Introduction to R Tidyverse

Andrew Redd, PhD. VINCI Helpdesk R Expert

Tidyverse Overview

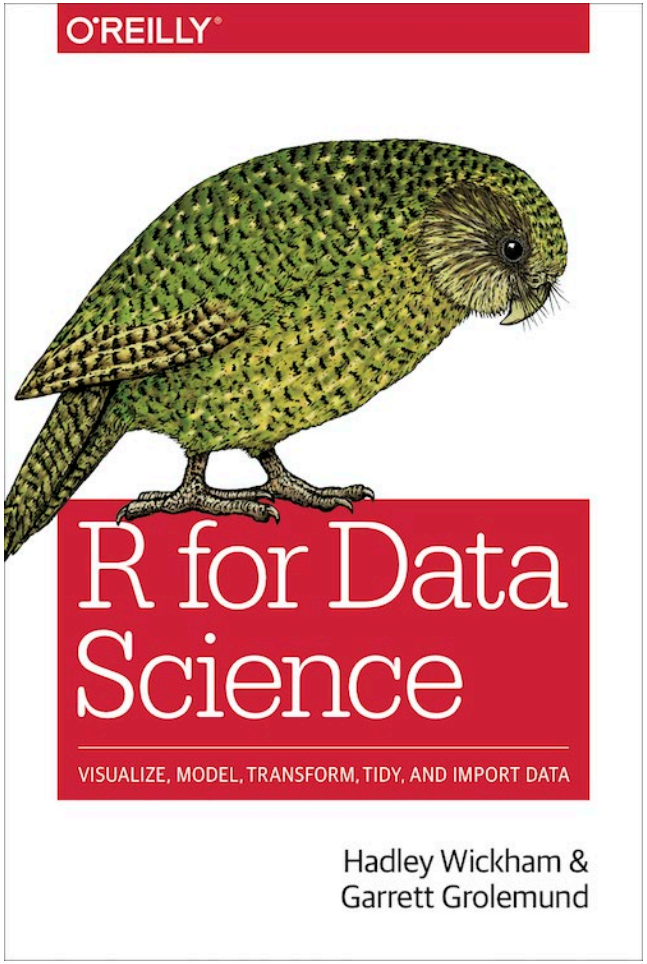
R For Data Science

What is the Tidyverse?

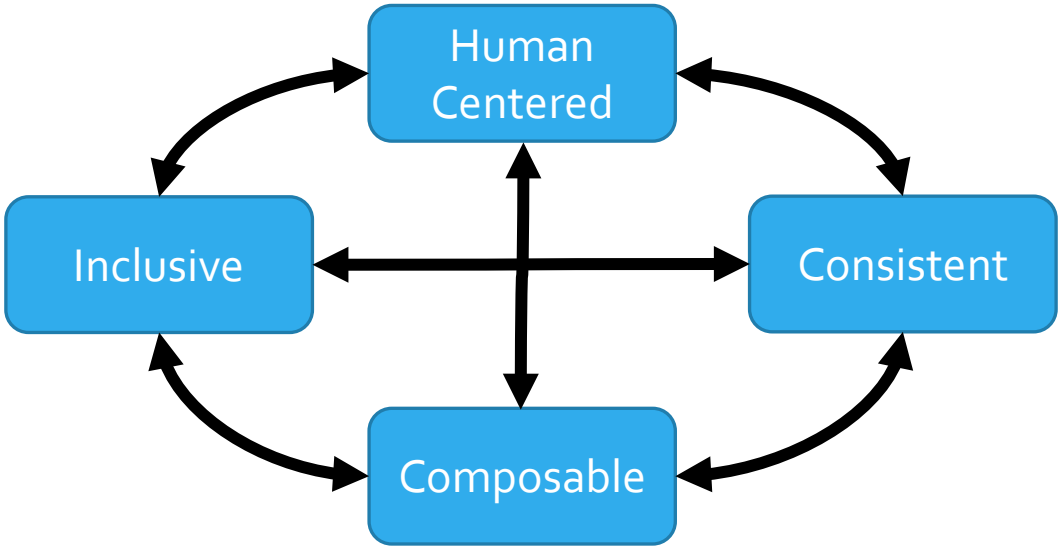
- Cohesive set of packages to handle common aspects of data analysis.
  - Import
  - Wrangle
  - Program
  - Visualize
  - Model
  - Communicate

