An Introduction to R Graphics

Part II—ggplot2

Dan Hall, Director of the SCC

Department of Statistics
Franklin College of Arts and Sciences
Statistical Consulting Center
UNIVERSITY OF GEORGIA

Table of Contents

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics.*
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.

- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.

- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.

- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics.*
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
    - Contains low-level graphics functions.
    - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
    - Mimics and extends trellis graphics from S and S-PLUS.
    - Characteristic feature is plots with multiple panels.
    - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
    - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
    - Sophisticated and powerful system. Not too hard to learn.
    - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics*.
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

Introduction

Having powerful and flexible systems for graphics is one of R's biggest strengths.

- Base Graphics. Contained in the `graphics` package distributed in base R.
- Grid graphics. `grid` package is distributed in base R.
  - Contains low-level graphics functions.
  - Useful as a platform for developing and implementing higher-level graphics functions and systems.
- Lattice Graphics. `lattice` package also distributed in base R.
  - Mimics and extends trellis graphics from S and S-PLUS.
  - Characteristic feature is plots with multiple panels.
  - Built on grid.
- ggplot2 Graphics. `ggplot2` package available on CRAN.
  - Based on Leland Wilkinson's ideas articulated in his book, *The Grammar of Graphics.*
  - Sophisticated and powerful system. Not too hard to learn.
  - Built on grid.

ggplot2 Basics

- Idea is that there are several different components that come together to produce a plot.
- These can be thought about and specified independently.
- Breaking a plot down into these components provides structure—a system—to the task of visualizing data.
- Arguably, this makes coding plots easier and more intuitive. Less arguably, it makes the software more flexible and powerful than other systems such as base graphics and lattice.
- `ggplot2` also takes care of some details automatically (like legends) and has nice defaults. These features simplify the task of coding a plot.

ggplot2 Basics

- Idea is that there are several different components that come together to produce a plot.
- These can be thought about and specified independently.
- Breaking a plot down into these components provides structure—a system—to the task of visualizing data.
- Arguably, this makes coding plots easier and more intuitive. Less arguably, it makes the software more flexible and powerful than other systems such as base graphics and lattice.
- ggplot2 also takes care of some details automatically (like legends) and has nice defaults. These features simplify the task of coding a plot.

ggplot2 Basics

- Idea is that there are several different components that come together to produce a plot.
- These can be thought about and specified independently.
- Breaking a plot down into these components provides structure—a system—to the task of visualizing data.
- Arguably, this makes coding plots easier and more intuitive. Less arguably, it makes the software more flexible and powerful than other systems such as base graphics and lattice.
- ggplot2 also takes care of some details automatically (like legends) and has nice defaults. These features simplify the task of coding a plot.

ggplot2 Basics

- Idea is that there are several different components that come together to produce a plot.
- These can be thought about and specified independently.
- Breaking a plot down into these components provides structure—a system—to the task of visualizing data.
- Arguably, this makes coding plots easier and more intuitive. Less arguably, it makes the software more flexible and powerful than other systems such as base graphics and lattice.
- ggplot2 also takes care of some details automatically (like legends) and has nice defaults. These features simplify the task of coding a plot.

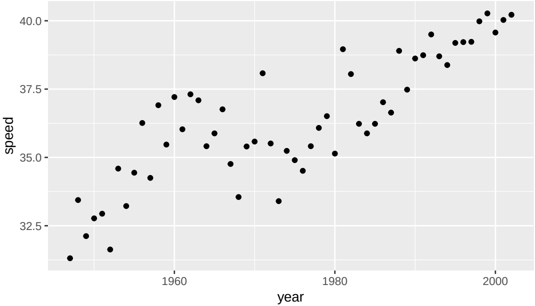
`ggplot2` Basics

- Idea is that there are several different components that come together to produce a plot.
- These can be thought about and specified independently.
- Breaking a plot down into these components provides structure—a system—to the task of visualizing data.
- Arguably, this makes coding plots easier and more intuitive. Less arguably, it makes the software more flexible and powerful than other systems such as base graphics and lattice.
- `ggplot2` also takes care of some details automatically (like legends) and has nice defaults. These features simplify the task of coding a plot.

ggplot2 Basics

- There are at least eight components that can be manipulated separately in `ggplot2`, but we start with the three most important:
  - A data frame;
  - One or more geometrical representations (`geom`);
  - A mapping of the data to aesthetic (`aes`) features of the `geom`.

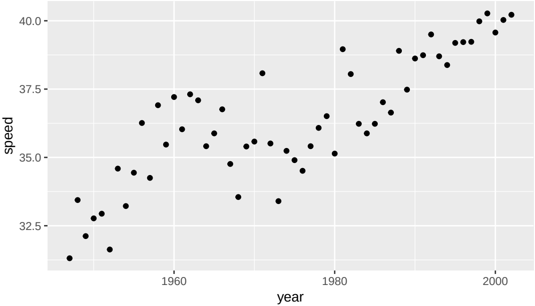
```r
# load ggplot2 and get some data:
require(ggplot2); data(Cars93,package="MASS"); source("https://tinyurl.com/une4s3g/getData_3.R")
# Now create a simple plot:
ggplot(tdf, mapping=aes(x=year,y=speed)) + geom_point()
```



ggplot2 Basics

- There are at least eight components that can be manipulated separately in `ggplot2`, but we start with the three most important:
  - A data frame;
  - One or more geometrical representations (`geom`);
  - A mapping of the data to aesthetic (`aes`) features of the `geom`.

