Exploratory Data Analysis with R

Regression and Time Series

Xuemao Zhang
East Stroudsburg University

October 21, 2022

Outline

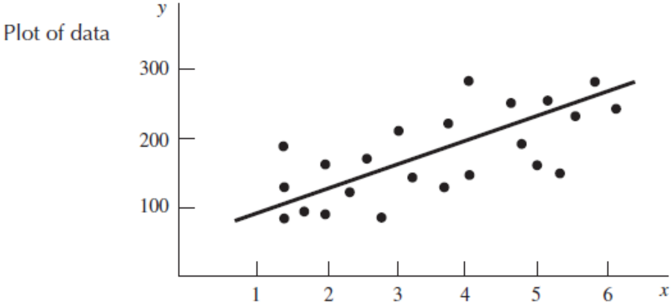
- Simple Linear Regression Models
  - Fit with a line
  - Fit with a curve: polynomial and local regression
- Logistic Regression
- Visualization of time series data
  - Time series
  - Area charts
  - Dummbbell charts
  - Slope graphs

SLR Models

When two variables are measured (not always but usually on a single experimental unit), the resulting data are called bivariate data (or Paired data). When both of the variables (X, Y) are quantitative, call the variable X - the *independent variable*, and Y - the *dependent variable*. A random sample is of the form

$$(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n).$$

Scatter plot can be used to check the relationship between X and Y. A typical scatter plot is like



SLR Models

Assume that visual examination of the scatter plot confirms that the points approximate a straight-line pattern

$$y = \beta_0 + \beta_1 x.$$

This model is called a **deterministic** mathematical model because it does not allow for any error in predicting y as a function of x.

However, the bivariate measurements that we observe do not generally fall exactly on a straight line, we choose to use a **probabilistic** model: for any fixed value of x,

$$E(Y|X = x) = \beta_0 + \beta_1 x.$$

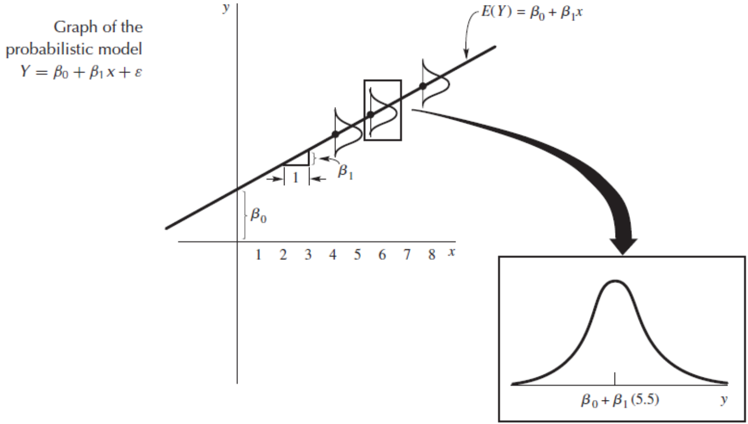
or, equivalently,

$$Y|_{X=x} = \beta_0 + \beta_1 x + \varepsilon,$$

where ε is a random variable possessing a specified probability distribution with mean 0.

For example, assume that ε's are independent normal random variables with mean 0 and common variance σ^2.

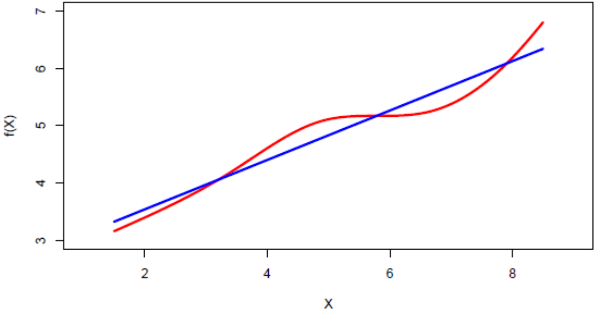
SLR Models



We estimate the population parameters β_0 and β_1 using sample information.

SLR Models

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of Y on the predictors $X_1, X_2, \ldots, X_p$ is linear.
- True regression functions are never linear!



- Although it may seem overly simplistic, linear regression is extremely useful both conceptually and practically.

SLR Models

Besides the estimation of the parameters β_0 and β_1, there are two more prediction problems.

- Prediction of $E(Y|X = x^*)$, the mean value of Y, for a fixed value of the independent variable $X = x^*$.
- Prediction of $Y|X = x^*$, a particular value of Y for a fixed value of the independent variable $X = x^*$.
- Note that $Y|X = x^*$ is a random variable, so it is harder to predict comared to the estimation of $E(Y|X = x^*)$.

SLR Models

- Consider the data set `Auto` in package `ISLR`, A data frame with 392 observations on 9 variables.

