
## qsec    1        NA        NA        NA
## cyl    NA 1.0000000 0.9020329 0.8324475
## disp   NA 0.9020329 1.0000000 0.7909486
## hp     NA 0.8324475 0.7909486 1.0000000
```

Now with pairwise complete calculation:

```
cor(to_correlate_miss, use="pairwise.complete.obs")
```

```
##            qsec        cyl       disp         hp
## qsec  1.0000000 -0.5961232 -0.4483569 -0.7313008
## cyl  -0.5961232  1.0000000  0.9020329  0.8324475
## disp -0.4483569  0.9020329  1.0000000  0.7909486
## hp   -0.7313008  0.8324475  0.7909486  1.0000000
```

7.11 Single-predictor (simple) regression

Next, let's turn to 'simple' linear regression (one predictor, one outcome), then scale to multiple regression (many predictors, one outcome). The standard linear regression model is implemented by the `lm` function in R. The `lm` function uses ordinary least squares (OLS) which estimates the parameter by minimizing the squared residuals.

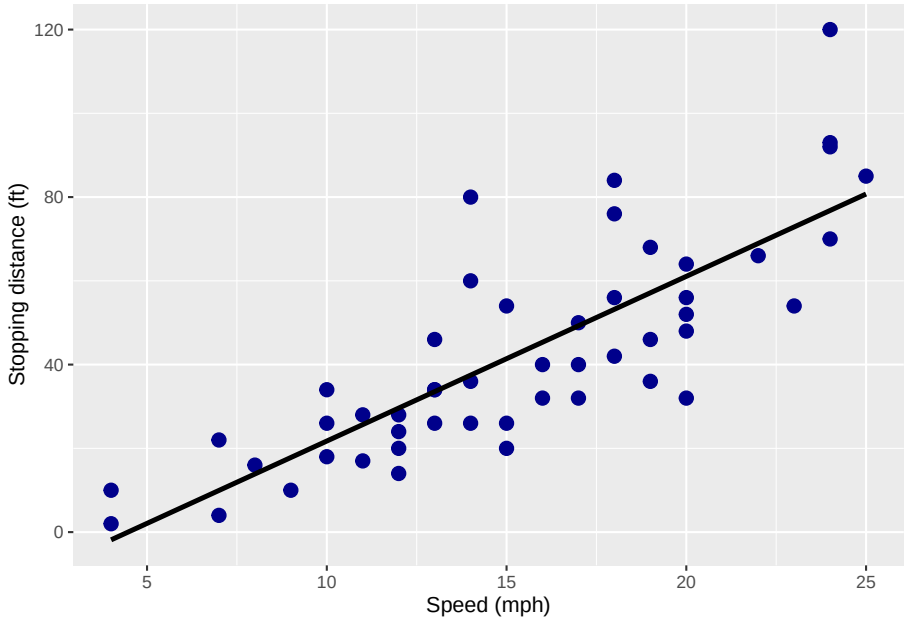
In simple regression, we are interested in a relationship of the form:

$$Y = B_0 + B_1 X$$

where Y is the dependent variable (criterion) and X is the predictor (covariate). The intercept is represented by $B0$ and the slope for the X predictor by $B1$.

Let's take a look at the simple case of stopping distance (braking) as a function of car speed.

```
ggplot(cars, aes(x=speed, y=dist)) +
  geom_point(color='darkblue', size = 3) +
  geom_smooth(method=lm, se=FALSE, fullrange=TRUE, color='black', size=1.2) +
  labs(x="Speed (mph)", y="Stopping distance (ft)")
```



When conducting regression, we typically try to capture linear relationships among variables. We can introduce higher-order polynomial terms (e.g., quadratic models) or splines (more flexible shapes), but this beyond the scope here.

Fortunately, this relationship looks quite linear! The faster the car, the longer it takes to brake.

In R regression models, we use the `~` operator to denote 'regressed on'. It's not especially intuitive, but we say the criterion is regressed on the predictor. Here, if we think speed is a key cause of stopping distance, we'd say 'braking distance regressed on speed' or 'speed predicts braking distance.'

In formula terms, this is `dist ~ speed`, which we pass as the first argument to `lm()`.

```
lm_cars <- lm(dist ~ speed, data=cars)
summary(lm_cars)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
```

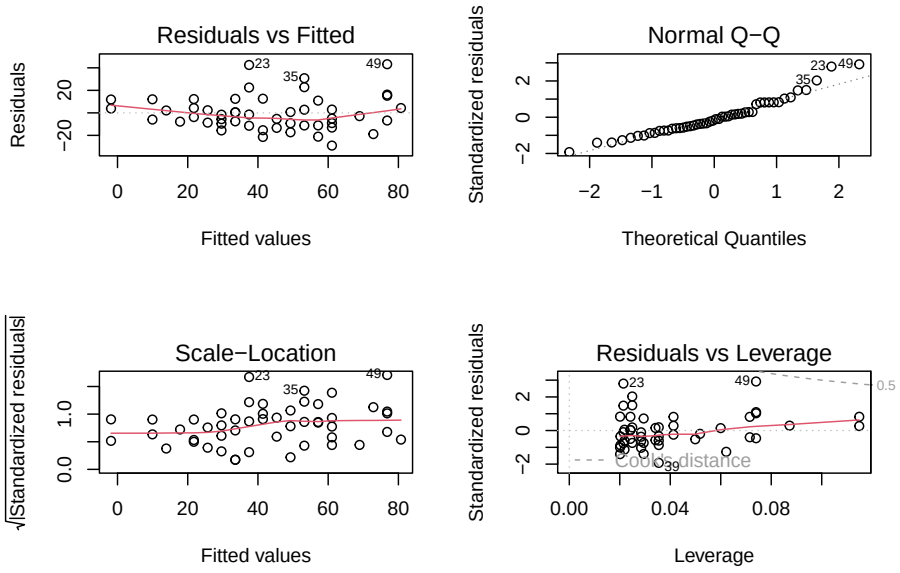
```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

The output contains individual parameter estimates of the model (here, just the intercept and slope), their standard errors, significance tests, and p-values (one degree of freedom). We also get global information such as the sum of squared errors and the coefficient of determination (R^2).