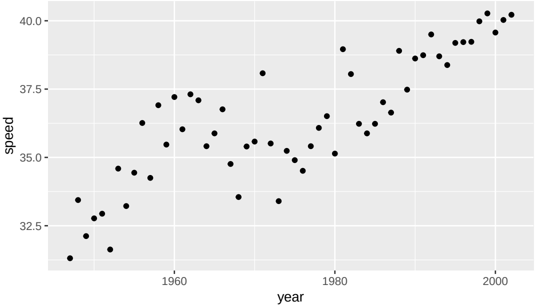
```r
# load ggplot2 and get some data:
require(ggplot2); data(Cars93,package="MASS"); source("https://tinyurl.com/une4s3g/getData_3.R")
# Now create a simple plot:
ggplot(tdf, mapping=aes(x=year,y=speed)) + geom_point()
```



ggplot2 Basics

- There are at least eight components that can be manipulated separately in `ggplot2`, but we start with the three most important:
  - A data frame;
  - One or more geometrical representations (`geom`);
  - A mapping of the data to aesthetic (`aes`) features of the `geom`.

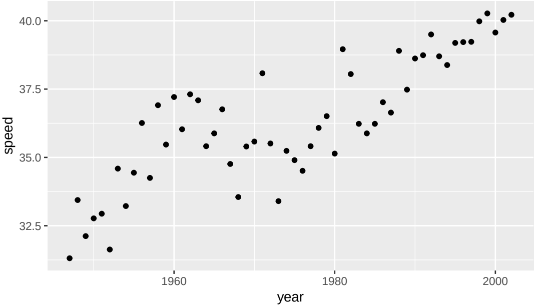
```r
# load ggplot2 and get some data:
require(ggplot2); data(Cars93,package="MASS"); source("https://tinyurl.com/une4s3g/getData_3.R")
# Now create a simple plot:
ggplot(tdf, mapping=aes(x=year,y=speed)) + geom_point()
```



ggplot2 Basics

- There are at least eight components that can be manipulated separately in `ggplot2`, but we start with the three most important:
  - A data frame;
  - One or more geometrical representations (`geom`);
  - A mapping of the data to aesthetic (`aes`) features of the `geom`.

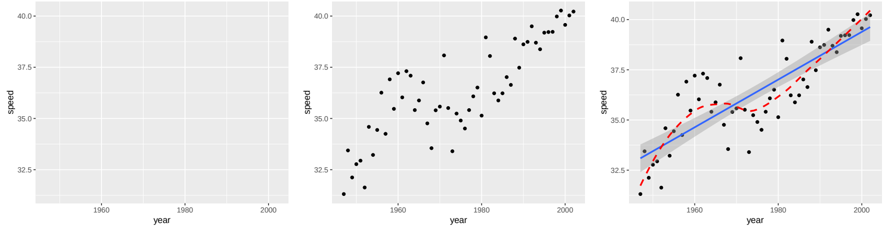
```r
# load ggplot2 and get some data:
require(ggplot2); data(Cars93,package="MASS"); source("https://tinyurl.com/une4s3g/getData_3.R")
# Now create a simple plot:
ggplot(tdf, mapping=aes(x=year,y=speed)) + geom_point()
```



ggplot2 Basics

- Let’s re-draw the plot step-by-step and we’ll add fitted curves.
- Notice ‘`ggplot()` just draws axes where `aes()` identifies the x and y variables.
- Additional features are added (literally, with a `+` operator).
- There are many `geom` functions.
    - `geom_point()` adds points. `geom_smooth()` adds a lowess curve (the default) and a least squares fit.
    - The x and y variables are inherited from the call to `ggplot()`.

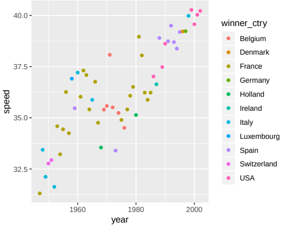
```
p1 <- ggplot(tdf, mapping=aes(x=year,y=speed)); p1
p2 <- p1 + geom_point(); p2
p3 <- p2 + geom_smooth(method=lm) + geom_smooth(color="red", linetype="dashed", se=FALSE); p3
```

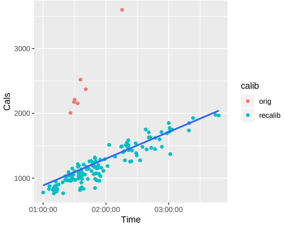


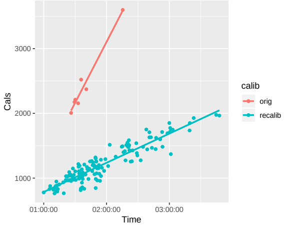
Aesthetics

- Aesthetic mappings always involve data. They determine how data influences the features of the plot.
- In many plots a single aesthetic mapping will be made in the initial call to ‘`ggplot()`. Subsequent functions (such as `geom` functions) **inherit** the mapping by default, but can also have their own `aes()` mappings to accomplish certain effects.

```r
ggplot(tdf, mapping=aes(x=year,y=speed,color=winner_ctry)) + geom_point()
ggplot(fred, mapping=aes(x=Time,y=Cals)) +
  geom_point(mapping=aes(color=calib)) + geom_smooth(method=lm,se=FALSE)
ggplot(fred, mapping=aes(x=Time,y=Cals,color=calib)) +
  geom_point() + geom_smooth(method=lm,se=FALSE)
```







