### Tidyverse

The tidyverse is a powerful collection of R packages that are actually data tools for transforming and visualizing data. All packages of the tidyverse share an underlying philosophy and common APIs.

The core packages are:

- **ggplot2**, which implements the grammar of graphics. You can use it to visualize your data.
- **dplyr** is a grammar of data manipulation. You can use it to solve the most common data manipulation challenges.
- **tidyr** helps you to create tidy data or data where each variable is in a column; each observation is a row and each value is a cell.
- **readr** is a fast and friendly way to read rectangular data.
- **stringr** provides a cohesive set of functions designed to make working with strings as easy as possible.
- **forcats** provide a suite of useful tools that solve common problems with factors.

You can install the complete tidyverse with:

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

Then, load the core tidyverse and make it available in your current R session by running:

```r
> library(tidyverse)
```

**Note:** there are many other tidyverse packages with more specialised usage. They are not loaded automatically with library(tidyverse), so you'll need to load each one with its own call to library().

### Useful Functions

- **tidyverse_conflicts()**: Conflicts between tidyverse and other packages.
- **tidyverse_deps()**: List all tidyverse dependencies.
- **tidyverse_logo()**: Get tidyverse logo, using ASCII or unicode characters.
- **tidyverse_packages()**: List all tidyverse packages.
- **tidyverse_update()**: Update tidyverse packages.

###Loading in the data

- **library(datasets)**: Load the datasets package.
- **library(gapminder)**: Load the gapminder package.
- **attach(iris)**: Attach iris data to the R search path.

### dplyr

**Filter**

- `filter()` allows you to select a subset of rows in a data frame.

  ```r
  > iris %>%
  > filter(Species=="virginica")
  > filter(Species=="virginica", Sepal.Length > 6)
  ```

**Arrange**

- `arrange()` sorts the observations in a dataset in ascending or descending order based on one of its variables.

  ```r
  > iris %>%
  > arrange(Sepal.Length)
  > arrange(desc(Sepal.Length))
  ```

**Mutate**

- `mutate()` allows you to update or create new columns of a data frame.

  ```r
  > iris %>%
  > mutate(Sepal.Length=Sepal.Length*10)
  ```

**Summarize**

- `summarize()` allows you to turn many observations into a single data point.

  ```r
  > iris %>%
  > summarize(medianSL=median(Sepal.Length),
  > maxSL=max(Sepal.Length))
  ```

You can also summarize multiple variables at once:

```r
> iris %>%
> filter(Species=="virginica") %>%
> summarize(medianSL=median(Sepal.Length),
> maxSL=max(Sepal.Length))
```

**Find median and max sepal length Filter for virginica then summarize the median sepal length**

### ggplot2

**Scatter plot**

- Scatter plots allow you to compare two variables within your data. To do this with ggplot2, you use `geom_point()`.

  ```r
  > iris_small <- iris %>%
  > filter(Sepal.Length > 5)
  > ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width)) +
  > geom_point()
  ```

**Additional Aesthetics**

- **Color**

  ```r
  > ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width,
  > color=Species)) + geom_point()
  ```

- **Size**

  ```r
  > ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width,
  > size=Sepal.Length)) + geom_point()
  ```

**Faceting**

- Faceting allows you to select a subset of rows in a data frame.

  ```r
  > by_species <- iris %>%
  > group_by(Species) %>%
  > summarize(medianPL=median(Petal.Length))
  ```

**Line Plots**

- **Bar Plots**

  ```r
  > by_species <- iris %>%
  > group_by(Species) %>%
  > summarize(medianPL=median(Petal.Length))
  ```

**Histograms**

- **Box Plots**

  ```r
  > ggplot(iris_small, aes(x=Petal.Length)) +
  > geom_boxplot()
  ```

**Box Plots**

- **Box Plots**

  ```r
  > ggplot(iris_small, aes(x=Petal.Length)) +
  > geom_boxplot()
  ```