7.12 Regression diagnostics

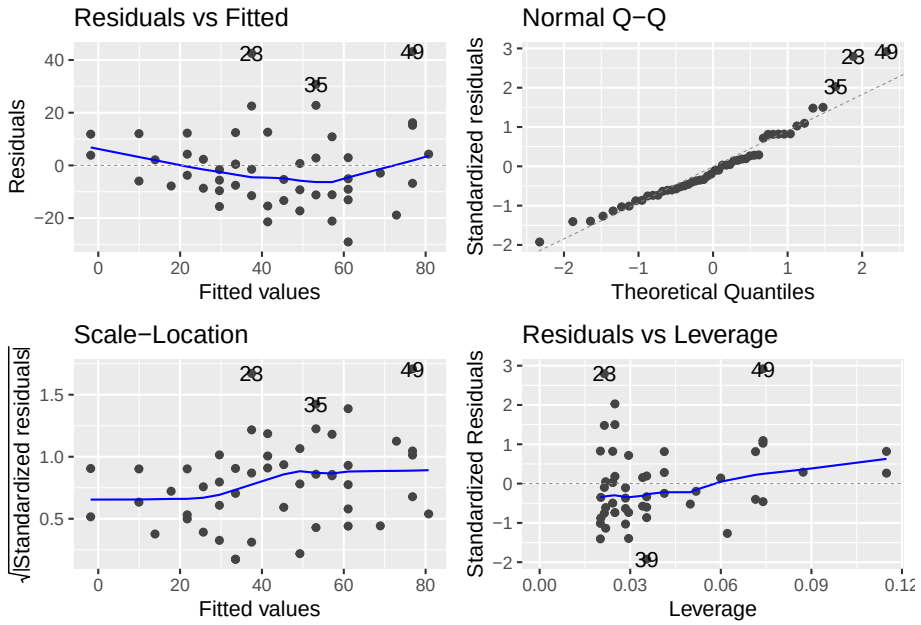
We can also get useful diagnostic plots for free using the `plot()` function:

```
par(mfrow=c(2,2))
plot(lm_cars)
```



The `ggfortify` package also provides an `autoplot` function that gives similar diagnostics within a handy ggplot-based graph.

```
autoplot(lm_cars)
```



7.13 Bootstrap estimates and confidence intervals

Using functionality from the `car` and `boot` packges, we can easily get estimates of the regression coefficients and standard errors using nonparametric bootstrapping, which relaxes the normal theory assumption on the standard errors and, therefore, the significance tests. Likewise, the model does not assume normally distributed error.

Nonparametric bootstrapping approximates the sampling distribution for a statistic of interest (e.g., a slope) by resampling the existing data with replacement many times and examining the resulting density.

```
system.time(lm_cars.boot <- Boot(lm_cars, R=2000))
```

```
##    usuário   sistema decorrido
##       2.29      0.05      2.37
```

```
summary(lm_cars.boot, high.moments=TRUE)
```

```
##
## Number of bootstrap replications R = 2000
##             original  bootBias  bootSE  bootMed bootSkew bootKurtosis
## (Intercept) -17.5791  0.226401 5.79888 -16.9703 -0.30316      0.41996
## speed         3.9324 -0.023637 0.41637   3.9049  0.21044      0.14774
```

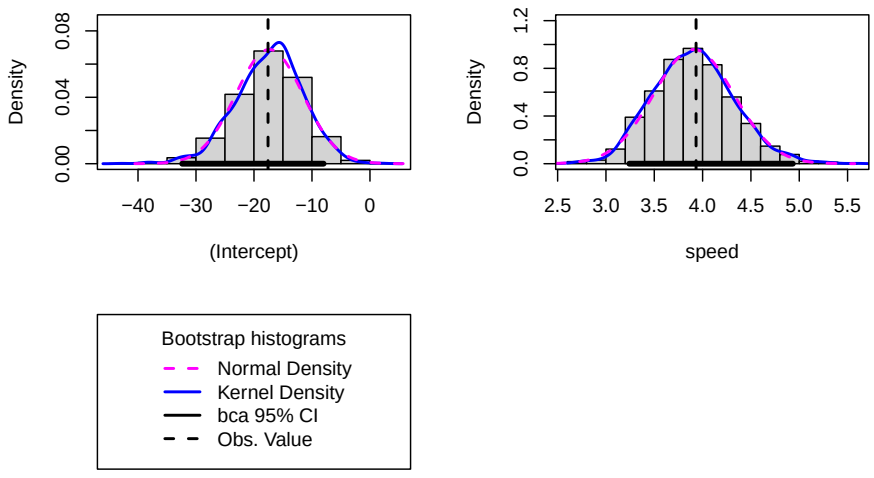
We can use the object to obtain 95% bootstrapped confidence intervals using the 'bias corrected, accelerated' method (aka bca).

```
confint(lm_cars.boot, level=.95, type="bca")
```

```
## Bootstrap bca confidence intervals
##
##                  2.5 %    97.5 %
## (Intercept) -32.379303 -7.993205
## speed         3.241629  4.933584
```

And we can easily compare the bootstrapped and standard OLS models:

```
hist(lm_cars.boot, legend="separate")
```



Notice that the `speed` regression coefficient is slightly positively skewed. Additional details are provided in John Fox's useful book "An R Companion to Applied Regression" (2nd edition): https://socialsciences.mcmaster.ca/jfox/Books/Companion/appendix/Appendix-Bootstrapping.pdf.

7.14 Multiple regression

We can easily extend to larger regression models by adding terms to the right side of the formula. For example, in the `mtcars` dataset (car performance statistics from 1974 Motor Trend), we could examine the extent to which the gas mileage (`mpg`) is a function of both gross horsepower (`hp`) and transmission (`am`, where 0 is 'automatic' and 1 is 'manual').

