Introduction to Data Wrangling

`tidyr`; `dplyr`

Dr Josh Hodge

Intended Learning Outcomes

By the end of this session students will be able to:

- Rearrange and extract variables and observations
- Calculate new variables within a dataframe
- Summarise data into group observations

Working Directory

- Specific directory or folder where files will be read from and written to.

`getwd()`	Returns file path of current working directory.
`setwd()`	Needs the file path of where you want to set your working directory.
`list.files() or dir()`	Lists all the files or folder in your working directory.

Importing Data

- Imports from the working directory

```
my_data<-read.csv("filename.csv")

my_data<-read.table("filename.csv", sep=",")

my_data<-read.table("filename.txt", sep="/t")
```

Other arguments: `header=TRUE/FALSE` **Take first row as column names**

`row.names=1` **Take first column as row names**

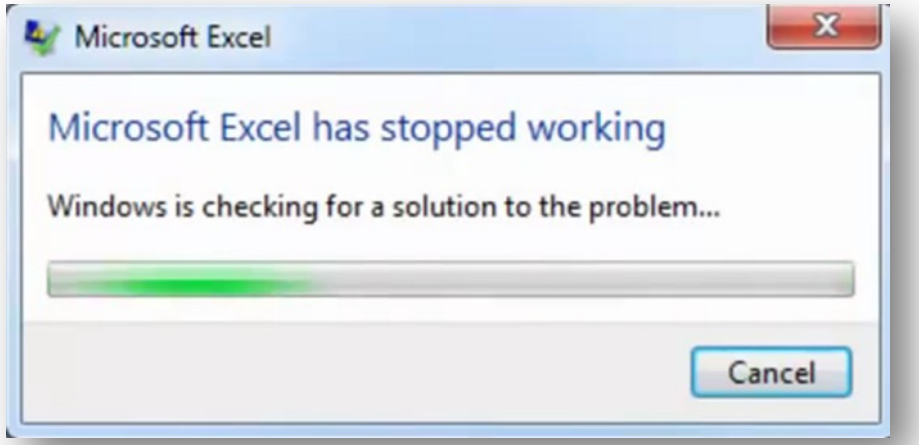
Exporting Data

- Exports to the working directory

```
write.csv(my_data, “filename.csv”)
```

Why is data wrangling important?

- 50-80% of your time



- Suitability to use with a particular software

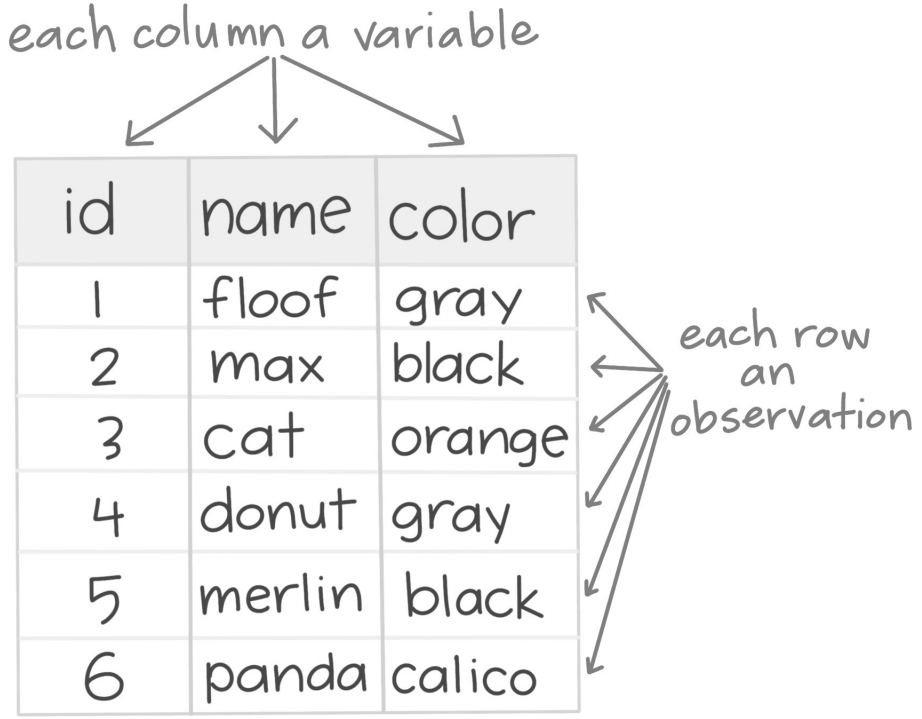
- Tidy Data in R

"TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."

