Data Wrangling with R's tidyverse

Wilfried Cools & Lara Stas

2023-12-13

square.research.vub.be

Compiled on R 4.3.1

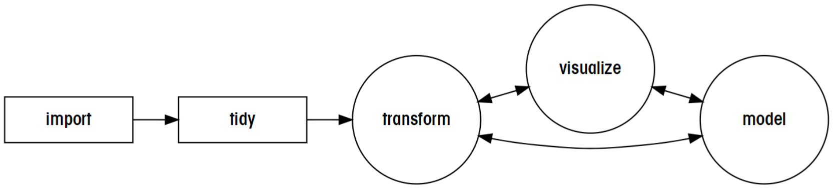
❗ What-Why-Who

This site aims to introduce researchers to data manipulation in R with the `dplyr`, `tidyr`, and `stringr` packages of the `tidyverse` ecosystem.
Our target audience is primarily the research community at VUB / UZ Brussel, those who have some basic experience in R and want to know more.
We invite you to help improve this document by sending us feedback: wilfried.cools@vub.be

Key Message

- Data manipulation can prepare data and/or its summarizing statistics
    - for modeling purposes
    - for visualisation purposes
- Data manipulation is inherent to data analysis, not just a precursor
    - no -fit's all data representation-
        - note: raw data should not be altered and kept safe
    - flexible use of data manipulation
        - supports more informative and complete modeling
        - elicits better visualisation of data and statistics
- Data manipulation is best done with coding
    - efficiently and correctly process data and statistics
    - maintain structure and transparency, to support reproducibility
- Data manipulation is easier and more intuitive when maintaining tidy data.

    - tidy data: meaning appropriately mapped into structure
        - each row an observation as research unit,
        - each column a variable as property,
        - each cell a particular value, linking row to column
        - note: data can be split into multiple tables (relational data).
    - aim for tidy data registration (avoid tedious manipulations)
- Workflow (Hadley Wickham):



R's tidyverse packages: dplyr and tidyr

- Current focus on `dplyr` and `tidyr` on manipulating and tidying data in the `tidyverse` eco-system (Hadley Wickham etal.)
- Data manipulation can be done in base R, or other packages
- `dplyr` and `tidyr`, the current defaults
    - inspired heavily on relational database logic
    - developed purposefully
        - largely consistent
        - well appreciated defaults
        - easy and intuitive to build (if you get it)
        - without loosing much flexibility
- `dplyr` and `tidyr`, part of the `tidyverse` ecosystem includes:
    - `ggplot2` for visualizing data and statistics [check Visualization]
    - `stringr` for dealing with texts
    - `forcats` for dealing with factors
    - …

Convenient cheat sheets at https://rstudio.com/resources/cheatsheets/.

Getting ahead of ourselves with dplyr

toy dataset

- The infamous `mtcars` data are used.
  - observe it's structure with `str( )` and first 6 observations `head( )` function.
  - note: available data with `data( )`
- Have a `tidyverse` look at the data with `glimpse( )`

```
glimpse(mtcars)
```

```
Rows: 32
Columns: 11
$ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
$ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
$ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
$ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
$ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,~
$ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
$ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18~
$ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,~
$ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,~
$ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
$ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

- Have a `tidyverse` look at the data with `slice_head( )`

```
mtcars %>% slice_head(n=6)
```

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

examplary data manipulation

- Get the minimum value of the ratio of `mpg` over `hp` for each combination of `am` and `cyl`.
  - take the `mtcars` data,
  - select variables `mpg`, `cyl`, `hp`, `am`, and rename `hp` to `hpow`,
  - subset rows where `hpow` bigger than 3.5,
  - create new variable `mpgr` as the ratio `mpg` on `hpow`,
  - summarize `mpgr` as the minimum for every combination of `cyl` and `am`,
  - and reshape the result into a table with one row per `cyl`-value (4,6,8) and a column for each `am` value (0,1),
  - with column variable names renamed to `am0` and `am1`.

```
mtcars %>%
    select(mpg, cyl, hpow=hp, am) %>%
    filter(hpow > 3.5) %>%
    mutate(mpgr = mpg/hpow) %>%
    group_by(cyl, am) %>%
    summarize(min=min(mpgr)) %>%
    pivot_wider(names_from=am,
        values_from=min) %>%
    select(cyl,am0=`0`,am1=`1`)
```

cyl	am0	am1
4	0.2216495	0.1963303
6	0.1560976	0.1125714
8	0.1068571	NA

`dplyr` package, functions to manipulate data

- `dplyr` reflects the `apply` function in base R
  - `d` is for data frames
- Focus on manipulating data frames (tibbles):
  - subsetting, altering, summarizing, ordering, combining, reshaping
- The main -verbs- (see example above)

  - `filter( )` : conditional selection of cases
  - `select( )` : conditional selection of variables, allows reordering and renaming
  - `mutate( )` : creation of new variables based on existing variables
  - `summarise( )` : reduce sets of values to single values

- The verb to structure data (see example above)
  - `group_by( )` : internal grouping, undo with ungroup( )
  - works preceding main verbs

- The verbs to enhance control on scope (advanced)
  - `across( )` : new way of scoping (instead of *_it, *_at, *_all)
    - works for selection in `mutate( ) and summarize( )`

- Additional `dplyr` verbs:
  - `arrange` : ordering of cases
  - `sample_n` and `sample_frac` : random sampling
  - `slice`, `transmute`, `rename`, `relocate`, ...

- Verbs to extend data
  - `bind_rows` and `bind_cols` : append data of same structure
  - `left_join`, `right_join`, `inner_join`, `full_join`, `semi_join` and `anti_join` : join data using indicator variable(s)

- Note: only the core of `dplyr` is discussed, much more is possible

group_by( )

- Grouping prepares data for group specific operations

intro

- Get a glimpse of the data as before,
  - number of rows and columns
    - in tidy data: observations and variables
  - number of groups, and grouping variables
    - 4 groups: 2 am x 2 vs
    - Note: grouping structure part of `glimpse`-output

- The width is set for presentation purposes

```
tst <- mtcars %>% group_by(am,vs)
glimpse(tst,width=40)
```

```
Rows: 32
Columns: 11
Groups: am, vs [4]
$ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 1~
$ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4~
$ disp <dbl> 160.0, 160.0, 108.0, 258.~
$ hp   <dbl> 110, 110, 93, 110, 175, 1~
$ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3~
$ wt   <dbl> 2.620, 2.875, 2.320, 3.21~
$ qsec <dbl> 16.46, 17.02, 18.61, 19.4~
$ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1~
$ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0~
$ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4~
$ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2~
```

- Actions on grouped data are grouped too,
    - e.g., a frequency table, count the number of observations (`count( )`)
    - grouped data result in grouped counts

```
tst %>% count()
```

am	vs	n
0	0	12
0	1	7
1	0	6
1	1	7

- Remove grouping with `ungroup( )`
    - this is good practice to avoid unwanted effects !

```
tst <- tst %>% ungroup( )
tst %>% count()
```

n

- Alternatively, overwrite the initial grouping
  - the last grouping is used by default
  - additional arguments, for example .add and .drop, can change that
    - a first groups by `vs`

    - a second groups by `am` and `vs`

```
mtcars %>% group_by(am) %>% group_by(vs)
mtcars %>% group_by(am) %>% group_by(vs, .add=TRUE)
```

- Transformed variables can also be used for grouping
  - e.g., cutting the `mpg` in 3 groups with `cut( )` then use `count( )` Notice the intervals that are created.

```
tst <- mtcars %>% group_by(mpg3 = cut(mpg, 3))
tst %>% count()
```

mpg3	n
<10.4,18.2]	14
<18.2,26.1]	13
<26.1,33.9]	5

exercises

- Embedded within the next sections

filter( )

- Filtering returns rows using matching conditions

intro

- Get a subset of rows that includes only those rows with `mpg` above 30

```
mtcars %>% filter(mpg > 30)
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2

- Include more than just one condition,
  - take only rows with `mpg` above 20 AND `qsec` below or equal to 18
  - note: consecutive filtering achieves the same.
    - & for `and`
    - for `or`
    - ! for `not`

```
mtcars %>% filter(mpg > 20 & qsec <= 18)
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2

- More complex conditions can be specified
  - take rows with `mpg` above 30 OR `qsec` below 20 AND `am` equal to 0
  - all the rules of logic apply, parentheses included

```
mtcars %>% filter(mpg > 30 | (qsec > 20 & am==0))
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2

33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2

- Grouping also works here
  - get all `distinct` values of `cyl` per level of `gear`
  - note: this selects the first unique rows

```
mtcars %>% group_by(gear) %>% distinct(cyl)
```

gear	cyl
4	6
4	4
3	6
3	8
3	4
5	4
5	8
5	6

exercises

- The `starwars` dataset is already part of tidyverse, so you should have it available !

```
data(starwars)
```

- Have a glimpse at the data, what data types are included ?

```
glimpse(starwars)
```

```
Rows: 87
Columns: 14
$ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Or~
$ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 180, 2~
$ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 84.0, 77.~
$ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", N~
$ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "light", "~
$ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue",~
$ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0, 57.0, ~
$ sex        <chr> "male", "none", "none", "male", "female", "male", "female",~
$ gender     <chr> "masculine", "masculine", "masculine", "masculine", "femini~
$ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "T~
$ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Huma~
$ films      <list> <"The Empire Strikes Back", "Revenge of the Sith", "Return~
$ vehicles   <list> <"Snowspeeder", "Imperial Speeder Bike">, <>, <>, <>, "Imp~
$ starships  <list> <"X-wing", "Imperial shuttle">, <>, <>, "TIE Advanced x1",~
```

- Do note, different data types are included in the tibble (data frame)
  - `chr` for characters, `int` for integers, `dbl` for doubles, we miss the `lgl` for a boolean
  - notice that even a vector of type `list` can be included.
- Filter the rows to subset the data and retain only characters with light skin and brown eye color

```
starwars %>% filter(skin_color == "light", eye_color == "brown")
```

```
# A tibble: 7 x 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Leia Org~    150    49 brown      light      brown             19 fema~ femin~
2 Biggs Da~    183    84 black      light      brown             24 male  mascu~
3 Cordé        157    NA brown      light      brown             NA fema~ femin~
4 Dormé        165    NA brown      light      brown             NA fema~ femin~
5 Raymus A~    188    79 brown      light      brown             NA male  mascu~
6 Poe Dame~     NA    NA brown      light      brown             NA male  mascu~
7 Padmé Am~    165    45 brown      light      brown             46 fema~ femin~
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Arrange the data according the character's height, largest on top ! (google it!!)