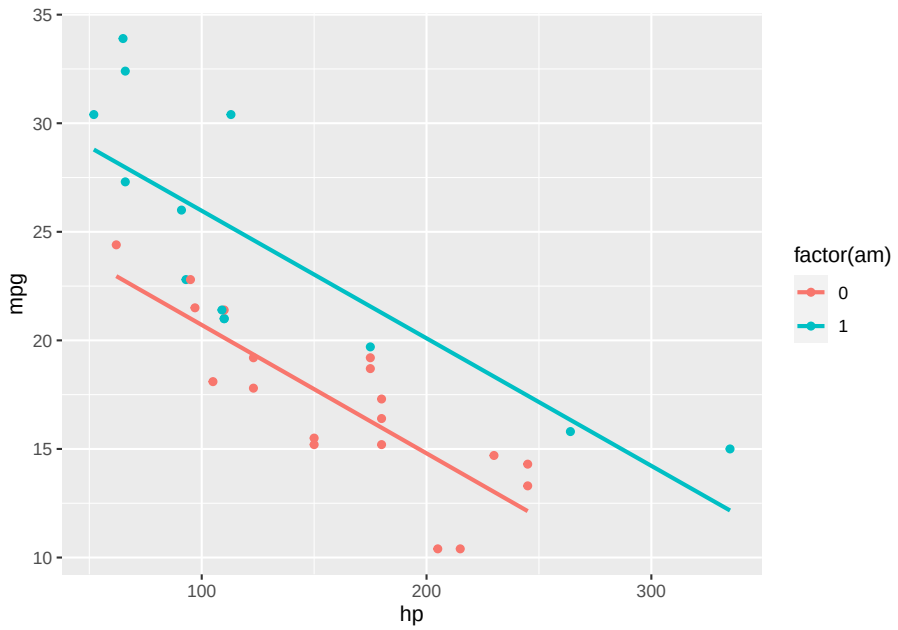
```
mpg_model <- lm(mpg ~ hp + am, mtcars)
summary(mpg_model)
```

```
##
## Call:
## lm(formula = mpg ~ hp + am, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3843 -2.2642  0.1366  1.6968  5.8657
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 26.584914   1.425094  18.655  < 2e-16 ***
## hp          -0.058888   0.007857  -7.495 2.92e-08 ***
## am           5.277085   1.079541   4.888 3.46e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.909 on 29 degrees of freedom
## Multiple R-squared:  0.782,  Adjusted R-squared:  0.767
## F-statistic: 52.02 on 2 and 29 DF,  p-value: 2.55e-10
```

It appears that these are both influential predictors. We could examine the relationship graphically.

7.15 Visualizing regression data

```
ggplot(mtcars, aes(x=hp, y=mpg, color=factor(am))) + geom_point() +
  stat_smooth(method=lm, se=FALSE)
```



There doesn't appear to be any meaningful interaction between horsepower and transmission type.

7.16 Getting results into a tidy, useful format

Note that the `broom` package is very useful for extracting global and specific statistics from many models in R, including regression models. The introductory vignette provides a number of useful examples: https://cran.r-project.org/web/packages/broom/vignettes/broom.html. Here, what if we want to save the global statistics and parameter estimates into data.frame objects?

We can use the `glance` function to get the global model statistics.

```
broom::glance(mpg_model)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squa~1 sigma stati~2  p.value    df logLik   AIC   BIC devia~3
```

```
##       <dbl>        <dbl> <dbl>   <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.782        0.767  2.91    52.0 2.55e-10     2  -78.0  164.  170.    245.
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

And the `tidy` function yields the parameter table

```
broom::tidy(mpg_model)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  26.6      1.43        18.7  1.07e-17
## 2 hp           -0.0589   0.00786     -7.50 2.92e- 8
## 3 am            5.28     1.08         4.89 3.46e- 5
```

As can imagine (and saw earlier in the functional programming overview), the ability to extract regression statistics into a tidy data.frame is a boon to scaling your analyses to multiple models and datasets.

7.17 Modeling interactions

We can use the `*` operator in R to ask that both the constituent variables and their interaction(s) are entered into the model. For example:

```
int_model <- lm(mpg ~ hp*wt + am, mtcars)
summary(int_model)
```

```
##
## Call:
## lm(formula = mpg ~ hp * wt + am, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9845 -1.6580 -0.7407  1.4362  4.5266
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.452241   5.280731   9.365 5.69e-10 ***
## hp          -0.119303   0.026550  -4.494 0.000119 ***
## wt          -8.100558   1.789325  -4.527 0.000108 ***
## am           0.125107   1.333431   0.094 0.925942
```

```
## hp:wt        0.027488   0.008473   3.244 0.003130 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.192 on 27 degrees of freedom
## Multiple R-squared:  0.8848, Adjusted R-squared:  0.8677
## F-statistic: 51.84 on 4 and 27 DF,  p-value: 2.765e-12
```

This model includes individual effects of horsepower (`hp`) and weight (`wt`), as well as their interation (`hp:wt`). This highlights that the asterisk operator `*` will compute all possible interations among the specified predictors. For example, `a*b*c*d` will generate all effets up through and including the `a x b x c x d` interation. By contrast, if you wish to specify a given interaction manually/directly, use the colon operator (e.g., `a:b`). The downside of the colon operator is that it doesn't guarantee that the corresponding lower-level effects are included, which is usually a sane default position. As a reminder, you should essentially never include an interation without including the lower level effects, because this can misassign the variance.

Note that with weight (`wt`) in the model, as well the horsepower x weight interaction, the automatic/manual transmission distinction is no longer significant (not even close). Let's take a look at this interaction. For a two-way continuous x continuous interation, we typically separate one predictor into low (-1SD), medium (mean), and high (+1SD) levels, and plot separate lines for each.

```
#handy 2-way interation plotting function from jtools.
interact_plot(int_model, pred = "hp", modx = "wt")
```

What do we see? At higher horsepower, gas mileage suffers regardless of the weight of the car. At lower horsepower, car weight makes a big difference (lower weight, higher mpg).

7.18 Contrasts in regression

(Some of the code and text here has been adapted from Russell Lenth's excellent `emmeans` documentation: https://cran.r-project.org/web/packages/emmeans/)

One of the handiest packages in the R regression universe is `emmeans`, which can provide the 'expected marginal means' (em means), as well as a host of other contrasts and comparisons. In particular, it is very easy to test simple slopes and pairwise differences. Furthermore, the package works with `multcomp` to handle correction for multiple comparisons. See the longer documentation here.

Let's look at the concentration of leucine in a study of pigs who were fed differing levels of protein in the diet (9, 12, 15, and 18%) and different protein sources

(fish, soybean, milk). The concentration has a long positive tail, so here we log transform it to normalize things somewhat.

```
data(pigs, package="emmeans")
pigs <- pigs %>% mutate(log_conc=log(conc), percent_fac=factor(percent))
pigs.lm <- lm(log_conc ~ source + percent_fac, data = pigs)
summary(pigs.lm)
```

```
##
## Call:
## lm(formula = log_conc ~ source + percent_fac, data = pigs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18858 -0.05980 -0.02158  0.08120  0.24601
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.22029    0.05361  60.067  < 2e-16 ***
## sourcesoy      0.27277    0.05293   5.153 3.19e-05 ***
## sourceskim     0.40228    0.05416   7.428 1.50e-07 ***
## percent_fac12  0.17955    0.05608   3.202 0.003960 **
## percent_fac15  0.21740    0.05973   3.640 0.001370 **
## percent_fac18  0.29985    0.06762   4.434 0.000191 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1151 on 23 degrees of freedom
## Multiple R-squared:  0.7566, Adjusted R-squared:  0.7037
## F-statistic:  14.3 on 5 and 23 DF,  p-value: 2.054e-06
```

This output is hard to look at because there are many dummy codes and we have to infer the reference condition for each factor (usually alphabetical). Also, we do not have an intuitive sense of the expected means in each condition because they depend on the sum of the intercept and the specific dummy code for the condition interest, averaging over the other factor.

We can obtain the expected means for each condition.

7.19 Expected means for protein source

