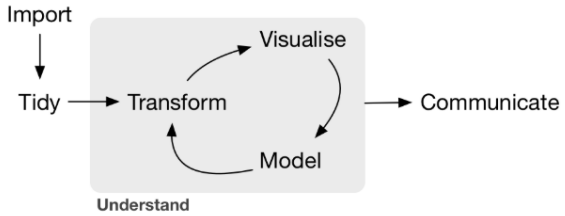
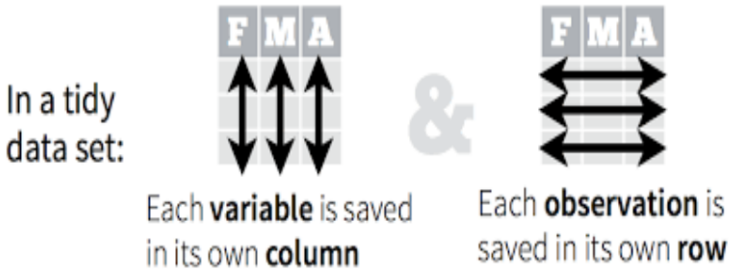
Data Transformation dplyr

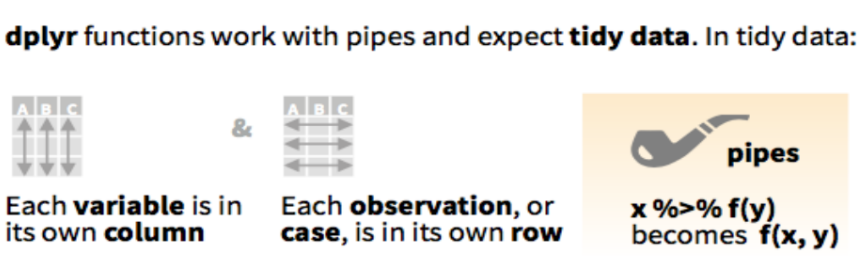
Dplyr

Dplyr : library(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 row





Dplyr …

Extract Cases: Manipulate Cases/ Rows

Row functions return a subset of rows as a new table. Use a variant that ends in_ for non standard evaluation friendly code

filter() - used to subset data with matching logical conditions.

filter(mydata, Index == "A")

filter(mydata6, Index %in% c("A", "C"))

#%in% operator can be used to select multiple items.

#Contains : looking for records wherein column state contains 'Ar' in their name

filter(mydata6, grepl("Ar", State))

filter(mydata6, Index %in% c("A", "C") & Y2002 >= 1300000)

filter(mydata6, Index %in% c("A", "C") | Y2002 >= 1300000)

filter(mydata6, !Index %in% c("A", "C"))

Manipulate Rows : Distinct

Remove Duplicate Rows based on all the variables (Complete Row) The distinct function is used to eliminate duplicates.

distinct(mydata)

#Remove Duplicate Rows based on a variable

distinct(mydata, Index, .keep_all= TRUE)

.keep_all function is used to retain all other variables in the output data frame.

Remove Duplicates Rows based on multiple variables In the example below, we are using two variables - Index, Y2010 to determine uniqueness.

x2 = distinct(mydata, Index, Y2010, .keep_all= TRUE)

Row Manipulations : sample rows

sample_frac function returns randomly N% of rows. In the example below, it returns randomly 10% of rows.

sample_frac(mydata,0.1)

sample_n function selects random rows from a data frame (or table). The second parameter of the function tells R the number of rows to select.

sample_n(mydata,3)

Rows : Sort

Sort Data by Multiple Variables : default sorting order of **arrange() function** is ascending . for descending order : desc(columnname)

```
arrange(mtcars, mpg)

arrange(mtcars, mpg, wt)

arrange(mtcars, desc(mpg), wt)
```

Manipulate Variables : Extract Variables

select(data , ...)
.... : Variables by name or by function

rename(data , new_name = old_name)
rename(mydata, Index1=Index)

select(mydata, starts_with("Y"))
select(mydata, -starts_with("Y"))
select(mydata, contains("I"))

select(mydata, State, everything())
keeps variable **'State'** in the front and the remaining variables follow that.

minus sign before a variable tells R to drop the variable.

mydata = select(mydata, -Index, -State)
mydata = select(mydata, -c(Index,State))

Do Function

Use : Compute within groups

do(data_frame, expressions_to_apply_to_each_group)

The dot (.) is required to refer to a data frame.

