R For Data Science *Cheat Sheet*

Tidyverse for Beginners

Learn More R for Data Science Interactively at www.datacamp.com

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 end each value is a cell.
- **readr** is a fast and friendly way to read rectangular data.
- **purrr** enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors.
- **tibble** is a modern re-imaginging of the data frame.

You can install the complete tidyverse with:

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

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

```
> 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.

`> iris %>%` `  filter(Species=="virginica")`	Select iris data of species "virginica"

`> iris %>%`
`  filter(Species=="virginica",`
`  Sepal.Length > 6)`
	Select iris data of species "virginica" and sepal length greater than 6.

Arrange

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

`> iris %>%` `  arrange(Sepal.Length)`	Sort in ascending order of sepal length
`> iris %>%` `  arrange(desc(Sepal.Length))`	Sort in descending order of sepal length

Combine multiple `dplyr` verbs in a row with the pipe operator `%>%`:

`> iris %>%`
`  filter(Species=="virginica") %>%`
`  arrange(desc(Sepal.Length))`
	Filter for species "virginica" then arrange in descending order of sepal length

Mutate

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

`> iris %>%` `  mutate(Sepal.Length=Sepal.Length*10)`	Change Sepal.Length to be in millimeters
`> iris %>%` `  mutate(SLMm=Sepal.Length*10)`	Create a new column called SLMm

Combine the verbs `filter()`, `arrange()`, and `mutate()`:

```
> iris %>%
  filter(Species=="Virginica") %>%
  mutate(SLMm=Sepal.Length*10) %>%
  arrange(desc(SLMm))
```

Summarize

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

`> iris %>%` `  summarize(medianSL=median(Sepal.Length))`	Summarize to find the median sepal length

`> iris %>%`
`  filter(Species=="virginica") %>%`
`  summarize(medianSL=median(Sepal.Length))`
	Filter for virginica then summarize the median sepal length

You can also summarize multiple variables at once:

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

`group_by()` allows you to summarize within groups instead of summarizing the entire dataset:

`> iris %>%`
`  group_by(Species) %>%`
`  summarize(medianSL=median(Sepal.Length),`
`  maxSL=max(Sepal.Length))`
	Find median and max sepal length of each species

`> iris %>%`
`  filter(Sepal.Length>6) %>%`
`  group_by(Species) %>%`
`  summarize(medianPL=median(Petal.Length),`
`  maxPL=max(Petal.Length))`
	Find median and max petal length of each species with sepal length > 6

ggplot2

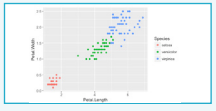
Scatter plot

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

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

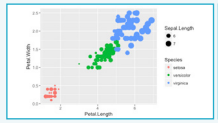
Additional Aesthetics

- Color



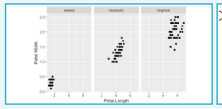
```
> ggplot(iris_small, aes(x=Petal.Length,
                         y=Petal.Width,
                         color=Species)) +
  geom_point()
```

- Size



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

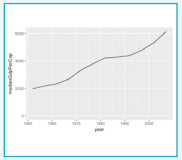
Faceting



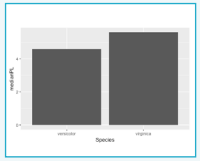
```
> ggplot(iris_small, aes(x=Petal.Length,
                         y=Petal.Width)) +
  geom_point()+
  facet_wrap(~Species)
```

Line Plots

```
> by_year <- gapminder %>%
  group_by(year) %>%
  summarize(medianGdpPerCap=median(gdpPercap))
> ggplot(by_year, aes(x=year,
                      y=medianGdpPerCap))+
  geom_line()+
  expand_limits(y=0)
```



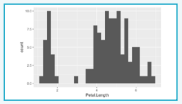
Bar Plots



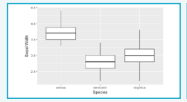
```
> by_species <- iris %>%
  filter(Sepal.Length>6) %>%
  group_by(Species) %>%
  summarize(medianPL=median(Petal.Length))
> ggplot(by_species, aes(x=Species,
                         y=medianPL)) +
  geom_col()
```

Histograms

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



Box Plots



```
> ggplot(iris_small, aes(x=Species,
                         y=Sepal.Width))+
  geom_boxplot()
```

DataCamp
Learn R for Data Science Interactively