The Book

- Free online (https://r4ds.had.co.nz)
- Also available in print



Unifying Principles

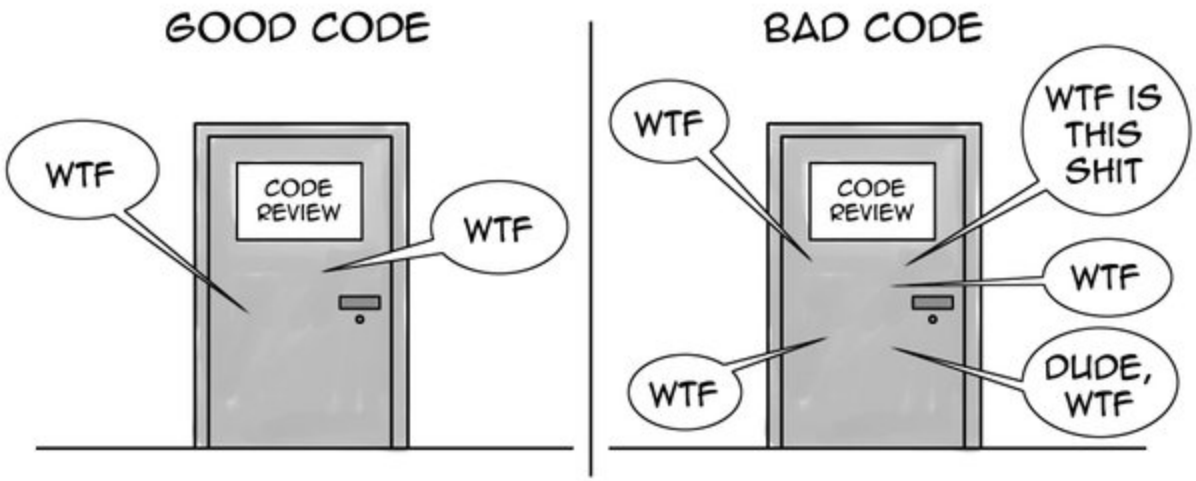


Related Work

- The Unix philsophy
- The Zen of Python
- Design Principles Behind Smalltalk

Principle: Human Centered

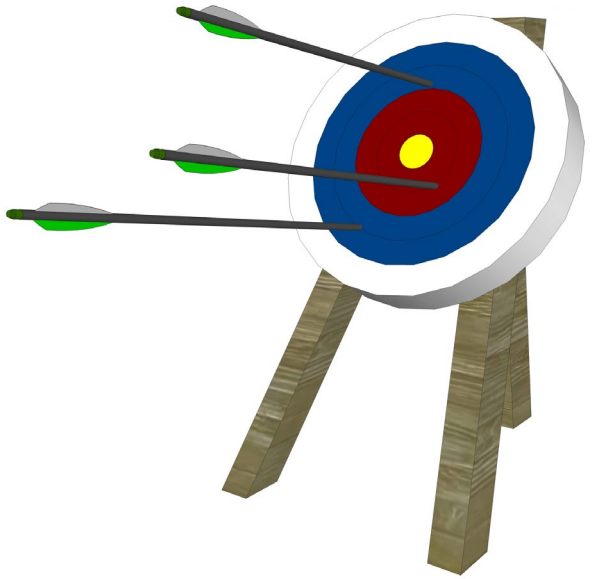
- Optimize for people, then the computer, not the other way around.
- Minimize cognitive load
- **Consistency** to minimize special cases
- **Composable** to break the task into bite sized pieces
- **Inclusively** design with all people in mind