Filter Data within a Categorical Variable

pull top 2 rows from 'A', 'C' and 'I' categories of variable Index.

t = mydata %>% filter(Index %in% c("A", "C","I")) %>% group_by(Index) %>% do(head(. , 2))

Selecting 3rd Maximum Value by Categorical Variable

third maximum value of variable Y2015 by variable Index. The following code first selects only two variables Index and Y2015. Then it filters the variable Index with 'A', 'C' and 'I' and then it groups the same variable and sorts the variable Y2015 in descending order. At last, it selects the third row.

```
t = mydata %>% select(Index, Y2015) %>%  filter(Index %in% c("A", "C","I")) %>%  group_by(Index) %>%  do(arrange(.,desc(Y2015))) %>%  slice(3)
```

Selecting cases

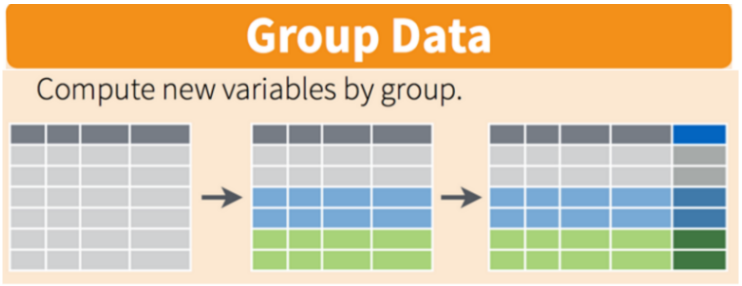
Helpers	Description
starts_with()	Starts with a prefix
ends_with()	Ends with a prefix
contains()	Contains a literal string
matches()	Matches a regular expression
num_range()	Numerical range like x01, x02, x03.
one_of()	Variables in character vector.
everything()	All variables.

Summaries

```
mtcars %>% summarise(mean(mpg), max(wt))
mtcars %>% summarise_all(mean)
mtcars %>% select(wt, gear)%>% summarise_all(c("min", "max"))
mtcars %>% summarise_all(funs(med = median))
mtcars %>% summarise_if(is.numeric, mean, na.rm = TRUE)
iris %>% summarise_if(is.numeric, mean, na.rm = TRUE)
#specific columns
mtcars %>% summarise_at(c("mpg", "wt"), mean, na.rm = TRUE)
```

Group Data

	am	`mean(mpg)`
	<dbl>	<dbl>
1	0	17.1
2	1	24.4

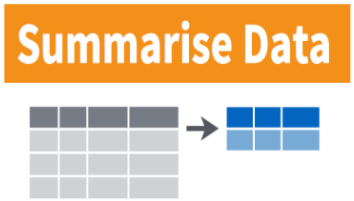



```
mtcars %>% group_by(am)
#nothing - just separation
mtcars %>% group_by(am) %>% summarise(mean(mpg))
```

	am	gear	`mean(mpg)`
	<dbl>	<dbl>	<dbl>
1	0	3	16.1
2	0	4	21.0
3	1	4	26.3
4	1	5	21.4

```
#subgroups
mtcars %>% group_by(am, gear) %>% summarise(mean(mpg))
```

Summarise with Group




```
mtcars %>% group_by(am) %>% summarise(mean(mpg), max(wt))
```

	am	`mean(mpg)`	`max(wt)`
	<dbl>	<dbl>	<dbl>
1	0	17.1	5.42
2	1	24.4	3.57

```
mtcars %>% group_by(am, gear) %>% summarise_all(mean)
```

```
mtcars %>% group_by(am) %>% summarise(mean(mpg), max(wt))
```


```
mtcars %>% group_by(am) %>% summarise(mean(mpg), max(wt))
```

Summary Functions

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output

COUNTS:
n() - number of values/ rows
n_distinct() - # of uniques
sum(!is.na() - # no of non-NAs

POSITION/ ORDER:
first() - first value
last() - last values
nth() - value in nth location of vector

RANK:
quantile() - nth quantile
min() - minimum value
max() - maximum value

LOCATION:
mean() - mean, also mean(!is.na()
median() - median

SPREAD:
IQR() - inter quartile range
mad() - mean absolute deviation
sd() - standard deviation
var() - variance

Row Names

Tidy data does not use row names, which store a variable outside of the columns. To work with the row names, first move them into columns

```
library(tibble)
```

tibble::rownnames_to_column() - Move row names into col

```
df= mtcars
df = tibble::rownames_to_column(df, var='cars')
```

```
has_rownames()
remove_rownames()
```

Subset Variables

Subset Variables (Columns)

`select()` to subset the data on variables or columns

```
#can select multiple columns just with a comma.
select(mtcars, mpg, wt)
```

```
#use - to deselect columns
select(mtcars, -am, -vs)
```

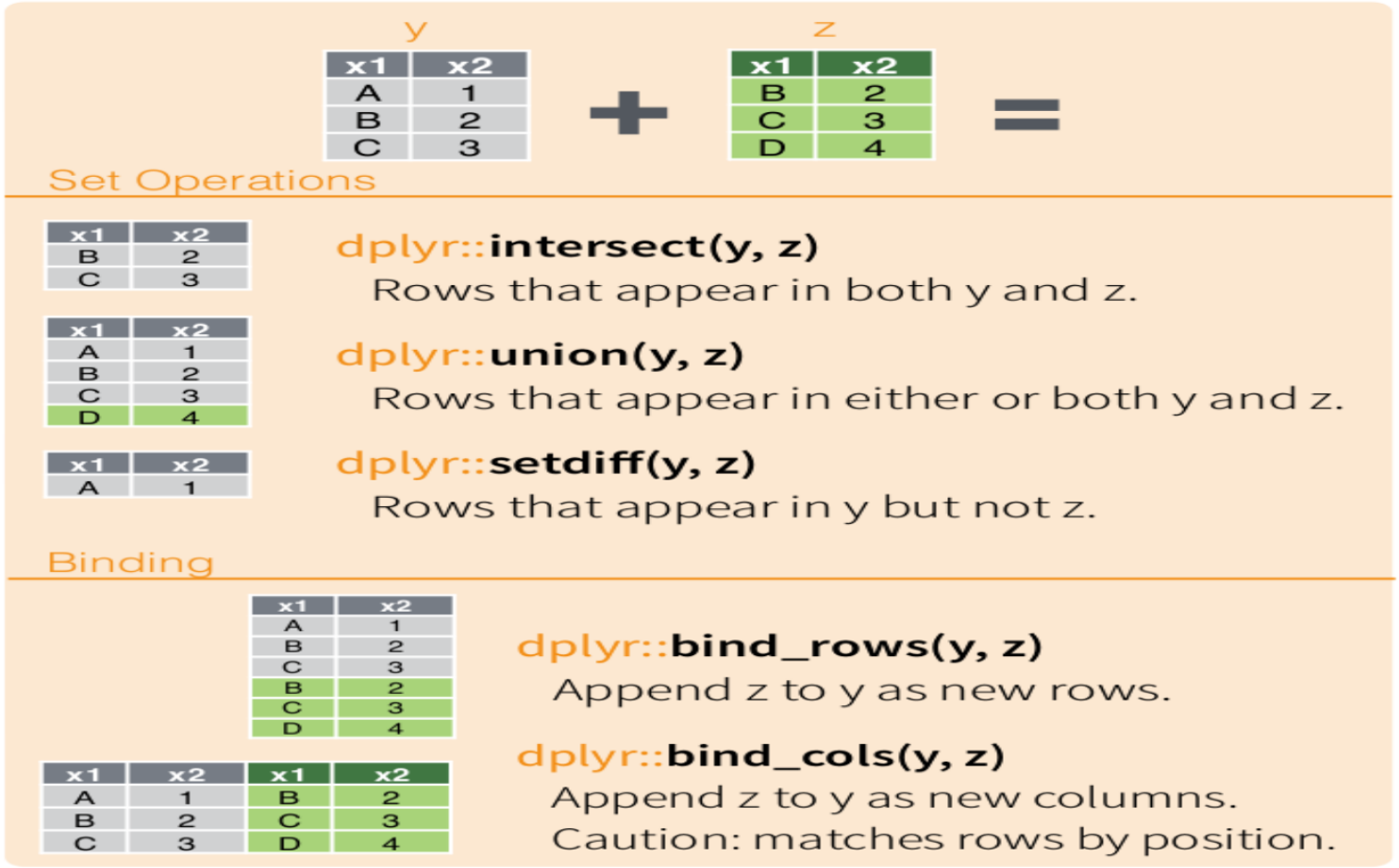
Make New Variable

Make New Variables

```
# add column with row number
 mtcars %>%    mutate(index = 1:nrow(mtcars),,
```

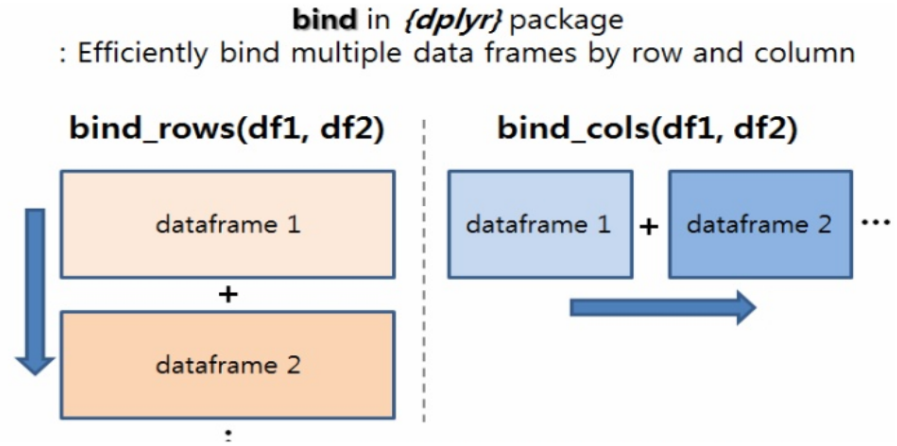
```
# add column with multiple mileage by 3
 mtcars %>%    mutate(mpg2 = mpg * 2)
```

Join Rows/Columns & Data Frames



bind_cols() / bind_rows()

Combine Data Frames



bind_rows(..., .id = NULL)
..)

- paste tables below each other as they are
- bind_rows(…, .id = NULL) Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names

bind_cols(...)

- paste tables beside each other as they are
- Returns tables placed side by side as a single table.
- BE SURE THAT ROWS ALIGN.

intersect/ union / set_diff

- **intersect(x, y, …)**
  Rows that appear in both x and y.

- **setdiff(x, y, …)**
  Rows that appear in x but not y.

- **union(x, y, …)**
  Rows that appear in x or y.
  (Duplicates removed).
- **union_all()**
  retains duplicates

Use **setequal()** to test whether two data sets contain the exact same rows (in any order)

Combine Data Sets

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

dplyr::**left_join(a, b, by = "x1")**

Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

dplyr::**right_join(a, b, by = "x1")**

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

dplyr::**inner_join(a, b, by = "x1")**

Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

dplyr::**full_join(a, b, by = "x1")**

Join data. Retain all values, all rows.

Vector Functions

Mutate and transmutate apply vectorised functions to columns to create new columns.

Vectorised Functions take vectors as input and return vectors of the same length as output

dplyr::lag() - offset elements by 1
dplyr::lead() - offset elements by -1

cummall() - cumulative all
cumany() - cumulative any
cummax() - cumulative max
cummean() - cummulative mean

cumprod() - cumulative product
cumsum() - cumulative sum

Vector Functions 2

Math:

+, -, *, / , %/%, %% - arithmetic

log(), log2(), log10 - logs

<, <=, >, >=, !=, == logical comparisons

Rankings:

cume_dist() - proportion of all values

dense_rank() - rank with ties, min, no gaps

min_rank() - rank with ties = min

ntile - bins into n bins

percent_rank() - mn_rank scaled to [0,1]

row_number() - rank with ties =”first”

Misc:

between() - x>= left & x<= right

case_when() - multi-case if_else()

coalesce() - first non NA values by element across set of vectors

if_else() - elementwise if + else()

na_if() - replace specific na values

Misc:

pmax() - element wise max()

pmin() - element wise min

recode() - vectorised switch

recode_factor() - vectorised switch for factors

Summaries & mutate multiple columns

summarise_all(), mutate_all() and transmute_all() apply the functions to all (non-grouping) columns.

summarise_at(), mutate_at() and transmute_at() allow you to select columns using the same name-based select_helpers just like with select().

summarise_if(), mutate_if() and transmute_if() operate on columns for which a predicate returns TRUE

Imp dplyr functions

Important dplyr Functions to remember

dplyr Function	Description	Equivalent SQL
select()	Selecting columns (variables)	SELECT
filter()	Filter (subset) rows.	WHERE
group_by()	Group the data	GROUP BY
summarise()	Summarise (or aggregate) data	-
arrange()	Sort the data	ORDER BY
join()	Joining data frames (tables)	JOIN
mutate()	Creating New Variables	COLUMN ALIAS

Further Reading..

http://ohi-science.org/data-science-training/dplyr.html

https://www.listendata.com/2016/08/dplyr-tutorial.html

https://www.red-gate.com/simple-talk/sql/reporting-services/using-r-to-explore-data-by-analysis-for-sql-professionals/

http://rpubs.com/williamsurles/293454

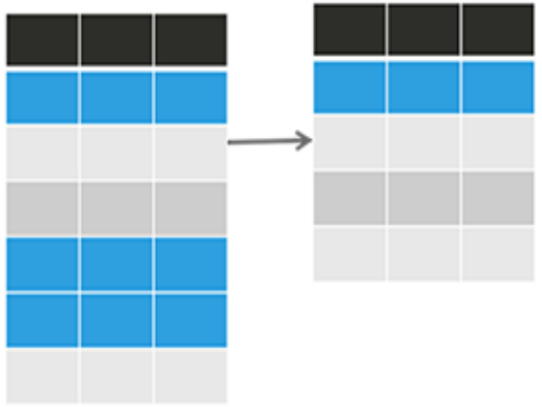
http://www3.econ.muni.cz/~137451/files/dplyr.html

Subsetting Data Frame Rows in R

- **filter()**: Select rows based on some criteria
- **sample_n()** and **sample_frac()**: Select random rows
- **top_n()**: Select top elements based on values

dplyr R package

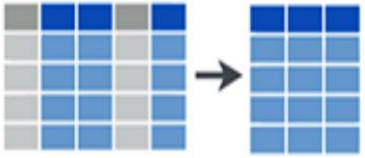
Removing Duplicate Data in R



- **duplicated()**: Find duplicate elements (R base)
- **unique()**: Keep only unique elements (R base)
- **dplyr::distinct()**: Keep only unique elements (more efficient than **unique()**)

R base and dplyr

Subsetting Data Frame Columns in R



- **select()**: Select columns by name or helper functions
- **Helper functions:** starts_with(), ends_with(), contains(), matches(), one_of()

dplyr R package

Computing and Adding new Variable(s) to a Data Frame

- **dplyr::mutate()**: Computes and adds new variables. Preserves existing variables.
- **dplyr::transmute()**: Computes new variable(s). Drops existing variables.
- **transform()**: R base function similar to dplyr::mutate().

R base and dplyr


Visualize
ggplot2 package
Tidy
Transform
dplyr package
tidyr package
Model
(trial and error)

Thanks