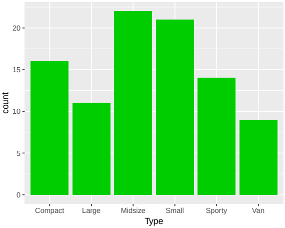
Aesthetics

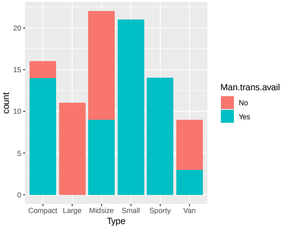
- There are several aesthetic features that can be mapped or set.
- Some roughly correspond to graphical parameters in base, but they differ and graphical parameters are not used in `ggplot2`.
    - `colour` (or `color`) and `fill`: Can take numbers or names with same values as used in base graphics.
    - `linetype`: Corresponds to and takes same values as `lty` in base graphics.
    - `size`: Width in mm. Corresponds to `lwd` in base.
    - `linejoin` and `lineend`: affect appearance of line joins (corners) and ends. Hard to see these effects unless you're using wide lines.
    - `shape`: Controls plotting symbols. Corresponds to and takes same values as `pch` in base graphics.
    - `family`: Controls font. Choices are `"sans"`, `"serif"`, or `"mono"`. Others can be implemented via secondary packages.
    - `fontface`: Controls font appearance. Choices are `"plain"`, `"bold"`, `"italic"`, `"bold.italic"`.
    - `hjust`, `vjust`: Control justification. Each take a number $\in [0, 1]$ or a string (`"top"`, `"middle"`, `"bottom"`, `"left"`, `"center"`, `"right"`).

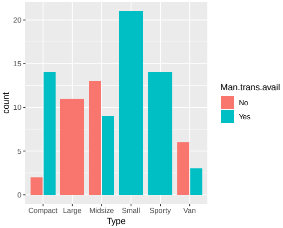
Mapping vs Setting Aesthetic Features

- Specifying a feature inside `aes()` *maps* the feature to a variable. To *set* the feature to a constant value, use it outside `aes()`.
- Bar charts are implemented with `geom_bar()`.
    - The 1st plot shows a univariate distribution, with `fill` set to a constant.
    - In the 2nd and 3rd plots, it is mapped to show a joint distribution.
    - Notice only `x` is specified in `aes()` for these plots.

```
ggplot(Cars93, aes(x=Type)) + geom_bar(fill="green3") # color and fill are set to constant values here (not in aes() function)
ggplot(Cars93, aes(x=Type,fill=Man.trans.avail)) + geom_bar() # stacked bars
# Previous line gives same result as next one (comented out):
# ggplot(Cars93, aes(x=Type)) + geom_bar(aes(fill=Man.trans.avail)) # stacked bars
ggplot(Cars93, aes(x=Type,fill=Man.trans.avail)) + geom_bar(position="dodge2") # clustered bars
```







geom Functions

- There are many functions that add geometric features (layers) to a plot.
- The ones below are part of ggplot2. Others are available in secondary packages.

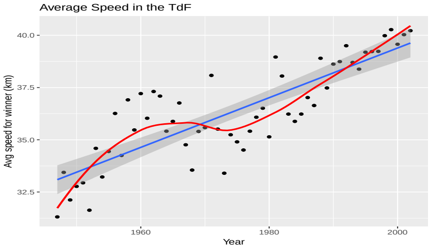
```
  geom_abline geom_density_2d  geom_linerange      geom_rug
    geom_area  geom_density2d        geom_map  geom_segment
     geom_bar    geom_dotplot       geom_path       geom_sf
   geom_bin2d   geom_errorbar      geom_point geom_sf_label
   geom_blank  geom_errorbarh geom_pointrange  geom_sf_text
 geom_boxplot    geom_freqpoly    geom_polygon   geom_smooth
     geom_col        geom_hex         geom_qq    geom_spoke
 geom_contour  geom_histogram    geom_qq_line     geom_step
   geom_count      geom_hline   geom_quantile     geom_text
geom_crossbar     geom_jitter     geom_raster     geom_tile
   geom_curve      geom_label       geom_rect   geom_violin
 geom_density       geom_line     geom_ribbon    geom_vline
```