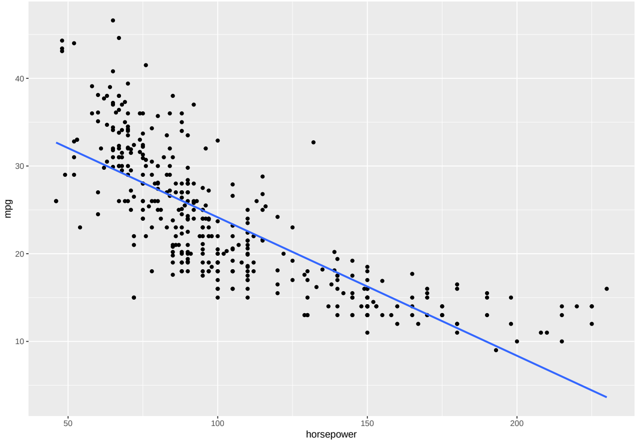
```
library(ISLR)
data("Auto", package="ISLR");
str(Auto);
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..
```

SLR Models

- Scatter plot with regression line

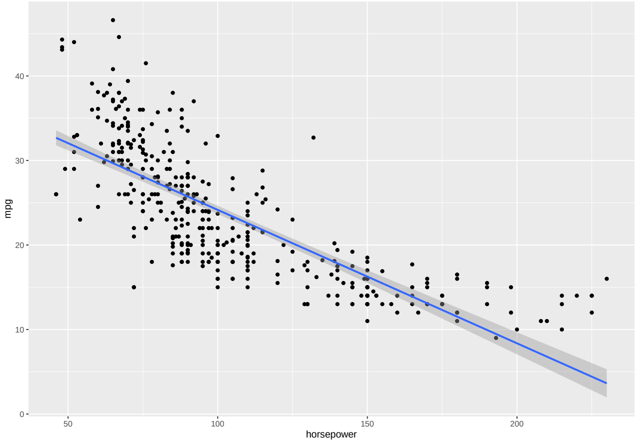
```r
ggplot(data=Auto, aes(x=horsepower,y=mpg))+
  geom_point()+
  geom_smooth(method=lm,formula = y~x, se=FALSE)
```



SLR Models

- Scatter plot with regression line
  - with confidence band

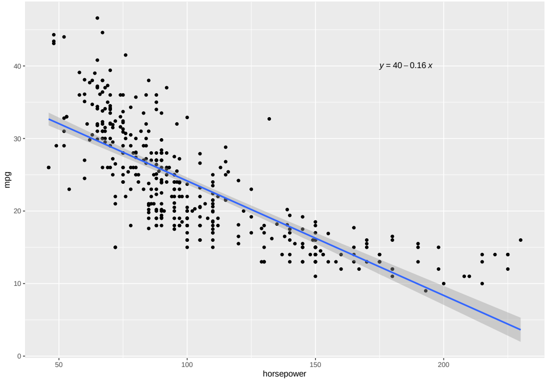
```
ggplot(data=Auto, aes(x=horsepower,y=mpg))+
  geom_point()+
  geom_smooth(method=lm,formula = y~x)
```



SLR Models

- Scatter plot with regression line
  - package ggpubr can let us add the regression equation

```
library(ggpubr);
ggplot(data=Auto, aes(x=horsepower,y=mpg))+
  geom_point()+
  geom_smooth(method=lm,formula = y~x)+
  stat_regline_equation(formula=y ~ x,label.x = 175,label.y =40 );
```



SLR Models

- Scatter plot with regression line
  - with prediction confidence band (predicting **individual** Y). Again, we need to fit the lm model first

```
model1 = lm(mpg~horsepower, data=Auto)
temp_var=predict(model1, interval="prediction")
```

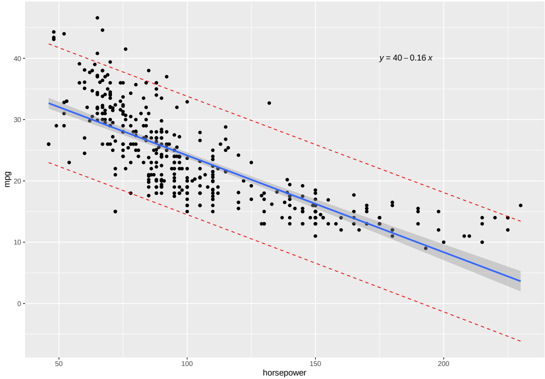
```
## Warning in predict.lm(model1, interval = "prediction"): predictions on c
```

```
new_df = cbind(Auto, temp_var)
str(new_df)
```

```
## 'data.frame':    392 obs. of  12 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 3
##  $ fit         : num  19.4 13.9 16.3 16.3 17.8 ...
```

