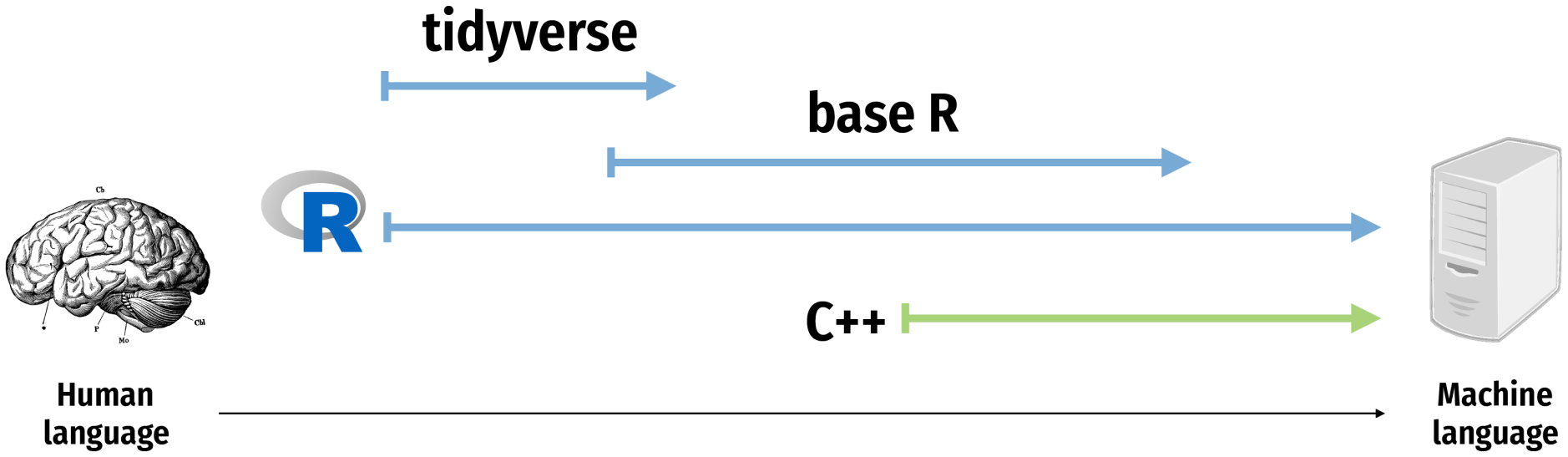
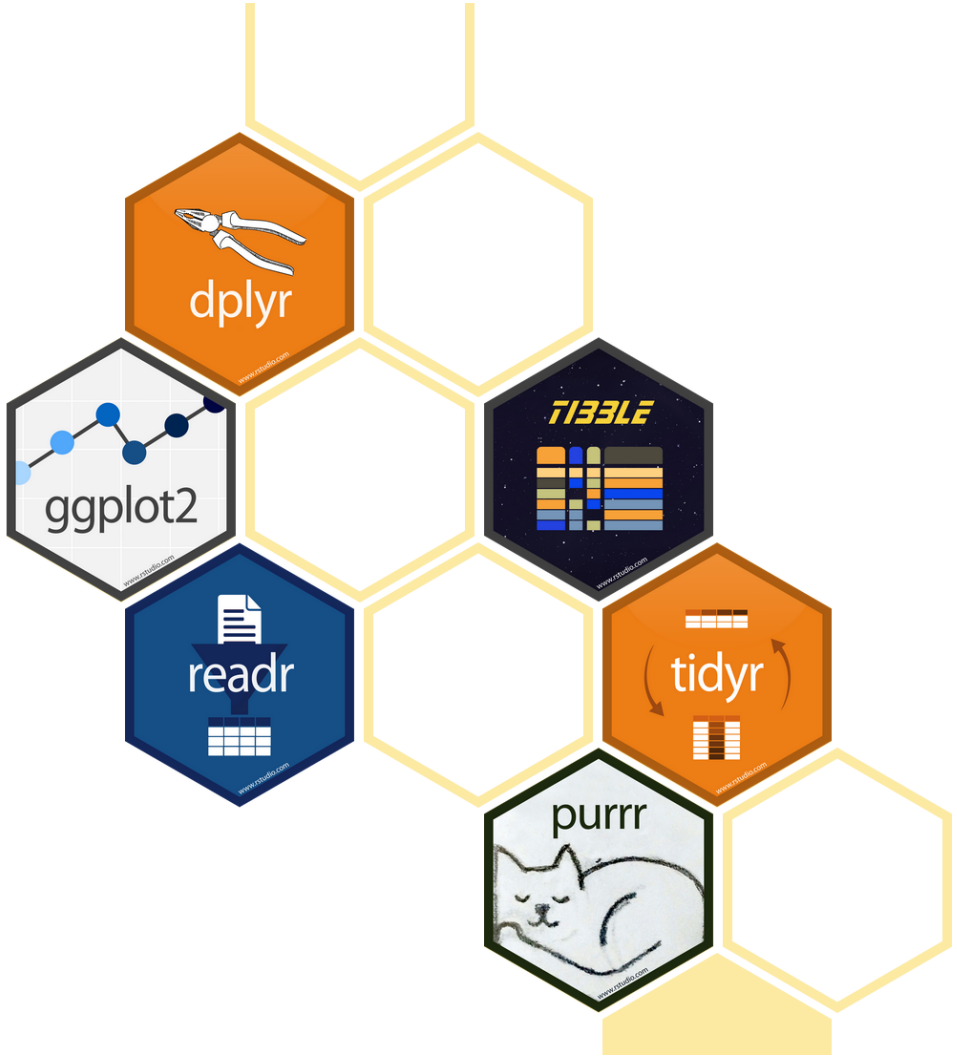
# Transform data with dplyr

