

# The dplyr Package in R

Write less and faster code

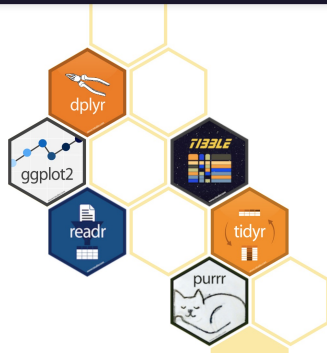
Selina Hofstetter

VAM 1 - MT 2018

October 17, 2018

# The tidyverse package

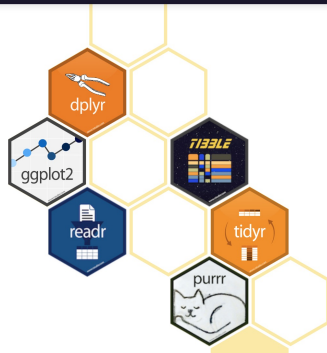
Tidyverse



- Created by [Hadley Wickham](#)

# The tidyverse package

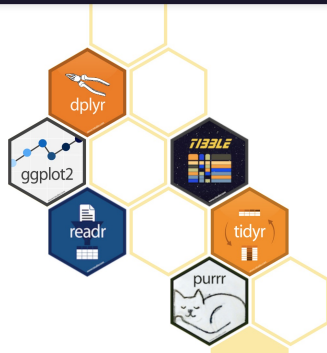
## Tidyverse



- Created by [Hadley Wickham](#)
- Helps you to organise, manipulate and analyse your data

# The tidyverse package

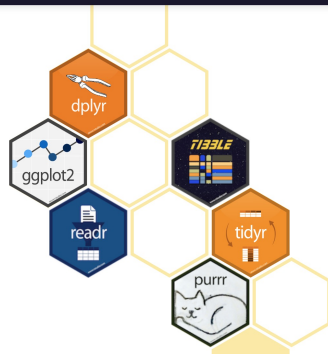
## Tidyverse



- Created by [Hadley Wickham](#)
- Helps you to organise, manipulate and analyse your data
- (Probably at most times) faster than your own code

# The tidyverse package

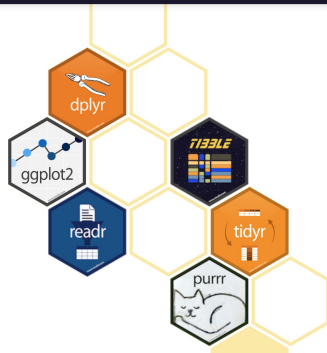
## Tidyverse



- Created by [Hadley Wickham](#)
- Helps you to organise, manipulate and analyse your data
- (Probably at most times) faster than your own code
- Clean and transparent, easy for others to read your code

# The tidyverse package

## Tidyverse



- Created by [Hadley Wickham](#)
- Helps you to organise, manipulate and analyse your data
- (Probably at most times) faster than your own code
- Clean and transparent, easy for others to read your code
- Requires you to write less

# The tidyverse package



- Created by [Hadley Wickham](#)
- Helps you to organise, manipulate and analyse your data
- (Probably at most times) faster than your own code
- Clean and transparent, easy for others to read your code
- Requires you to write less
- My tip: Always think first how you could write code with the help of tidyverse (specifically: dplyr) - avoid long loops!

# Your *dplyr* toolbox:

Five verbs to **manipulate** your data with:

- `mutate()` allows you to generate new variables

## Example:

```
# Mutate: Generate a new variable and store it in mydata:  
mydata <- mydata %>% mutate(percvote = voteshare*100)  
# Check if the new variable "percvote" is now there:  
names(mydata)
```



# Your *dplyr* toolbox:

Five verbs to **manipulate** your data with:

- `mutate()` allows you to generate new variables
- `select()` lets you select certain variables in your dataset

## Example:

```
# Select: Select certain variables in your data:
mydata %>% select(percvote)
# Use as_tibble() or head() to get a more compact view of the selected data:
mydata %>% as_tibble() %>% select(percvote)
mydata %>% select(percvote) %>% head()
# Or a range of variables:
mydata %>% as_tibble() %>% select(setting:year)
```

# Your *dplyr* toolbox:

Five verbs to **manipulate** your data with:

- `mutate()` allows you to generate new variables
- `select()` lets you select certain variables in your dataset
- `filter()` enables you to filter out observations of a certain value in your dataset

## Example:

```
# Filter: Filter out observations of a certain value in your data:  
# For example, get all percentage voteshares above 50%:  
mydata %>% filter(percvote > 50)
```

# Your *dplyr* toolbox:

Five verbs to **manipulate** your data with:

- `mutate()` allows you to generate new variables
- `select()` lets you select certain variables in your dataset
- `filter()` enables you to filter out observations of a certain value in your dataset
- `summarise()` allows you to for example count or calculate the mean value of groups of observations in your data

## Example:

```
# Summarise: Aggregate/summarise your data:  
# For example, counting the number of observations in topcan:  
topcan %>% summarise(n())  
# Use group_by to count the observations within a specific group of topcan  
# For example within each type of election:  
topcan %>% group_by(setting) %>% summarise(obs=n())  
# Or calculate the mean voteshare within each type of election:  
topcan %>% group_by(setting) %>% summarise(mean(percvote))
```

# Your *dplyr* toolbox:

Five verbs to **manipulate** your data with:

- `mutate()` allows you to generate new variables
- `select()` lets you select certain variables in your dataset
- `filter()` enables you to filter out observations of a certain value in your dataset
- `summarise()` allows you to for example count or calculate the mean value of groups of observations in your data
- `arrange()` sorts your data, e.g. alphabetically when you have country-level data

## Example:

```
# Arrange: Sorts your data:  
# For example, alphabetically for elections, then for states and then descending in years within states:  
topcan <- topcan %>% arrange(setting, state, -year)  
# Have another look at the data, now sorted:  
topcan %>% View()
```