THE ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE

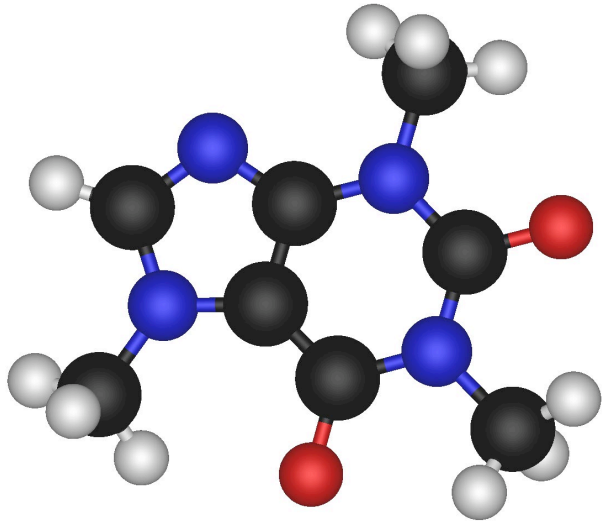
Principle: Consistent

- Consistency minimizes the cognitive load.
- Interfaces
  - Functions should be **composable**.
  - should feel "functional" for **humans**
    - Embrace functional programming
- Data Structures
  - Standardized format, the 'tibble'
- Consistency > Performance



Principle: Composable

- Complex from many simple pieces
  - Molecules from atoms
- Decompose large problems into smaller tractable ones.
- Do only one thing but do it well.
- Align input and output
- Dense feature matrices



Principle: Inclusive

- Tidyverse is a community in the larger R ecosystem
- Not everyone is like me
  - Education
  - Background
  - Skill level
  - Language
  - Visually or otherwise impaired

%>%

The Pipe

%>% - The pipe

- Embodies the principles of Human Centric and Composability
  - Operations read from left to right as a workflow, not inside to outside
  - Avoids nesting
  - Minimizes local variables
  - Simplifies adding/removing steps in a workflow

Basic Usage

Pipe format

- `x %>% f`
- `x %>% f(y)`
- `x %>% f %>% g %>% h`
- Placeholder
  - `x %>% f(y, .)`
  - `x %>% f(y, z=.)`

Traditional

- `f(x)`
- `f(x, y)`
- `h(g(f(x)))`
- `f(x,y)`
- `f(y, z=x)`

Import

Getting data into R

Core Tidyverse: readr

- **`read_csv()`**: comma separated (CSV) files
- **`read_tsv()`**: tab separated files
- **`read_delim()`**: general delimited files
- **`read_fwf()`**: fixed width files
- **`read_table()`**: tabular files where columns are separated by white-space.
- **`read_log()`**: web log files

readr Example

Input

```
readr::read_csv("data/confirmed.csv")
```

Output

```
-- Column specification --------------------------------
cols(
  .default = col_double(),
  `Province/State` = col_character(),
  `Country/Region` = col_character()
)
i Use `spec()` for the full column specifications.
```

readr v. base

readr

- consistent
  - col_types, col_names
- faster
- strings as string
- automatic progress bar

base

- inconsistent
  - colClasses, header
- slow
- strings are factors

Foreign data

- SAS: `read_sas()` for *.sas7bdat & *.sas7bcat
- SPSS: `read_sav()` & `read_por()`
- Stata: `read_dta()`

- `Handles labels & missing values`

- read_xlsx()
- `read_xls()`
- `read_excel()`

VINCI

Databases

- https://db.rstudio.com/
- Common interface defined by DBI
- Multiple backends
  - odbc (covers MS SQL Server, used by VINCI)
    - In house package VINCI facilitates connections to VINCI ORD databases
  - SQLite
  - Many more

Web data

(Not really applicable on VINCI but for completeness …)

- Foundational
  - `xml2` – XML data
  - `jsonlite` – JSON conversion
- httr for web APIs
- `googlesheets4` via the Sheets API v4
- `googledrive`
- `rvest` – scraping html

Data

what is data?