SLR Models

```
ggplot(data=new_df, aes(x=horsepower,y=mpg))+
  geom_point()+
  geom_line(aes(y=lwr), color = "red", linetype = "dashed")+
 geom_line(aes(y=upr), color = "red", linetype = "dashed")+
  geom_smooth(method=lm,formula = y~x)+
  stat_regline_equation(formula=y ~ x,label.x = 175,label.y =40 );
```

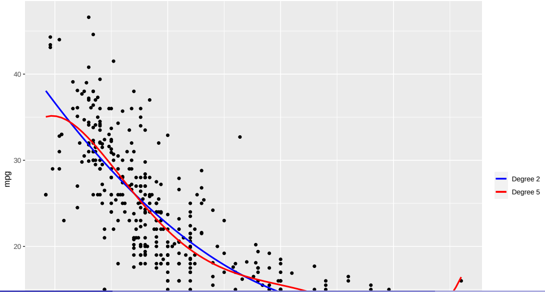


SLR Models - polynomial regression

- `mpg` and `horsepower` reveals a nonlinear relationship
- We consider a polynomial regression model first

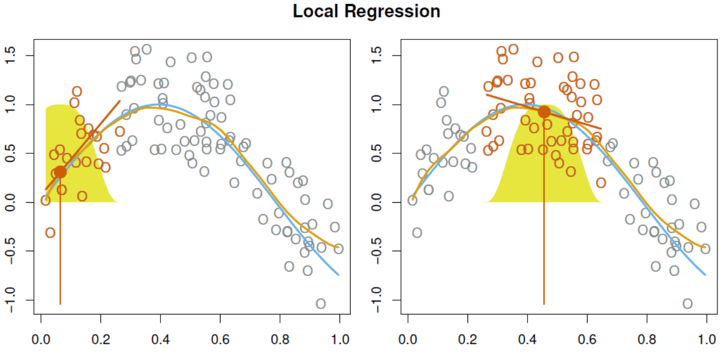
$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \varepsilon_i$$

```
ggplot(data=Auto, aes(x=horsepower,y=mpg))+
geom_point()+
geom_smooth(method=lm,formula = y~poly(x, 2),
aes(color="Degree 2"), se=FALSE)+
geom_smooth(method=lm,formula = y~poly(x, 5),
aes(color="Degree 5"), se=FALSE)+
scale_colour_manual(name="", values=c("blue", "red"))
```



SLR Models - local regression

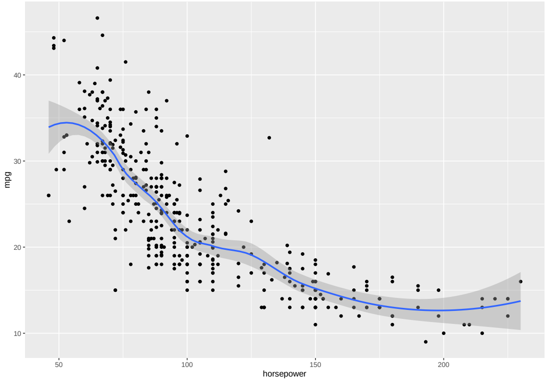
- Local regression is a different approach for fitting flexible non-linear functions, which involves computing the fit at a target point x_0 using only the nearby observations.



SLR Models - local regression

- In order to perform local regression, we use the `loess()` function;
- To visualize the fit, we use `method='loess'` in `geom_smooth`
  - span specifies the percentage of observations in each neighborhood

```
ggplot(data=Auto, aes(x=horsepower,y=mpg))+
geom_point()+
geom_smooth(method='loess',span=0.5,formula=y~x);
```



Logistic Regression

- Example: Credit Card Default

```
library(ISLR)
data("Default", package = "ISLR")
str(Default)
```

```
## 'data.frame':    10000 obs. of  4 variables:
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ..
##  $ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ..
##  $ balance: num  730 817 1074 529 786 ...
##  $ income : num  44362 12106 31767 35704 38463 ...
```

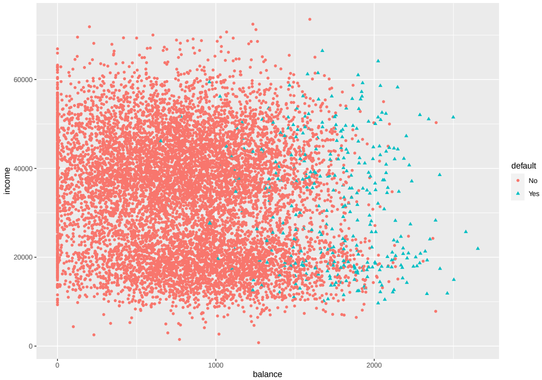
```
head(Default)
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

Logistic Regression

- Example: Credit Card Default

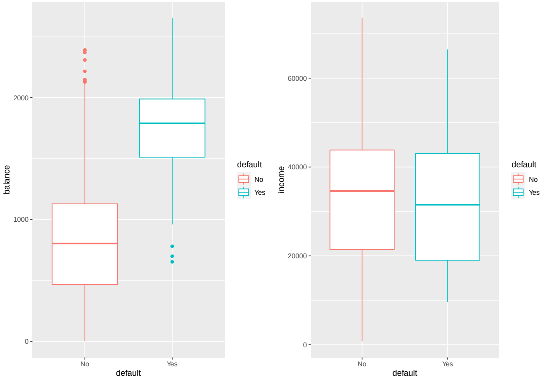
```
library(ggplot2);
ggplot(data=Default, aes(x=balance, y=income,
                         color=default, shape=default))+
  geom_point();
```



Logistic Regression

- Example: Credit Card Default

