# R syntax comparison
Cheat Sheet

SMITH COLLEGE

## Syntax

**Syntax** is the set of rules that govern what code works and doesn't work. Most programming languages offer one standardized syntax, but R has many.

**Most people use some combination of all the syntaxes available to them.**

1. Dollar sign syntax uses the dollar sign to locate a variable within a dataset. It is expected by most **base** R functions.
2. Formula syntax uses the **data=** argument at the end of a list of function arguments. The formula syntax is used by modeling functions like **lm()**, **lattice** graphics like **xyplot()**, and **mosaic** summary statistics like **mean()**.
3. Tidyverse syntax uses data as the first argument to function calls. It is used by the packages **dplyr** and **tidyr**, among others. The associated graphics library is **ggplot2**.

## Dollar sign syntax

```
goal(data$x, data$y)
```

**Summary statistics:**

one continuous variable:

```
mean(mtcars$mpg)
```

one categorical variable:

```
table(mtcars$cyl)
```

two categorical variables:

```
table(mtcars$cyl, mtcars$am)
```

one continuous, one categorical:

```
mean(mtcars$mpg[mtcars$cyl==4])
mean(mtcars$mpg[mtcars$cyl==6])
mean(mtcars$mpg[mtcars$cyl==8])
```

**Plotting:**

one continuous variable:

```
hist(mtcars$disp)

boxplot(mtcars$disp)
```

one categorical variable:

```
barplot(table(mtcars$cyl))
```

two continuous variables:

```
plot(mtcars$disp, mtcars$mpg)
```

two categorical variables:

```
mosaicplot(table(mtcars$am, mtcars$cyl))
```

one continuous, one categorical:

```
histogram(mtcars$disp[mtcars$cyl==4])
histogram(mtcars$disp[mtcars$cyl==6])
histogram(mtcars$disp[mtcars$cyl==8])

boxplot(mtcars$disp[mtcars$cyl==4])
boxplot(mtcars$disp[mtcars$cyl==6])
boxplot(mtcars$disp[mtcars$cyl==8])
```

**Wrangling:**

subsetting:

```
mtcars[mtcars$mpg>30, ]
```

making a new variable:

```
mtcars$efficient[mtcars$mpg>30] <- TRUE
mtcars$efficient[mtcars$mpg<30] <- FALSE
```

## Formula syntax

```
goal(y~x|z, data=data, group=w)
```

**Summary statistics:**

one continuous variable:

```
mosaic::mean(~mpg, data=mtcars)
```

one categorical variable:

```
mosaic::tally(~cyl, data=mtcars)
```

two categorical variables:

```
mosaic::tally(cyl~am, data=mtcars)
```

one continuous, one categorical:

```
mosaic::mean(mpg~cyl, data=mtcars)
```

tilde

**Plotting:**

one continuous variable:

```
xyplot::histogram(~disp, data=mtcars)

xyplot::bwplot(~disp, data=mtcars)
```

one categorical variable:

```
mosaic::bargraph(~cyl, data=mtcars)
```

two continuous variables:

```
xyplot::xyplot(mpg~disp, data=mtcars)
```

two categorical variables:

```
mosaic::bargraph(~am, data=mtcars,
    group=cyl)
```

one continuous, one categorical:

```
xyplot::histogram(~disp|cyl, data=mtcars)

xyplot::bwplot(cyl~disp, data=mtcars)
```

## Tidyverse syntax

```
data %>% goal(x)
```

**Summary statistics:**

one continuous variable:

```
mtcars %>% dplyr::summarize(mean(mpg))
```

one categorical variable:

```
mtcars %>% dplyr::group_by(cyl) %>%
    dplyr::summarize(n())
```

the pipe

two categorical variables:

```
mtcars %>% dplyr::group_by(cyl, am) %>%
    dplyr::summarize(n())
```

one continuous, one categorical:

```
mtcars %>% dplyr::group_by(cyl) %>%
    dplyr::summarize(mean(mpg))
```

**Plotting:**

one continuous variable:

```
ggplot2::qplot(x=mpg, data=mtcars,
    geom = “histogram”)

ggplot2::qplot(y=disp, x=1, data=mtcars,
    geom=“boxplot”)
```

one categorical variable:

```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```

two continuous variables:

```
ggplot2::qplot(x=disp, y=mpg, data=mtcars,
    geom=“point”)
```

two categorical variables:

```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar")
        + facet_grid(.~am)
```

one continuous, one categorical:

```
ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
        geom=“boxplot”)

ggplot2::qplot(x=disp, data=mtcars, geom = "histogram")
        + facet_grid(.~cyl)
```

**Wrangling:**

subsetting:

```
mtcars %>%
    dplyr::filter(mpg>30)
```

making a new variable:

```
mtcars <- mtcars %>%
    dplyr::mutate(efficient = if_else(mpg>30,
    TRUE, FALSE))
```

Updated: 2/17