Tidy Data

Tidy data is data where:

1. Every column is variable.
2. Every row is an observation.
3. Every cell is a single value.

Core Tidyverse: tibble

- Abstracts data structure
- lazy, do less
- surly, complain more
- force cleaner & clearer code

Wrangle

Transform and tidy

Core Tidyverse: dplyr

- Data manipulation
- The core 'verbs'
  - mutate()
  - select()
  - filter()
  - summarize()
  - arrange()
  - group_by()

dplyr details: other 1-table verbs

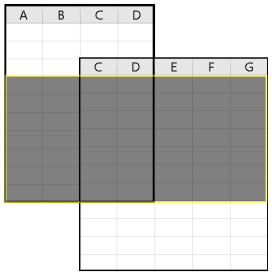
- count observations → `count()` and `tally()`
  - `count()` + `mutate()` → `add_count()` and `add_tally()`
- Subsetting:
  - `distinct()`
  - `slice()`, `slice_(head|tail|min|max|sample)`
- Column manipulations
  - moving columns → `relocate()`
  - renaming → `rename()` or `rename_with()`
- `mutate() + select()` → `transmute()`

dplyr details: 2-table verbs

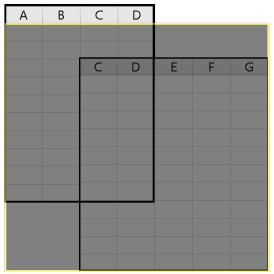
- Naïve Combine
  - bind_rows() and bind_cols()
- Set operations (removes duplicates)
  - intersect(), union(), union_all(), and setdiff()
- Joins (SQL like)
  - continued on next page.

SQL Style Table Joins

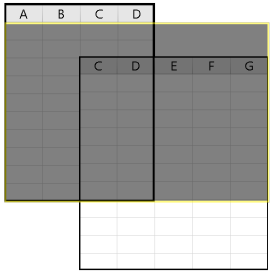
- inner_join()



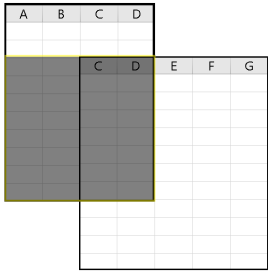
- full_join()



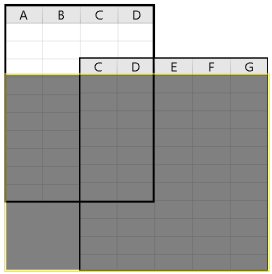
- left_join()



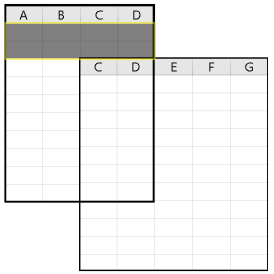
- semi_join()



- right_join()



- anti_join()



dplyr (common) vector functions

- case_when()/na_if()
- coalesce()
- if_else()
- n()/n_distinct()

- order_by() – adds window context
  - lag() and lead()
  - first()/nth()/last()
  - row_number()
  - ntile()/min_rank()/dense_rank()/percent_rank()/cume_dist()

backends

- dtplyr – Data.table, efficient large in-memory data.
- **dbplyr – connection to DBI compatible tables**
- sparklyr – Apache Spark

remote specific

- copy_to()
- collapse() – collapse operations into SQL.
- compute() – get results, leave on server
- collect() – get results, bring data locally
- ident() & sql() for escaping table names and explicit SQL.
- explain()/show_query()

Core Tidyverse: tidyr

- reshaping
  - pivot_longer()/gather()
  - pivot_wider()/spread()
  - uncount()
- Nesting
  - nest()/unnest()
- Strings
  - extract()/separate()
- Missing Values
  - complete()/expand()/crossing()
  - drop_na()/replace_na()/fill()

Type Specific

Most packages in tidyverse handle dates correctly implicitly

- Dates, eg. mdy_hms(), year(), leap_year(), now(), …
- Intervals – specific start/end date/time
- duration – physical time independent of calendar/clock time
- period – changes in calendar/clock time

- String (character) manipulation

- Categorical data
- Can hold any binary data

Program

Core Tidyverse: purrr

- Map/reduce operations
- Replaces built in functions apply, Map, Reduce
- Type consistent
- map_*(), map2_*(), pmap_*(), imap_*()

Glue

- Data + Strings = Glue
- Context dependent strings

```
library(glue)
name <- "Leonardo"
glue('My name is {name}.')
#> My name is Leonardo.
```