```
library(gridExtra);
p1=ggplot(data=Default, aes(x=default, y=balance, color=default))+
  geom_boxplot();
p2=ggplot(data=Default, aes(x=default, y=income, color=default))+
  geom_boxplot();
grid.arrange(p1, p2, ncol=2);
```



Logistic Regression

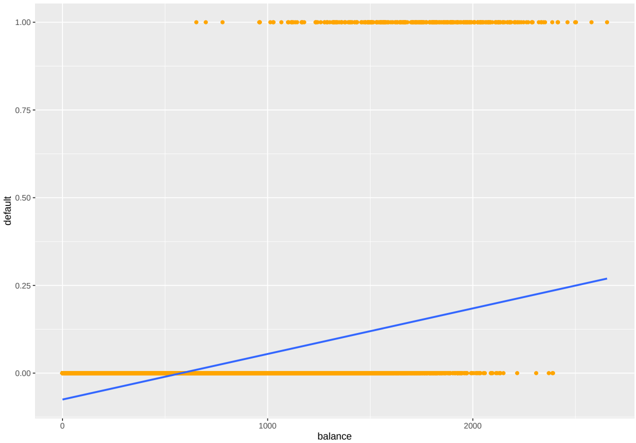
- Can we use Linear Regression?

Suppose $Y = \begin{cases} 0 & \text{if No} \\ 1 & \text{if Yes} \end{cases}$, Can we simply perform a linear regression of Y on X and classify as Yes if $\hat{Y} > 0.5$?

```
library(dplyr)
Default$default1=ifelse(Default$default=="No",0,1)
Default%>%select(-default)%>%rename(default=default1)->Default
```

Logistic Regression

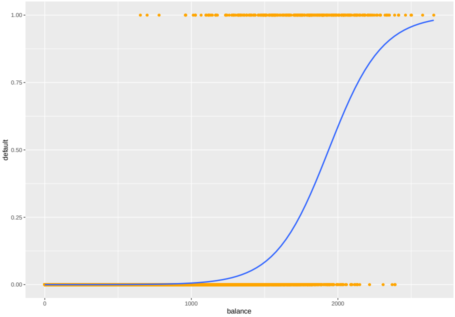
```
ggplot(data=Default, aes(x=balance,y=default))+
  geom_point(color="orange")+
  geom_smooth(method=lm, formula=y~x,se=FALSE);
```



Logistic Regression

- In this case of a binary outcome, linear regression might produce probabilities less than zero or bigger than one. So it can not give a good estimate of $E(Y|X=x) = Pr(Y=1|X=x)$. Logistic regression is more appropriate.

```
ggplot(data=Default, aes(x=balance,y=default))+
  geom_point(color="orange")+
  stat_smooth(method="glm", method.args=list(family="binomial"),
              se=FALSE)
```



Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the binay responses setting.
- We consider the case that $y \in \{0, 1\}$
- Let $p(X) = Pr(Y = 1|X)$
  - For example, we want to use biomarker level to predict probability of cancer.
- Logistic regression uses the form

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

  - $p(X)$ will lie between 0 and 1.
- Furthermore,

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X.$$

  - This function of $p(X)$ is called the logit or log odds (by log we mean natural log : ln).

Logistic Regression

- We use maximum likelihood to estimate the parameters
- Most statistical packages can fit linear logistic regression models by maximum likelihood. In R we use the `glm` function.

```
fit1=glm(formula=default~balance,family=binomial,data=Default);
summary(fit1)$coefficients;
```

```
##                  Estimate   Std. Error   z value      Pr(>|z|)
## (Intercept) -10.651330614 0.3611573721 -29.49221 3.623124e-191
## balance       0.005498917 0.0002203702  24.95309 1.976602e-137
```

Logistic Regression

- Logistic Regression with Several Variables
- Suppose that there are p predictors: $X_1, \ldots, X_p$.
- Just like before

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

- And just like before

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

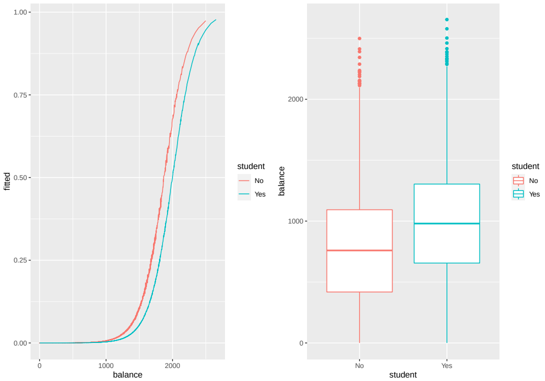
Logistic Regression