```
starwars %>% arrange(desc(height))
```

```
# A tibble: 87 x 14
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Yarael ~    264    NA none       white      yellow          NA   male  mascu~
 2 Tarfful     234   136 brown      brown      blue            NA   male  mascu~
 3 Lama Su     229    88 none       grey       black           NA   male  mascu~
 4 Chewbac~    228   112 brown      unknown    blue           200   male  mascu~
 5 Roos Ta~    224    82 none       grey       orange          NA   male  mascu~
 6 Grievous    216   159 none       brown, wh~ green, y~       NA   male  mascu~
 7 Taun We     213    NA none       grey       black           NA   fema~ femin~
 8 Rugor N~    206    NA none       green      orange          NA   male  mascu~
 9 Tion Me~    206    80 none       grey       black           NA   male  mascu~
10 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
# i 77 more rows
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Who is smallest (comes on top after arranging) ?

```
starwars %>% arrange(height)
```

```
# A tibble: 87 x 14
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Yoda         66    17 white      green      brown            896 male  mascu~
 2 Ratts T~     79    15 none       grey, blue unknown           NA male  mascu~
 3 Wicket ~     88    20 brown      brown      brown              8 male  mascu~
 4 Dud Bolt     94    45 none       blue, grey yellow            NA male  mascu~
 5 R2-D2        96    32 <NA>       white, bl~ red               33 none  mascu~
 6 R4-P17       96    NA none       silver, r~ red, blue         NA none  femin~
 7 R5-D4        97    32 <NA>       white, red red               NA none  mascu~
 8 Sebulba     112    40 none       grey, red  orange            NA male  mascu~
 9 Gasgano     122    NA none       white, bl~ black             NA male  mascu~
10 Watto       137    NA black      blue, grey yellow            NA male  mascu~
# i 77 more rows
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Slice the data and keep only the 5th to 10th observation ! (`?slice`)

```
starwars %>% slice(5:10)
```

```
# A tibble: 6 x 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Leia Org~    150    49 brown      light      brown             19 fema~ femin~
2 Owen Lars    178   120 brown, gr~ light      blue              52 male  mascu~
```

```
3 Beru Whi~    165    75 brown      light      blue              47 fema~ femin~
4 R5-D4         97    32 <NA>       white, red red               NA none  mascu~
5 Biggs Da~    183    84 black      light      brown             24 male  mascu~
6 Obi-Wan ~    182    77 auburn, w~ fair       blue-gray         57 male  mascu~
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Slice the first 2 observations for each gender (group your data) !
  - what other functions are discussed at `?slice_head` ?

```
starwars %>% group_by(gender) %>% slice_head(n=2)
```

```
# A tibble: 6 x 14
# Groups:   gender [3]
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Leia Org~    150    49 brown      light      brown             19 fema~ femin~
2 Beru Whi~    165    75 brown      light      blue              47 fema~ femin~
3 Luke Sky~    172    77 blond      fair       blue              19 male  mascu~
4 C-3PO        167    75 <NA>       gold       yellow           112 none  mascu~
5 Ric Olié     183    NA brown      fair       blue              NA <NA>  <NA>
6 Quarsh P~    183    NA black      dark       brown             62 <NA>  <NA>
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Use `slice_sample( )` to randomly select 5 observations !

```
starwars %>% slice_sample(n = 5)
```

```
# A tibble: 5 x 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Owen Lars    178   120 brown, gr~ light      blue              52 male  mascu~
2 Mace Win~    188    84 none       dark       brown             72 male  mascu~
3 Ki-Adi-M~    198    82 white      pale       yellow            92 male  mascu~
4 Beru Whi~    165    75 brown      light      blue              47 fema~ femin~
5 Bail Pre~    191    NA black      tan        brown             67 male  mascu~
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Use `slice_max( )` to select 3 observations with highest values on `height` !

```
starwars %>% slice_max(height, n = 3)
```

```
# A tibble: 3 x 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Yarael P~    264    NA none       white      yellow            NA male  mascu~
2 Tarfful      234   136 brown      brown      blue              NA male  mascu~
3 Lama Su      229    88 none       grey       black             NA male  mascu~
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Get the top 3 (highest mass) for each species !
  - ignore characters with missing data for `mass`
  - note, -not missing- are those who are not `!` missing `is.na()`

```
starwars %>% group_by(species) %>% filter(!is.na(mass)) %>% slice_max(mass, n = 3)
```

```
# A tibble: 40 x 14
# Groups:   species [32]
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Ratts T~     79    15 none       grey, blue unknown           NA male  mascu~
 2 Dexter ~    198   102 none       brown      yellow            NA male  mascu~
 3 Ki-Adi-~    198    82 white      pale       yellow            92 male  mascu~
 4 Zam Wes~    168    55 blonde     fair, gre~ yellow            NA fema~ femin~
 5 IG-88       200   140 none       metal      red               15 none  mascu~
 6 C-3PO       167    75 <NA>       gold       yellow           112 none  mascu~
 7 R2-D2        96    32 <NA>       white, bl~ red               33 none  mascu~
 8 R5-D4        97    32 <NA>       white, red red               NA none  mascu~
 9 Sebulba     112    40 none       grey, red  orange            NA male  mascu~
10 Wicket ~     88    20 brown      brown      brown              8 male  mascu~
# i 30 more rows
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

select( )

- Extract columns (variables) by name (or position), rename and/or reorder them

intro

- Select the variable `mpg`
  - notice that even with one column, the result remains a dataframe (not a vector), this is tidyverse policy !
- An operation on a data with a certain type should result in data of the same type.
  - if you take one column from a matrix you have a one column matrix, not a vector.
  - if you take one column from a data frame, again, you end up with a one-column data frame, not a vector.

```
mtcars %>% select(mpg)
```

mpg
21.0
21.0
22.8
21.4
18.7
18.1

- To retrieve a vector with `dplyr` use `pull( )`

- Specific operations allow for changing the data types

```
mtcars %>% pull(mpg)
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
[31] 15.0 21.4
```

- Extract columns `qsec` and `mpg` (top 6 observations)
  - note: more than one column can be considered jointly, their order is specified as such

```
mtcars %>% select(qsec,mpg)
```

qsec	mpg
16.46	21.0
17.02	21.0
18.61	22.8
19.44	21.4
17.02	18.7
20.22	18.1

- Extract the third and first column (top 6)
  - note: columns can be extracted by their position

```
mtcars %>% select(3,1)
```

disp	mpg
160	21.0
160	21.0
108	22.8
258	21.4
360	18.7
225	18.1

- Remove columns at third to sixth position (top 6)
  - note: to remove, use a negation, but it is either keep or remove not both

```
mtcars %>% select(-c(3:6))
```

mpg	cyl	qsec	vs	am	gear	carb
21.0	6	16.46	0	1	4	4
21.0	6	17.02	0	1	4	4
22.8	4	18.61	1	1	4	1
21.4	6	19.44	1	0	3	1
18.7	8	17.02	0	0	3	2
18.1	6	20.22	1	0	3	1

- **helper functions** can facilitate selections
- Use partial string matching with `contains( )`

  - extract columns with names that include the string `ar` (show 6)

```
mtcars %>% select(contains('ar'))
```

gear	carb
4	4
4	4
4	1
3	1
3	2
3	1

- Use regular expressions with `matches( )`
  - extract columns with names that include the string `ar` but with at least one element before and after it

```
mtcars %>% select(matches('.ar.'))
```

carb
4
4
1
1
2
1

- Variables can be renamed during selection
  - rename the `cyl` into `cyl468` to reflect its values
  - same for `vs` and `am`, and select it together with `mpg`

```
mtcars %>% select(mpg,cyl468=cyl,vs01=vs,am01=am)
```

mpg	cyl468	vs01	am01
21.0	6	0	1
21.0	6	0	1
22.8	4	1	1
21.4	6	1	0

18.7	8	0	0
18.1	6	1	0

- Rename the `cyl`, `vs` and `am` directly

```
mtcars %>% rename(cyl468=cyl,vs01=vs,am01=am)
```

mpg	cyl468	disp	hp	drat	wt	qsec	vs01	am01	gear	carb
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

- Note that a `select( )` will include the grouping variables by default

- Grouping variables are identified with `group_cols( )`

- Create a grouping by `vs` and `am`, and extract only those columns

```
mtcars %>% group_by(vs,am) %>% select(group_cols( ))
```

0 - 0
0 - 1
1 - 0
1 - 1

exercises

- The `starwars` dataset is probably still loaded into your workspace !

```
data(starwars)
```

- Select the columns hair, skin and eye color !

```
starwars %>% select(hair_color, skin_color, eye_color)
```

```
# A tibble: 87 x 3
   hair_color    skin_color  eye_color
   <chr>         <chr>       <chr>
 1 blond         fair        blue
 2 <NA>          gold        yellow
 3 <NA>          white, blue red
 4 none          white       yellow
 5 brown         light       brown
 6 brown, grey   light       blue
 7 brown         light       blue
 8 <NA>          white, red  red
 9 black         light       brown
10 auburn, white fair        blue-gray
# i 77 more rows
```

- Use the `:` operator for consecutive columns hair and eye color !

```
starwars %>% select(hair_color:eye_color)
```

```
# A tibble: 87 x 3
   hair_color    skin_color  eye_color
   <chr>         <chr>       <chr>
 1 blond         fair        blue
 2 <NA>          gold        yellow
 3 <NA>          white, blue red
 4 none          white       yellow
 5 brown         light       brown
 6 brown, grey   light       blue
 7 brown         light       blue
 8 <NA>          white, red  red
 9 black         light       brown
10 auburn, white fair        blue-gray
# i 77 more rows
```

- Remove these columns instead of selecting them !

```
starwars %>% select(-(hair_color:eye_color))
```

```
# A tibble: 87 x 11
   name    height  mass birth_year sex   gender homeworld species films vehicles
   <chr>    <int> <dbl>      <dbl> <chr> <chr>  <chr>     <chr>   <lis> <list>
 1 Luke S~    172    77       19   male  mascu~ Tatooine  Human   <chr> <chr>
 2 C-3PO      167    75      112   none  mascu~ Tatooine  Droid   <chr> <chr>
 3 R2-D2       96    32       33   none  mascu~ Naboo     Droid   <chr> <chr>
 4 Darth ~    202   136       41.9 male  mascu~ Tatooine  Human   <chr> <chr>
 5 Leia O~    150    49       19   fema~ femin~ Alderaan  Human   <chr> <chr>
 6 Owen L~    178   120       52   male  mascu~ Tatooine  Human   <chr> <chr>
 7 Beru W~    165    75       47   fema~ femin~ Tatooine  Human   <chr> <chr>
 8 R5-D4       97    32       NA   none  mascu~ Tatooine  Droid   <chr> <chr>
 9 Biggs ~    183    84       24   male  mascu~ Tatooine  Human   <chr> <chr>
10 Obi-Wa~    182    77       57   male  mascu~ Stewjon   Human   <chr> <chr>
# i 77 more rows
# i 1 more variable: starships <list>
```

- Select all columns with a name ending with `color` (check help files on helper functions, use ?language) !

```
starwars %>% select(ends_with("color"))
```

```
# A tibble: 87 x 3
   hair_color    skin_color  eye_color
   <chr>         <chr>       <chr>
 1 blond         fair        blue
 2 <NA>          gold        yellow
 3 <NA>          white, blue red
 4 none          white       yellow
 5 brown         light       brown
 6 brown, grey   light       blue
 7 brown         light       blue
 8 <NA>          white, red  red
 9 black         light       brown
10 auburn, white fair        blue-gray
# i 77 more rows
```

- Use select to rename `homeworld` to `home_world` !

```
starwars %>% select(home_world = homeworld)
```

```
# A tibble: 87 x 1
   home_world
   <chr>
 1 Tatooine
 2 Tatooine
 3 Naboo
 4 Tatooine
 5 Alderaan
 6 Tatooine
 7 Tatooine
 8 Tatooine
 9 Tatooine
10 Stewjon
# i 77 more rows
```

- Do the same with the `rename( )` function !

```
starwars %>% rename(home_world = homeworld)
```

```
# A tibble: 87 x 14
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
 2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
 3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
 4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
 5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
 6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
 7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
 8 R5-D4        97    32 <NA>       white, red red             NA   none  mascu~
 9 Biggs D~    183    84 black      light      brown           24   male  mascu~
10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male  mascu~
# i 77 more rows
# i 5 more variables: home_world <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Select only the numeric variables, use `where( )` and `is.numeric( )` !
  - Maybe check the ?language again

```
starwars %>% select(where(is.numeric))
```

```
# A tibble: 87 x 3
   height  mass birth_year
    <int> <dbl>      <dbl>
 1    172    77       19
 2    167    75      112
 3     96    32       33
 4    202   136       41.9
 5    150    49       19
 6    178   120       52
 7    165    75       47
 8     97    32       NA
 9    183    84       24
10    182    77       57
# i 77 more rows
```

- Select only those variables with names `height`, `mass` and/or `size`, with `any_of( )` !

```
starwars %>% select(any_of(c('height','mass','size')))
```

```
# A tibble: 87 x 2
   height  mass
    <int> <dbl>
 1    172    77
 2    167    75
 3     96    32
 4    202   136
 5    150    49
 6    178   120
 7    165    75
 8     97    32
 9    183    84
10    182    77
# i 77 more rows
```

mutate( )

- Create new variables based on existing ones

intro

- A new variable (column) mpg2 can be created by mpg value squared

```
mtcars %>% mutate(mpg2=mpg^2)
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	mpg2
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	441.00
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	441.00
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	519.84
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	457.96
18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	349.69
18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	327.61

- The original value can also be overwritten
  - e.g., the mpg can be assigned the values of mpg squared

```
mtcars %>% mutate(mpg=mpg^2)
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
441.00	6	160	110	3.90	2.620	16.46	0	1	4	4
441.00	6	160	110	3.90	2.875	17.02	0	1	4	4
519.84	4	108	93	3.85	2.320	18.61	1	1	4	1
457.96	6	258	110	3.08	3.215	19.44	1	0	3	1
349.69	8	360	175	3.15	3.440	17.02	0	0	3	2
327.61	6	225	105	2.76	3.460	20.22	1	0	3	1

- Based on multiple variables, e.g., NEWVAR can represent the mpg value multiplied by the vs value

- The convenient everything function is a short-cut to every column not explicitely mentioned

```
mtcars %>% mutate(NEWVAR=mpg*vs) %>% select(NEWVAR,everything())
```

NEWVAR	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0.0	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
0.0	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
22.8	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1

21.4	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
0.0	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
18.1	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

- A new variable can be created based on a newly created variable as well
  - e.g., `NEWVAR` is the `mpg` value multiplied by the `vs` value and this new variable is divided by the `disp` variable

```
mtcars %>% mutate(NEWVAR=mpg*vs,NEWVAR2=NEWVAR/disp) %>% select(NEWVAR,NEWVAR2,everything(
```

NEWVAR	NEWVAR2	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0.0	0.00000000	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
0.0	0.00000000	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
22.8	0.21111111	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
21.4	0.08294574	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
0.0	0.00000000	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
18.1	0.08044444	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

- ***window functions*** facilitate the automation of mutations (google for dplyr window functions).
  - e.g., add a column with the cumulative sum of `mpg` using `cumsum( )`

```
mtcars %>% mutate(NEWVAR=cumsum(mpg))
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	NEWVAR
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	21.0
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	42.0
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	64.8
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	86.2
18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	104.9
18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	123.0

- Add a column with indicator whether the `mpg` is between 20 and 22

```
mtcars %>% mutate(NEWVAR=between(mpg,20,22)) %>% select(NEWVAR,everything())
```

NEWVAR	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
TRUE	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
TRUE	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
FALSE	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
TRUE	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
FALSE	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
FALSE	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

- Add a row number dependent on the rank of `mpg` values, with the `rownumber( )` function. When arranged by `mpg` this is more clear.

```
mtcars %>% mutate(id=row_number(mpg)) %>% select(id,everything())
mtcars %>% mutate(id=row_number(mpg)) %>% arrange(mpg) %>% select(id,everything())
```

id	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	10.4	8	472	205	2.93	5.250	17.98	0	0	3	4
2	10.4	8	460	215	3.00	5.424	17.82	0	0	3	4
3	13.3	8	350	245	3.73	3.840	15.41	0	0	3	4
4	14.3	8	360	245	3.21	3.570	15.84	0	0	3	4
5	14.7	8	440	230	3.23	5.345	17.42	0	0	3	4
6	15.0	8	301	335	3.54	3.570	14.60	0	1	5	8

- Grouping variables can group the operations

- To create a ranking within groups, `vs` and `am`, `row_number( )` can be used again
  - Notice, for each combination of `vs` and `am`, there will be a 1 (first), 2 (second)... for `id` (not all shown, only 6 per combination are shown).

```
mtcars %>% group_by(vs,am) %>% mutate(id=row_number(mpg))
```

mpg	cyl	disp	hp	drat	wt	qsec	gear	carb	id
0 - 0									
18.7	8	360.0	175	3.15	3.440	17.02	3	2	11
14.3	8	360.0	245	3.21	3.570	15.84	3	4	4
16.4	8	275.8	180	3.07	4.070	17.40	3	3	9
17.3	8	275.8	180	3.07	3.730	17.60	3	3	10
15.2	8	275.8	180	3.07	3.780	18.00	3	3	6

10.4	8	472.0	205	2.93	5.250	17.98	3	4	1
0 - 1									
21.0	6	160.0	110	3.90	2.620	16.46	4	4	4
21.0	6	160.0	110	3.90	2.875	17.02	4	4	5
26.0	4	120.3	91	4.43	2.140	16.70	5	2	6
15.8	8	351.0	264	4.22	3.170	14.50	5	4	2
19.7	6	145.0	175	3.62	2.770	15.50	5	6	3
15.0	8	301.0	335	3.54	3.570	14.60	5	8	1
1 - 0									
21.4	6	258.0	110	3.08	3.215	19.44	3	1	4
18.1	6	225.0	105	2.76	3.460	20.22	3	1	2
24.4	4	146.7	62	3.69	3.190	20.00	4	2	7
22.8	4	140.8	95	3.92	3.150	22.90	4	2	6
19.2	6	167.6	123	3.92	3.440	18.30	4	4	3
17.8	6	167.6	123	3.92	3.440	18.90	4	4	1
1 - 1									
22.8	4	108.0	93	3.85	2.320	18.61	4	1	2
32.4	4	78.7	66	4.08	2.200	19.47	4	1	6
30.4	4	75.7	52	4.93	1.615	18.52	4	2	4
33.9	4	71.1	65	4.22	1.835	19.90	4	1	7
27.3	4	79.0	66	4.08	1.935	18.90	4	1	3
30.4	4	95.1	113	3.77	1.513	16.90	5	2	5

exercises

- For the starwars data, create a new variable `height_m` with `height` divided by 100 !

```
starwars %>% mutate(height_m = height / 100) %>% select(height_m, height, everything( ))
```

```
# A tibble: 87 x 15
   height_m height name    mass hair_color skin_color eye_color birth_year sex
      <dbl>  <int> <chr>  <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1     1.72    172 Luke ~    77 blond      fair       blue            19   male
 2     1.67    167 C-3PO     75 <NA>       gold       yellow         112   none
 3     0.96     96 R2-D2     32 <NA>       white, bl~ red             33   none
 4     2.02    202 Darth~   136 none       white      yellow          41.9 male
 5     1.5     150 Leia ~    49 brown      light      brown           19   fema~
 6     1.78    178 Owen ~   120 brown, gr~ light      blue            52   male
 7      1.65    165 Beru ~    75 brown      light      blue            47    fema~
 8      0.97     97 R5-D4     32 <NA>       white, red red             NA    none
 9      1.83    183 Biggs~    84 black      light      brown           24    male
10      1.82    182 Obi-W~    77 auburn, w~ fair       blue-gray       57    male
# i 77 more rows
# i 6 more variables: gender <chr>, homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>
```

- Note, the `select` function also defines the order, again the `everything` function avoids explicitely naming all variables
- Create the same new variable, but also define BMI as `mass` / `height_m` to the power 2 !

```r
starwars %>% mutate(height_m = height / 100, BMI = mass / (height_m^2)) %>% select(BMI, ev
```

```
# A tibble: 87 x 16
     BMI name      height  mass hair_color skin_color eye_color birth_year sex
   <dbl> <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1  26.0 Luke Sky~    172    77 blond      fair       blue             19    male
 2  26.9 C-3PO        167    75 <NA>       gold       yellow          112    none
 3  34.7 R2-D2         96    32 <NA>       white, bl~ red              33    none
 4  33.3 Darth Va~    202   136 none       white      yellow           41.9 male
 5  21.8 Leia Org~    150    49 brown      light      brown            19    fema~
 6  37.9 Owen Lars    178   120 brown, gr~ light      blue             52    male
 7  27.5 Beru Whi~    165    75 brown      light      blue             47    fema~
 8  34.0 R5-D4         97    32 <NA>       white, red red              NA    none
 9  25.1 Biggs Da~    183    84 black      light      brown            24    male
10  23.2 Obi-Wan ~    182    77 auburn, w~ fair       blue-gray        57    male
# i 77 more rows
# i 7 more variables: gender <chr>, homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>, height_m <dbl>
```

- Use `transmute` to repeat the above mutation but keep only `height_m` and `BMI` !

```r
starwars %>% transmute(height_m = height / 100, BMI = mass / (height_m^2))
```

```
# A tibble: 87 x 2
   height_m   BMI
      <dbl> <dbl>
 1     1.72  26.0
 2     1.67  26.9
 3     0.96  34.7
 4     2.02  33.3
 5     1.5   21.8
 6     1.78  37.9
 7     1.65  27.5
 8     0.97  34.0
 9     1.83  25.1
10     1.82  23.2
# i 77 more rows
```

- Create a new variable with the z-score of height (zcore = (value-mean)/sd) !

```
starwars %>% mutate(zscore=(height-mean(height,na.rm=T))/sd(height,na.rm=T)) %>% select(zs
```

```
# A tibble: 87 x 15
    zscore name    height  mass hair_color skin_color eye_color birth_year sex
     <dbl> <chr>    <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 -0.0678 Luke S~    172    77 blond      fair       blue              19   male
 2 -0.212  C-3PO      167    75 <NA>       gold       yellow           112   none
 3 -2.25   R2-D2       96    32 <NA>       white, bl~ red               33   none
 4  0.795  Darth ~    202   136 none       white      yellow            41.9 male
 5 -0.701  Leia O~    150    49 brown      light      brown             19   fema~
 6  0.105  Owen L~    178   120 brown, gr~ light      blue              52   male
 7 -0.269  Beru W~    165    75 brown      light      blue              47   fema~
 8 -2.22   R5-D4       97    32 <NA>       white, red red               NA   none
 9  0.249  Biggs ~    183    84 black      light      brown             24   male
10  0.220  Obi-Wa~    182    77 auburn, w~ fair       blue-gray         57   male
# i 77 more rows
# i 6 more variables: gender <chr>, homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>
```

- Now create that z-score per species !

```
starwars %>% group_by(species) %>% mutate(zscore=(height-mean(height,na.rm=T))/sd(height,n
```

```
# A tibble: 87 x 15
# Groups:   species [38]
   zscore species height name    mass hair_color skin_color eye_color birth_year
    <dbl> <chr>    <int> <chr>  <dbl> <chr>      <chr>      <chr>          <dbl>
 1 -0.371 Human      172 Luke ~    77 blond      fair       blue             19
 2  0.728 Droid      167 C-3PO     75 <NA>       gold       yellow         112
 3 -0.716 Droid       96 R2-D2     32 <NA>       white, bl~ red             33
 4  2.02  Human      202 Darth~   136 none       white      yellow          41.9
 5 -2.13  Human      150 Leia ~    49 brown      light      brown           19
 6  0.108 Human      178 Owen ~   120 brown, gr~ light      blue            52
 7 -0.929 Human      165 Beru ~    75 brown      light      blue            47
 8 -0.696 Droid       97 R5-D4     32 <NA>       white, red red             NA
 9  0.507 Human      183 Biggs~    84 black      light      brown           24
10  0.427 Human      182 Obi-W~    77 auburn, w~ fair       blue-gray       57
# i 77 more rows
# i 6 more variables: sex <chr>, gender <chr>, homeworld <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Create a gender indicator that replaces the `male` and `female` labels with `m` and `f` (use `recode( )`) !

```
starwars %>% mutate(new_value=recode(sex,'male'='m','female'='f')) %>% select(new_value, s
```

```
# A tibble: 87 x 15
   new_value sex   name  height  mass hair_color skin_color eye_color birth_year
   <chr>     <chr> <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl>
 1 m         male  Luke~    172    77 blond      fair       blue            19
 2 none      none  C-3PO    167    75 <NA>       gold       yellow         112
 3 none      none  R2-D2     96    32 <NA>       white, bl~ red             33
 4 m         male  Dart~    202   136 none       white      yellow          41.9
 5 f         fema~ Leia~    150    49 brown      light      brown           19
 6 m         male  Owen~    178   120 brown, gr~ light      blue            52
 7 f         fema~ Beru~    165    75 brown      light      blue            47
 8 none      none  R5-D4     97    32 <NA>       white, red red             NA
 9 m         male  Bigg~    183    84 black      light      brown           24
10 m         male  Obi--~   182    77 auburn, w~ fair       blue-gray       57
# i 77 more rows
# i 6 more variables: gender <chr>, homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>
```

- Create a gender indicator that, when sex is ‘none’ uses the species values and otherwise keeps the sex specification (use `ifelse( )`))!

```
starwars %>% mutate(new_value=ifelse(sex=='none',species,sex)) %>% select(new_value, speci
# A tibble: 87 x 15
   new_value species sex    name     height  mass hair_color skin_color eye_color
   <chr>     <chr>   <chr>  <chr>     <int> <dbl> <chr>      <chr>      <chr>
 1 male      Human   male   Luke S~     172    77 blond      fair       blue
 2 Droid     Droid   none   C-3PO       167    75 <NA>       gold       yellow
 3 Droid     Droid   none   R2-D2        96    32 <NA>       white, bl~ red
 4 male      Human   male   Darth ~     202   136 none       white      yellow
 5 female    Human   female Leia O~     150    49 brown      light      brown
 6 male      Human   male   Owen L~     178   120 brown, gr~ light      blue
 7 female    Human   female Beru W~     165    75 brown      light      blue
 8 Droid     Droid   none   R5-D4        97    32 <NA>       white, red red
 9 male      Human   male   Biggs ~     183    84 black      light      brown
10 male      Human   male   Obi-Wa~     182    77 auburn, w~ fair       blue-gray
# i 77 more rows
# i 6 more variables: birth_year <dbl>, gender <chr>, homeworld <chr>,
#   films <list>, vehicles <list>, starships <list>
```

summarize( )

- Reduce sets of values into their summaries, based on grouped data.

intro

- A new variable (column) is created based on an existing one by summarizing, condensing the data
  - e.g., the mean of all `mpg` values can be obtained

```
mtcars %>% summarize(myAverage=mean(mpg))
```

myAverage
20.09062

- Multiple summaries can be obtained jointly, the mean and standard deviation of all `mpg` values can be obtained
  - you could do that for multiple variables, and also include `disp`

```
mtcars %>% summarize(myAvMpg=mean(mpg),mySdMpg=sd(mpg),myAvDisp=mean(disp),mySdDisp=sd(dis
```

myAvMpg	mySdMpg	myAvDisp	mySdDisp
20.09062	6.026948	230.7219	123.9387

- Grouping variables are very natural to use with `summarize( )`
- The mean of all `mpg` values can be obtained for each level of `vs`

```
mtcars %>% group_by(vs) %>% summarize(myAverage=mean(mpg))
```

vs	myAverage
0	16.61667
1	24.55714

- The mean and standard deviation can be obtained for multiple variables too,
  - e.g., an average and standard deviation of `mpg` and `disp` for each group

```
mtcars %>% group_by(vs,am) %>% summarize(myAvMpg=mean(mpg),mySdMpg=sd(mpg),myAvDisp=mean(d
```

am	myAvMpg	mySdMpg	myAvDisp	mySdDisp
0				
0	15.05000	2.774396	357.6167	71.82349
1	19.75000	4.008865	206.2167	95.23362
1				
0	20.74286	2.471071	175.1143	49.13072
1	28.37143	4.757701	89.8000	18.80213

- The total number of observations within a group
  - e.g., `vs`, can be obtained with `n( )`, or using the special verb `count( )`

```
mtcars %>% group_by(vs) %>% count( )
mtcars %>% group_by(vs) %>% summarize(mycount=n( ))
```

vs	mycount
0	18
1	14

- Making use of **summary functions**, summarizing can be more automated
  - e.g., the number of distinct values in a vector for each combination of `vs` and `am` can be obtained with `n_distinct( )`, and the third number of each group with can be obtained with `nth( )`

```
mtcars %>% group_by(vs,am) %>% summarize(nrDist=n_distinct(mpg),`3th`=nth(mpg,3))
```

am	nrDist	3th
0		
0	10	16.4
1	5	26.0
1		
0	7	24.4
1	6	30.4

exercises

- Summarize the `height` into the average height (some missing values need to be dealt with, check `?mean`) !

```
starwars %>% summarise(height = mean(height, na.rm = TRUE))
```

```
# A tibble: 1 x 1
  height
   <dbl>
1   174.
```

- Repeat the above, but include the average `mass` per `species` and `sex` !

```
starwars %>% group_by(species,sex) %>% summarize(height = mean(height, na.rm = TRUE), mass
# A tibble: 41 x 4
# Groups:   species [38]
   species   sex    height  mass
   <chr>     <chr>   <dbl> <dbl>
 1 Aleena    male      79   15
 2 Besalisk  male     198  102
 3 Cerean    male     198   82
 4 Chagrian  male     196  NaN
 5 Clawdite  female   168   55
 6 Droid     none     131.  69.8
 7 Dug       male     112   40
 8 Ewok      male      88   20
 9 Geonosian male     183   80
10 Gungan    male     209.  74
# i 31 more rows
```

across( )

- Scoping a verb, the `across( )` function allows for `summarize( )` or `mutate( )` operations on a set of variables

intro

- Select variables by either explicitly naming them or by extraction using dedicated functions
- Turn to the `mtcars` data again, take a `glimpse` to remind yourself what the data looks like
- Select `mpg`, `cyl`, `am` and `vs`
- Turn both `am` and `vs` into a factor before calling the structure with `glimpse( )`

```
mtcars %>% select(mpg,cyl,am,vs) %>% mutate(across(c('am','vs'),factor)) %>% glimpse( )
```

```
Rows: 32
Columns: 4
$ mpg <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8, ~
$ cyl <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8, ~
$ am  <fct> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, ~
$ vs  <fct> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, ~
```

- A factor is made from all variables in between `cyl` and `vs` with a `:` operator

```
mtcars %>% select(mpg,cyl,am,vs) %>% mutate(across(cyl:vs,factor)) %>% glimpse( )
```

```
Rows: 32
Columns: 4
$ mpg <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8, ~
$ cyl <fct> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8, ~
$ am  <fct> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, ~
$ vs  <fct> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, ~
```

- A factor is made from all variables that contain the letter combination `ar`

```
mtcars %>% select(mpg,cyl,gear,carb) %>% mutate(across(contains("ar"),factor)) %>% glimpse
```

```
Rows: 32
Columns: 4
$ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
$ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
$ gear <fct> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
$ carb <fct> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

- The `accross( )` function allows for applying a list of functions
  - e.g., for the first and third variable, a function is applied to obtain a median, a mean and an sd
  - use is made of `tidyverse` short-cuts `~` to indicate a function is used and `.x` that works as a container for the variables used in that function.

```
descr <- list(
  md = ~median(.x, na.rm = TRUE),
  av = ~mean(.x, na.rm = TRUE),
  sd = ~sd(.x, na.rm = TRUE)
)
mtcars %>% mutate(across(c(1,3), descr))
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	mpg_md	mpg_av	mpg_sd	disp_
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	19.2	20.09062	6.026948	1
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	19.2	20.09062	6.026948	1
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	19.2	20.09062	6.026948	1
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	19.2	20.09062	6.026948	1
18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	19.2	20.09062	6.026948	1
18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	19.2	20.09062	6.026948	1

- Making use of **helper functions**
  - the same as for `select( )`, selections can be more automated
- Helper functions include among others `all_of( )`, `where( )`, `matches( )`, `starts_with( )`
  - are possible to use within `mutate( )` and `summarize( )`

```
descr <- list(
  md = ~median(.x, na.rm = TRUE),
  av = ~mean(.x, na.rm = TRUE),
  sd = ~sd(.x, na.rm = TRUE)
)
mtcars %>% mutate(across(c(1,3), descr))
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	mpg_md	mpg_av	mpg_sd	disp_
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	19.2	20.09062	6.026948	1
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	19.2	20.09062	6.026948	1
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	19.2	20.09062	6.026948	1
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	19.2	20.09062	6.026948	1
18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	19.2	20.09062	6.026948	1
18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	19.2	20.09062	6.026948	1

exercises

- For starwars data, request the minimum and maximum values of the numeric variables (use `where( )`).
  - Note that you need to deal with missing values

```
min_max <- list(
  min = ~min(.x, na.rm = TRUE),
  max = ~max(.x, na.rm = TRUE)
)
starwars %>% summarise(across(where(is.numeric), min_max))
```

```
# A tibble: 1 x 6
  height_min height_max mass_min mass_max birth_year_min birth_year_max
       <int>      <int>    <dbl>    <dbl>          <dbl>          <dbl>
1         66        264       15     1358              8            896
```

join( )

- Datafiles can be combined using common variables that serve as key (cfr. relational databases).

intro

- Methods differ primarily in how they deal with mismatches in key variable values
- Assume a cylinder specific datafile, `mtcyl`, with a 2 cylinder but no 8 cylinder unlike the `mtcars` (4,6,8)

```
mtcyl <- tribble(
~cyl,~type,
2,'small',
4,'medium',
6,'large'
)
```

- Combine the `mtcars` and `mtcyl` but ignore the irrelevant `cyl` equal to 2 (not part of `mtcars`), with a `left_join( )`
  - Notice that `cyl` equal to 8 turns out missing, because it is not specified in the -right- datafile (`mtcyl`)

```
mtcars %>% left_join(mtcyl) %>% select(cyl,type,everything())
```

```
Joining with ‘by = join_by(cyl)‘
```

cyl	type	mpg	disp	hp	drat	wt	qsec	vs	am	gear	carb
6	large	21.0	160	110	3.90	2.620	16.46	0	1	4	4
6	large	21.0	160	110	3.90	2.875	17.02	0	1	4	4
4	medium	22.8	108	93	3.85	2.320	18.61	1	1	4	1
6	large	21.4	258	110	3.08	3.215	19.44	1	0	3	1
8	NA	18.7	360	175	3.15	3.440	17.02	0	0	3	2
6	large	18.1	225	105	2.76	3.460	20.22	1	0	3	1

- Combine the `mtcars` and `mtcyl` but ignore the `cyl` equal to 8 because it lacks information on type, with a `right_join( )`
  - Notice that `cyl` equal to 2 is included, but turns out missing for most variables because it is not specified in the -left- datafile

```
mtcars %>% right_join(mtcyl) %>% arrange(cyl) %>% select(cyl,type,everything())
```

```
Joining with ‘by = join_by(cyl)‘
```

cyl	type	mpg	disp	hp	drat	wt	qsec	vs	am	gear	carb
2	small	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	medium	22.8	108.0	93	3.85	2.320	18.61	1	1	4	1
4	medium	24.4	146.7	62	3.69	3.190	20.00	1	0	4	2
4	medium	22.8	140.8	95	3.92	3.150	22.90	1	0	4	2
4	medium	32.4	78.7	66	4.08	2.200	19.47	1	1	4	1
4	medium	30.4	75.7	52	4.93	1.615	18.52	1	1	4	2

- Combine the `mtcars` and `mtcyl` for only those observations with the linking variable `cyl` in both files, with a `right_join( )`
  - Notice no missing values, but some data is not included

```
mtcars %>% inner_join(mtcyl) %>% arrange(cyl) %>% select(cyl,type,everything())
```

```
Joining with ‘by = join_by(cyl)‘
```

cyl	type	mpg	disp	hp	drat	wt	qsec	vs	am	gear	carb
4	medium	22.8	108.0	93	3.85	2.320	18.61	1	1	4	1

4	medium	24.4	146.7	62	3.69	3.190	20.00	1	0	4	2
4	medium	22.8	140.8	95	3.92	3.150	22.90	1	0	4	2
4	medium	32.4	78.7	66	4.08	2.200	19.47	1	1	4	1
4	medium	30.4	75.7	52	4.93	1.615	18.52	1	1	4	2
4	medium	33.9	71.1	65	4.22	1.835	19.90	1	1	4	1

- Combine the `mtcars` and `mtcyl` keeping all available information, with a `full_join( )` showing selected rows 1 to 3, 5, 7 and 33

```
mtcars %>% full_join(mtcyl) %>% slice(c(1:3,5,7,33))
```

```
Joining with ‘by = join_by(cyl)‘
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	type
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	large
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	large
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	medium
18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	NA
14.3	8	360	245	3.21	3.570	15.84	0	0	3	4	NA
NA	2	NA	NA	NA	NA	NA	NA	NA	NA	NA	small

- Other types of join exist, like `semi_join( )`, `nest_join( )`, `anti_join()`, which are described in the help files.

exercises

- Two mini tibbles `band_members` and `band_instruments` are probably loaded into your workspace automatically as part of the tidyverse !

```
band_members
```

```
# A tibble: 3 x 2
  name  band
  <chr> <chr>
1 Mick  Stones
2 John  Beatles
3 Paul  Beatles
```

```
band_instruments
```

```
# A tibble: 3 x 2
  name  plays
  <chr> <chr>
1 John  guitar
2 Paul  bass
3 Keith guitar
```

- Combine the two, left/right/inner/full !

```
band_members %>% inner_join(band_instruments)
```

```
Joining with ‘by = join_by(name)‘
```

```
# A tibble: 2 x 3
  name  band    plays
  <chr> <chr>   <chr>
1 John  Beatles guitar
2 Paul  Beatles bass
```

```
band_members %>% left_join(band_instruments)
```

```
Joining with ‘by = join_by(name)‘
```

```
# A tibble: 3 x 3
  name  band    plays
  <chr> <chr>   <chr>
1 Mick  Stones  <NA>
2 John  Beatles guitar
3 Paul  Beatles bass
```

```
band_members %>% right_join(band_instruments)
```

```
Joining with ‘by = join_by(name)‘
```

```
# A tibble: 3 x 3
  name  band    plays 
  <chr> <chr>   <chr> 
1 John  Beatles guitar
2 Paul  Beatles bass  
3 Keith <NA>    guitar
```

```
band_members %>% full_join(band_instruments)
```

```
Joining with ‘by = join_by(name)‘
```

```
# A tibble: 4 x 3
  name  band    plays 
  <chr> <chr>   <chr> 
1 Mick  Stones  <NA>  
2 John  Beatles guitar
3 Paul  Beatles bass  
4 Keith <NA>    guitar
```

- Try out the same with `semi_join( )` and `anti_join( )` and interpret what happens !

```
band_members %>% semi_join(band_instruments)
```

```
Joining with ‘by = join_by(name)‘
```

```
# A tibble: 2 x 2
  name  band   
  <chr> <chr>  
1 John  Beatles
2 Paul  Beatles
```

```
band_members %>% anti_join(band_instruments)
```

```
Joining with ‘by = join_by(name)‘
```

```
# A tibble: 1 x 2
  name  band
  <chr> <chr>
1 Mick  Stones
```

dplyr exercises, catching up

- Compare the structure of the mtcars data with a glimpse at that data.

```
glimpse(mtcars)
```

```
Rows: 32
Columns: 11
$ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
$ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
$ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
$ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
$ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,~
$ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
$ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18~
$ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,~
$ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,~
$ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
$ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

- Compare a select of mpg with a pull of mpg.

```
mtcars %>% select(mpg) %>% my_gt(6)
```

mpg
21.0
21.0
22.8
21.4
18.7
18.1

```
mtcars %>% pull(mpg)
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
[31] 15.0 21.4
```

Check the help file and select the second before last column, but -pull- it from the data frame so that it turns into a vector

```
mtcars %>% pull(-3)
```

```
 [1] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1
```

- Select all columns except the `am`.

```
mtcars %>% select(-am)
```

```
                     mpg cyl  disp  hp drat    wt  qsec vs gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0    3    2
```

```
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1    4    2
```

- Select all columns except the `am` and `vs`.

```
mtcars %>% select(-am,-vs)
```

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

```
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60    4    2
```

- Keep only columns `mpg`, `cyl` and `disp`, but rename `mpg` to `miles_gallon`.

```
mtcars %>% select(miles_gallon=mpg,cyl,disp)
```

```
                    miles_gallon cyl  disp
Mazda RX4                   21.0   6 160.0
Mazda RX4 Wag               21.0   6 160.0
Datsun 710                  22.8   4 108.0
Hornet 4 Drive              21.4   6 258.0
Hornet Sportabout           18.7   8 360.0
Valiant                     18.1   6 225.0
Duster 360                  14.3   8 360.0
Merc 240D                   24.4   4 146.7
Merc 230                    22.8   4 140.8
Merc 280                    19.2   6 167.6
Merc 280C                   17.8   6 167.6
Merc 450SE                  16.4   8 275.8
Merc 450SL                  17.3   8 275.8
Merc 450SLC                 15.2   8 275.8
Cadillac Fleetwood          10.4   8 472.0
Lincoln Continental         10.4   8 460.0
Chrysler Imperial           14.7   8 440.0
Fiat 128                    32.4   4  78.7
Honda Civic                 30.4   4  75.7
Toyota Corolla              33.9   4  71.1
Toyota Corona               21.5   4 120.1
Dodge Challenger            15.5   8 318.0
AMC Javelin                 15.2   8 304.0
Camaro Z28                  13.3   8 350.0
Pontiac Firebird            19.2   8 400.0
Fiat X1-9                   27.3   4  79.0
Porsche 914-2               26.0   4 120.3
Lotus Europa                30.4   4  95.1
Ford Pantera L              15.8   8 351.0
Ferrari Dino                19.7   6 145.0
Maserati Bora               15.0   8 301.0
```

```
Volvo 142E                   21.4   4 121.0
```

- Using rename not all variables need to be mentioned explicitely. Change only mpg to miles_gallon.

```
mtcars %>% rename(miles_gallon=mpg)
```

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

- Keep only the consecutive columns in between disp and wt (use a :), additionally add `mpg` as a last column.

- It is possible to pipe also base R functions, try it and pipe the solution above through `names` to get the variable names.

```
mtcars %>% select(disp:wt,mpg) %>% names
```

```
[1] "disp" "hp"   "drat" "wt"   "mpg"
```

- Create a variable for the row names. Maybe check `rownames_to_column`.

```
mtcars %>% rownames_to_column('type')
```

```
                  type  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1            Mazda RX4 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
2        Mazda RX4 Wag 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
3           Datsun 710 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
4       Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
5    Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
6              Valiant 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
7           Duster 360 14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
8            Merc 240D 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
9             Merc 230 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
10            Merc 280 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
11           Merc 280C 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
12          Merc 450SE 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
13          Merc 450SL 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
14         Merc 450SLC 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
15  Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
16 Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
17   Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
18            Fiat 128 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
19         Honda Civic 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
20      Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
21       Toyota Corona 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
22    Dodge Challenger 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
23         AMC Javelin 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
24          Camaro Z28 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
25    Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
26           Fiat X1-9 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
27       Porsche 914-2 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
```

```
28        Lotus Europa 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
29      Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
30        Ferrari Dino 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
31       Maserati Bora 15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
32          Volvo 142E 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

- Change the `mpg` (miles per gallon) into `kpl` (kilometers per liter) with 1 mpg is 0.425 km/l, using `mutate( )`.

```
mtcars %>% mutate(kpl=mpg*.425)
```

```
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb     kpl
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4  8.9250
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4  8.9250
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1  9.6900
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1  9.0950
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2  7.9475
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1  7.6925
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4  6.0775
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2 10.3700
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2  9.6900
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4  8.1600
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4  7.5650
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3  6.9700
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3  7.3525
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3  6.4600
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4  4.4200
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4  4.4200
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4  6.2475
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1 13.7700
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2 12.9200
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1 14.4075
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1  9.1375
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2  6.5875
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2  6.4600
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4  5.6525
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2  8.1600
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1 11.6025
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2 11.0500
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2 12.9200
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4  6.7150
```

```
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6  8.3725
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8  6.3750
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2  9.0950
```

- Select about 10% of the observations, check the help file on using `sample_frac( )`.
  - run this code multiple times to see what happens

```
mtcars %>% sample_frac(.1)
```

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Dodge Challenger 15.5   8  318 150 2.76 3.520 16.87  0  0    3    2
Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
```

```
mtcars %>% sample_frac(.1)
```

```
               mpg cyl  disp  hp drat   wt qsec vs am gear carb
Merc 280C     17.8   6 167.6 123 3.92 3.44 18.9  1  0    4    4
Merc 230      22.8   4 140.8  95 3.92 3.15 22.9  1  0    4    2
Porsche 914-2 26.0   4 120.3  91 4.43 2.14 16.7  0  1    5    2
```

- Select the 10th to 15th row, check the help file on using `slice( )`

```
mtcars %>% slice(10:15)
```

```
                    mpg cyl  disp  hp drat   wt  qsec vs am gear carb
Merc 280           19.2   6 167.6 123 3.92 3.44 18.30  1  0    4    4
Merc 280C          17.8   6 167.6 123 3.92 3.44 18.90  1  0    4    4
Merc 450SE         16.4   8 275.8 180 3.07 4.07 17.40  0  0    3    3
Merc 450SL         17.3   8 275.8 180 3.07 3.73 17.60  0  0    3    3
Merc 450SLC        15.2   8 275.8 180 3.07 3.78 18.00  0  0    3    3
Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.25 17.98  0  0    3    4
```

- Select only the distinct combinations, for variables `am` and `vs`

```
mtcars %>% distinct(cyl,vs,am)
```

```
                  cyl vs am
Mazda RX4           6  0  1
```

```
Datsun 710          4  1  1
Hornet 4 Drive      6  1  0
Hornet Sportabout   8  0  0
Merc 240D           4  1  0
Porsche 914-2       4  0  1
Ford Pantera L      8  0  1
```

- You only get three variables, check the help files to determine how to keep all variables (for each first observation of that combination)

```
mtcars %>% distinct(cyl,vs,am,.keep_all=T)
```

```
                   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
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Porsche 914-2     26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
```

- Filter the data to retain only cases with mpg > 20 and hp above or equal to 110

```
mtcars %>% filter(mpg>20, hp>=110)
```

```
                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
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

- Filter the data to retain only the Datsun 710

```
mtcars %>% rownames_to_column('type') %>% filter(type=='Datsun 710')
```

```
        type  mpg cyl disp hp drat   wt  qsec vs am gear carb
1 Datsun 710 22.8   4  108 93 3.85 2.32 18.61  1  1    4    1
```

Getting ahead of ourselves again, with tidier and friends

toy dataset

- A dataset can be read in, for example using the `read_delim()` function

- Just copy-paste data from notepad, excel or another spreadsheet program
- The copy-pasted table can be assigned to the `myrepeated` object

```
myrepeated <- read_delim(clipboard(),delim='\t')
```

- The `clipboard()` function is just one way, you can also specify a path to the data
  - the delimiter is `\t` or TABs

  - type `?read_delim` to get details on more possibilities
- Because you do not have it, it is included already

```
(myrepeated <- tribble(
  ~id, ~`t1 score`, ~`t1 posit`, ~`t2 score`, ~`t2 posit`, ~`t3 score`, ~`t3 posit`,
  "id1",1,'x', NA,'y',4,'x',
  "id2",2,'y',3,'x',NA,NA,
  "id3",1,'x',2,'y',5,'x'
))
```

```
# A tibble: 3 x 7
  id    ‘t1 score‘ ‘t1 posit‘ ‘t2 score‘ ‘t2 posit‘ ‘t3 score‘ ‘t3 posit‘
  <chr>      <dbl> <chr>           <dbl> <chr>           <dbl> <chr>
1 id1            1 x                  NA y                   4 x
2 id2            2 y                   3 x                  NA <NA>
3 id3            1 x                   2 y                   5 x
```

```
Joining with ‘by = join_by(id, time)‘
```

examplary data tidying

- Having it read in, it is tidied, turned into 2 files joined after separating cell contents
  - make a dataset without the posit variables, and one without the score variables, and pivot the score or posit values from columns to rows identified by a new variable `type`

  - disentangle the values in `type` in two parts: `time` and `type`
  - recombine the two datasets after removing the new variable `type` from at least one of them
  - remove all rows with missing values in either the variable `score` or `posit`

```
scores <- myrepeated %>%
    select(id,`t1 score`,`t2 score`,`t3 score`) %>%
    pivot_longer(-id,names_to='type',values_to='score')
positions <- myrepeated %>%
    select(id,`t1 posit`,`t2 posit`,`t3 posit`) %>%
    pivot_longer(-id,names_to='type',values_to='posit')

scores <- scores %>%
    separate(type,c('time','type'))
positions <- positions %>%
    separate(type,c('time','type'))

joined <- scores %>%
    select(-type) %>% full_join(positions)

longform <- joined %>%
    select(-type) %>% filter(!is.na(score),!is.na(posit))
```

id	time	score	posit
id1	t1	1	x
id1	t3	4	x
id2	t1	2	y
id2	t2	3	x
id3	t1	1	x
id3	t2	2	y
id3	t3	5	x

- It is possible to switch back to a wider data representation
  - e.g., to calculate correlations (maybe fill in the missing values NA as 0 values)

```
longform %>% pivot_wider(names_from=c(time),values_from=c(score,posit),values_fill=list(sc
```

```
# A tibble: 3 x 7
  id    score_t1 score_t3 score_t2 posit_t1 posit_t3 posit_t2
  <chr>    <dbl>    <dbl>    <dbl> <chr>    <chr>    <chr>
```

```
1 id1          1        4        0 x        x        <NA>
2 id2          2        0        3 y        <NA>     x
3 id3          1        5        2 x        x        y
```

`tidyr` and import packages, functions to read and tidy data

- `tidier` combines a few functions to tidy up the data
  - a core idea at the origin of the development of the tidyverse
- By enforcing structure on the data, functions defined to operate on that data can be made much more consistent too
- `readr` combines a few functions to read in data, stored externally, in text format, excel, spss, …
- The `tidier` and `readr` packages:
  - focus on importing data and making it tidy
    - the data has to be brought into the R workspace
    - the data has to be tidy for efficient further processing
  - use to create tidy data
    - a row for each research unit
    - a columns for each variable
    - a cell that links a research unit to a variable
  - requires
    - pivoting data into longer or wider form
    - creating pure variables
- The main -verbs- (see example above)
  - `pivot_wider( )` and `pivot_longer( )`: turn multiple columns or rows into one, making datafiles longer or wider
  - `separate( )` and `extract( )`: create multiple columns from one column using delimiters or regular expressions

pivot_*()

- Turning long form data into wide form and vise versa, is called pivoting.

intro

- In tidy data each research unit is assigned to a row, in a tidy dataframe (tibble)
  - what is the research unit depends on the research question and can change (eg., test score → student)
- Contrary to univariate data representation, a multivariate data representation can be useful and be more intuitive
- To change research units or to switch between uni -and multivariate, data can be pivoted, turned wider or longer
- Pivoting from wider to longer
  - column headers are turned into values of an identifier column
  - values over different columns are combined into new column
  - the identifier column and values column require names
- The `iris` dataset, with 4 values for each unit within each species, is pivoted
  - Notice, the k column headers turn into nxk cell values to serve as identifiers
- Beware: without a unique identifier for each row, the clustering of columns' information is lost
  - a unique identifier per row should typically be added before pivoting
  - given a unique identifier, it should be removed from the pivoted variables

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
long_iris_no_identifier <- iris %>% pivot_longer(-Species,names_to='type',values_to='score
long_iris_with_identifier <- iris %>% mutate(id=1:n()) %>% pivot_longer(-c(id,Species),nam
```

Species	type	score
setosa	Sepal.Length	5.1
setosa	Sepal.Width	3.5
setosa	Petal.Length	1.4

setosa	Petal.Width	0.2
setosa	Sepal.Length	4.9
setosa	Sepal.Width	3.0

Species	id	type	score
setosa	1	Sepal.Length	5.1
setosa	1	Sepal.Width	3.5
setosa	1	Petal.Length	1.4
setosa	1	Petal.Width	0.2
setosa	2	Sepal.Length	4.9
setosa	2	Sepal.Width	3.0

- Pivoting from longer to wider
  - column headers are created from values in an identifier column
  - values within a values column are aligned over different columns
  - the identifier column and values column must be specified
- Without adding a row specific identifier before pivoting the `iris` dataset from wide to long
  - no information would be available to assign values to a particular row
  - many values are forced into one single cell
- Note that long-er and wide-r is used
  - expresses that data can be long for certain aspects and wide for others

```r
long_iris_no_identifier %>% pivot_wider(values_from=score,names_from=type)
```

```
Warning: Values from ‘score‘ are not uniquely identified; output will contain list-cols.
* Use ‘values_fn = list‘ to suppress this warning.
* Use ‘values_fn = {summary_fun}‘ to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} %>%
  dplyr::group_by(Species, type) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
# A tibble: 3 x 5
  Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
  <fct>      <list>       <list>      <list>       <list>
```

```
1 setosa     <dbl [50]>   <dbl [50]>  <dbl [50]>   <dbl [50]>
2 versicolor <dbl [50]>   <dbl [50]>  <dbl [50]>   <dbl [50]>
3 virginica  <dbl [50]>   <dbl [50]>  <dbl [50]>   <dbl [50]>
```

- To pivot from longer to wider form, a column is spread out over multiple columns and along with it the values
  - new column names are extracted from a column, typically with a limited set of labels
  - values to populate the newly constructed columns are extracted from a column too

```
long_iris_with_identifier %>% group_by(type) %>% mutate(id=1:n()) %>% pivot_wider(values_f
```

Species	id	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	1	5.1	3.5	1.4	0.2
setosa	2	4.9	3.0	1.4	0.2
setosa	3	4.7	3.2	1.3	0.2
setosa	4	4.6	3.1	1.5	0.2
setosa	5	5.0	3.6	1.4	0.2
setosa	6	5.4	3.9	1.7	0.4

exercises

- Pivot the `world_bank_pop` dataset from the `tidyr` package, to have univariate data for the scores over the different years

```
world_bank_pop %>% pivot_longer(-c(country,indicator),values_to='scores',names_to='year')
```

```
# A tibble: 19,152 x 4
   country indicator   year  scores
   <chr>   <chr>       <chr>  <dbl>
 1 ABW     SP.URB.TOTL 2000   41625
 2 ABW     SP.URB.TOTL 2001   42025
 3 ABW     SP.URB.TOTL 2002   42194
 4 ABW     SP.URB.TOTL 2003   42277
 5 ABW     SP.URB.TOTL 2004   42317
 6 ABW     SP.URB.TOTL 2005   42399
 7 ABW     SP.URB.TOTL 2006   42555
 8 ABW     SP.URB.TOTL 2007   42729
 9 ABW     SP.URB.TOTL 2008   42906
10 ABW     SP.URB.TOTL 2009   43079
```

```
# i 19,142 more rows
```

- Use the `us_rent_income` dataset, also part of the `tidyr` package, and remove variable `moe` before pivoting the estimates to wide form

```
(us_rent_income %>% select(-moe) %>% pivot_wider(values_from=estimate,names_from=variable)
```

```
# A tibble: 52 x 4
   GEOID NAME                 income  rent
   <chr> <chr>                 <dbl> <dbl>
 1 01    Alabama               24476   747
 2 02    Alaska                32940  1200
 3 04    Arizona               27517   972
 4 05    Arkansas              23789   709
 5 06    California            29454  1358
 6 08    Colorado              32401  1125
 7 09    Connecticut           35326  1123
 8 10    Delaware              31560  1076
 9 11    District of Columbia  43198  1424
10 12    Florida               25952  1077
# i 42 more rows
```

- Verify what happens when you did not remove `moe`

```
(us_rent_income %>% pivot_wider(values_from=estimate,names_from=variable))
```

```
# A tibble: 104 x 5
   GEOID NAME         moe income  rent
   <chr> <chr>      <dbl>  <dbl> <dbl>
 1 01    Alabama      136  24476    NA
 2 01    Alabama        3     NA   747
 3 02    Alaska       508  32940    NA
 4 02    Alaska        13     NA  1200
 5 04    Arizona      148  27517    NA
 6 04    Arizona        4     NA   972
 7 05    Arkansas     165  23789    NA
 8 05    Arkansas       5     NA   709
 9 06    California   109  29454    NA
10 06    California     3     NA  1358
# i 94 more rows
```

- It is possible to include multiple variables to pivot wide, jointly, use a vector of variables that includes estimate and moe, and see what happens

```
(us_rent_income %>% pivot_wider(values_from=c(estimate,moe),names_from=variable))
```

```
# A tibble: 52 x 6
   GEOID NAME                 estimate_income estimate_rent moe_income moe_rent
   <chr> <chr>                          <dbl>         <dbl>      <dbl>    <dbl>
 1 01    Alabama                        24476           747        136        3
 2 02    Alaska                         32940          1200        508       13
 3 04    Arizona                        27517           972        148        4
 4 05    Arkansas                       23789           709        165        5
 5 06    California                     29454          1358        109        3
 6 08    Colorado                       32401          1125        109        5
 7 09    Connecticut                    35326          1123        195        5
 8 10    Delaware                       31560          1076        247       10
 9 11    District of Columbia           43198          1424        681       17
10 12    Florida                        25952          1077         70        3
# i 42 more rows
```

separate( ) / unite( )

- Splitting up information within a variable, or combining information over variables, to ensure cell values to offer one and only one piece of relevant information

intro

- Each variable should consist of one type of information, in a tidy dataframe (tibble)
  - variables that combine information should often be split
  - variables that provide no meaningful information by themselves should be removed, sometimes united
- Columns (variables) can be split and united
- The long form iris data shows a type that consists of both Petal/Sepal and Length/Width, the can be separated

```
long_iris_with_identifier %>% separate(type,c('PS','lw'))
```

Species	id	PS	lw	score
setosa	1	Sepal	Length	5.1
setosa	1	Sepal	Width	3.5
setosa	1	Petal	Length	1.4
setosa	1	Petal	Width	0.2
setosa	2	Sepal	Length	4.9
setosa	2	Sepal	Width	3.0

- On the contrary, variables can also be united
- Separated columns can be combined, using a separator dash in this case (default is underscore)

```
long_iris_separated %>% unite('myType',PS:lw,sep='-')
```

```
# A tibble: 600 x 4
   Species    id myType       score
   <fct>   <int> <chr>        <dbl>
 1 setosa      1 Sepal-Length   5.1
 2 setosa      1 Sepal-Width    3.5
 3 setosa      1 Petal-Length   1.4
 4 setosa      1 Petal-Width    0.2
 5 setosa      2 Sepal-Length   4.9
 6 setosa      2 Sepal-Width    3
 7 setosa      2 Petal-Length   1.4
 8 setosa      2 Petal-Width    0.2
 9 setosa      3 Sepal-Length   4.7
10 setosa      3 Sepal-Width    3.2
# i 590 more rows
```

- The `tidyr` package includes other functions for more involved programming and simulation studies
- Notice in particular `expand( )`, `crossover( )`, `nesting( )`, best check the helpfile.

```
?expand
```

exercises

- Turn the row names of the `mtcars` data to a variable called type using the `rownames_to_column( )` function

  - it consists of car type information (maybe use `type`), car sub-type (`subtype`) and sub-type specification (`spec`)
  - look into the `fill` argument to ensure the the pieces of information are read in from right to left

```
mtcars %>% rownames_to_column('type') %>%
    separate(type,c("type","subtype","spec"),fill='right') %>%
    ungroup() %>% my_gt(6)
```

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

- Separate the type variable to isolate information on the type on one hand, and the rest on the other

```
mtcars %>% rownames_to_column('type') %>%
    separate(type,c("type","subtype","spec"),fill='right') %>%
    unite("subtype",c("subtype","spec")) %>% ungroup() %>% my_gt(6)
```

type	subtype	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda	RX4_NA	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda	RX4_Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun	710_NA	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet	4_Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet	Sportabout_NA	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	NA_NA	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Import data with `readr`, `readxl` or `haven`

- when using your own data, they have to be imported into the workspace
- Data that are saved as R objects in a workspace (`*.RData`) can be loaded with the `load()` function
- Data that need to be imported from elsewhere require dedicated functions (packages)

readr

- The readr package in tidyverse deals with the basic data, like comma separated or tab-delimited data

intro

- The primary function in readr is read_delim( ) which imports tabular data with a delimiter as specified
- Note that a path to the data may need to be specified, in absolute terms or relative to the current working directory

```
getwd()
setwd(readClipboard())
setwd('../../my_sub_dir_2_levels_up')
```

- A delimiter should be specified, \t for tabs

- ?read_delim offers information on how to set many different arguments and gain flexibility to read in data
- In this current working directory should have a tab-delimited file named repeated.txt

```
myrepeated <- read_delim(file='repeated.txt',delim='\t') # if
```

- Data can be copy pasted in using the clipboard( ) instead of a path, or a path can be asked for interactively with file.choose( )

```
myrepeated <- read_delim(clipboard(),delim='\t')
myrepeated <- read_delim(file.choose(),delim='\t')
```

readxl

- The readxl package in tidyverse deals with the notorious excel files

intro

- The primary function in `readxl` is `read_excel( )` which imports tabular data from an excel file
- Note that a path to the data may need to be specified, in absolute terms or relative to the current working directory
- The `example_data_set.xlsx` if it would exist in current working directory could be read in, possibly having assigned a particular sheet
- Interesting arguments are the sheet to read from, or the number of rows to skip
- `?read_excel` offers information on the many arguments that add flexibility for reading in data

```
read_excel('example_data_set.xlsx', sheet='my_data', skip=1)
```

`haven`

- The `haven` package in tidyverse deals with the data stored as part of one of the main statistical software, like SAS, spss and Stata

intro

- For SPSS, with *.sav files, Data is simply read, using default parameters `read_sav( )` reads SPSS stored data
- The `haven` package is not automatically loaded with tidyverse
- Let's first get the path to the iris data as an example
- `?read_sav` for more information on the available arguments

```
library(haven)
path_to_spss_examplary_data <- system.file("examples", "iris.sav", package = "haven")
read_sav(path_to_spss_examplary_data)
```

```
# A tibble: 150 x 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <dbl+lbl>
 1          5.1         3.5          1.4         0.2 1 [setosa]
 2          4.9         3            1.4         0.2 1 [setosa]
 3          4.7         3.2          1.3         0.2 1 [setosa]
 4          4.6         3.1          1.5          0.2 1 [setosa]
 5          5           3.6          1.4          0.2 1 [setosa]
 6          5.4         3.9          1.7          0.4 1 [setosa]
 7          4.6         3.4          1.4          0.3 1 [setosa]
 8          5           3.4          1.5          0.2 1 [setosa]
 9          4.4         2.9          1.4          0.2 1 [setosa]
10          4.9         3.1          1.5          0.1 1 [setosa]
# i 140 more rows
```

- For SAS, with for example *.sas7bdat files, data is read using default parameters
- Let's again get the path to the iris data as an example
- `?read_sas` for more information on the available arguments

```
path <- system.file("examples", "iris.sas7bdat", package = "haven")
read_sas(path)
```

- For Stata, with for example *.dta files, data is read using default parameters
- Let's again get the pat to the iris data as an example
- `?read_dta` for more information on the available arguments

```
path <- system.file("examples", "iris.dta", package = "haven")
read_dta(path)
```

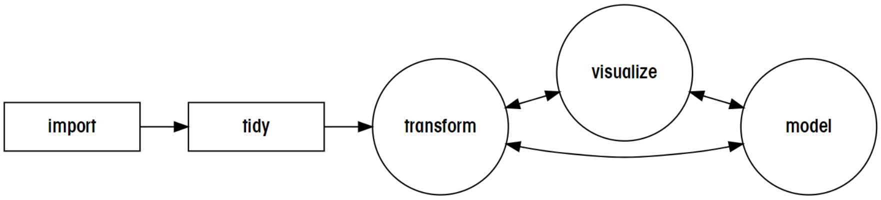
To write any of the files, use the `write_` prefix, for `dta`, `sas` and `sav`
To write the `mtcars` into sas format.

```
write_sas(mtcars,'mytryinSAS.sas7bdat')
```

Last remarks

Current page provides a primer on data manipulation, tidying data and the importing of data, which are the main steps in preparation of most real data analyses and visualizations.

It is strongly advised to play with the techniques discussed above to get some proficiency in using it, as it would add significantly to the flexibility of whatever you want to further do with your data.



Other tidyverse packages exist, and within the same framework many more are being developed. The consistency within the tidyverse ecosystem should give you a push though, to study the other packages yourself when of interest.

Base R still is a proper alternative to the tidyverse ecosystem, so be aware that others may do things differently.