```
pigs.emm.s <- emmeans(pigs.lm, "source")
print(pigs.emm.s)
```

```
##  source emmean     SE df lower.CL upper.CL
##  fish     3.39 0.0367 23     3.32     3.47
##  soy      3.67 0.0374 23     3.59     3.74
##  skim     3.80 0.0394 23     3.72     3.88
##
## Results are averaged over the levels of: percent_fac
## Confidence level used: 0.95
```

7.20 Expected means for protein level

```
pigs.emm.p <- emmeans(pigs.lm, "percent_fac")
print(pigs.emm.p)
```

```
##  percent_fac emmean     SE df lower.CL upper.CL
##  9             3.45 0.0409 23     3.36     3.53
##  12            3.62 0.0384 23     3.55     3.70
##  15            3.66 0.0437 23     3.57     3.75
##  18            3.75 0.0530 23     3.64     3.85
##
## Results are averaged over the levels of: source
## Confidence level used: 0.95
```

7.21 Means in each cell of the factorial design

```
print(emmeans(pigs.lm, ~source*percent_fac))
```

```
##  source percent_fac emmean     SE df lower.CL upper.CL
##  fish   9             3.22 0.0536 23     3.11     3.33
##  soy    9             3.49 0.0498 23     3.39     3.60
##  skim   9             3.62 0.0501 23     3.52     3.73
##  fish   12            3.40 0.0493 23     3.30     3.50
##  soy    12            3.67 0.0489 23     3.57     3.77
##  skim   12            3.80 0.0494 23     3.70     3.90
##  fish   15            3.44 0.0548 23     3.32     3.55
##  soy    15            3.71 0.0507 23     3.61     3.82
##  skim   15            3.84 0.0549 23     3.73     3.95
##  fish   18            3.52 0.0547 23     3.41     3.63
##  soy    18            3.79 0.0640 23     3.66     3.93
##  skim   18            3.92 0.0646 23     3.79     4.06
##
## Confidence level used: 0.95
```

7.22 Pairwise comparisons among protein sources

If we wanted to compare the pairwise differences in the effect of protein source on leucine concentration while controlling for protein percentage (and potentially other variables we add to the model), we could use the `pairs` function:

```
pig_pairs <- pairs(pigs.emm.s)
print(pig_pairs)
```

```
##  contrast    estimate     SE df t.ratio p.value
##  fish - soy    -0.273 0.0529 23  -5.153  0.0001
##  fish - skim   -0.402 0.0542 23  -7.428  <.0001
##  soy - skim    -0.130 0.0530 23  -2.442  0.0570
##
## Results are averaged over the levels of: percent_fac
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Note that you can get a sense of the contrasts being tested by `emmeans` by examining the `@linfct` slot of the object. I've learned *a lot* by examining these contrast matrices and thinking about how to setup a (focal) contrast of interest. Also note that you get p-value adjustment for free (here, Tukey's HSD method).

Contrasts for the predicted mean level of leucine contrast for each protein source, controlling for protein percentage.

```
pigs.emm.s@linfct
```

```
##      (Intercept) sourcesoy sourceskim percent_fac12 percent_fac15 percent_fac18
## [1,]           1         0          0          0.25          0.25          0.25
## [2,]           1         1          0          0.25          0.25          0.25
## [3,]           1         0          1          0.25          0.25          0.25
```

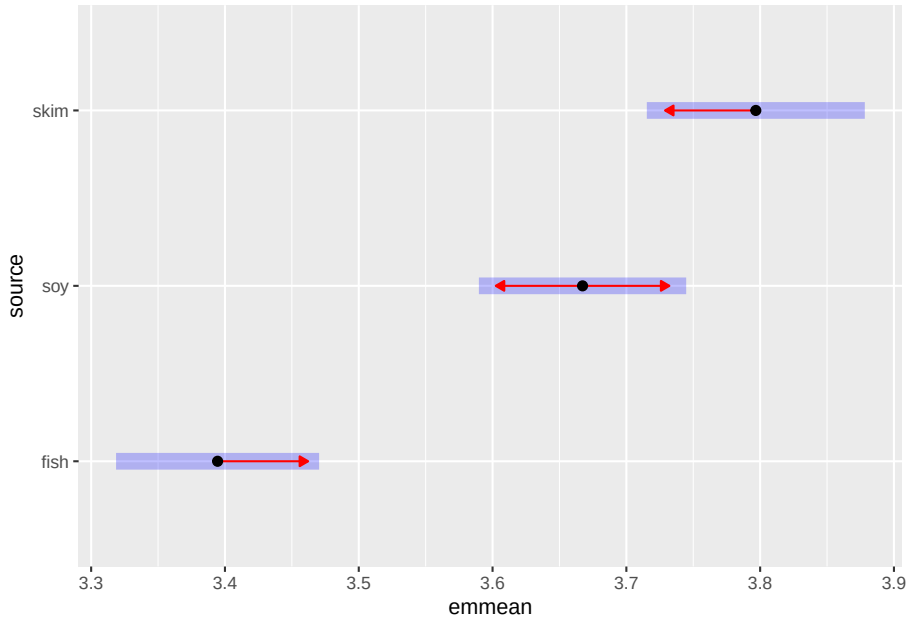
What are the pairwise contrasts for the protein sources?

```
pig_pairs@linfct
```

```
##      (Intercept) sourcesoy sourceskim percent_fac12 percent_fac15 percent_fac18
## [1,]           0        -1          0             0             0             0
## [2,]           0         0         -1             0             0             0
## [3,]           0         1         -1             0             0             0
```

The `emmeans` package also provides useful plots to understand pairwise differences:

```
plot(pigs.emm.s, comparisons = TRUE)
```



The blue bars are confidence intervals for the EMMs, and the red arrows are for the comparisons among them. If an arrow from one mean overlaps an arrow from another group, the difference is not significant, based on the adjust setting (which defaults to "tukey"). (Note: Don't ever use confidence intervals for EMMs to perform comparisons; they can be very misleading.)