```
fit2=glm(formula=default~balance+income+student,
         family=binomial,data=Default);
summary(fit2)$coefficients;
```

```
##                  Estimate   Std. Error    z value      Pr(>|z|)
## (Intercept) -1.086905e+01 4.922555e-01 -22.080088 4.911280e-108
## balance      5.736505e-03 2.318945e-04  24.737563 4.219578e-135
## income       3.033450e-06 8.202615e-06   0.369815  7.115203e-01
## studentYes  -6.467758e-01 2.362525e-01  -2.737646  6.188063e-03
```

Logistic Regression

```
fitted=fit2$fitted.values;
Default=cbind(Default, fitted);
p1=ggplot(data=Default, aes(x=balance, y=fitted, color=student))+
  geom_line();
p2=ggplot(data=Default,aes(x=student, y=balance, color=student))+
  geom_boxplot();
grid.arrange(p1,p2,ncol=2);
```

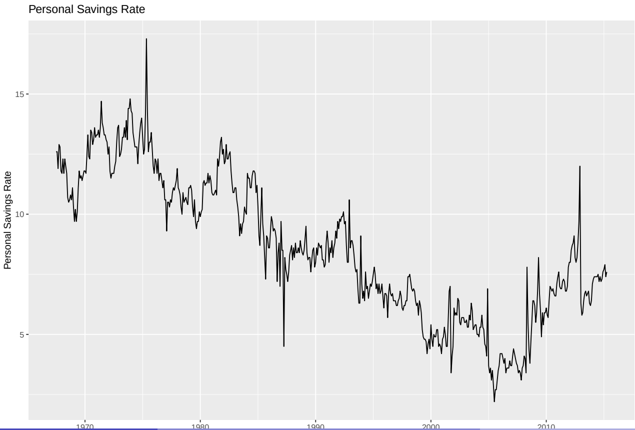


Visualizing time series data

- A graph can be a powerful vehicle for displaying change over time. The most common time-dependent graph is the time series line graph.
- A **time series** is a set of quantitative values obtained at successive time points. The intervals between time points (e.g., hours, days, weeks, months, or years) are usually equal.
- Consider the Economics time series that come with the `ggplot2` package. It contains US monthly economic data collected from January 1967 through January 2015.

Visualizing time series data

```
library(ggplot2)
ggplot(economics, aes(x = date, y = psavert)) +
geom_line() +
labs(title = "Personal Savings Rate",
x = "Date",
y = "Personal Savings Rate");
```



Visualizing time series data

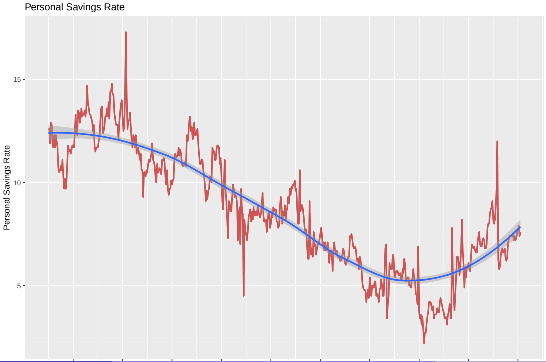
- The `scale_x_date` function can be used to reformat dates

```
library(ggplot2)
ggplot(economics, aes(x = date, y = psavert)) +
geom_line() +
  scale_x_date(date_breaks='5 years',date_labels="%Y") +
labs(title = "Personal Savings Rate",
x = "Date",
y = "Personal Savings Rate");
```

Visualizing time series data

- We can fit the time-series data using local regression

```
library(ggplot2)
ggplot(economics, aes(x = date, y = psavert)) +
geom_line(color = "indianred3",size=1) +
  scale_x_date(date_breaks='5 years',date_labels="%Y") +
  geom_smooth(method="loess")+
labs(title = "Personal Savings Rate",
x = "Date", y = "Personal Savings Rate");
```



Visualizing time series data

- The quantmod package for R is designed to assist the quantitative trader in the development, testing, and deployment of statistically based trading models.

- get apple (AAPL) closing prices from Jan 1, 2015

```
library(quantmod)
library(dplyr);
getSymbols("AAPL",return.class = "data.frame",
from="2015-01-01");
```

```
## [1] "AAPL"
```

```
apple=AAPL%>%mutate( Date=as.Date(row.names(.)) )%>%
dplyr::select(Date,AAPL.Close)%>%
  rename(Close = AAPL.Close)%>%mutate(Company = "Apple");
head(apple)
```