–HADLEY WICKHAM

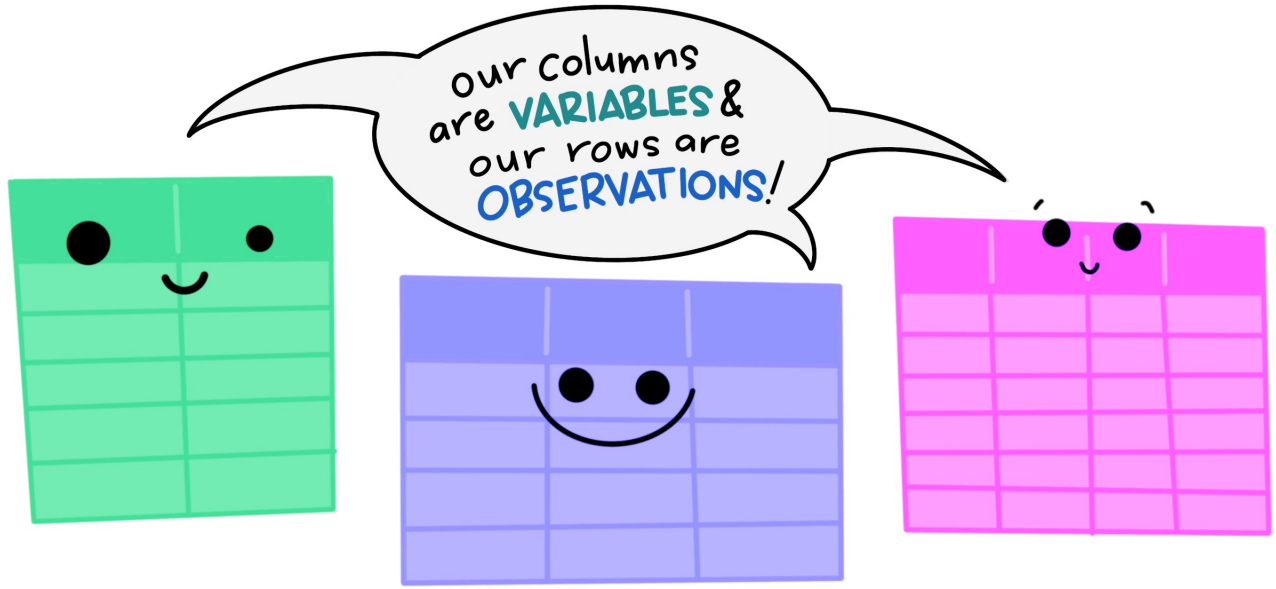
In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement



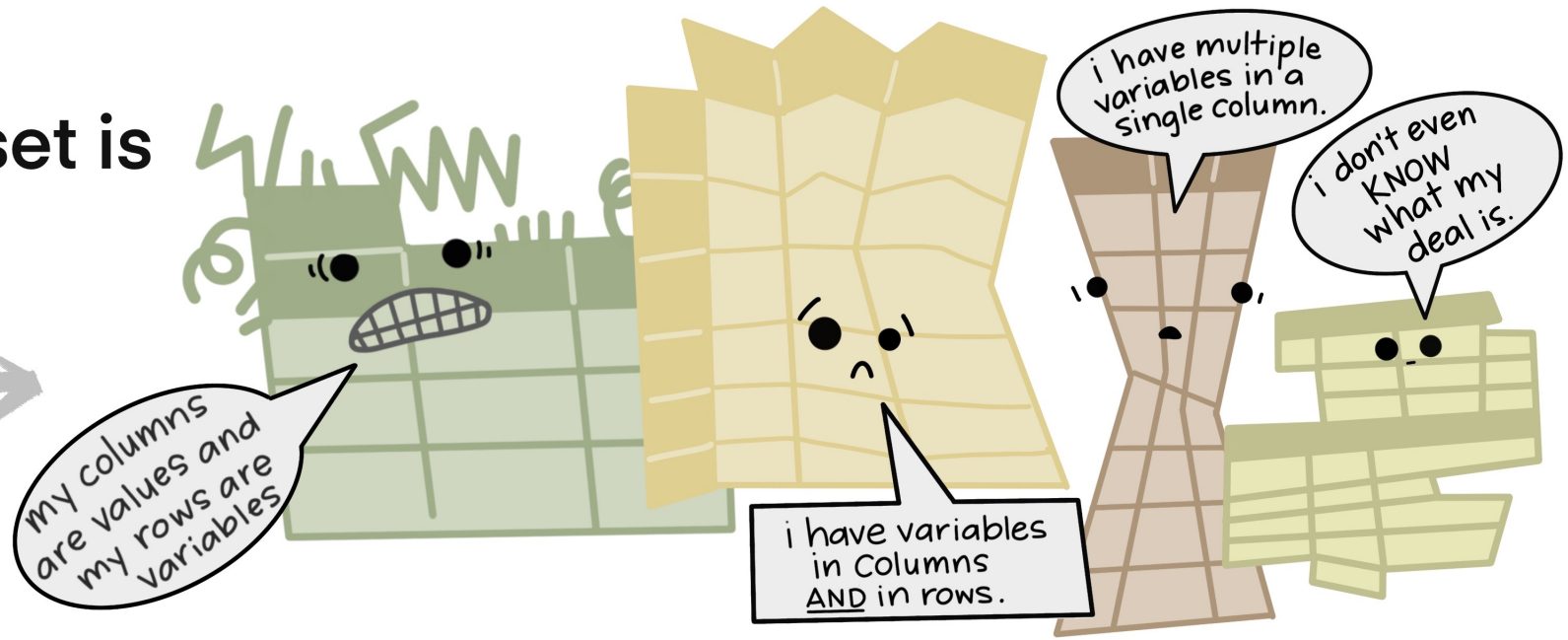
Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