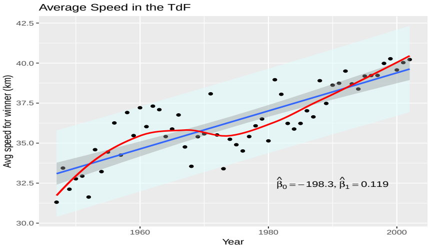
geom Functions

- geoms will use the data set specified in the ggplot() function or can use a different data set.

```
frame <- ggplot(tdf, aes(x=year,y=speed)) +
  xlab("Year") + ylab("Avg speed for winner (km)") + ggtitle("Average Speed in the TdF")
frame + geom_point() + geom_smooth(method="lm") + geom_smooth(se=FALSE,color="red")

# Here we add more features using geoms that use data from other data frames:
tdf.lm1.pi <- predict( lm(speed~year,data=tdf), interval="prediction") #don't worry about warning
pi.df <- data.frame(year=tdf$year, l.plim=tdf.lm1.pi[,2], u.plim=tdf.lm1.pi[,3])
txt.df <- data.frame(x=1990, y=32.5, txt = "list(hat(beta)[0]==-198.3,hat(beta)[1]==0.119 )" )
frame + geom_ribbon(data=pi.df, mapping=aes(x=year,ymin=l.plim,ymax=u.plim),
                    inherit.aes=F, fill="lightcyan", alpha=.5) + # alpha controls transparancy
  geom_point() + geom_smooth(method="lm") + geom_smooth(se=FALSE,color="red") +
  geom_text(data=txt.df,mapping=aes(x,y,label=txt),parse=T) # can avoid making txt.df with annotate() fn commented out below
#  annotate(geom="text", x=1990, y=32.5, label = "list(hat(beta)[0]==-198.3,hat(beta)[1]==0.119 )",parse=T )
```

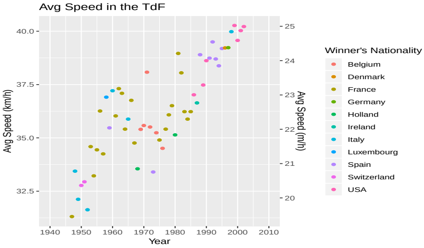




Modifying Axes and Scales

- In the previous example I used `ggtitle()` for a title and `xlab()` and `ylab()` for axis labels.
    - A more general function, `labs()`, can add title, subtitle, figure caption, and labels for aesthetics, which are useful because they appear in the legend.
- There are `xlim()` and `ylim()` functions to control the ranges of the axes.
- Functions like `xlab()` and `xlim()` are convenient, but can be replaced by a `scale` function.

```
ggplot(tdf, mapping=aes(year, speed, color=winner_ctry)) + geom_point() + labs(color="Winner's Nationality",title="Avg Speed in the TdF") +
  scale_x_continuous(name="Year", limits=c(1940,2010), breaks=seq(1940,2010,by=10), labels=as.character(seq(1940,2010,by=10))) +
  scale_y_continuous("Avg Speed (km/h)", sec.axis=sec_axis(~.*.6214,name="Avg Speed (m/h)"))
```



Scales

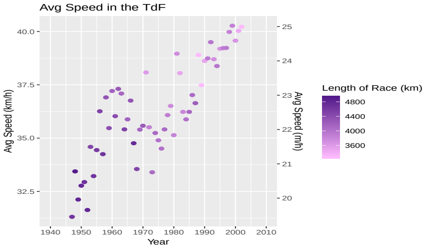
- As mentioned previously, `ggplot2` builds plots by combining components that can be manipulated separately. Scales are one of these components.
- The plot components are
  - data frames,
  - geometrical representations,
  - aesthetic mappings,
  - scales,
  - statistics from the data to be mapped,
  - position adjustments,
  - a coordinate system,
  - a faceting scheme.
- In addition, the overall appearance and some specific features are controlled by a `theme`.

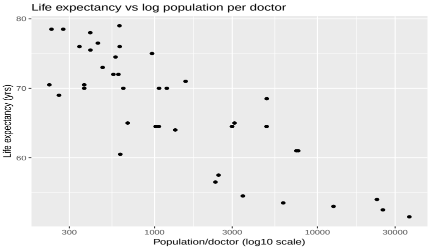
Scales

- Scales are functions that control the mapping from data to an aesthetic.
    - Every aesthetic has one; default scales are used but can be overridden/modified by using a `scale` function or functions like `xlab()` and `xlim()`.

```
ggplot(tdf, mapping=aes(year, speed, color=distance)) + geom_point(shape=19,size=2) + ggtitle("Avg Speed in the TdF") +
  scale_x_continuous(name="Year", limits=c(1940,2010), breaks=seq(1940,2010,by=10), labels=as.character(seq(1940,2010,by=10))) +
  scale_y_continuous("Avg Speed (km/h)", sec.axis=sec_axis(~.*.6214,name="Avg Speed (m/h)")) +
  scale_color_gradient("Length of Race (km)",low="Plum1",high="purple4")

ggplot(tvData, mapping=aes(x=popPerMD,y=lifeExpect)) + geom_point() +
  labs(title="Life expectancy vs log population per doctor",y="Life expectancy (yrs)") +
  scale_x_log10("Population/doctor (log10 scale)")
```





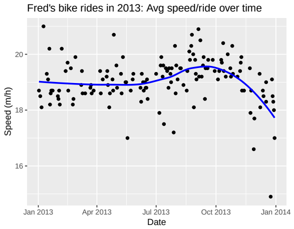
Scales

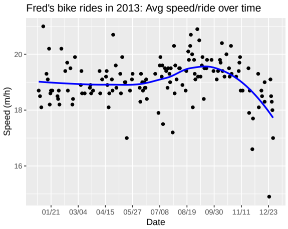
- Below we see the default scale for a variable of class `"Date"` (left).
- Next we modify that scale with the `scale_x_date()` function (middle).
- Finally, legends are generated only for mapped aesthetic features, so if we want to identify different `geoms` we have to map their aesthetics and create a suitable scale for those mappings (right).