7.23 Pairwise differences and simple slopes in regression

Returning to the iris dataset (from the parallel R examples), consider a model in which we examine the association between petal length and width across species. Here, we regress petal length on petal width, species (three levels), and their interaction.

```
fitiris <- lm(Petal.Length ~ Petal.Width * Species, data = iris)
summary(fitiris)
```

```
##
```

```
## Call:
## lm(formula = Petal.Length ~ Petal.Width * Species, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.84099 -0.19343 -0.03686  0.16314  1.17065
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.3276     0.1309  10.139  < 2e-16 ***
## Petal.Width                     0.5465     0.4900   1.115   0.2666
## Speciesversicolor               0.4537     0.3737   1.214   0.2267
## Speciesvirginica                2.9131     0.4060   7.175 3.53e-11 ***
## Petal.Width:Speciesversicolor   1.3228     0.5552   2.382   0.0185 *
## Petal.Width:Speciesvirginica    0.1008     0.5248   0.192   0.8480
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3615 on 144 degrees of freedom
## Multiple R-squared:  0.9595, Adjusted R-squared:  0.9581
## F-statistic: 681.9 on 5 and 144 DF,  p-value: < 2.2e-16
```

```
car::Anova(fitiris, type="III") #overall effects of predictors in the model
```

```
## Anova Table (Type III tests)
##
## Response: Petal.Length
##                      Sum Sq  Df  F value    Pr(>F)
## (Intercept)         13.4329   1 102.8050 < 2.2e-16 ***
## Petal.Width          0.1625   1   1.2438 0.2665889
## Species              6.7474   2  25.8196 2.614e-10 ***
## Petal.Width:Species  2.0178   2   7.7213 0.0006525 ***
## Residuals           18.8156 144
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that this yields a categorical (species) x continuous (petal width) interaction. The output from `car::Anova` indicates that the interaction is significant, but we need more detailed guidance on how the slope for petal width is moderated by species. We can visualize the interaction as follows:

```
interact_plot(fitiris, pred = "Petal.Width", modx = "Species")
```

In a simple slopes test, we might wish to know whether the slope for `Petal.Width` is non-zero in each species individually. Let's start by getting the estimated marginal means for each species.

```
emmeans(fitiris, ~Species)
```

```
##  Species    emmean     SE  df lower.CL upper.CL
##  setosa       1.98 0.4699 144     1.05     2.91
##  versicolor   4.02 0.0609 144     3.90     4.14
##  virginica    5.02 0.1636 144     4.69     5.34
##
## Confidence level used: 0.95
```

And pairwise differences between species:

```
pairs(emmeans(fitiris, ~Species))
```

```
##  contrast               estimate    SE  df t.ratio p.value
##  setosa - versicolor      -2.040 0.474 144  -4.306  0.0001
##  setosa - virginica       -3.034 0.498 144  -6.097  <.0001
##  versicolor - virginica   -0.994 0.175 144  -5.692  <.0001
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Transitioning to petal width, because we are interested its linear effect (slope), we use the `emtrends` function to estimate the slope in each species individually. In terms of simple slopes, we test whether the Petal.Width slope is non-zero in each Species. The `infer` argument in the summary of `emtrends` requests t-tests and p-values for the slopes.

```
summary(emtrends(model = fitiris, ~Species, var="Petal.Width"), infer=TRUE)
```

Finally, we could examine pairwise differences between slopes among species.

```
pairs(emtrends(model = fitiris, ~Species, var="Petal.Width"))
```

7.24 A few other emmeans features

In the pigs dataset, we have treated protein percentage as a fact (9, 12, 15, or 18 percent). If we keep this representation (as opposed to entering percentage as continuous), we can easily get orthogonal polynomial contrasts in `emmeans`. For example, is the effect of protein percent linearly related to leucine, or might it be quadratic or cubic?

```
pigs.emm.p <- emmeans(pigs.lm, "percent_fac")
ply <- contrast(pigs.emm.p, "poly")
ply
```

```
##  contrast  estimate     SE df t.ratio p.value
##  linear      0.9374 0.2106 23   4.452  0.0002
##  quadratic  -0.0971 0.0883 23  -1.099  0.2830
##  cubic       0.1863 0.1877 23   0.992  0.3313
##
## Results are averaged over the levels of: source
```

```
coef(ply) #show the contrast coefficients
```

```
##               percent_fac c.1 c.2 c.3
## percent_fac9            9  -3   1  -1
## percent_fac12          12  -1  -1   3
## percent_fac15          15   1  -1  -3
## percent_fac18          18   3   1   1
```

There is a lot more on probing interations here: https://cran.r-project.org/web/packages/emmeans/vignettes/interactions.html.

Finally, we can examine effects in multivariate regression models (i.e., multiple DVs). Here, we can examine the sales of two varieties of oranges as a function of their prices, the day, and store. Sales for both oranges are modeled at once (`sales1` and `sales2`) to get a sense of price interdependence.

```
org.int <- lm(cbind(sales1, sales2) ~ price1 * price2 + day + store, data = oranges)
summary(org.int)
```

```
## Response sales1 :
##
## Call:
## lm(formula = sales1 ~ price1 * price2 + day + store, data = oranges)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4646 -2.8230  0.4627  1.5425  6.8493
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   67.91880   30.97512   2.193  0.03920 *
## price1        -1.41364    0.60170  -2.349  0.02819 *
```

```
## price2        -0.52938    0.68462  -0.773  0.44760
## day2           0.19984    2.57232   0.078  0.93878
## day3           7.57474    2.52069   3.005  0.00652 **
## day4           2.90480    2.52012   1.153  0.26143
## day5           9.68284    2.56659   3.773  0.00105 **
## day6           5.51804    2.53842   2.174  0.04076 *
## store2         1.58480    2.91591   0.544  0.59225
## store3        -0.61385    3.01569  -0.204  0.84057
## store4         2.54945    3.03622   0.840  0.41012
## store5         1.83524    2.86046   0.642  0.52777
## store6         7.01866    2.76358   2.540  0.01867 *
## price1:price2  0.01369    0.01367   1.001  0.32755
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.213 on 22 degrees of freedom
## Multiple R-squared:  0.761,  Adjusted R-squared:  0.6197
## F-statistic: 5.387 on 13 and 22 DF,  p-value: 0.0002804
##
##
## Response sales2 :
##
## Call:
## lm(formula = sales2 ~ price1 * price2 + day + store, data = oranges)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.5004 -1.8640 -0.1674  2.7834  9.9762
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -47.88395   38.70597  -1.237  0.22909
## price1          1.69081    0.75187   2.249  0.03487 *
## price2          0.89372    0.85549   1.045  0.30751
## day2           -2.03219    3.21433  -0.632  0.53375
## day3           10.10986    3.14981   3.210  0.00404 **
## day4            3.96694    3.14910   1.260  0.22097
## day5            7.72972    3.20716   2.410  0.02475 *
## day6            5.06292    3.17196   1.596  0.12472
## store2          6.73894    3.64367   1.849  0.07787 .
## store3          2.05515    3.76835   0.545  0.59099
## store4          5.68186    3.79401   1.498  0.14845
## store5          6.40091    3.57438   1.791  0.08710 .
## store6          5.14117    3.45332   1.489  0.15075
## price1:price2  -0.03199    0.01708  -1.872  0.07450 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.265 on 22 degrees of freedom
## Multiple R-squared:  0.7783, Adjusted R-squared:  0.6473
## F-statistic:  5.94 on 13 and 22 DF,  p-value: 0.0001367
```

Using `emmeans`, we can test the difference in sales of the two orange varieties as a function of the price of the first type (`price1`).

```
#test the pairwise differences in price1 slopes between varieties.
emtrends(org.int, pairwise ~ variety, var = "price1", mult.name = "variety")
```

```
## $emtrends
##  variety price1.trend    SE df lower.CL upper.CL
##  sales1        -0.749 0.171 22   -1.104   -0.394
##  sales2         0.138 0.214 22   -0.306    0.582
##
## Results are averaged over the levels of: day, store
## Confidence level used: 0.95
##
## $contrasts
##  contrast        estimate   SE df t.ratio p.value
##  sales1 - sales2   -0.887 0.24 22  -3.690  0.0013
##
## Results are averaged over the levels of: day, store
```

Capítulo 8

Data Science r packages

```
library(caret)
library(glmnet)
library(rpart)
library(rpart.plot)
```

```
ls("package:caret")
```

```
##   [1] "anovaScores"           "avNNet"                "bag"
##   [4] "bagControl"            "bagEarth"              "bagEarthStats"
##   [7] "bagFDA"                "best"                  "BoxCoxTrans"
##  [10] "calibration"           "caretFuncs"            "caretGA"
##  [13] "caretSA"               "caretSBF"              "caretTheme"
##  [16] "cforestStats"          "checkConditionalX"     "checkInstall"
##  [19] "checkResamples"        "class2ind"             "classDist"
##  [22] "cluster"               "compare_models"        "confusionMatrix"
##  [25] "confusionMatrix.train" "contr.dummy"           "contr.ltfr"
##  [28] "createDataPartition"   "createFolds"           "createModel"
##  [31] "createMultiFolds"      "createResample"        "createTimeSlices"
##  [34] "ctreeBag"              "defaultSummary"        "dotPlot"
##  [37] "downSample"            "dummyVars"             "expandParameters"
##  [40] "expoTrans"             "extractPrediction"     "extractProb"
##  [43] "F_meas"                "featurePlot"           "filterVarImp"
##  [46] "findCorrelation"       "findLinearCombos"      "flatTable"
##  [49] "gafs"                  "gafs.default"          "gafs_initial"
##  [52] "gafs_lrSelection"      "gafs_raMutation"       "gafs_rwSelection"
##  [55] "gafs_spCrossover"      "gafs_tourSelection"    "gafs_uCrossover"
##  [58] "gafsControl"           "gamFormula"            "gamFuncs"
##  [61] "gamScores"             "getModelInfo"          "getSamplingInfo"
```

```
##  [64] "getTrainPerf"          "ggplot.gafs"           "ggplot.safs"
##  [67] "groupKFold"            "hasTerms"              "icr"
##  [70] "index2vec"             "ipredStats"            "knn3"
##  [73] "knn3Train"             "knnreg"                "knnregTrain"
##  [76] "ldaBag"                "ldaFuncs"              "ldaSBF"
##  [79] "learning_curve_dat"    "lift"                  "lmFuncs"
##  [82] "lmSBF"                 "LPH07_1"               "LPH07_2"
##  [85] "lrFuncs"               "MAE"                   "maxDissim"
##  [88] "MeanSD"                "minDiss"               "mnLogLoss"
##  [91] "modelCor"              "modelLookup"           "multiClassSummary"
##  [94] "nbBag"                 "nbFuncs"               "nbSBF"
##  [97] "nearZeroVar"           "negPredValue"          "nnetBag"
## [100] "nullModel"             "nzv"                   "oneSE"
## [103] "outcome_conversion"    "panel.calibration"     "panel.lift"
## [106] "panel.lift2"           "panel.needle"          "pcaNNet"
## [109] "pickSizeBest"          "pickSizeTolerance"     "pickVars"
## [112] "plot.gafs"             "plot.rfe"              "plot.safs"
## [115] "plot.train"            "plotClassProbs"        "plotObsVsPred"
## [118] "plsBag"                "plsda"                 "posPredValue"
## [121] "postResample"          "precision"             "predict.bagEarth"
## [124] "predict.gafs"          "predict.train"         "predictionFunction"
## [127] "predictors"            "preProcess"            "print.train"
## [130] "probFunction"          "progress"              "prSummary"
## [133] "R2"                    "recall"                "resampleHist"
## [136] "resamples"             "resampleSummary"       "resampleWrapper"
## [139] "rfe"                   "rfeControl"            "rfeIter"
## [142] "rfFuncs"               "rfGA"                  "rfSA"
## [145] "rfSBF"                 "rfStats"               "RMSE"
## [148] "safs"                  "safs_initial"          "safs_perturb"
## [151] "safs_prob"             "safsControl"           "sbf"
## [154] "sbfControl"            "sbfIter"               "sensitivity"
## [157] "SLC14_1"               "SLC14_2"               "sortImp"
## [160] "spatialSign"           "specificity"           "splsda"
## [163] "sumDiss"               "summary.bagEarth"      "svmBag"
## [166] "thresholder"           "tolerance"             "train"
## [169] "trainControl"          "treebagFuncs"          "treebagGA"
## [172] "treebagSA"             "treebagSBF"            "twoClassSim"
## [175] "twoClassSummary"       "upSample"              "var_seq"
## [178] "varImp"                "well_numbered"
```

```
ls("package:glmnet")
```

```
##  [1] "assess.glmnet"    "bigGlm"           "buildPredmat"     "Cindex"
##  [5] "coef.glmnet"      "coef.relaxed"     "confusion.glmnet" "coxgrad"
##  [9] "coxnet.deviance"  "cv.glmnet"        "glmnet"           "glmnet.control"
```

```
## [13] "glmnet.measures"  "makeX"            "na.replace"       "na_sparse_fix"
## [17] "predict.glmnet"   "predict.relaxed"  "prepareX"         "print.cv.glmnet"
## [21] "relax.glmnet"     "rmult"            "roc.glmnet"       "stratifySurv"
```

```
ls("package:rpart")
```

```
##  [1] "car.test.frame" "car90"          "cu.summary"     "kyphosis"
##  [5] "meanvar"        "na.rpart"       "path.rpart"     "plotcp"
##  [9] "post"           "printcp"        "prune"          "prune.rpart"
## [13] "rpart"          "rpart.control"  "rpart.exp"      "rsq.rpart"
## [17] "snip.rpart"     "solder"         "solder.balance" "stagec"
## [21] "xpred.rpart"
```

```
ls("package:rpart.plot")
```

```
## [1] "prp"               "ptitanic"          "rpart.plot"
## [4] "rpart.predict"     "rpart.rules"       "show.prp.palettes"
```