# The tidyverse

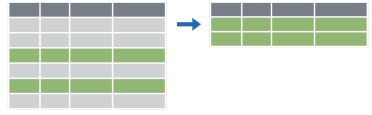


# The tidyverse



# dplyr: verbs for manipulating data

Extract rows with `filter()`



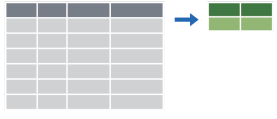
Extract columns with `select()`

Arrange/sort rows with `arrange()`

Make new columns with `mutate()`

Make group summaries with

`group_by() %>% summarize()`



# `filter()`

**Extract rows that meet some sort of test**

```
filter(DATA, ...)
```

- **`DATA`** = Data frame to transform
- **`...`** = One or more tests

`filter()` returns each row for which the test is TRUE

```
filter(gapminder, country == "Denmark")
```

| country | continent | year |
|---|---|---|
| Afghanistan | Asia | 1952 |
| Afghanistan | Asia | 1957 |
| Afghanistan | Asia | 1962 |
| Afghanistan | Asia | 1967 |
| Afghanistan | Asia | 1972 |
| ... | ... | ... |

| country | continent | year |
|---|---|---|
| Denmark | Europe | 1952 |
| Denmark | Europe | 1957 |
| Denmark | Europe | 1962 |
| Denmark | Europe | 1967 |
| Denmark | Europe | 1972 |
| Denmark | Europe | 1977 |

# Logical tests

| Test | Meaning | Test | Meaning |
|---|---|---|---|
| `x < y` | Less than | `x %in% y` | In (group membership) |
| `x > y` | Greater than | `is.na(x)` | Is missing |
| `==` | Equal to | `!is.na(x)` | Is not missing |
| `x <= y` | Less than or equal to | | |
| `x >= y` | Greater than or equal to | | |
| `x != y` | Not equal to | | |

# Your turn #1: Filtering

**Use `filter()` and logical tests to show...**

1. The data for Canada
2. All data for countries in Oceania
3. Rows where the life expectancy is greater than 82

03:00

```
filter(gapminder, country == "Canada")
```

```
filter(gapminder, continent == "Oceania")
```

```
filter(gapminder, lifeExp > 82)
```