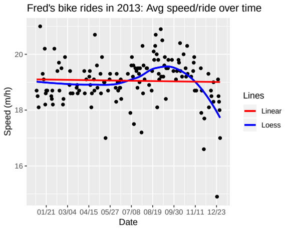
```
fred1 <-  ggplot(fred, mapping=aes(x=date,y=AvgSpd)) + geom_point() + geom_smooth(se=F,color="blue") +
  labs(title="Fred's bike rides in 2013: Avg speed/ride over time",y="Speed (m/h)",x="Date"); fred1

fred2 <- fred1 + scale_x_date(date_labels="%m/%d",date_breaks="6 weeks",limits=as.Date(c("2013-01-01","2013-12-31"))); fred2

fred2 + geom_smooth(se=F,mapping=aes(color="blue")) + geom_smooth(method="lm",se=F,mapping=aes(color="red")) +
  scale_colour_identity(name="Lines", breaks=c("red","blue"), labels=c("Linear","Loess"), guide="legend")
```







Coordinate Systems

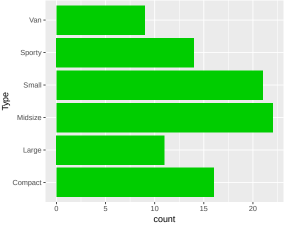
- The default and most common coordinate system is implemented in the `coord_cartesian()` function.
    - That is, in all our plots so far, there has been an implicit `+coord_cartesian()` added to our code.
- Other useful coordinate functions are
    - `coord_fixed()`, `coord_equal()`: implement fixed aspect ratio Cartesion coordinates.
    - `coord_flip()`: reverses the x and y variables.
    - `coord_map()`: for maps.
    - `coord_polar()`: polar coordinates.
    - `coord_trans()`: implements transformed Cartesian coordinates.

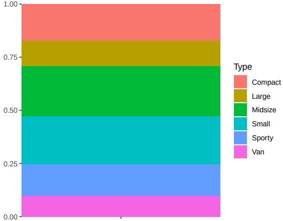
Coordinate Systems—Examples

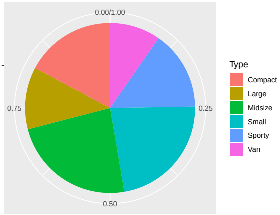
- Examples of `coord_flip()` and `coord_polar()`:

```
ggplot(Cars93, aes(x=Type)) + geom_bar(fill="green3") + coord_flip()

g <- ggplot(Cars93, aes(x="", fill=Type)) + geom_bar(width=1,position="fill") +
  scale_x_discrete(NULL, expand = c(0, 0)) + scale_y_continuous(NULL, expand = c(0, 0)); g
g + coord_polar(theta="y",start=0)
```





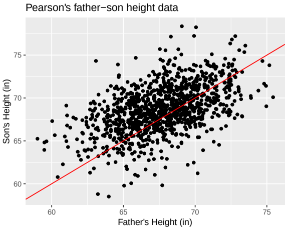


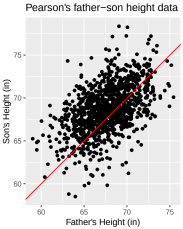
Coordinate Systems—Examples

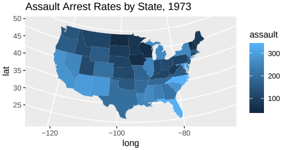
- Examples of `coord_equal()` and `coord_map()`. Note that the aspect ratio in these presentation slides is distorted.

```r
fs <- UsingR::father.son # Pearson's father-son height data
fs1 <- ggplot(fs,aes(x=fheight,y=sheight)) + geom_point() + geom_abline(slope=1,intercept=0,color="red")  +
  labs(x="Father's Height (in)",y="Son's Height (in)",title="Pearson's father-son height data"); fs1
fs1 + coord_equal()

# Example from help page for map_data() function from ggplot2 package:
states <- map_data("state"); arrests <- USArrests
names(arrests) <- tolower(names(arrests)); arrests$region <- tolower(rownames(USArrests))
choro <- merge(states, arrests, sort = FALSE, by = "region"); choro <- choro[order(choro$order), ]
ggplot(choro, aes(long, lat)) + geom_polygon(aes(group = group, fill = assault)) +
  coord_map("albers",  lat0 = 45.5, lat1 = 29.5) + ggtitle("Assault Arrest Rates by State, 1973")
```