More Magrittr

- Building functions

```

```

- Exposition pipe %$%

```

```

- `Tee pipe %T>%, Returns LHS`

```

```

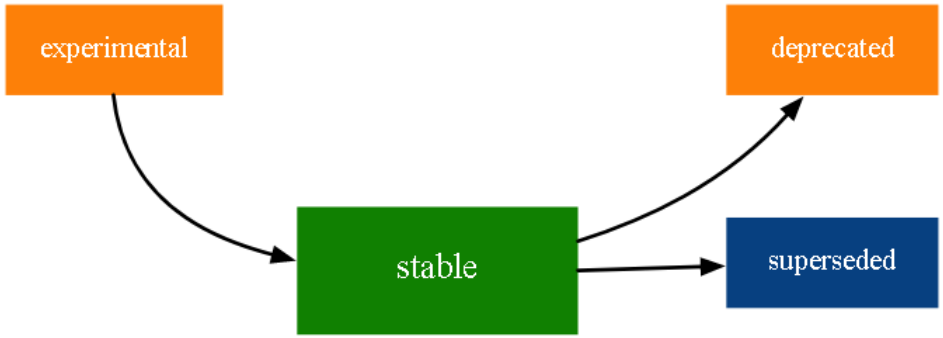
- `Assign-back pipe`

```

```

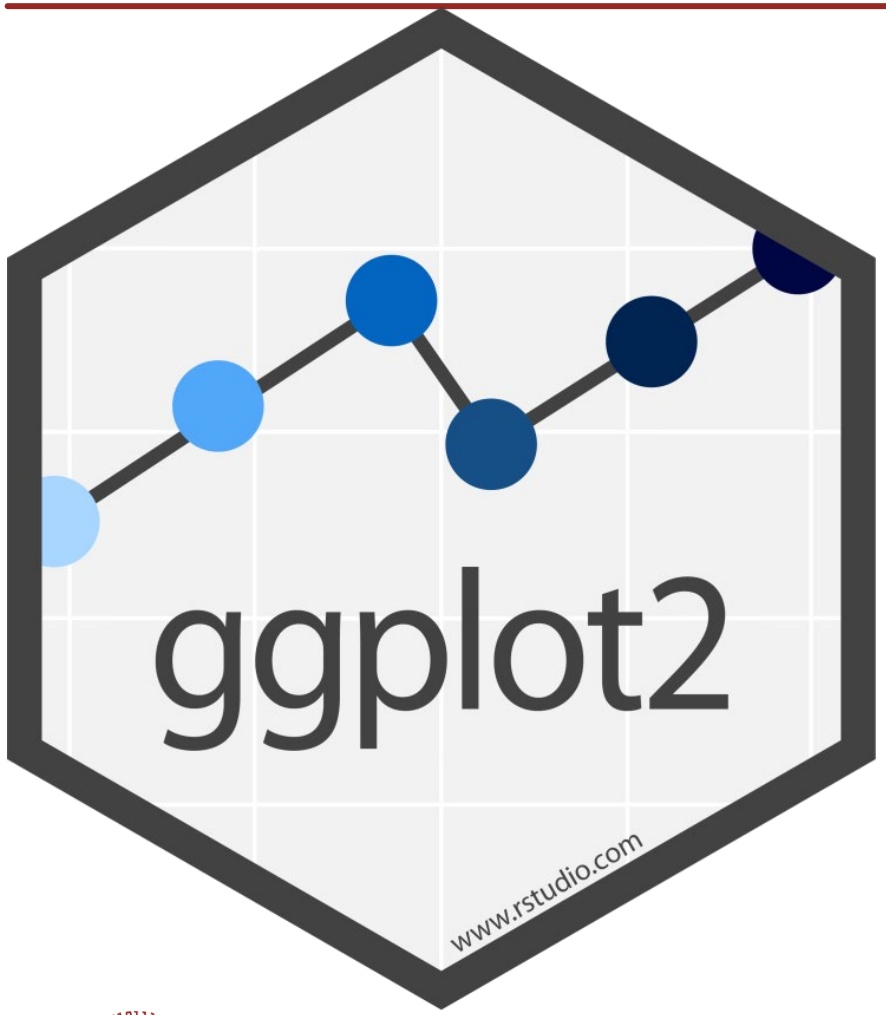
Lifecycle

- Stage management
  - Experimental lifecycle experimental
  - Stable lifecycle stable
  - Deprecated lifecycle deprecated
  - Superseded lifecycle superseded