8.1 knn

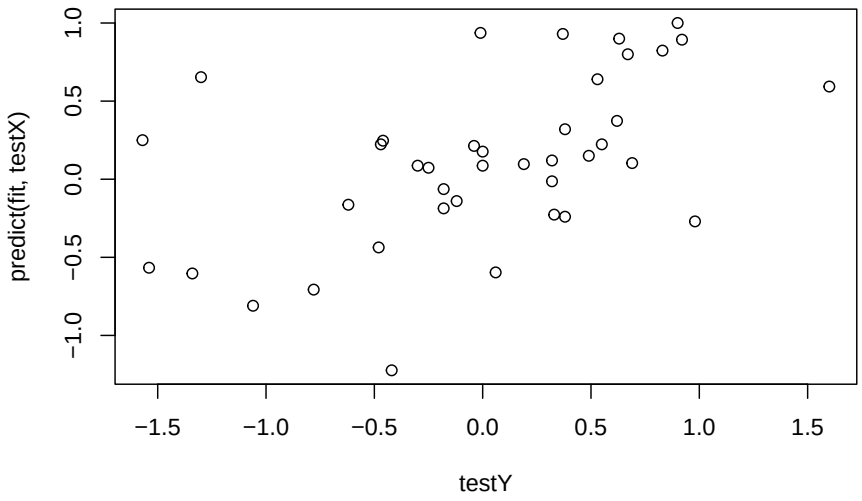
```
data(BloodBrain)

inTrain <- createDataPartition(logBBB, p = .8)[[1]]

trainX <- bbbDescr[inTrain,]
trainY <- logBBB[inTrain]

testX <- bbbDescr[-inTrain,]
testY <- logBBB[-inTrain]

fit <- knnreg(trainX, trainY, k = 3)
fit
```

```
## 3-nearest neighbor regression model
```

```
summary(fit)
```

```
##         Length Class  Mode
## learn   2      -none- list
## k       1      -none- numeric
## theDots 0      -none- list
```

```
plot(testY, predict(fit, testX))
```




```
irisFit1 <- knn3(Species ~ ., iris)
irisFit1
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##     setosa versicolor  virginica
##         50         50         50
```

```
names(irisFit1)
```

```
## [1] "learn"   "k"       "terms"   "xlevels" "theDots"
```

```
irisFit2 <- knn3(as.matrix(iris[, -5]), iris[,5])
irisFit2
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##     setosa versicolor  virginica
##         50         50         50
```

```
data(iris3)
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
knn3Train(train, test, cl, k = 5, prob = TRUE)
```

```
##  [1] "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s" "s"
## [20] "s" "s" "s" "s" "s" "s" "c" "c" "v" "c" "c" "c" "c" "c" "v" "c" "c" "c" "c"
## [39] "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "v" "c" "c" "v" "v" "v" "v"
## [58] "v" "c" "v" "v" "v" "v" "c" "v" "v" "v" "v" "v" "v" "v" "v" "v" "v" "v" "v"
## attr(,"prob")
##               c s         v
##  [1,] 0.0000000 1 0.0000000
##  [2,] 0.0000000 1 0.0000000
##  [3,] 0.0000000 1 0.0000000
##  [4,] 0.0000000 1 0.0000000
##  [5,] 0.0000000 1 0.0000000
##  [6,] 0.0000000 1 0.0000000
##  [7,] 0.0000000 1 0.0000000
##  [8,] 0.0000000 1 0.0000000
##  [9,] 0.0000000 1 0.0000000
## [10,] 0.0000000 1 0.0000000
## [11,] 0.0000000 1 0.0000000
## [12,] 0.0000000 1 0.0000000
## [13,] 0.0000000 1 0.0000000
## [14,] 0.0000000 1 0.0000000
## [15,] 0.0000000 1 0.0000000
## [16,] 0.0000000 1 0.0000000
## [17,] 0.0000000 1 0.0000000
## [18,] 0.0000000 1 0.0000000
## [19,] 0.0000000 1 0.0000000
## [20,] 0.0000000 1 0.0000000
## [21,] 0.0000000 1 0.0000000
## [22,] 0.0000000 1 0.0000000
## [23,] 0.0000000 1 0.0000000
## [24,] 0.0000000 1 0.0000000
## [25,] 0.0000000 1 0.0000000
## [26,] 1.0000000 0 0.0000000
## [27,] 1.0000000 0 0.0000000
## [28,] 0.4000000 0 0.6000000
## [29,] 1.0000000 0 0.0000000
## [30,] 1.0000000 0 0.0000000
## [31,] 1.0000000 0 0.0000000
## [32,] 1.0000000 0 0.0000000
## [33,] 1.0000000 0 0.0000000
```

```
## [34,] 0.4000000 0 0.6000000
## [35,] 0.8000000 0 0.2000000
## [36,] 1.0000000 0 0.0000000
## [37,] 1.0000000 0 0.0000000
## [38,] 1.0000000 0 0.0000000
## [39,] 1.0000000 0 0.0000000
## [40,] 1.0000000 0 0.0000000
## [41,] 1.0000000 0 0.0000000
## [42,] 1.0000000 0 0.0000000
## [43,] 1.0000000 0 0.0000000
## [44,] 1.0000000 0 0.0000000
## [45,] 1.0000000 0 0.0000000
## [46,] 1.0000000 0 0.0000000
## [47,] 1.0000000 0 0.0000000
## [48,] 1.0000000 0 0.0000000
## [49,] 1.0000000 0 0.0000000
## [50,] 1.0000000 0 0.0000000
## [51,] 0.0000000 0 1.0000000
## [52,] 0.8000000 0 0.2000000
## [53,] 0.6000000 0 0.4000000
## [54,] 0.0000000 0 1.0000000
## [55,] 0.0000000 0 1.0000000
## [56,] 0.0000000 0 1.0000000
## [57,] 0.0000000 0 1.0000000
## [58,] 0.0000000 0 1.0000000
## [59,] 0.6666667 0 0.3333333
## [60,] 0.2000000 0 0.8000000
## [61,] 0.0000000 0 1.0000000
## [62,] 0.0000000 0 1.0000000
## [63,] 0.0000000 0 1.0000000
## [64,] 0.6000000 0 0.4000000
## [65,] 0.0000000 0 1.0000000
## [66,] 0.0000000 0 1.0000000
## [67,] 0.0000000 0 1.0000000
## [68,] 0.0000000 0 1.0000000
## [69,] 0.0000000 0 1.0000000
## [70,] 0.0000000 0 1.0000000
## [71,] 0.0000000 0 1.0000000
## [72,] 0.2000000 0 0.8000000
## [73,] 0.0000000 0 1.0000000
## [74,] 0.0000000 0 1.0000000
## [75,] 0.2000000 0 0.8000000
```