The standard structure of tidy data means that "tidy datasets are all alike..."



"...but every messy dataset is messy in its own way."

–HADLEY WICKHAM



Wide vs Long Dataframes

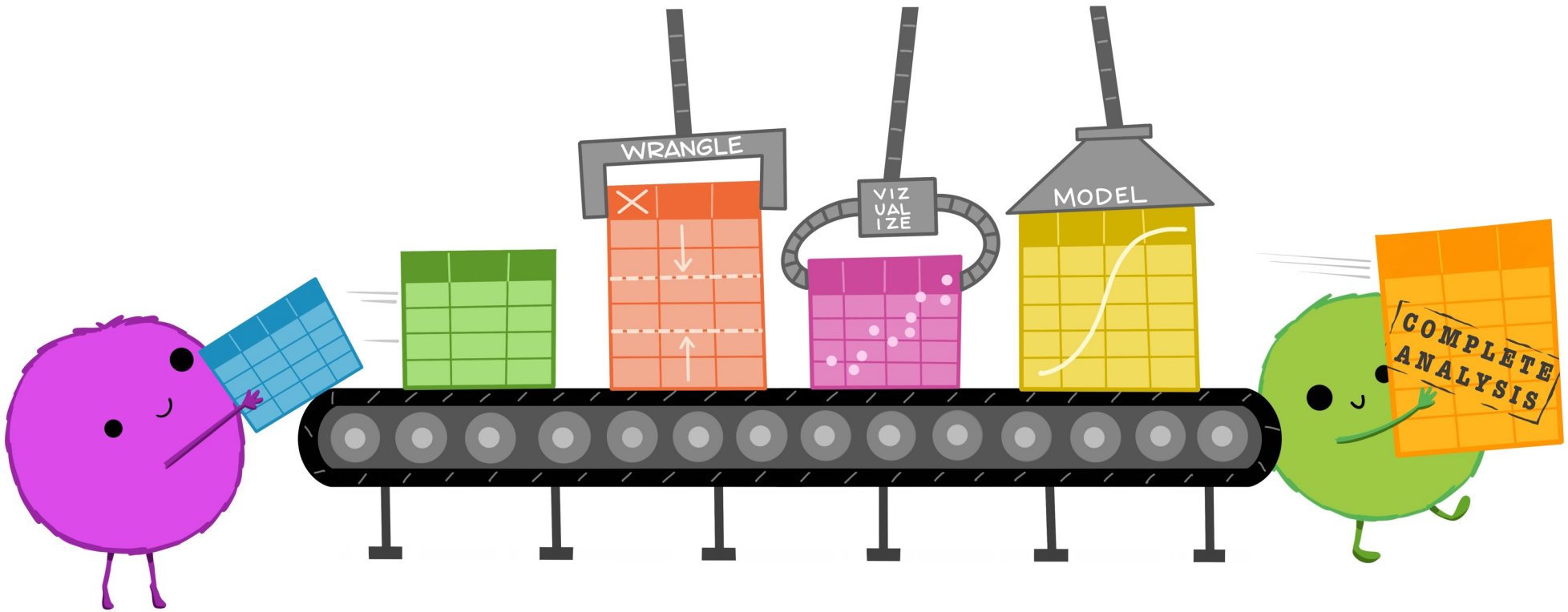
Site	SpDiv.1991	SpDiv.1992
1a	4.97	5.29
1b	5.92	5.39
1c	2.16	2.70
1d	1.03	1.42

Site	Year	SpDiv
1a	1991	4.97
1b	1991	5.92
1c	1991	2.16
1d	1991	1.03
1a	1992	5.29
1b	1992	5.39
1c	1992	2.70
1d	1992	1.42