Pearson's father−son height data


Pearson's father−son height data

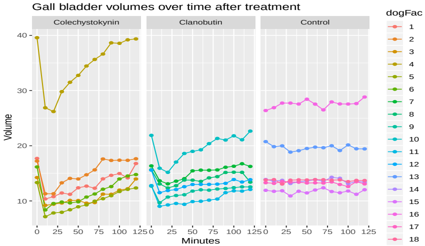

Assault Arrest Rates by State, 1973

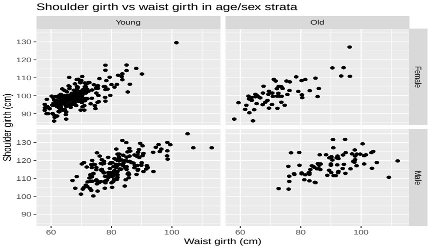
Facetting

- Facetting shows conditional relationships in paneled plots.
  - **facet_wrap()** builds plots at the level(s) of the conditioning variable(s) and adds them row-wise (**dir=h**) or column-wise (**dir=v**) to an array of plots.
  - With **facet_grid()**, plots in the grid are conditioned on a row value and a column value. These can each be values of a single variable or of combinations of variables.

```
gall$dogFac <- factor(gall$dogno); gall$trtfac <- factor(gall$trt,labels=c("Colechystokynin","Clanobutin","Control"))
ggplot(gall, aes(min,volume,group=dogFac,color=dogFac)) + geom_line() +  geom_point() +
  xlab("Minutes") + ylab("Volume") +  ggtitle("Gall bladder volumes over time after treatment") + facet_wrap(~trtfac, ncol = 3)

bodyDat$over34 <- factor(as.numeric(bodyDat$age>34),labels=c("Young","Old"))
ggplot(bodyDat,aes(x=waist_girth,y=should_girth)) + ggtitle("Shoulder girth vs waist girth in age/sex strata") +
  ylab("Shoulder girth (cm)") + xlab("Waist girth (cm)") +  geom_point() +  facet_grid(genderFac~over34)
```

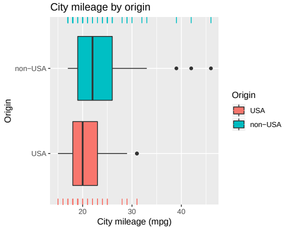


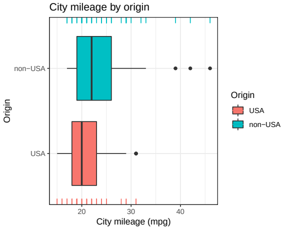


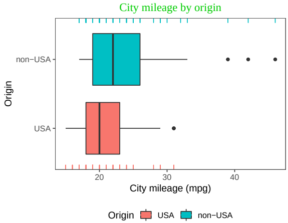
Themes

- The overall appearance of a plot is controlled by a theme. Themes control background color, font size and color, legend position, and much more.
- There are several complete themes in `ggplot2` with more in `ggthemes`.
    - The default theme is `theme_grey()`, but switching themes is easy by using other theme functions.
    - Alternatively, change specific elements in the current theme with `theme()`.

```
bp <- ggplot(Cars93) + geom_boxplot(aes(x = Origin, y = MPG.city, fill = Origin)) +
  ggtitle("City mileage by origin") + ylab("City mileage (mpg)") + coord_flip() +
  geom_rug(data=Cars93[Cars93$Origin!="USA",],mapping=aes(x=NULL,y=MPG.city,color=Origin), sides="t") +
  geom_rug(data=Cars93[Cars93$Origin=="USA",],mapping=aes(x=NULL,y=MPG.city,color=Origin), sides="b") +
  scale_colour_discrete(drop=FALSE) ; bp
bp + theme_bw()
bp + ggthemes::theme_few() + theme(legend.position="bottom", plot.title=element_text(family="serif",color="green3",hjust=.5))
```



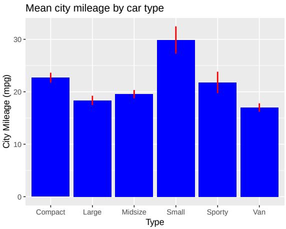


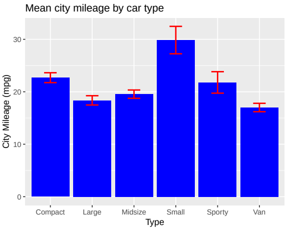


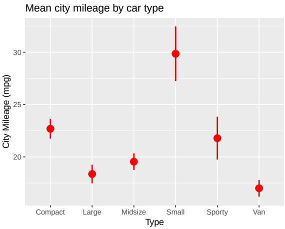
Examples—Bar Plots Showing Statistics by Group

- We've seen examples of bar plots showing (joint) distributions. For that we used `geom_bar()`.
- Here we use `geom_col()` for bar plots showing statistics by group (with error bars).

```r
mean.arr <- tapply(Cars93$MPG.city,Cars93$Type,mean); se.arr <- tapply(Cars93$MPG.city,Cars93$Type,function(x) sqrt(var(x)/length(x)))
df <- data.frame(Type=names(mean.arr),mn=mean.arr,se=se.arr)
p1 <- ggplot(df, aes(Type,mn,ymin = mean.arr-1.96*se.arr, ymax = mean.arr+1.96*se.arr)) +
  geom_col(fill="blue") + labs(title="Mean city mileage by car type",x="Type",y="City Mileage (mpg)")
p1 + geom_linerange(color="red",size=.8) # one style of error bar
p1 + geom_errorbar(color="red",width=.3,size=.8) # another style of error bar
# A better choice is to omit the bars entirely:
ggplot(df, aes(Type,mn,ymin = mean.arr-1.96*se.arr, ymax = mean.arr+1.96*se.arr)) +
  labs(title="Mean city mileage by car type",x="Type",y="City Mileage (mpg)") + geom_pointrange(color="red",size=.8)
```