Vizualize

Core Tidyverse: ggplot2



- Grammar of Graphics
  - aesthetics
  - layers
  - scales
  - coordinates
- Is a presentation unto itself
  - Watch for future CyberSeminars on ggplot2

VINCI

Model

Tidymodels

- Also, a topic that deserves its own presentation
- Packages
  - **rsample**: efficient data splitting (train/test) and resampling
  - **parsnip**: unifying interface of common models
  - **recipes**: data pre-processing & feature engineering
  - **workflows**: pre-processing + modeling + post-processing
  - **tune**: hyperparameter optimization
  - **yardstick**: performance metrics
  - **broom**: tidying output from models
  - and more. see tidymodels.org/packages

VINCI

Wrap-up

Where to go now

Tidyverse.org

Tidyverse

Packages Blog Learn Help Contribute

R packages for data science

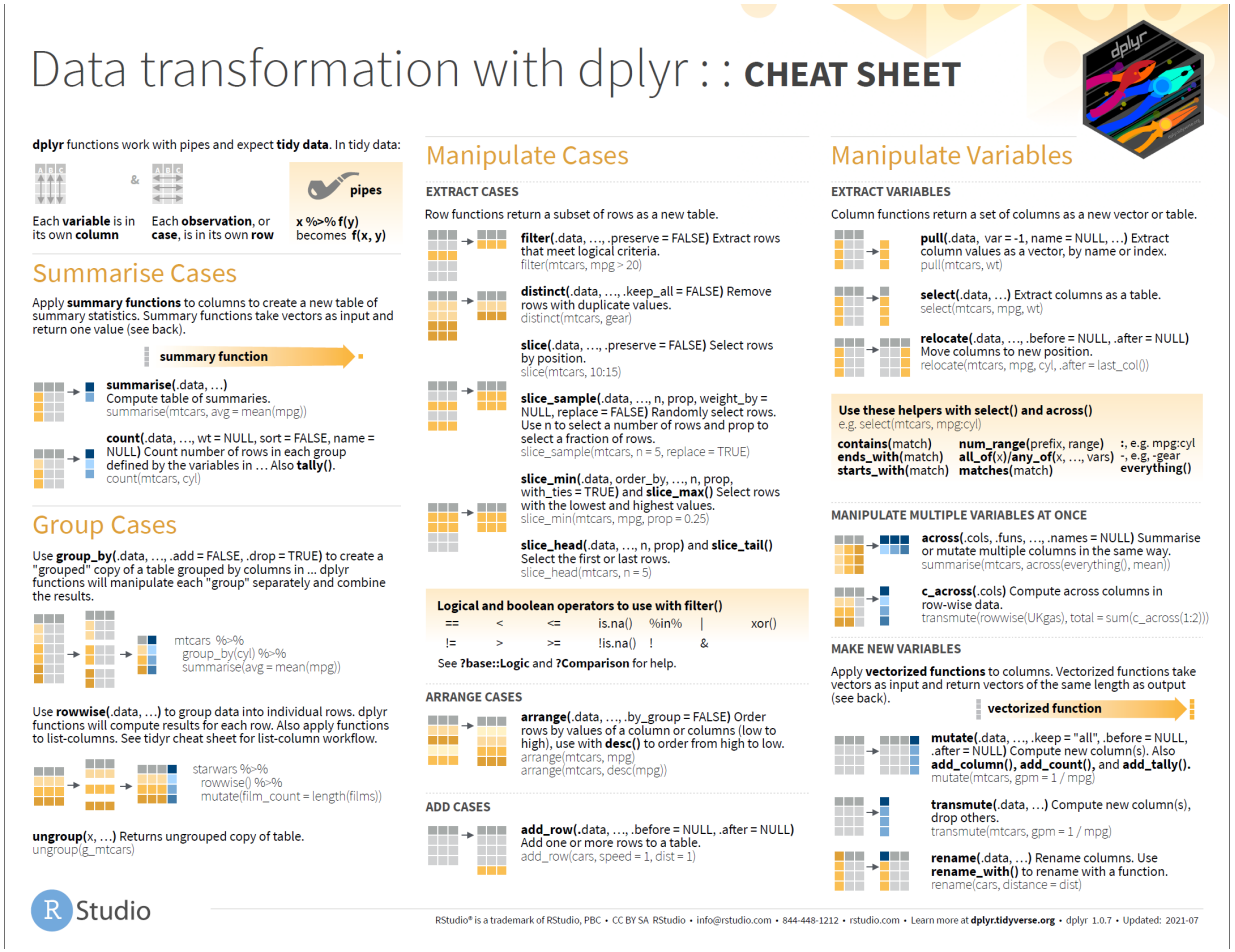
The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