`tidyr`

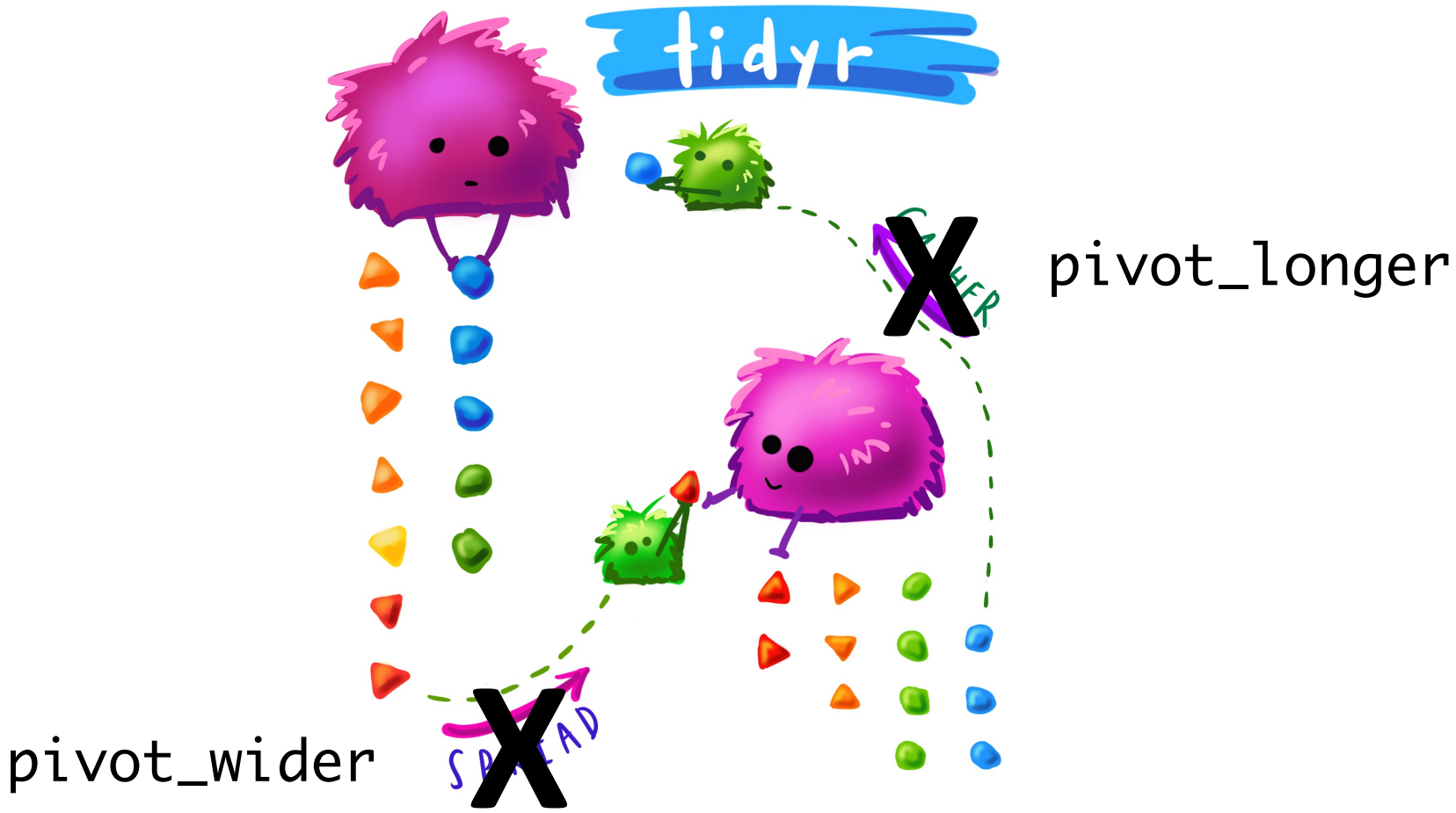
https://github.com/rstudio/cheatsheets/raw/master/data-import.pdf

1. Each variable is saved in its own column
2. Each observation is saved in its own row


WRANGLE
VIZ
UAL
IZE
MODEL
COMPLETE
ANALYSIS

Wide vs Long Dataframes

Wide vs Long Dataframes



pivot_wider and pivot_longer

Produces long dataframes:

```
data<-pivot_longer(dataframe,names_to="NewColumnName",
values_to="NewColumnName", cols=vector of column names)
```

Produces wide dataframes:

```
data<-pivot_wider(dataframe, names_from="FactorColumnName",
                  values_from="MeasureColumnName")
```

dplyr

dplyr

1. Arrange observations `?arrange`
2. Extract variables and observations `?select; ?filter`
3. Make new variables `?mutate`
4. Make groupie observations `?group_by; ?summarise`

Data and Dependencies

```
install.packages("dplyr")

library(dplyr)

data<-read.csv("/R/fertility.csv", header=T)
```

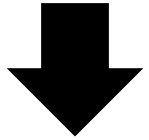
Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos

- AFC: Antral follicle count
- E2: Fertility level
- Gn: Gonadotropin level

arrange()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-arrange(fertility,Age)
```

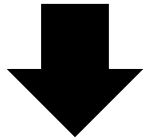


Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
21	21-25	27	27	3.8	32	859	150	1350	11	5
23	21-25	9	9.5	6.9	30	996	300	2175	9	5
23	21-25	7	7	5.7	38	1213	150	1500	12	1
24	21-25	14	14	3.8	54	1706	225	2100	13	7

arrange()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-arrange(fertility,desc(Age))
```

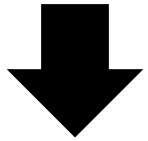


Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
46	46-50	7	7	3.9	63	1541	450	3600	7	4
46	46-50	2	2	9.4	65	1151	450	4050	7	2
45	41-45	9	10	7.6	32	290	525	5625	5	3
44	41-45	20	20	3.8	48	1667	225	1800	15	10

select()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-select(fertility,Age, FSH, Oocytes, Embryos)
```

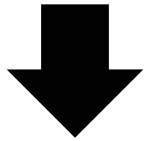


Age	FSH	Oocytes	Embryos
40	5.3	25	13
37	7.1	7	6
40	4.9	27	15
40	3.9	9	4

select()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-select(fertility,Age:MeanAFC)
```

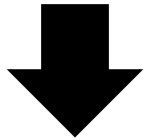


Age	AgeGroup	LowAFC	MeanAFC
40	36-40	40	51.5
37	36-40	41	41
40	36-40	38	41
40	36-40	36	37.5

select()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-select(fertility,-FSH)
```



Age	AgeGroup	LowAFC	MeanAFC	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	45	1427	300	2700	25	13
37	36-40	41	41	53	802	225	1800	7	6
40	36-40	38	41	40	4533	450	4850	27	15
40	36-40	36	37.5	26	1804	300	2700	9	4

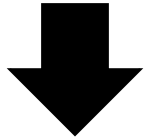
Useful `select()` Functions

`-`	Everything but
`:`	A range
`contains()`	Columns whose name contains a character string
`ends_with`	Columns whose name ends with a string
`everything()`	Every column
`matches()`	Columns whose name matches a regular express
`num_range()`	Columns names x1, x2, x3
`starts_with()`	Columns whose name starts with a character string

select()

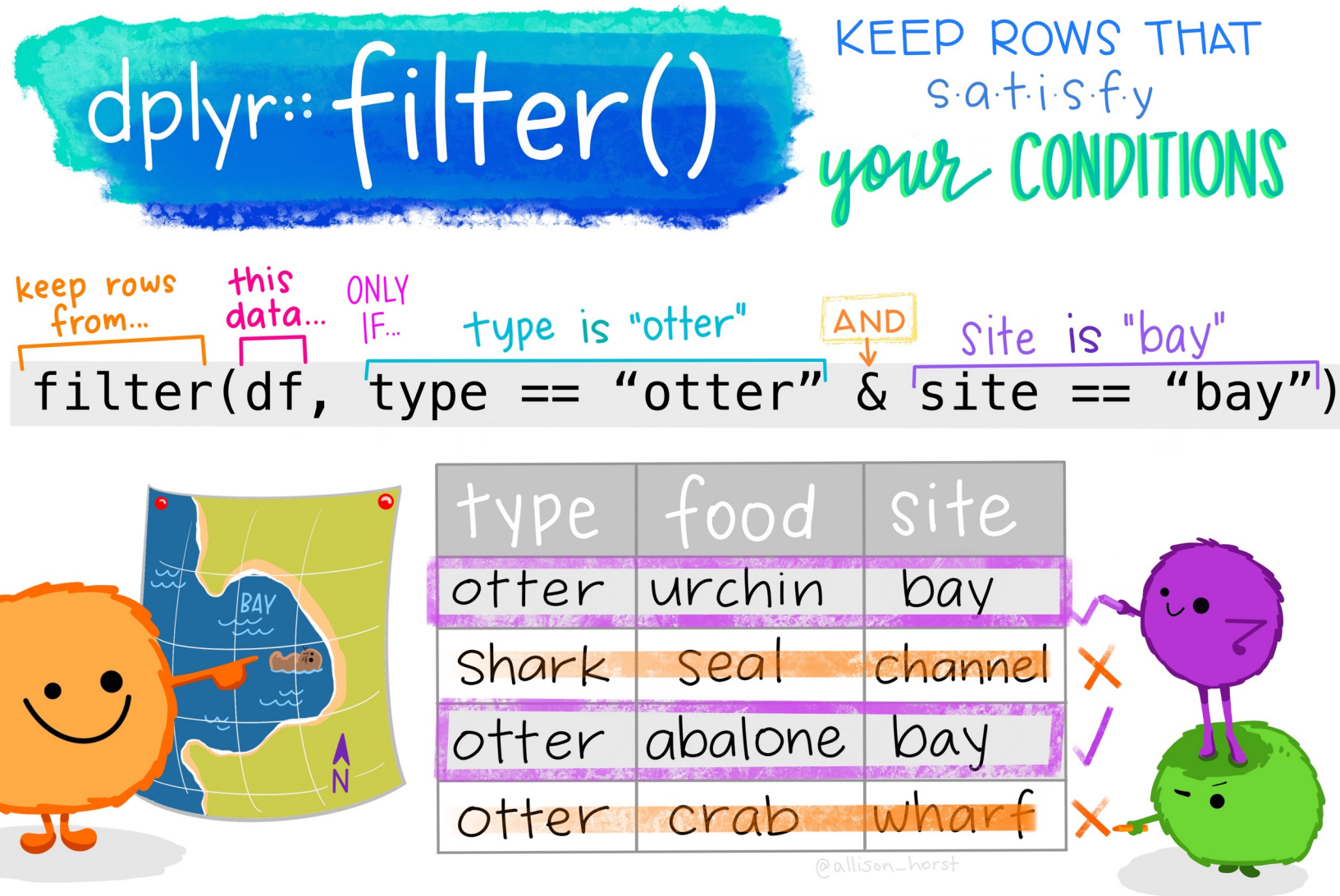
Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-select(fertility,contains("AFC"))
```