# `filter()` with multiple conditions

**Extract rows that meet *every* test**

```
filter(gapminder, country == "Denmark", year > 2000)
```

```
filter(gapminder, country == "Denmark", year > 2000)
```

| country | continent | year |
|---|---|---|
| Afghanistan | Asia | 1952 |
| Afghanistan | Asia | 1957 |
| Afghanistan | Asia | 1962 |
| Afghanistan | Asia | 1967 |
| Afghanistan | Asia | 1972 |
| … | … | … |

| country | continent | year |
|---|---|---|
| Denmark | Europe | 2002 |
| Denmark | Europe | 2007 |

# Boolean operators

| Operator | Meaning |
|---|---|
| `a & b` | and |
| `a \| b` | or |
| `!a` | not |

# Default is "and"

**These do the same thing:**

```
filter(gapminder, country == "Denmark", year > 2000)
```

```
filter(gapminder, country == "Denmark" & year > 2000)
```

# Your turn #2: Filtering

**Use `filter()` and Boolean logical tests to show...**

1. Canada before 1970
2. Countries where life expectancy in 2007 is below 50
3. Countries where life expectancy in 2007 is below 50 and are not in Africa

03:00

```
filter(gapminder, country == "Canada", year < 1970)
```

```
filter(gapminder, year == 2007, lifeExp < 50)
```

```
filter(gapminder, year == 2007, lifeExp < 50,
       continent != "Africa")
```

# Common mistakes

## Collapsing multiple tests into one

```
filter(gapminder, 1960 < year < 1980)
```

```
filter(gapminder,
       year > 1960, year < 1980)
```

## Using multiple tests instead of `%in%`

```
filter(gapminder,
       country == "Mexico",
       country == "Canada",
       country == "United States")
```

```
filter(gapminder,
       country %in% c("Mexico", "Canada",
                      "United States"))
```

# Common syntax

**Every dplyr verb function follows the same pattern**

**First argument is a data frame; returns a data frame**

```
VERB(DATA, ...)
```

- `VERB` = dplyr function/verb
- `DATA` = Data frame to transform
- `...` = Stuff the verb does

# mutate()

## Create new columns

```
mutate(DATA, ...)
```

- **DATA** = Data frame to transform
- **...** = Columns to make

```
mutate(gapminder, gdp = gdpPercap * pop)
```

| country | year | gdpPercap | pop |
|---|---|---|---|
| Afghanistan | 1952 | 779.4453145 | 8425333 |
| Afghanistan | 1957 | 820.8530296 | 9240934 |
| Afghanistan | 1962 | 853.10071 | 10267083 |
| Afghanistan | 1967 | 836.1971382 | 11537966 |
| Afghanistan | 1972 | 739.9811058 | 13079460 |
| ... | ... | ... | ... |

| country | year | ... | gdp |
|---|---|---|---|
| Afghanistan | 1952 | ... | 6567086330 |
| Afghanistan | 1957 | ... | 7585448670 |
| Afghanistan | 1962 | ... | 8758855797 |
| Afghanistan | 1967 | ... | 9648014150 |
| Afghanistan | 1972 | ... | 9678553274 |
| Afghanistan | 1977 | ... | 11697659231 |

```
mutate(gapminder, gdp = gdpPercap * pop,
                  pop_mil = round(pop / 1000000))
```

| country | year | gdpPercap | pop |
|---|---|---|---|
| Afghanistan | 1952 | 779.4453145 | 8425333 |
| Afghanistan | 1957 | 820.8530296 | 9240934 |
| Afghanistan | 1962 | 853.10071 | 10267083 |
| Afghanistan | 1967 | 836.1971382 | 11537966 |
| Afghanistan | 1972 | 739.9811058 | 13079460 |
| ... | ... | ... | ... |

| country | year | ... | gdp | pop_mil |
|---|---|---|---|---|
| Afghanistan | 1952 | ... | 6567086330 | 8 |
| Afghanistan | 1957 | ... | 7585448670 | 9 |
| Afghanistan | 1962 | ... | 8758855797 | 10 |
| Afghanistan | 1967 | ... | 9648014150 | 12 |
| Afghanistan | 1972 | ... | 9678553274 | 13 |
| Afghanistan | 1977 | ... | 11697659231 | 15 |