```
##                  Date   Close Company
## 2015-01-02 2015-01-02 27.3325   Apple
## 2015-01-05 2015-01-05 26.5625   Apple
## 2015-01-06 2015-01-06 26.5650   Apple
## 2015-01-07 2015-01-07 26.9375   Apple
```

Visualizing time series data

- get facebook (META; It was FB) closing prices from Jan 1, 2015

```
getSymbols("META",return.class = "data.frame",
from="2015-01-01");
```

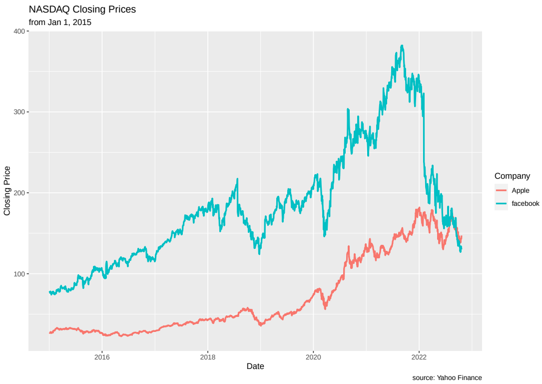
```
## [1] "META"
```

```
facebook=META%>%mutate( Date=as.Date(row.names(.)) )%>%
dplyr::select(Date,META.Close)%>%
  rename(Close = META.Close)%>%mutate(Company = "facebook");
head(facebook)
```

```
##                  Date Close  Company
## 2015-01-02 2015-01-02 78.45 facebook
## 2015-01-05 2015-01-05 77.19 facebook
## 2015-01-06 2015-01-06 76.15 facebook
## 2015-01-07 2015-01-07 76.15 facebook
## 2015-01-08 2015-01-08 78.18 facebook
## 2015-01-09 2015-01-09 77.74 facebook
```

Visualizing time series data

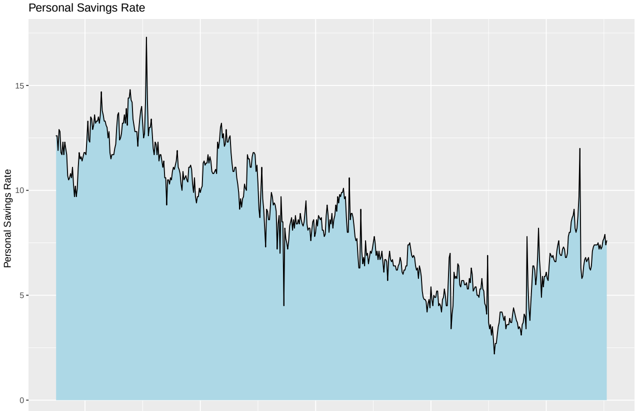
```
# combine data for both companies
mseries <- rbind(apple, facebook);
ggplot(mseries, aes(x=Date, y= Close, color=Company)) +
geom_line(size=1) +
  labs(title = "NASDAQ Closing Prices",
       subtitle = "from Jan 1, 2015",
     caption = "source: Yahoo Finance",y = "Closing Price")
```



Area charts

- A simple area chart is basically a line graph, with a fill from the line to the x-axis.

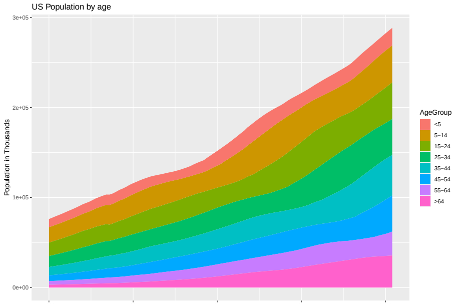
```
ggplot(economics, aes(x=date, y=psavert)) +
geom_area(fill="lightblue", color="black") +
labs(title = "Personal Savings Rate",
x = "Date",y = "Personal Savings Rate")
```



Area charts

- A **stacked area chart** can be used to show differences between groups over time.
  - Consider the uspopage dataset from the gcookbook package (Data for "R Graphics Cookbook"). We'll plot the age distribution of the US population from 1900 and 2002.

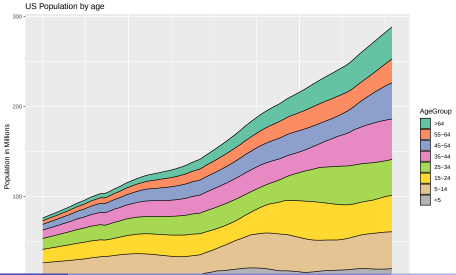
```
library(gcookbook)
data(uspopage, package="gcookbook")
ggplot(uspopage, aes(x=Year,y=Thousands,fill=AgeGroup)) +
  geom_area() +
labs(title="US Population by age",
x="Year",y="Population in Thousands");
```



Area charts

- Now let's
  - change the unit from thousands to millions
  - create black borders to highlight the difference between groups
  - reverse the order the groups to match increasing age using `forcats::fct_rev`
  - choose a different color scheme

```
uspopage$AgeGroup=forcats::fct_rev(uspopage$AgeGroup);
ggplot(uspopage, aes(x=Year,y=Thousands/1000,fill=AgeGroup)) +
  geom_area(color = "black") +
  scale_fill_brewer(palette = "Set2")+
labs(title="US Population by age",
x="Year",y="Population in Millions");
```



Area charts

- Apparently, the number of young children have not changed very much in the past 100 years.
- Stacked area charts are most useful when interest is on both
  - 1. group change over time
  - 2. overall change over time.
- Place the most important groups at the bottom of a stacked area chart.

Dummbbell charts

- **Dumbbell charts** are useful for displaying change between two time points for several groups or observations.
  - The `geom_dumbbell` function from the `ggalt` package is used.
- Using the `gapminder` dataset let's plot the change in life expectancy from 1952 to 2007 in the Americas. The dataset is in long format. We will need to convert it to wide format in order to create the dumbbell plot

```
library(gapminder)
data(gapminder, package = "gapminder");
unique(gapminder$year);
```

```
##  [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007
```

```
# subset data
plotdata_long <-   dplyr::filter(gapminder,continent=="Americas"
                  & year %in% c(1952, 2007)) %>%
dplyr::select(country, year, lifeExp);
```

Dummbbell charts

- convert data to wide format

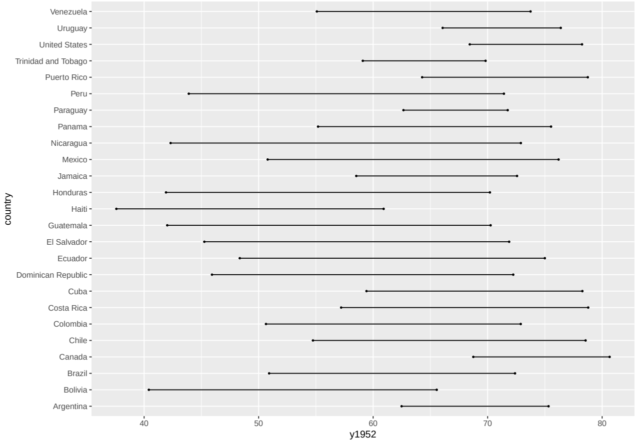
```
library(tidyr)
plotdata_wide <- pivot_wider(plotdata_long,
                   names_from=year, values_from=lifeExp)
names(plotdata_wide) <- c("country", "y1952", "y2007")
head(plotdata_wide);
```

```
## # A tibble: 6 x 3
##   country   y1952 y2007
##   <fct>     <dbl> <dbl>
## 1 Argentina  62.5  75.3
## 2 Bolivia    40.4  65.6
## 3 Brazil     50.9  72.4
## 4 Canada     68.8  80.7
## 5 Chile      54.7  78.6
## 6 Colombia   50.6  72.9
```

Dummbbell charts

- create dumbbell plot

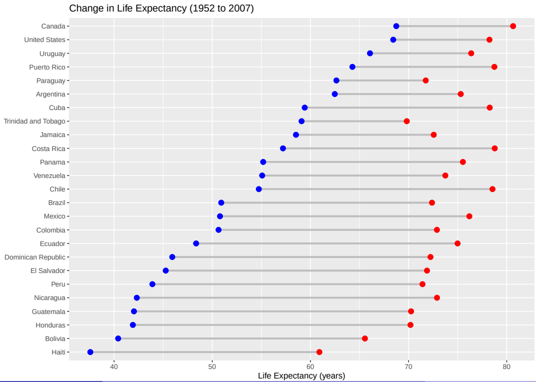
```
library(ggalt)
ggplot(plotdata_wide, aes(y=country,x=y1952, xend=y2007))+
  geom_dumbbell()
```



Dummbbell charts

- create dumbbell plot

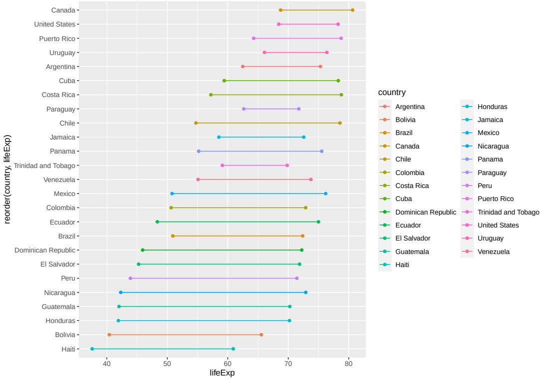
```
ggplot(plotdata_wide,#order factor country by y1952
aes(y = reorder(country, y1952),x = y1952,xend = y2007)) +
geom_dumbbell(size = 1.2,size_x = 3,size_xend = 3,colour = "grey",
      colour_x = "blue",colour_xend = "red")+
  labs(title = "Change in Life Expectancy (1952 to 2007)",
       x = "Life Expectancy (years)",y = "");
```



Dummbbell charts

- create dumbbell plot using ggplot2

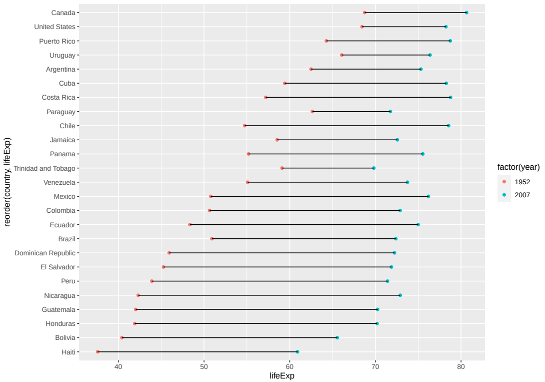
```
gapminder %>% dplyr::filter(continent=="Americas"
                  & year %in% c(1952, 2007)) %>%
dplyr::select(country, year, lifeExp) %>%
  ggplot(aes(x=lifeExp,y=reorder(country,lifeExp), color=country))+
  geom_point()+
  geom_line(aes(group=country))
```



Dummbbell charts

- create dumbbell plot using ggplot2

```
gapminder %>% dplyr::filter(continent=="Americas"
                  & year %in% c(1952, 2007)) %>%
dplyr::select(country, year, lifeExp) %>%
  ggplot(aes(x=lifeExp,y=reorder(country,lifeExp)))+
  geom_point(aes(color=factor(year)))+
  geom_line(aes(group=country))
```



Slope graphs

- When there are several groups and several time points, a slope graph can be helpful
- The function `newggslopegraph` in the package `CGPfunctions` makes the job much easier than using `ggplot2`.
- Function `newggslopegraph` arguments
  - data frame
  - time variable (which must be a factor)
  - numeric variable to be plotted
  - grouping variable (creating one line per group).

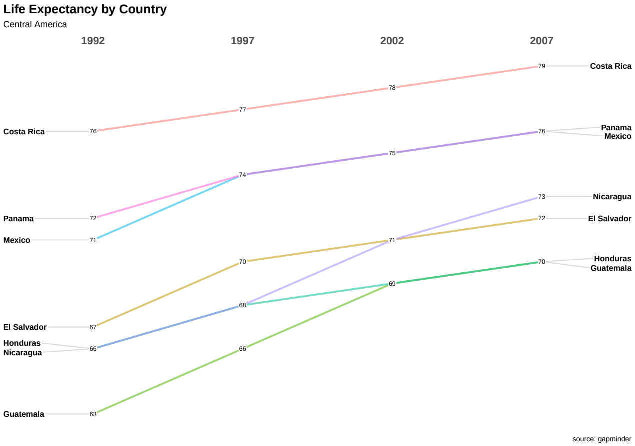
Slope graphs

- Let's plot life expectancy for seven Central American countries in 1992, 1997, 2002, and 2007. Again we'll use the gapminder data.

```
Americas=unique(gapminder[which(gapminder$continent==
                                  "Americas"),]$country)
write.csv(Americas, file = "Americas.csv");
countries=c("Mexico","Panama", "Costa Rica","Nicaragua",
            "Honduras", "El Salvador", "Guatemala","Belize")
df6 <-gapminder%>%filter(year %in% c(1992,1997,2002,2007) &
  country %in% countries)%>%
  mutate(year=factor(year), lifeExp=round(lifeExp));
```

Slope graphs

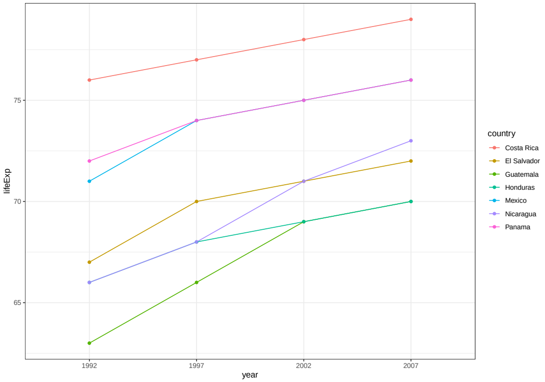
```r
library(CGPfunctions)
# create slope graph
newggslopegraph(df6, year, lifeExp, country) +
labs(title="Life Expectancy by Country",
subtitle="Central America",caption="source: gapminder")
```



Slope graphs

- using ggplot2

```
#levels(df6$year)
df6 %>% ggplot(aes(x=year,y=lifeExp,color=country))+
  geom_point()+
  geom_line(aes(group=country))
```



License