LowAFC	MeanAFC
40	51.5
41	41
38	41
36	37.5

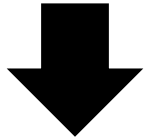
filter()


dplyr::filter()
KEEP ROWS THAT satisfy your CONDITIONS
keep rows from...
this data...
ONLY IF...
type is "otter"
AND
site is "bay"
filter(df, type == “otter” & site == “bay”)
BAY
N
type food site
otter urchin bay
shark seal channel
otter abalone bay
otter crab wharf
@allison_horst

filter()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-filter(fertility,Age<=30)
```



Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
30	26-30	36	36	4	49	2526	150	1500	19	12
29	26-30	35	35	3.9	67	3812	150	975	19	16
25	26-30	27	32	5	30	3458	125	825	22	13
30	26-30	18	31	4	33	2227	150	1463	27	18

Useful `filter()` Tests

`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Equal to
`!=`	Not equal to
`%in%`	Group membership
`!is.na`	Is not NA

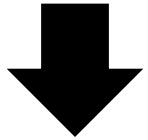
mutate()


dplyr::mutate
add Column(s),
keep existing.
A
B
C
1
5
10
0
2
-3
4
9
6
3
2
14
9
8
THE PLAN:
A
B
C
+
=
HORST '18

mutate()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-mutate(fertility,ratio = Embryos/Oocytes)
```

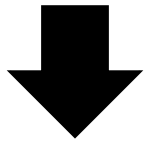


Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos	ratio
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13	0.52
37	36-40	41	41	7.1	53	802	225	1800	7	6	0.86
40	36-40	38	41	4.9	40	4533	450	4850	27	15	0.56
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4	0.44

summarise()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-summarise(fertility,mean.FSH=mean(FSH),sd.FSH=sd(FSH))
```



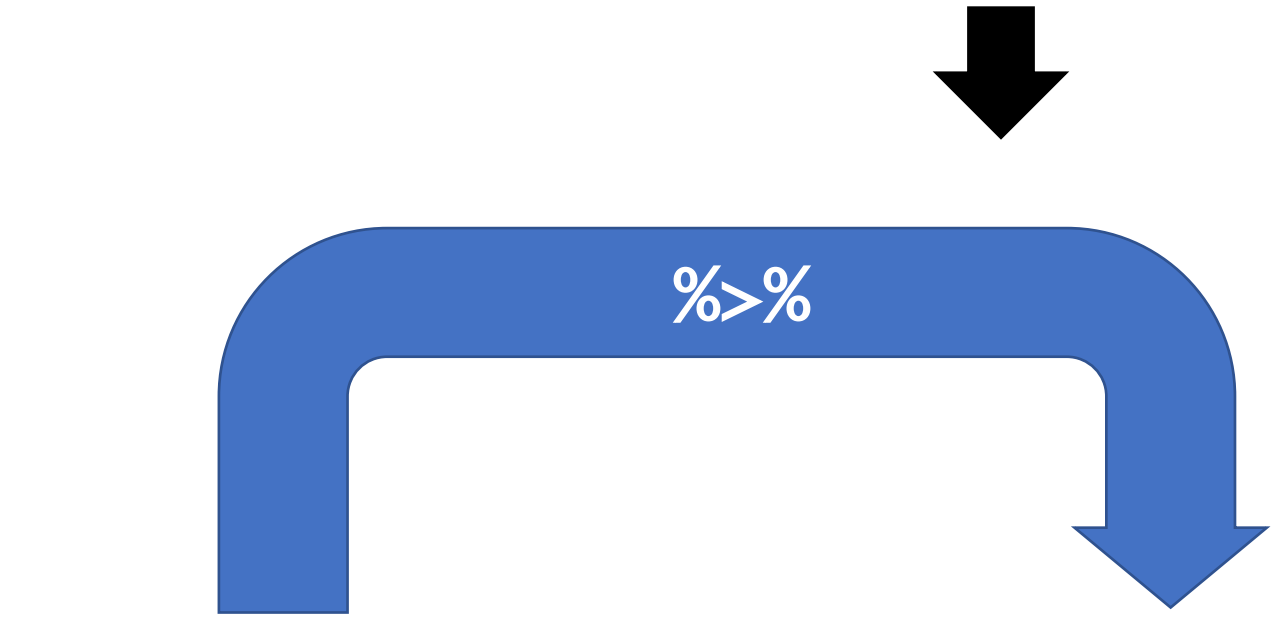
mean.FSH	sd.FSH
4.92	1.35

Useful `summarise()` Functions

`min()`	Minimum value
`max()`	Maximum value
`mean()`	Mean value
`median()`	Median value
`sum()`	Sum of Values
`var()`	Variance of a vector
`sd()`	Standard deviation of a vector
`n()`	Number of values in a vector

The Pipe Operator %>%