Mean city mileage by car type


Mean city mileage by car type

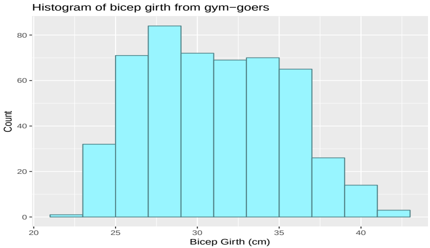

Mean city mileage by car type

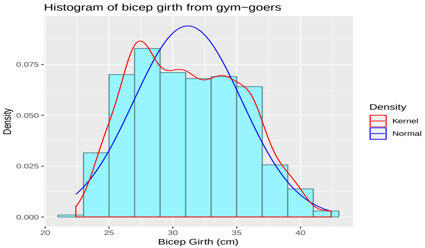
Examples—Histograms

- By default, `geom_histogram()` uses too many bins so always choose a binning scheme with one or more of the arguments `binwidth`, `bins`, `center`, `boundary`, `breaks`, and `closed`.

```
ggplot(bodyDat, aes(bicep_girth)) + geom_histogram(binwidth=2,color="cadetblue4",fill="cadetblue1") +
  xlab("Bicep Girth (cm)") +  ylab("Count") + ggtitle("Histogram of bicep girth from gym-goers")

ggplot(bodyDat, aes(x = bicep_girth)) +
  geom_histogram(aes(y =..density..),binwidth=2,color="cadetblue4",fill="cadetblue1") +
  xlab("Bicep Girth (cm)") + ylab("Density") + ggtitle("Histogram of bicep girth from gym-goers") +
  stat_function(aes(color="blue"),fun = dnorm,
                args = list(mean = mean(bodyDat$bicep_girth), sd = sd(bodyDat$bicep_girth))) +
  geom_density(aes(color="red"),adjust=.8) + # adjust value multiplies the default bandwidth
  scale_colour_identity(name="Density", breaks=c("red","blue"), labels=c("Kernel","Normal"), guide="legend")
```

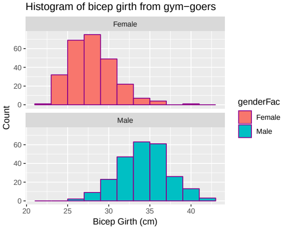

Histogram of bicep girth from gym−goers

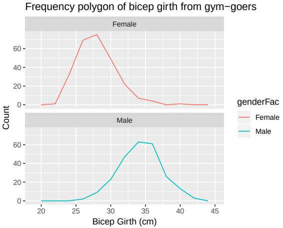

Histogram of bicep girth from gym−goers

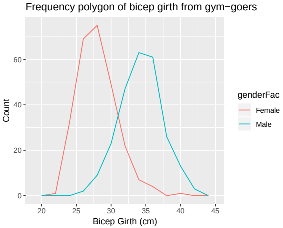
Examples—Histograms vs Frequency Polygons

- Recall that the bicep data are from men and women. Best to examine distribution by sex.

```
ggplot(bodyDat, aes(bicep_girth,fill=genderFac)) + geom_histogram(binwidth=2,color="darkmagenta") +
  xlab("Bicep Girth (cm)") +  ylab("Count") + ggtitle("Histogram of bicep girth from gym-goers") +
  facet_wrap(~genderFac, ncol = 1)
# Now use frequency polygon instead of histogram
fp1 <- ggplot(bodyDat, aes(bicep_girth,color=genderFac)) + geom_freqpoly(binwidth=2) +
  xlab("Bicep Girth (cm)") +  ylab("Count") + ggtitle("Frequency polygon of bicep girth from gym-goers")
fp1 + facet_wrap(~genderFac, ncol = 1)
# No need to use panels. Polygons look good when superimposed in the same panel
fp1
```


Histogram of bicep girth from gym−goers

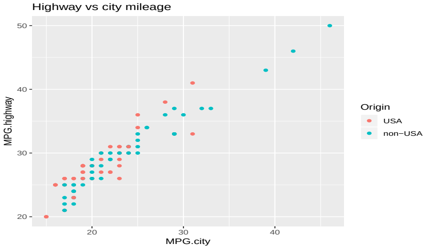

Frequency polygon of bicep girth from gym−goers

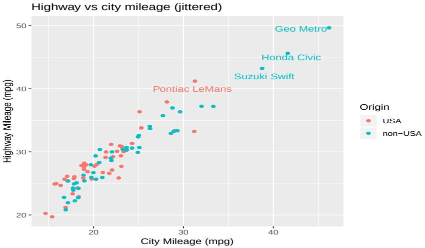

Frequency polygon of bicep girth from gym−goers

Examples—Scatterplots with Jittering and Labels

- To avoid overplotting it is useful to *jitter* the points.
- Points can be labeled with `geom_text()`. Need to adjust position of labels slightly. Adjustment can be done (and often must be done) in multiple ways.

```
ggplot(Cars93, aes(x=MPG.city,y=MPG.highway,color=Origin)) + geom_point() + ggtitle("Highway vs city mileage")
ggplot(Cars93, aes(x=MPG.city,y=MPG.highway,color=Origin)) +
  geom_jitter() + labs(x="City Mileage (mpg)",y="Highway Mileage (mpg)", title="Highway vs city mileage (jittered)") +
  geom_text(aes(label = Make),show.legend=F, data=Cars93[Cars93$MPG.city>45,], vjust = "inward", hjust = "inward") +
  geom_text(aes(label = Make),show.legend=F, data=Cars93[Cars93$MPG.highway>40 & Cars93$MPG.city<45,],
          nudge_y = -1)
```