# `ifelse()`

## Do conditional tests within `mutate()`

```
ifelse(TEST,
       VALUE_IF_TRUE,
       VALUE_IF_FALSE)
```

- **`TEST`** = A logical test
- **`VALUE_IF_TRUE`** = What happens if test is true
- **`VALUE_IF_FALSE`** = What happens if test is false

```
mutate(gapminder,
       after_1960 = ifelse(year > 1960, TRUE, FALSE))
```

```
mutate(gapminder,
       after_1960 = ifelse(year > 1960,
                           "After 1960",
                           "Before 1960"))
```

# Your turn #3: Mutating

**Use `mutate()` to...**

1. Add an `africa` column that is TRUE if the country is on the African continent
2. Add a column for logged GDP per capita (hint: use `log()`)
3. Add an `africa_asia` column that says “Africa or Asia” if the country is in Africa or Asia, and “Not Africa or Asia” if it’s not

03:00

```
mutate(gapminder, africa = ifelse(continent == "Africa",
                                  TRUE, FALSE))
```

```
mutate(gapminder, log_gdpPercap = log(gdpPercap))
```

```
mutate(gapminder,
       africa_asia =
         ifelse(continent %in% c("Africa", "Asia"),
                "Africa or Asia",
                "Not Africa or Asia"))
```

# What if you have multiple verbs?

## Make a dataset for just 2002 *and* calculate logged GDP per capita

## Solution 1: Intermediate variables

```
gapminder_2002 <- filter(gapminder, year == 2002)

gapminder_2002_log <- mutate(gapminder_2002,
                             log_gdpPercap = log(gdpPercap))
```

# What if you have multiple verbs?

## Make a dataset for just 2002 *and* calculate logged GDP per capita

## Solution 2: Nested functions

```
filter(mutate(gapminder_2002,
              log_gdpPercap = log(gdpPercap)),
       year == 2002)
```

# What if you have multiple verbs?

**Make a dataset for just 2002 *and* calculate logged GDP per capita**

**Solution 3: Pipes!**

**The `%>%` operator (pipe) takes an object on the left and passes it as the first argument of the function on the right**

```
gapminder %>% filter(_, country == "Canada")
```

# What if you have multiple verbs?

**These do the same thing!**

```
filter(gapminder, country == "Canada")
```

```
gapminder %>% filter(country == "Canada")
```

# What if you have multiple verbs?

**Make a dataset for just 2002 *and* calculate logged GDP per capita**

**Solution 3: Pipes!**

```
gapminder %>%
  filter(year == 2002) %>%
  mutate(log_gdpPercap = log(gdpPercap))
```

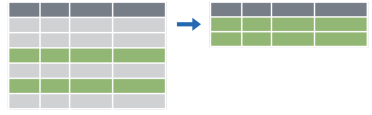
# %>%

```
leave_house(get_dressed(get_out_of_bed(wake_up(me, time =
"8:00"), side = "correct"), pants = TRUE, shirt = TRUE), car
= TRUE, bike = FALSE)
```

```
me %>%
  wake_up(time = "8:00") %>%
  get_out_of_bed(side = "correct") %>%
  get_dressed(pants = TRUE, shirt = TRUE) %>%
  leave_house(car = TRUE, bike = FALSE)
```

# dplyr: verbs for manipulating data

Extract rows with `filter()`



Extract columns with `select()`

Arrange/sort rows with `arrange()`

Make new columns with `mutate()`

Make group summaries with

`group_by() %>% summarize()`