Learn the tidyverse

Cheat Sheets



Data transformation with dplyr : : CHEAT SHEET

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

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(.data, …, wt = NULL, sort = FALSE, name = NULL) Count number of rows in each group defined by the variables in … Also **tally()**.
count(mtcars, cyl)

Group Cases

Use **group_by**(.data, …, .add = FALSE, .drop = TRUE) to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.

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

Use **rowwise**(.data, …) to group data into individual rows. dplyr functions will compute results for each row. Also apply functions to list-columns. See tidyr cheat sheet for list-column workflow.

```
starwars %>%
  rowwise() %>%
  mutate(film_count = length(films))
```

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

Manipulate Cases

EXTRACT CASES

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

filter(.data, …, .preserve = FALSE) Extract rows that meet logical criteria.
filter(mtcars, mpg > 20)

distinct(.data, …, .keep_all = FALSE) Remove rows with duplicate values.
distinct(mtcars, gear)

slice(.data, …, .preserve = FALSE) Select rows by position.
slice(mtcars, 10:15)

slice_sample(.data, …, n, prop, weight_by = NULL, replace = FALSE) Randomly select rows. Use n to select a number of rows and prop to select a fraction of rows.
slice_sample(mtcars, n = 5, replace = TRUE)

slice_min(.data, order_by, …, n, prop, with_ties = TRUE) and **slice_max()** Select rows with the lowest and highest values.
slice_min(mtcars, mpg, prop = 0.25)

slice_head(.data, …, n, prop) and **slice_tail()** Select the first or last rows.
slice_head(mtcars, n = 5)

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, …, .by_group = FALSE) 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(cars, speed = 1, dist = 1)

Manipulate Variables

EXTRACT VARIABLES

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

pull(.data, var = -1, name = NULL, …) Extract column values as a vector, by name or index.
pull(mtcars, wt)

select(.data, …) Extract columns as a table.
select(mtcars, mpg, wt)

relocate(.data, …, .before = NULL, .after = NULL) Move columns to new position.
relocate(mtcars, mpg, cyl, .after = last_col())

Use these helpers with select() and across()
e.g. select(mtcars, mpg:cyl)

contains(match)
ends_with(match)
starts_with(match)
num_range(prefix, range)
all_of(x)/**any_of**(x, …, vars)
matches(match)
:, e.g. mpg:cyl
-, e.g, -gear
everything()

MANIPULATE MULTIPLE VARIABLES AT ONCE

across(.cols, .funs, …, .names = NULL) Summarise or mutate multiple columns in the same way.
summarise(mtcars, across(everything(), mean))

c_across(.cols) Compute across columns in row-wise data.
transmute(rowwise(UKgas), total = sum(c_across(1:2)))

MAKE NEW VARIABLES

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

vectorized function

mutate(.data, …, .keep = "all", .before = NULL, .after = NULL) Compute new column(s). Also **add_column()**, **add_count()**, and **add_tally()**.
mutate(mtcars, gpm = 1 / mpg)

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

rename(.data, …) Rename columns. Use **rename_with()** to rename with a function.
rename(cars, distance = dist)

RStudio

Other Resources

- VINCIpedia
  - R Academy
- VINCI Helpdesk
- VINCI Training & Office Hours, Most Wednesdays 3-4 Eastern
- https://education.rstudio.com/
- https://www.tidyverse.org/
- https://r4ds.had.co.nz/index.html