8.2 rpart

```
fit <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
fit
```

```
## n= 81
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 81 17 absent (0.79012346 0.20987654)
##    2) Start>=8.5 62  6 absent (0.90322581 0.09677419)
##      4) Start>=14.5 29  0 absent (1.00000000 0.00000000) *
##      5) Start< 14.5 33  6 absent (0.81818182 0.18181818)
##       10) Age< 55 12  0 absent (1.00000000 0.00000000) *
##       11) Age>=55 21  6 absent (0.71428571 0.28571429)
##         22) Age>=111 14  2 absent (0.85714286 0.14285714) *
##         23) Age< 111 7  3 present (0.42857143 0.57142857) *
##    3) Start< 8.5 19  8 present (0.42105263 0.57894737) *
```

```
fit2 <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis,
              parms = list(prior = c(.65,.35), split = "information"))
fit2
```

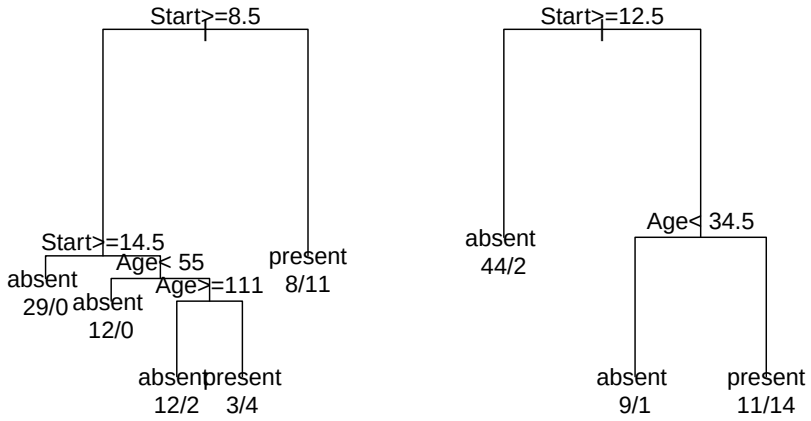
```
## n= 81
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 81 28.350000 absent (0.65000000 0.35000000)
##   2) Start>=12.5 46  3.335294 absent (0.91563089 0.08436911) *
##   3) Start< 12.5 35 16.453120 present (0.39676840 0.60323160)
##     6) Age< 34.5 10  1.667647 absent (0.81616742 0.18383258) *
##     7) Age>=34.5 25  9.049219 present (0.27932897 0.72067103) *
```

```
fit3 <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis,
              control = rpart.control(cp = 0.05))
fit3
```

```
## n= 81
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
```

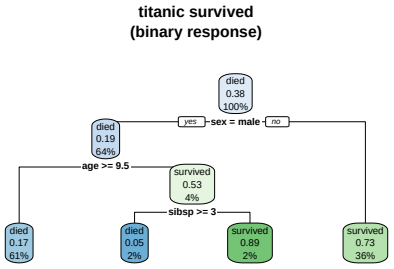
```
##
## 1) root 81 17 absent (0.79012346 0.20987654)
##   2) Start>=8.5 62  6 absent (0.90322581 0.09677419) *
##   3) Start< 8.5 19  8 present (0.42105263 0.57894737) *
```

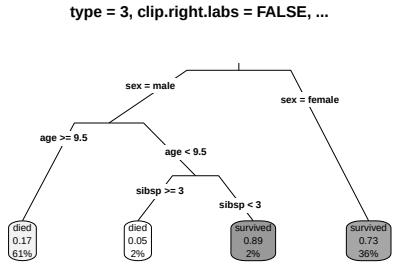
```
par(mfrow = c(1,2), xpd = NA) # otherwise on some devices the text is clipped
plot(fit)
text(fit, use.n = TRUE)
plot(fit2)
text(fit2, use.n = TRUE)
```

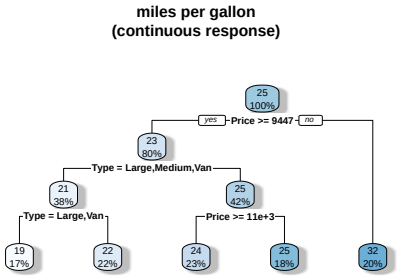


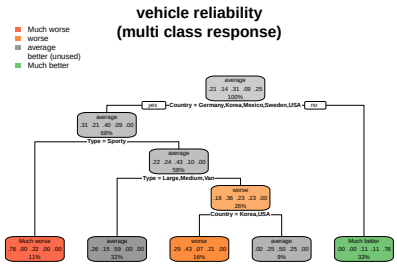

```
old.par <- par(mfrow=c(2,2))            # put 4 figures on one page

data(ptitanic)

#--------------------------------------------------------------------------

binary.model <- rpart(survived ~ ., data = ptitanic, cp = .02)
                                        # cp = .02 for small demo tree

rpart.plot(binary.model,
           main = "titanic survived\n(binary response)")

rpart.plot(binary.model, type = 3, clip.right.labs = FALSE,
```

```
           branch = .4,
           box.palette = "Grays",       # override default GnBu palette
           main = "type = 3, clip.right.labs = FALSE, ...\n")

#-------------------------------------------------------------------------------

anova.model <- rpart(Mileage ~ ., data = cu.summary)

rpart.plot(anova.model,
           shadow.col = "gray",         # add shadows just for kicks
           main = "miles per gallon\n(continuous response)\n")

#-------------------------------------------------------------------------------

multi.class.model <- rpart(Reliability ~ ., data = cu.summary)

rpart.plot(multi.class.model,
           main = "vehicle reliability\n(multi class response)")
```










```
par(old.par)
```

Referências Bibliográficas

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 Grolemund, 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.