

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 and 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-imaging of the data frame.



• **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:

```
> 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

<pre>> tidyverse_conflicts() > tidyverse_deps() > tidyverse_logo() > tidyverse_packages() > tidyverse_update()</pre>	<p>Conflicts between tidyverse and other packages</p> <p>List all tidyverse dependencies</p> <p>Get tidyverse logo, using ASCII or unicode characters</p> <p>List all tidyverse packages</p> <p>Update tidyverse packages</p>
---	---

Loading in the data

<pre>> library(datasets) > library(gapminder) > attach(iris)</pre>	<p>Load the datasets package</p> <p>Load the gapminder package</p> <p>Attach iris data to the R search path</p>
---	---

dplyr

Filter

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

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

Arrange

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

<pre>> iris %>% arrange(Sepal.Length) > iris %>% arrange(desc(Sepal.Length))</pre>	<p>Sort in ascending order of sepal length</p> <p>Sort in descending order of sepal length</p>
--	--

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

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

Mutate

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

<pre>> iris %>% mutate(Sepal.Length=Sepal.Length*10) > iris %>% mutate(SLMm=Sepal.Length*10)</pre>	<p>Change Sepal.Length to be in millimeters</p> <p>Create a new column called SLMm</p>
--	--

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

<pre>> iris %>% filter(Species=="virginica") %>% mutate(SLMm=Sepal.Length*10) %>% arrange(desc(SLMm))</pre>	
---	--

Summarize

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

<pre>> iris %>% summarize(medianSL=median(Sepal.Length)) > iris %>% filter(Species=="virginica") %>% summarize(medianSL=median(Sepal.Length))</pre>	<p>Summarize to find the median sepal length</p> <p>Filter for virginica then summarize the median sepal length</p>
--	---

You can also summarize multiple variables at once:

<pre>> iris %>% filter(Species=="virginica") %>% summarize(medianSL=median(Sepal.Length), maxSL=max(Sepal.Length))</pre>	
--	--

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

<pre>> iris %>% group_by(Species) %>% summarize(medianSL=median(Sepal.Length), maxSL=max(Sepal.Length)) > iris %>% filter(Sepal.Length>6) %>% group_by(Species) %>% summarize(medianPL=median(Petal.Length), maxPL=max(Petal.Length))</pre>	<p>Find median and max sepal length of each species</p> <p>Find median and max petal length of each species with sepal length > 6</p>
---	--

ggplot2

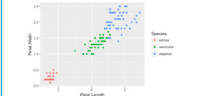
Scatter plot

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

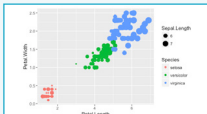
<pre>> iris_small <- iris %>% filter(Sepal.Length > 5) > ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width)) + geom_point()</pre>	<p>Compare petal width and length</p>
---	---------------------------------------

Additional Aesthetics

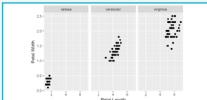
• Color

	<pre>> ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width, color=Species)) + geom_point()</pre>
---	--

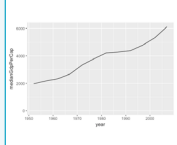
• Size

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


Faceting

	<pre>> ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width)) + geom_point() + facet_wrap(~Species)</pre>
---	--

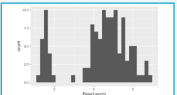
Line Plots

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

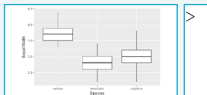
Bar Plots

	<pre>> 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()</pre>
---	--

Histograms

<pre>> ggplot(iris_small, aes(x=Petal.Length)) + geom_histogram()</pre>	
--	---

Box Plots

	<pre>> ggplot(iris_small, aes(x=Species, y=Sepal.Width)) + geom_boxplot()</pre>
---	--