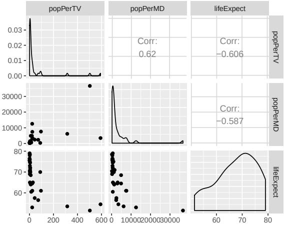


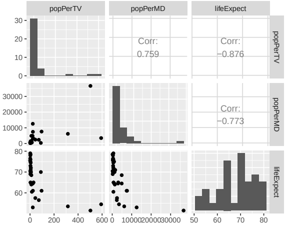


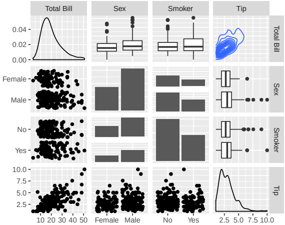
Examples—Plot Matrices

- The GGally package offers the ggpairs() function for plotting pairwise plot matrices.
- This yields scatter plot matrices when all variables are continuous (left and middle), but it works with variables of mixed scales (right).

```
require(GGally); ggpairs(na.omit(tvData), c("popPerTV","popPerMD","lifeExpect"))
# Default uses nonparametric densities on the diagonal and Pearson cors in upper tri, but these can be altered:
ggpairs(na.omit(tvData), columns=c("popPerTV","popPerMD","lifeExpect"),
        upper=list(continuous=wrap("cor",method="spearman")), diag=list(continuous=wrap("barDiag",bins=10)))
# Example from the ggpairs() help page involving the tips data set from reshape package.
data(tips, package = "reshape")
ggpairs(tips[,c("total_bill", "sex", "smoker", "tip")], columnLabels = c("Total Bill", "Sex", "Smoker", "Tip"),
  upper = list(continuous = "density", combo = "box_no_facet"),
  lower = list(continuous = "points", combo = "dot_no_facet") )
```



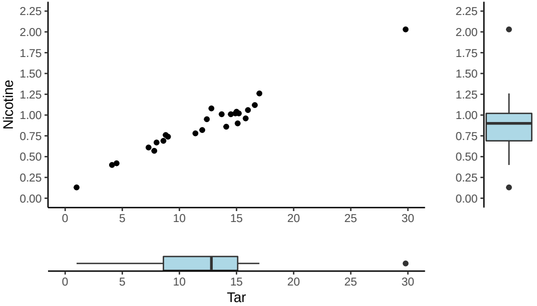




Examples—Multiple Plots per Page

- Multiple plots/page is implemented in several tools including `plot_grid()` of the `cowplot` package and `ggmatrix()` of the `GGally` package.
- Here's an example using `plot_grid()` in which alignment is important.

```
tar.nic.scatter <- ggplot(data=cigData, aes(x=tar,y=nicotine)) + geom_point() + theme_classic() +
  scale_x_continuous("",breaks=seq(0,30,by=5),limits=c(0,30)) + scale_y_continuous("Nicotine",breaks=seq(0,2.25,by=.25),limits=c(0,2.25))
tar.box <- ggplot(aes(y = tar), data = cigData) + geom_boxplot(fill = "lightblue") + theme_classic() + coord_flip() +
  scale_y_continuous("Tar",breaks=seq(0, 30, by=5),limits=c(0,30)) + scale_x_continuous(name=NULL,breaks=NULL)
nic.box <- ggplot(aes(y = nicotine), data = cigData) + geom_boxplot(fill = "lightblue") + theme_classic() +
  scale_y_continuous(name=NULL,breaks=seq(from=0, to=2.25, by=.25),limits=c(0,2.25)) + scale_x_continuous(name=NULL,breaks=NULL)
cowplot::plot_grid(tar.nic.scatter, nic.box, tar.box, align = "hv", ncol=2, nrow = 2, rel_widths = c(4, 1)/5, rel_heights = c(4, 1)/5)
```



Resources for Graphics in R

- Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis, Second Edition.* Springer.
  - Available through UGA Libraries for free.
- Friendly, M. (2018). Data Visualization in R, SCS Short Course. http://www.datavis.ca/courses/RGraphics/
  - The Session 4 slides focus on `ggplot2`.
- Tierney, L. (2019). STAT:4580 Data Visualizations and Data Technology. Course Notes.
- RStudio. Data Visualization with ggplot2:: Cheat Sheet. (All RStudio cheat sheets in a single PDF at this link.)
- `ggplot2` home page https://ggplot2.tidyverse.org/reference/

Thank You!

- If you need assistance with R or with selecting or implementing data visualizations to better understand your data, contact the SCC!
- We can help!

www.stat.uga/consulting

Finally…

- Holiday wishes, rendered with `ggplot2` and shamelessly stolen from the *Standard error* blog http://t-redactyl.io/: