Visualization and Data Structure

Principles of graph grammar and tidy data

Thiago de Paula Oliveira
thiago.oliveira@ed.ac.uk
https://prof-thiagooliveira.netlify.com

18th September 2021

@HighlanderLab

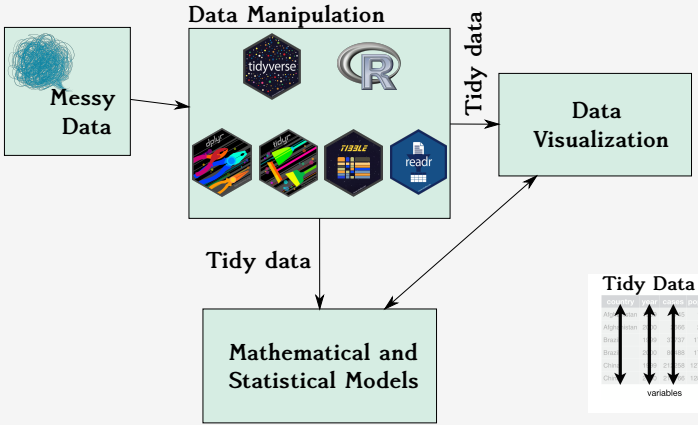
Contents

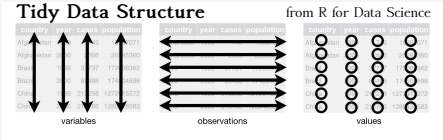
Pixabay

Concept of tidy data

Concept of tidy data

- **Data is often messy** → require organization and restructuring
- Tidy data sets: **makes data analysis easy!**





Common problems with messy data

- Column **headers are values**, not variable names

Farm	<$10k	$10k-20k	>$30k
Farm 1	100	50	72
Farm 2	102	54	61
Farm 3	101	50	58
Farm 4	96	50	87
Farm 5	102	47	88
Farm 6	96	50	95
Farm 7	101	53	77
Farm 8	100	50	80
Farm 9	97	49	103
Farm 10	104	54	94

Farm	Income	Freq
1	<$10k	100.00
2	<$10k	102.00
3	<$10k	101.00
4	<$10k	96.00
5	<$10k	102.00
6	<$10k	96.00
⋮	⋮	⋮

[30 rows × 3 columns]

Common problems with messy data

- Column **headers are values**, not variable names

Farm	<$10k	$10k-20k	>$30k
Farm 1	100	50	72
Farm 2	102	54	61
Farm 3	101	50	58
Farm 4	96	50	87
Farm 5	102	47	88
Farm 6	96	50	95
Farm 7	101	53	77
Farm 8	100	50	80
Farm 9	97	49	103
Farm 10	104	54	94

Farm	Income	Freq
1	<$10k	100.00
2	<$10k	102.00
3	<$10k	101.00
4	<$10k	96.00
5	<$10k	102.00
6	<$10k	96.00
⋮	⋮	⋮

[30 rows × 3 columns]

Common problems with messy data

- Multiple types of observational units are stored in the same table

id	year	month	Type	d1	d2	...
Ind 1	2020	01	tmin	35.82		...
Ind 1	2020	01	tmax	36.70		...
Ind 1	2020	02	tmin		35.92	...
Ind 1	2020	02	tmax		36.79	...
Ind 1	2020	03	tmin			...
Ind 1	2020	03	tmax			...
Ind 1	2020	04	tmin			...
Ind 1	2020	04	tmax			...
⋮	⋮	⋮	⋮	⋮	⋮	⋮

[20 rows × 35 columns]

id	date	type	value
Ind 1	2020-01-01	tmin	35.82
Ind 1	2020-01-01	tmax	36.70
Ind 1	2020-01-02	tmin	35.92
Ind 1	2020-01-02	tmax	36.79
Ind 1	2020-01-03	tmin	36.04
Ind 1	2020-01-03	tmax	36.83
Ind 1	2020-01-04	tmin	36.13
Ind 1	2020-01-04	tmax	36.90
Ind 1	2020-01-05	tmin	36.14
Ind 1	2020-01-05	tmax	36.92

[20 rows × 4 columns]

Common problems with messy data

- Multiple types of observational units are stored in the same table

id	year	month	Type	d1	d2	...
Ind 1	2020	01	tmin	35.82		...
Ind 1	2020	01	tmax	36.70		...
Ind 1	2020	02	tmin		35.92	...
Ind 1	2020	02	tmax		36.79	...
Ind 1	2020	03	tmin			...
Ind 1	2020	03	tmax			...
Ind 1	2020	04	tmin			...
Ind 1	2020	04	tmax			...
⋮	⋮	⋮	⋮	⋮	⋮	⋮

[20 rows × 35 columns]

id	date	type	value
Ind 1	2020-01-01	tmin	35.82
Ind 1	2020-01-01	tmax	36.70
Ind 1	2020-01-02	tmin	35.92
Ind 1	2020-01-02	tmax	36.79
Ind 1	2020-01-03	tmin	36.04
Ind 1	2020-01-03	tmax	36.83
Ind 1	2020-01-04	tmin	36.13
Ind 1	2020-01-04	tmax	36.90
Ind 1	2020-01-05	tmin	36.14
Ind 1	2020-01-05	tmax	36.92

[20 rows × 4 columns]

Common problems with messy data

- **Variables** are sorted **in both rows and columns**
- **Multiple variables** are sorted in only one column

id	date	type	value
Ind 1	2020-01-01	tmin	35.82
Ind 1	2020-01-01	tmax	36.70
Ind 1	2020-01-02	tmin	35.92
Ind 1	2020-01-02	tmax	36.79
Ind 1	2020-01-03	tmin	36.04
Ind 1	2020-01-03	tmax	36.83
Ind 1	2020-01-04	tmin	36.13
Ind 1	2020-01-04	tmax	36.90
Ind 1	2020-01-05	tmin	36.14
Ind 1	2020-01-05	tmax	36.92

[20 rows $\times$ 4 columns]

id	date	tmin	tmax
1	2020-01-01	35.82	36.70
1	2020-01-02	35.92	36.79
1	2020-01-03	36.04	36.83
1	2020-01-04	36.13	36.90
1	2020-01-05	36.14	36.92
1	2020-01-06	36.19	37.00
1	2020-01-07	36.30	37.07
1	2020-01-08	36.33	37.17
1	2020-01-09	36.56	37.32
1	2020-01-10	36.57	37.99

[10 rows $\times$ 5 columns]

Common problems with messy data

- **Variables** are sorted **in both rows and columns**
- **Multiple variables** are sorted in only one column

id	date	type	value
Ind 1	2020-01-01	tmin	35.82
Ind 1	2020-01-01	tmax	36.70
Ind 1	2020-01-02	tmin	35.92
Ind 1	2020-01-02	tmax	36.79
Ind 1	2020-01-03	tmin	36.04
Ind 1	2020-01-03	tmax	36.83
Ind 1	2020-01-04	tmin	36.13
Ind 1	2020-01-04	tmax	36.90
Ind 1	2020-01-05	tmin	36.14
Ind 1	2020-01-05	tmax	36.92

[20 rows $\times$ 4 columns]

id	date	tmin	tmax
1	2020-01-01	35.82	36.70
1	2020-01-02	35.92	36.79
1	2020-01-03	36.04	36.83
1	2020-01-04	36.13	36.90
1	2020-01-05	36.14	36.92
1	2020-01-06	36.19	37.00
1	2020-01-07	36.30	37.07
1	2020-01-08	36.33	37.17
1	2020-01-09	36.56	37.32
1	2020-01-10	36.57	37.99

[10 rows $\times$ 5 columns]

Data Import

Data Import

readr package

Reading Tabular Data

- `read_delim("file.txt", delim = ",")`
- `read_csv("file.csv")`
- `read_tsv("file.tsv")`

Writing Files

- `write_csv(data,"file.csv")`
- `write_tsv(data,"file.csv")`

Reading other formats

- `read_rds("file.rds")`
- `read_log("file.RData")`: reads Apache style log files.

Writing `rds` Data

- `write_rds(data, "file.RDS")`

Converting `data.frame` to `tibble`

- `as_tibble(data)`

Data Import

readr package

Reading Tabular Data

- `read_delim("file.txt", delim = ",")`
- `read_csv("file.csv")`
- `read_tsv("file.tsv")`

Writing Files

- `write_csv(data,"file.csv")`
- `write_tsv(data,"file.csv")`

Reading other formats

- `read_rds("file.rds")`
- `read_log("file.RData")`: reads Apache style log files.

Writing `rds` Data

- `write_rds(data, "file.RDS")`

Converting `data.frame` to `tibble`

- `as_tibble(data)`

Data Import

readr package

Reading Tabular Data

- `read_delim("file.txt", delim = ",")`
- `read_csv("file.csv")`
- `read_tsv("file.tsv")`

Writing Files

- `write_csv(data,"file.csv")`
- `write_tsv(data,"file.csv")`

Reading other formats

- `read_rds("file.rds")`
- `read_log("file.RData")`: reads Apache style log files.

Writing `rds` Data

- `write_rds(data, "file.RDS")`

Converting `data.frame` to `tibble`

- `as_tibble(data)`

Difference between `tibble()` and `data.frame()`

- There are two main differences in the usage of a data frame vs a tibble:
  - printing: refined print method that **shows only the first 10 rows, and all the columns that fit on screen**

```
# A tibble: 10,000 × 7
     ind father mother year  sex   phenotype herd
   <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
 1     1     NA     NA 0     M          37.7 E
 2     2     NA     NA 0     M          35.8 B
 3     3     NA     NA 0     M          28.4 A
 4     4     NA     NA 0     M          33.6 D
 5     5     NA     NA 0     M          32.9 A
 6     6     NA     NA 0     M          31.6 A
 7     7     NA     NA 0     M          38.8 A
 8     8     NA     NA 0     M          33.3 E
 9     9     NA     NA 0     M          39.0 E
10    10     NA     NA 0     M          46.0 C
# … with 9,990 more rows
```

  - subsetting: if you try to access a variable that does not exist, you'll get an warning message returning `NULL`

```
> cbp$yea
NULL
Warning message:
Unknown or uninitialised column: `yea`.
```

Data Structure

Pivoting: 'long form'

- It **makes datasets longer** by increasing the number of rows and decreasing the number of columns

```
R> dataEx1 <- readRDS("dataEx1.RDS")
```

	<$10k	$10k-20k	>$30k
Farm 1	100	50	72
Farm 2	102	54	61
Farm 3	101	50	58
Farm 4	96	50	87
Farm 5	102	47	88
Farm 6	96	50	95
Farm 7	101	53	77
Farm 8	100	50	80
Farm 9	97	49	103
Farm 10	104	54	94

```
R> dataEx1 %>%
+   rownames_to_column(var = "Farm") %>%
+   pivot_longer(cols = 2:4, names_to = "Income",
+                values_to = "Freq")
```

Pivoting: 'long form'

- It **makes datasets longer** by increasing the number of rows and decreasing the number of columns

```
R> dataEx1 <- readRDS("dataEx1.RDS")
```

	<$10k	$10k-20k	>$30k
Farm 1	100	50	72
Farm 2	102	54	61
Farm 3	101	50	58
Farm 4	96	50	87
Farm 5	102	47	88
Farm 6	96	50	95
Farm 7	101	53	77
Farm 8	100	50	80
Farm 9	97	49	103
Farm 10	104	54	94

```
R> dataEx1 %>%
+   rownames_to_column(var = "Farm") %>%
+   pivot_longer(cols = 2:4, names_to = "Income",
+                values_to = "Freq")
```

```
# A tibble: 30 × 3
   Farm   Income    Freq
   <chr>  <chr>    <dbl>
 1 Farm 1 <$10k      100
 2 Farm 1 $10k-20k    50
 3 Farm 1 >$30k       72
 4 Farm 2 <$10k      102
 5 Farm 2 $10k-20k    54
 6 Farm 2 >$30k       61
 7 Farm 3 <$10k      101
 8 Farm 3 $10k-20k    50
 9 Farm 3 >$30k       58
10 Farm 4 <$10k       96
# … with 20 more rows
```

Pivoting: 'wide form'

- It **makes a dataset wider** by increasing the number of columns and decreasing the number of rows

```
R> dataEx2 <- readRDS("dataEx2.RDS")
```

```
# A tibble: 10 × 4
   id    date       type  value
   <chr> <date>     <fct> <dbl>
 1 Ind 1 2020-01-01 tmin   35.8
 2 Ind 1 2020-01-01 tmax   36.7
 3 Ind 1 2020-01-02 tmin   35.9
 4 Ind 1 2020-01-02 tmax   36.8
 5 Ind 1 2020-01-03 tmin   36.0
 6 Ind 1 2020-01-03 tmax   36.8
 7 Ind 1 2020-01-04 tmin   36.1
 8 Ind 1 2020-01-04 tmax   36.9
 9 Ind 1 2020-01-05 tmin   36.1
10 Ind 1 2020-01-05 tmax   36.9
```

```
R> dataEx2 %>%
+   pivot_wider(values_from = value,
+               names_from = type)
```

Pivoting: 'wide form'

- It **makes a dataset wider** by increasing the number of columns and decreasing the number of rows

```
R> dataEx2 <- readRDS("dataEx2.RDS")
```

```
# A tibble: 10 × 4
   id    date       type  value
   <chr> <date>     <fct> <dbl>
 1 Ind 1 2020-01-01 tmin   35.8
 2 Ind 1 2020-01-01 tmax   36.7
 3 Ind 1 2020-01-02 tmin   35.9
 4 Ind 1 2020-01-02 tmax   36.8
 5 Ind 1 2020-01-03 tmin   36.0
 6 Ind 1 2020-01-03 tmax   36.8
 7 Ind 1 2020-01-04 tmin   36.1
 8 Ind 1 2020-01-04 tmax   36.9
 9 Ind 1 2020-01-05 tmin   36.1
10 Ind 1 2020-01-05 tmax   36.9
```

```
R> dataEx2 %>%
+   pivot_wider(values_from = value,
+               names_from = type)
```

```
# A tibble: 10 × 4
   id    date       tmin  tmax
   <chr> <date>     <dbl> <dbl>
 1 Ind 1 2020-01-01  35.8  36.7
 2 Ind 1 2020-01-02  35.9  36.8
 3 Ind 1 2020-01-03  36.0  36.8
 4 Ind 1 2020-01-04  36.1  36.9
 5 Ind 1 2020-01-05  36.1  36.9
 6 Ind 1 2020-01-06  36.2  37.0
 7 Ind 1 2020-01-07  36.3  37.1
 8 Ind 1 2020-01-08  36.3  37.2
 9 Ind 1 2020-01-09  36.6  37.3
10 Ind 1 2020-01-10  36.6  38.0
```

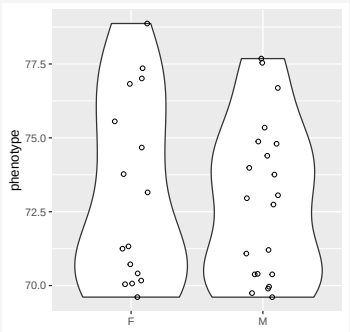
Simulated data: Dairy Cattle Breeding Programme (cbp data)

- Data is comprised of founders (with phantom parents) and phenotyped individuals (on the Milk Yield), where for each one, we have information of parents, sex, and herd.

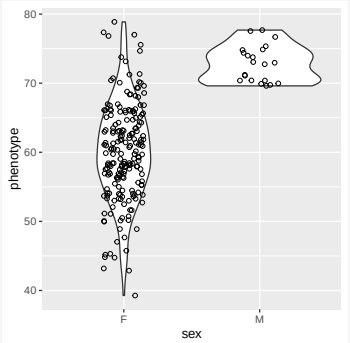
Ind	Father	Mother	Year	Sex	Herd	Pheno
1	NA	NA	0	M	E	37.67
2	NA	NA	0	M	B	35.80
3	NA	NA	0	M	A	28.43
4	NA	NA	0	M	D	33.63
⋮	⋮	⋮	⋮	⋮	⋮	
9997	8342	8677	9	F	A	58.43
9998	8088	8510	9	F	E	58.17
9999	8449	8685	9	F	D	56.72
10000	8296	8854	9	F	B	67.08

Manipulate Cases: `filter()`

```
R> threshold <- with(cbp, mean(phenotype) +
+                          2*sd(phenotype))
R> cbp %>%
+   filter(herd %in% c("A", "E")) %>%
+   filter(year == "9" & phenotype > threshold) %>%
+   ggplot(aes(y = phenotype, x = sex)) +
+    geom_violin() +
+    geom_jitter(shape = 1, width = 0.15)
```




```
R> cbp %>%
+   filter(herd %in% c("A", "E")) %>%
+   filter(year == "9" & (phenotype > threshold |
+                         sex == "F")) %>%
+   ggplot(aes(y = phenotype, x = sex)) +
+    geom_violin() +
+    geom_jitter(shape = 1, width = 0.15)
```



Manipulate Cases: `distinct()`

- Select only unique/distinct rows from a data frame.
- **Faster** than `unique.data.frame()`

```
R> cbp %>%
+   distinct(dplyr::across(contains("r")))
```

```
# A tibble: 8,186 × 4
   father mother year  herd
    <dbl>  <dbl> <fct> <fct>
 1     NA     NA 0     E
 2     NA     NA 0     B
 3     NA     NA 0     A
 4     NA     NA 0     D
 5     NA     NA 0     C
 6    418    692 1     E
 7    461    614 1     B
 8    195    524 1     A
 9    198    768 1     D
10    122    537 1     A
# … with 8,176 more rows
```

```
R> cbp %>%
+   distinct(sex)
```

Manipulate Cases: `distinct()`

- Select only unique/distinct rows from a data frame.
- **Faster** than `unique.data.frame()`

```
R> cbp %>%
+   distinct(dplyr::across(contains("r")))
```

```
# A tibble: 8,186 × 4
   father mother year  herd
    <dbl>  <dbl> <fct> <fct>
 1     NA     NA 0     E
 2     NA     NA 0     B
 3     NA     NA 0     A
 4     NA     NA 0     D
 5     NA     NA 0     C
 6    418    692 1     E
 7    461    614 1     B
 8    195    524 1     A
 9    198    768 1     D
10    122    537 1     A
# … with 8,176 more rows
```

```
R> cbp %>%
+   distinct(sex)
```

```
# A tibble: 2 × 1
  sex
  <fct>
1 M
2 F
```

Manipulate Cases: `slice()`

- It allows you to **select, remove, and duplicate rows**.

```
R> cbp %>% slice(1:3)
```

```
# A tibble: 3 × 7
    ind father mother year  sex   phenotype herd
  <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
1     1     NA     NA 0     M          37.7 E
2     2     NA     NA 0     M          35.8 B
3     3     NA     NA 0     M          28.4 A
```

```
R> cbp %>% slice((n()-5L):n())
```

```
# A tibble: 6 × 7
    ind father mother year  sex   phenotype herd
  <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
1  9995   8463   8849 9     F          71.3 E
2  9996   8013   8628 9     F          61.9 E
3  9997   8342   8677 9     F          58.4 A
4  9998   8088   8510 9     F          58.2 E
5  9999   8449   8685 9     F          56.7 D
6 10000   8296   8854 9     F          67.1 B
```

```
R> cbp %>% slice_min(phenotype, n = 3)
```

```
R> cbp %>% slice_sample(n = 5)
```

Manipulate Cases: `slice()`

- It allows you to **select, remove, and duplicate rows**.

```
R> cbp %>% slice(1:3)
```

```
# A tibble: 3 × 7
    ind father mother year  sex   phenotype herd
  <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
1     1     NA     NA 0     M          37.7 E
2     2     NA     NA 0     M          35.8 B
3     3     NA     NA 0     M          28.4 A
```

```
R> cbp %>% slice((n()-5L):n())
```

```
# A tibble: 6 × 7
    ind father mother year  sex   phenotype herd
  <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
1  9995   8463   8849 9     F          71.3 E
2  9996   8013   8628 9     F          61.9 E
3  9997   8342   8677 9     F          58.4 A
4  9998   8088   8510 9     F          58.2 E
5  9999   8449   8685 9     F          56.7 D
6 10000   8296   8854 9     F          67.1 B
```

```
R> cbp %>% slice_min(phenotype, n = 3)
```

```
# A tibble: 3 × 7
    ind father mother year  sex   phenotype herd
  <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
1  1331    148    840 1     M          15.3 B
2  1316     62    551 1     M          20.2 D
3  1992    354    523 1     F          20.3 C
```

```
R> cbp %>% slice_sample(n = 5)
```

```
# A tibble: 5 × 7
    ind father mother year  sex   phenotype herd
  <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
1  3726   2367   2846 3     F          52.2 C
2  2087   1338   1820 2     M          59.7 D
3  2209   1402   1792 2     M          40.0 B
4  8205   7097   7904 8     M          72.5 B
5  2612   1129   1536 2     F          49.7 A
```

Manipulate Cases: `arrange()`

- **Orders the rows** of a data frame **by** the values of **selected columns**

```
R> cbp %>% arrange(desc(phenotype))
```

	ind	father	mother	year	sex	phenotype	herd
	<dbl>	<dbl>	<dbl>	<fct>	<fct>	<dbl>	<fct>
1	9069	8366	8725	9	M	82.1	C
2	9979	8305	8994	9	F	80.0	B
3	9492	8346	8985	9	M	79.1	B
4	9934	8470	8776	9	F	78.9	A
5	8650	7117	7726	8	F	78.6	B

```
R> cbp %>% arrange(desc(herd))
```

	ind	father	mother	year	sex	phenotype	herd
	<dbl>	<dbl>	<dbl>	<fct>	<fct>	<dbl>	<fct>
1	1	NA	NA	0	M	37.7	E
2	8	NA	NA	0	M	33.3	E
3	9	NA	NA	0	M	39.0	E
4	17	NA	NA	0	M	42.5	E
5	23	NA	NA	0	M	51.9	E

```
R> cbp %>% arrange(sex, phenotype, herd)
```

	ind	father	mother	year	sex	phenotype	herd
	<dbl>	<dbl>	<dbl>	<fct>	<fct>	<dbl>	<fct>
1	1992	354	523	1	F	20.3	C
2	618	NA	NA	0	F	20.8	C
3	1830	378	900	1	F	20.8	A
4	557	NA	NA	0	F	21.5	A
5	1663	102	983	1	F	21.9	B
6	800	NA	NA	0	F	22.2	A
7	1645	47	541	1	F	22.9	D

```
R> cbp %>% arrange(herd, phenotype, sex)
```

	ind	father	mother	year	sex	phenotype	herd
	<dbl>	<dbl>	<dbl>	<fct>	<fct>	<dbl>	<fct>
1	1830	378	900	1	F	20.8	A
2	557	NA	NA	0	F	21.5	A
3	95	NA	NA	0	M	22.2	A
4	800	NA	NA	0	F	22.2	A
5	1592	440	667	1	F	23.0	A
6	1192	355	642	1	M	24.7	A
7	1671	448	953	1	F	25.0	A

Manipulate Cases: add_row()

```
R> df <- tibble(x = 1:4, y = 20:23)
R> df %>% add_row(x = 3, y =21)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <dbl>
1     1    20
2     2    21
3     3    22
4     4    23
5     3    21
```

```
R> df %>% add_row(x = 3, y =21, .before = 4)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <dbl>
1     1    20
2     2    21
3     3    22
4     3    21
5     4    23
```

```
R> df %>% add_row(x = 3, .before = 4)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <int>
1     1    20
2     2    21
3     3    22
4     3    NA
5     4    23
```

Manipulate Cases: add_row()

```
R> df <- tibble(x = 1:4, y = 20:23)
R> df %>% add_row(x = 3, y =21)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <dbl>
1     1    20
2     2    21
3     3    22
4     4    23
5     3    21
```

```
R> df %>% add_row(x = 3, y =21, .before = 4)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <dbl>
1     1    20
2     2    21
3     3    22
4     3    21
5     4    23
```

```
R> df %>% add_row(x = 3, .before = 4)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <int>
1     1    20
2     2    21
3     3    22
4     3    NA
5     4    23
```

Manipulate Cases: add_row()

```
R> df <- tibble(x = 1:4, y = 20:23)
R> df %>% add_row(x = 3, y =21)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <dbl>
1     1    20
2     2    21
3     3    22
4     4    23
5     3    21
```

```
R> df %>% add_row(x = 3, y =21, .before = 4)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <dbl>
1     1    20
2     2    21
3     3    22
4     3    21
5     4    23
```

```
R> df %>% add_row(x = 3, .before = 4)
```

```
# A tibble: 5 × 2
      x     y
  <dbl> <int>
1     1    20
2     2    21
3     3    22
4     3    NA
5     4    23
```

Manipulate Variables: `pull()`

- `pull()` is similar to $

```
R> df %>% pull(var = x)
```

```
[1] 1 2 3 4
```

```
R> df %>% pull(var = x) %>% sum()
```

```
[1] 10
```

Manipulate Variables: `select()`

- **Select variables** in a data frame

```
R> cbp %>% select(ind:mother)
```

	ind	father	mother
	<dbl>	<dbl>	<dbl>
1	1	NA	NA
2	2	NA	NA
3	3	NA	NA
4	4	NA	NA
5	5	NA	NA

```
R> cbp %>%
+   select(starts_with(c("p","m")) & !ends_with("r"))
```

	phenotype
	<dbl>
1	37.7
2	35.8
3	28.4
4	33.6
5	32.9

```
R> cbp %>%
+   select(starts_with(c("p","m"))| ends_with("r"))
```

	phenotype	mother	father	year
	<dbl>	<dbl>	<dbl>	<fct>
1	37.7	NA	NA	0
2	35.8	NA	NA	0
3	28.4	NA	NA	0
4	33.6	NA	NA	0
5	32.9	NA	NA	0

```
R> cbp %>% select(contains("a"))
```

	father	year
	<dbl>	<fct>
1	NA	0
2	NA	0
3	NA	0
4	NA	0
5	NA	0

Manipulate Variables: `relocate()`

Change column positions

```
R> cbp %>%
+   relocate(mother, .before = father) %>%
+   relocate(year, herd, .after = sex)
```

```
# A tibble: 10,000 × 7
     ind mother father sex    year  herd  phenotype
   <dbl>  <dbl>  <dbl> <fct>  <fct> <fct>     <dbl>
 1     1     NA     NA M      0     E          37.7
 2     2     NA     NA M      0     B          35.8
 3     3     NA     NA M      0     A          28.4
 4     4     NA     NA M      0     D          33.6
 5     5     NA     NA M      0     A          32.9
 6     6     NA     NA M      0     A          31.6
 7     7     NA     NA M      0     A          38.8
 8     8     NA     NA M      0     E          33.3
 9     9     NA     NA M      0     E          39.0
10    10     NA     NA M      0     C          46.0
# … with 9,990 more rows
```

```
R> cbp %>% relocate(where(is.factor),
+                   .after = last_col())
```

```
# A tibble: 10,000 × 7
     ind father mother phenotype year  sex   herd
   <dbl>  <dbl>  <dbl>     <dbl> <fct> <fct> <fct>
 1     1     NA     NA      37.7 0     M     E
 2     2     NA     NA      35.8 0     M     B
 3     3     NA     NA      28.4 0     M     A
 4     4     NA     NA      33.6 0     M     D
 5     5     NA     NA      32.9 0     M     A
 6     6     NA     NA      31.6 0     M     A
 7     7     NA     NA      38.8 0     M     A
 8     8     NA     NA      33.3 0     M     E
 9     9     NA     NA      39.0 0     M     E
10    10     NA     NA      46.0 0     M     C
# … with 9,990 more rows
```