```
select(fertility,contains("AFC"))
```




```
fertility
```

```
select(____,contains("AFC"))
```

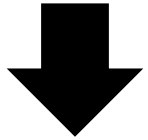
The Pipe Operator %>%

```
fertility%>%arrange(Age)

fertility%>%select(contains("AFC"))

fertility%>%filter(Age<=30)

fertility%>%mutate(ratio = Embryos/Oocytes)

fil1%>%summarise(mean.FSH=mean(FSH),sd.FSH=sd(FSH))
```

The Pipe Operator %>%

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<- fertility%>%filter(Age<=30)%>%select(Age:MeanAFC,-
AgeGroup)
```



Age	LowAFC	MeanAFC
30	36	36
29	35	35
25	27	32
30	18	31

group_by()

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
groups<-group_by(fertility, AgeGroup)
```

Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
21	21-25	27	27	3.8	32	859	150	1350	11	5
23	21-25	9	9.5	6.9	30	996	300	2175	9	5
23	21-25	7	7	5.7	38	1213	150	1500	12	1
24	21-25	14	14	3.8	54	1706	225	2100	13	7

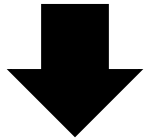
group_by()


(really important)
OTHER PARTS SOLD SEPARATELY!
dplyr::ungroup
DPLYR::group_by

group_by()+summarise()

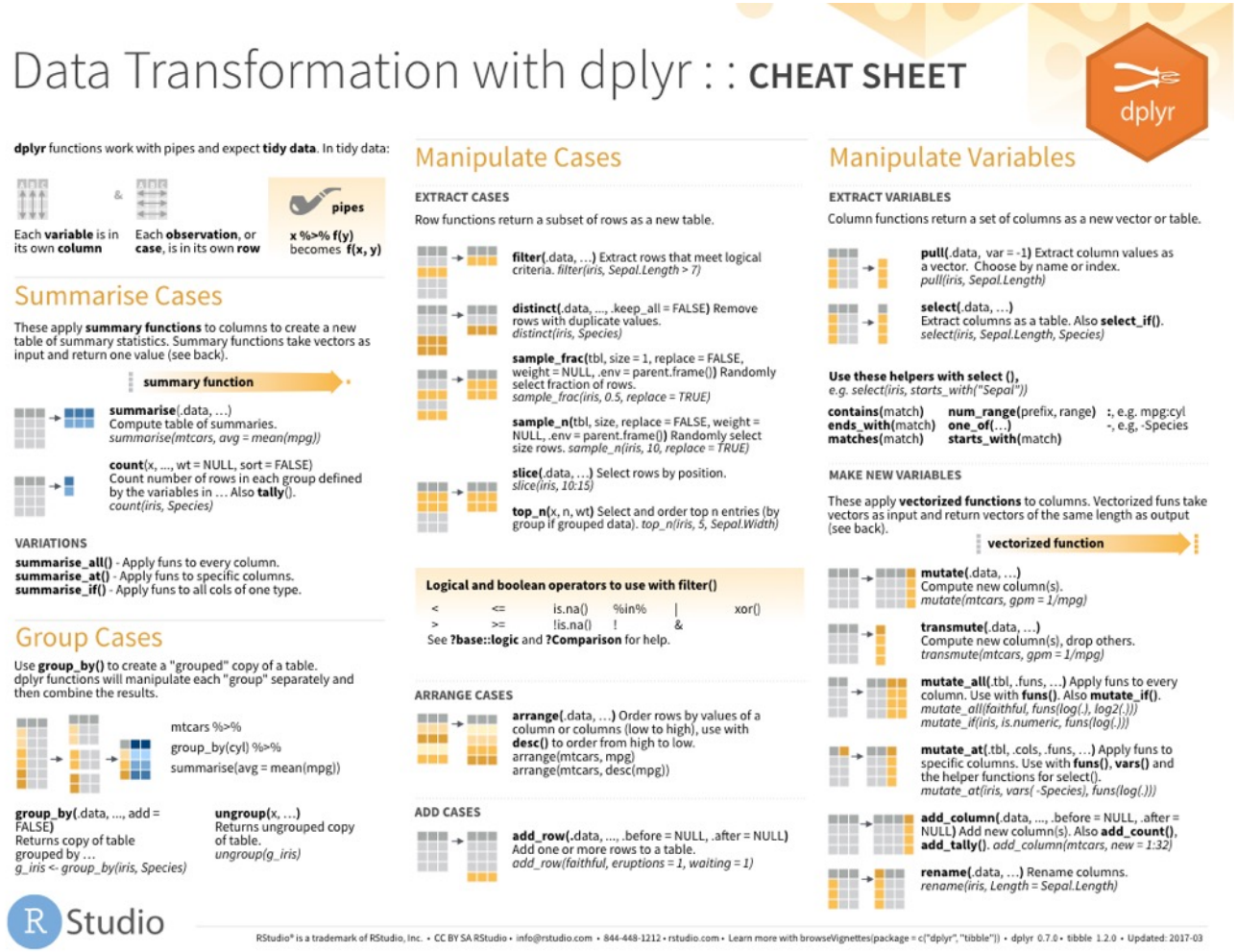
Age	AgeGroup	LowAFC	MeanAFC	FSH	E2	MaxE2	MaxDailyGn	TotalGn	Oocytes	Embryos
40	36-40	40	51.5	5.3	45	1427	300	2700	25	13
37	36-40	41	41	7.1	53	802	225	1800	7	6
40	36-40	38	41	4.9	40	4533	450	4850	27	15
40	36-40	36	37.5	3.9	26	1804	300	2700	9	4

```
new<-fertility%>%group_by(AgeGroup)%>%summarise(mean.FSH=mean(FSH),
                                                sd.FSH=sd(FSH))
```



AgeGroup	mean.FSH	sd.FSH
21-25	4.73	1.21
26-30	4.96	1.39
31-35	5.76	1.75
36-40	6.17	1.97
41-45	6.86	2.32
46-50	6.65	3.89

More dplyr



Data Transformation with dplyr : : CHEAT SHEET

dplyr

dplyr functions work with pipes and expect tidy data. In tidy data:

Each variable is in its own column. & Each observation, or case, is in its own row.

pipes: x %>% f(y) becomes f(x, y)

Summarise Cases

These apply summary functions to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …) Compute table of summaries. *summarise(mtcars, avg = mean(mpg))*

count(x, ..., wt = NULL, sort = FALSE) Count number of rows in each group defined by the variables in … Also tally(). *count(iris, Species)*

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use group_by() to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))

group_by(.data, ..., add = FALSE) Returns copy of table grouped by … *g_iris <- group_by(iris, Species)*

ungroup(x, …) Returns ungrouped copy of table. *ungroup(g_iris)*

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values. *distinct(iris, Species)*

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows. *sample_frac(iris, 0.5, replace = TRUE)*

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position. *slice(iris, 10:15)*

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	| xor()
>	>=	!is.na()	!	&

See ?base::logic and ?Comparison for help.

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with desc() to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL) Add one or more rows to a table. *add_row(faithful, eruptions = 1, waiting = 1)*

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index. *pull(iris, Sepal.Length)*

select(.data, …) Extract columns as a table. Also select_if(). *select(iris, Sepal.Length, Species)*

Use these helpers with select (), *e.g. select(iris, starts_with("Sepal"))*

contains(match) num_range(prefix, range) :, e.g. mpg:cyl
ends_with(match) one_of(…) -, e.g, -Species
matches(match) starts_with(match)

MAKE NEW VARIABLES

These apply vectorized functions to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, …) Compute new column(s). *mutate(mtcars, gpm = 1/mpg)*

transmute(.data, …) Compute new column(s), drop others. *transmute(mtcars, gpm = 1/mpg)*

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with funs(). Also mutate_if(). *mutate_all(faithful, funs(log(.), log2(.)))* *mutate_if(iris, is.numeric, funs(log(.)))*

mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with funs(), vars() and the helper functions for select(). *mutate_at(iris, vars(-Species), funs(log(.)))*

add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also add_count(), add_tally(). *add_column(mtcars, new = 1:32)*

rename(.data, …) Rename columns. *rename(iris, Length = Sepal.Length)*

RStudio

https://www.rstudio.com/resources/cheatsheets/#dplyr

What's next?

- Practical until end of the session

- Complete before next session:

15: Base Graphics