Manipulate Variables: `mutate()`

- Adds new variables and **preserves existing ones**
- **Variables can be removed** by setting their value to **NULL**

```
R> cbp %>% select(sex, phenotype) %>%
+   mutate(logPheno = log10(phenotype))
```

	sex <fct>	phenotype <dbl>	logPheno <dbl>
1	M	37.7	1.58
2	M	35.8	1.55
3	M	28.4	1.45
4	M	33.6	1.53
5	M	32.9	1.52

```
R> cbp %>% select(sex, phenotype) %>%
+   mutate(rankPheno = min_rank(desc(phenotype))) %>%
+   arrange(rankPheno)
```

	sex <fct>	phenotype <dbl>	rankPheno <int>
1	M	82.1	1
2	F	80.0	2
3	M	79.1	3
4	F	78.9	4
5	F	78.6	5

```
R> cbp %>%
+   mutate(across(!phenotype, as.factor))%>%
+   mutate(phenotype = NULL)
```

	ind <fct>	father <fct>	mother <fct>	year <fct>	sex <fct>	herd <fct>
1	1	NA	NA	0	M	E
2	2	NA	NA	0	M	B
3	3	NA	NA	0	M	A
4	4	NA	NA	0	M	D
5	5	NA	NA	0	M	A

```
R> cbp %>% select(sex, herd, phenotype) %>%
+   mutate(herdSex = sex:herd) %>%
+   relocate(herdSex, .before = phenotype)
```

	sex <fct>	herd <fct>	herdSex <fct>	phenotype <dbl>
1	M	E	M:E	37.7
2	M	B	M:B	35.8
3	M	A	M:A	28.4
4	M	D	M:D	33.6
5	M	A	M:A	32.9

Manipulate Variables: `transmute()`

- Adds new variables and **drops existing ones**

```
R> cbp %>% select(sex, phenotype) %>%
+   transmute(logPheno = log10(phenotype))
```

	logPheno
	<dbl>
1	1.58
2	1.55
3	1.45
4	1.53
5	1.52

```
R> cbp %>% select(sex, phenotype) %>%
+   transmute(rankPheno = min_rank(desc(phenotype))) %>%
+   arrange(rankPheno)
```

```
# A tibble: 5 × 1
  rankPheno
      <int>
1         1
2         2
3         3
4         4
5         5
```

```
R> cbp %>%
+   transmute(across(!phenotype, as.factor))%>%
+   mutate(phenotype = NULL)
```

	ind	father	mother	year	sex	herd
	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>
1	1	NA	NA	0	M	E
2	2	NA	NA	0	M	B
3	3	NA	NA	0	M	A
4	4	NA	NA	0	M	D
5	5	NA	NA	0	M	A

```
R> cbp %>% select(sex, herd, phenotype) %>%
+   transmute(herdSex = sex:herd)
```

	herdSex
	<fct>
1	M:E
2	M:B
3	M:A
4	M:D
5	M:A

Manipulate Variables: `across()`

- **Avoid using deprecated functions**: use `across()` instead of `mutate_at()` and `mutate_if()`.

```
R> cbp %>% mutate(across(phenotype, round, 2))
```

```
# A tibble: 10,000 × 7
     ind father mother year  sex   phenotype herd
   <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>
 1     1     NA     NA 0     M          37.7 E
 2     2     NA     NA 0     M          35.8 B
 3     3     NA     NA 0     M          28.4 A
 4     4     NA     NA 0     M          33.6 D
 5     5     NA     NA 0     M          32.9 A
 6     6     NA     NA 0     M          31.6 A
 7     7     NA     NA 0     M          38.8 A
 8     8     NA     NA 0     M          33.3 E
 9     9     NA     NA 0     M          39.0 E
10    10     NA     NA 0     M          46.0 C
# … with 9,990 more rows
```

```
R> fun <- function(x, na.rm = TRUE){
+    x - mean(x, na.rm = na.rm)
+  }
R> cbp %>%
+  mutate(year2 = as.numeric(year)-1) %>%
+  mutate(across(c(year2, phenotype), fun))
```

```
# A tibble: 10,000 × 8
     ind father mother year  sex   phenotype herd  year2
   <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct>  <dbl>
 1     1     NA     NA 0     M         -12.3 E       -4.5
 2     2     NA     NA 0     M         -14.2 B       -4.5
 3     3     NA     NA 0     M         -21.6 A       -4.5
 4     4     NA     NA 0     M         -16.4 D       -4.5
 5     5     NA     NA 0     M         -17.1 A       -4.5
 6     6     NA     NA 0     M         -18.4 A       -4.5
 7     7     NA     NA 0     M         -11.2 A       -4.5
 8     8     NA     NA 0     M         -16.7 E       -4.5
 9     9     NA     NA 0     M         -11.0 E       -4.5
10    10     NA     NA 0     M          -4.02 C      -4.5
# … with 9,990 more rows
```

Manipulate Variables: `rename()`

- `rename()` changes the names of individual variables
- `rename_with()` renames columns **using a function**

```
R> cbp %>% rename(Pheno = phenotype)
```

```
# A tibble: 5 × 7
    ind father mother year  sex   Pheno herd
  <dbl>  <dbl>  <dbl> <fct> <fct> <dbl> <fct>
1     1     NA     NA 0     M      37.7 E
2     2     NA     NA 0     M      35.8 B
3     3     NA     NA 0     M      28.4 A
4     4     NA     NA 0     M      33.6 D
5     5     NA     NA 0     M      32.9 A
```

```
R> cbp %>%
+   rename_with(~(toupper(gsub("r", "r2", .x,
+                              fixed = TRUE))))
```

```
# A tibble: 5 × 7
    IND FATHER2 MOTHER2 YEAR2 SEX   PHENOTYPE HER2D
  <dbl>   <dbl>   <dbl> <fct> <fct>     <dbl> <fct>
1     1      NA      NA 0     M          37.7 E
2     2      NA      NA 0     M          35.8 B
3     3      NA      NA 0     M          28.4 A
4     4      NA      NA 0     M          33.6 D
5     5      NA      NA 0     M          32.9 A
```

- Symbols `.` and `.x` means the same thing.
  - refers to **all the columns to which the formula in $\sim$ are applied**
- `funs()` has been soft-deprecated $\rightarrow$ $\sim$ (**purrr style formula**)

Group Cases: group_by()

- **Groups defined by variables**
- **Oerations** are performed "**by group**".

```
R> cbp %>%
+   group_by(herd) %>%
+   mutate(year2 = as.numeric(year)-1) %>%
+   mutate(across(c(year2, phenotype), fun))
```

```
# A tibble: 10,000 × 8
# Groups:   herd [5]
     ind father mother year  sex   phenotype herd  year2
   <dbl>  <dbl>  <dbl> <fct> <fct>     <dbl> <fct> <dbl>
 1     1     NA     NA 0     M         -12.4 E      -4.5
 2     2     NA     NA 0     M         -14.2 B      -4.5
 3     3     NA     NA 0     M         -21.6 A      -4.5
 4     4     NA     NA 0     M         -16.2 D      -4.5
 5     5     NA     NA 0     M         -17.2 A      -4.5
 6     6     NA     NA 0     M         -18.4 A      -4.5
 7     7     NA     NA 0     M         -11.2 A      -4.5
 8     8     NA     NA 0     M         -16.8 E      -4.5
 9     9     NA     NA 0     M         -11.1 E      -4.5
10    10     NA     NA 0     M          -4.11 C     -4.5
# … with 9,990 more rows
```

Statistics Summary

Count

- `dplyr::n()` - number of values or rows
- `dplyr::n_distinct()` - number of unique values
- `sum(!is.na())` - count number of non-NA's

Position

- `mean()` - mean of a sample
- `median()` - median of a sample

Rank

- `quantile()` - n-th quantile of the distribution
- `min()` - minimum value
- `max()` - maximum value

Count

- `dplyr::n()` - number of values or rows
- `dplyr::n_distinct()` - number of unique values
- `sum(!is.na())` - count number of non-NA's

Position

- `mean()` - mean of a sample
- `median()` - median of a sample

Rank

- `quantile()` - n-th quantile of the distribution
- `min()` - minimum value
- `max()` - maximum value

Count

- `dplyr::n()` - number of values or rows
- `dplyr::n_distinct()` - number of unique values
- `sum(!is.na())` - count number of non-NA's

Position

- `mean()` - mean of a sample
- `median()` - median of a sample

Rank

- `quantile()` - n-th quantile of the distribution
- `min()` - minimum value
- `max()` - maximum value

Summary Functions

Spread

- `IQR()` - Inter-Quartile Range
- `mad()` - median absolute deviation
- `sd()` - standard deviation
- `var()` - variance

Order

- `dplyr::first()` - first value
- `dplyr::last()` - last value
- `dplyr::nth()` - value in n-th location of vector

Summary Functions

Spread

- `IQR()` - Inter-Quartile Range
- `mad()` - median absolute deviation
- `sd()` - standard deviation
- `var()` - variance

Order

- `dplyr::first()` - first value
- `dplyr::last()` - last value
- `dplyr::nth()` - value in n-th location of vector

Summarise Cases: `summarise()`

```
R> s1 <- cbp %>% group_by(year, sex, herd) %>%
+   summarise( mean = mean(phenotype),
+              median = median(phenotype),
+              min = min(phenotype),
+              max = max(phenotype),
+              IQR = IQR(phenotype),
+              sd = sd(phenotype),
+              var = sd^2,
+              n = n()) %>%
+   mutate(across(where(is.numeric), round,1))
```

```
# A tibble: 100 × 11
# Groups:   year, sex [20]
   year  sex   herd   mean median   min   max   IQR    sd   var     n
   <fct> <fct> <fct> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 0     F     A      40.3   39.6  21.5  61.3  10.7   7.4  55.1   101
 2 0     F     B      40.6   40.8  24.9  56.7  10.2   6.9  47     114
 3 0     F     C      40.3   40.8  20.8  60.7   7.9   7.3  53.8    92
 4 0     F     D      39.1   39.8  24.3  56.7  10.6   7.1  51      94
 5 0     F     E      40.1   40.1  25.2  56.7   8     6.6  43.7    99
 6 0     M     A      41     40.6  22.2  55.2   9.6   7    49.3    99
 7 0     M     B      40.9   40.7  26.4  62.2   8     6.6  44      86
 8 0     M     C      40.7   39.1  21.6  57.9  10.8   7.7  59.8   108
 9 0     M     D      39.9   40.2  24.7  59.8   9.4   7    48.5   106
10 0     M     E      40.4   40.3  23.9  54.9   9.4   6.9  47.6   101
# … with 90 more rows
```

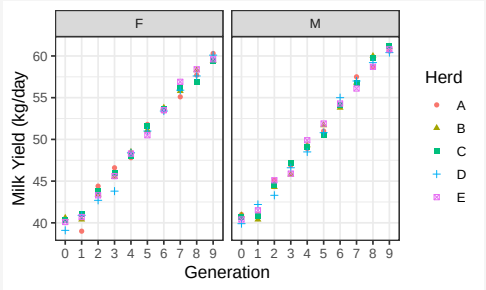
```
R> s1 %>%
+   ggplot(aes(y=mean, x =year)) +
+    geom_point(aes(shape =herd, colour = herd)) +
+    facet_wrap(~sex) +
+    labs(x = "Generation", y = "Milk Yield (kg/day",
+    colour = "Herd", shape = "Herd") +
+    theme_bw(base_size = 13)
```

Summarise Cases: summarise()

```
R> s1 <- cbp %>% group_by(year, sex, herd) %>%
+   summarise( mean = mean(phenotype),
+              median = median(phenotype),
+              min = min(phenotype),
+              max = max(phenotype),
+              IQR = IQR(phenotype),
+              sd = sd(phenotype),
+              var = sd^2,
+              n = n()) %>%
+   mutate(across(where(is.numeric), round,1))
```

```
# A tibble: 100 × 11
# Groups:   year, sex [20]
   year sex   herd   mean median   min   max   IQR    sd   var     n
   <fct> <fct> <fct> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 0    F     A      40.3   39.6  21.5  61.3  10.7   7.4  55.1   101
 2 0    F     B      40.6   40.8  24.9  56.7  10.2   6.9  47     114
 3 0    F     C      40.3   40.8  20.8  60.7   7.9   7.3  53.8    92
 4 0    F     D      39.1   39.8  24.3  56.7  10.6   7.1  51      94
 5 0    F     E      40.1   40.1  25.2  56.7   8     6.6  43.7    99
 6 0    M     A      41     40.6  22.2  55.2   9.6   7    49.3    99
 7 0    M     B      40.9   40.7  26.4  62.2   8     6.6  44      86
 8 0    M     C      40.7   39.1  21.6  57.9  10.8   7.7  59.8   108
 9 0    M     D      39.9   40.2  24.7  59.8   9.4   7    48.5   106
10 0    M     E      40.4   40.3  23.9  54.9   9.4   6.9  47.6   101
# … with 90 more rows
```

```
R> s1 %>%
+   ggplot(aes(y=mean, x =year)) +
+    geom_point(aes(shape =herd, colour = herd)) +
+    facet_wrap(~sex) +
+    labs(x = "Generation", y = "Milk Yield (kg/day",
+    colour = "Herd", shape = "Herd") +
+    theme_bw(base_size = 13)
```



Summarise Cases: summarise()

```
R> s1 %>%
+   filter(year %in% c("8","9") & max > 76 & min >40)
```

```
# A tibble: 7 × 11
# Groups:   year, sex [4]
  year  sex   herd   mean median   min   max   IQR    sd   var     n
  <fct> <fct> <fct> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 8     F     B      58.3   57.9  41.7  78.6   8.9   6.6  43.2   114
2 8     M     C      59.7   59.9  45.3  77.6   7.3   6    35.8   108
3 8     M     D      59.2   59.1  45.1  76.7   7.7   6.3  39.5   106
4 9     F     E      59.6   60.1  42.9  77.4   7.5   6.3  40.3    99
5 9     M     B      61.1   60.7  45    79.1  11.1   7.2  51.9    86
6 9     M     C      61.2   61    42.4  82.1   8.5   6.5  41.8   108
7 9     M     E      60.8   59.6  45.1  77.7   8.4   6.3  39.9   101
```

```
R> s1 %>%
+   filter(year %in% c("8","9") & max > 76 & min >40)
+   transmute(CV = sd / mean,
+             CV = scales::percent(CV)) %>%
+   mutate(across(where(is.numeric), round,1))
```

Summarise Cases: summarise()

```
R> s1 %>%
+   filter(year %in% c("8","9") & max > 76 & min >40)
```

```
# A tibble: 7 × 11
# Groups:   year, sex [4]
  year  sex   herd   mean median   min   max   IQR    sd   var     n
  <fct> <fct> <fct> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 8     F     B      58.3   57.9  41.7  78.6   8.9   6.6  43.2   114
2 8     M     C      59.7   59.9  45.3  77.6   7.3   6    35.8   108
3 8     M     D      59.2   59.1  45.1  76.7   7.7   6.3  39.5   106
4 9     F     E      59.6   60.1  42.9  77.4   7.5   6.3  40.3    99
5 9     M     B      61.1   60.7  45    79.1  11.1   7.2  51.9    86
6 9     M     C      61.2   61    42.4  82.1   8.5   6.5  41.8   108
7 9     M     E      60.8   59.6  45.1  77.7   8.4   6.3  39.9   101
```

```
R> s1 %>%
+   filter(year %in% c("8","9") & max > 76 & min >40)
+   transmute(CV = sd / mean,
+             CV = scales::percent(CV)) %>%
+   mutate(across(where(is.numeric), round,1))
```

```
# A tibble: 7 × 3
# Groups:   year, sex [4]
  year  sex   CV
  <fct> <fct> <chr>
1 8     F     11%
2 8     M     10.05%
3 8     M     10.64%
4 9     F     11%
5 9     M     11.78%
6 9     M     10.62%
7 9     M     10.36%
```

References

- Wickham, H. Tidy Data. **Journal of Statistical Software**, v. 59, no. 10, 2014

Useful links:

- `https://tidyr.tidyverse.org/`
- `https://tidyr.tidyverse.org/articles/pivot.html`
- `https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf`