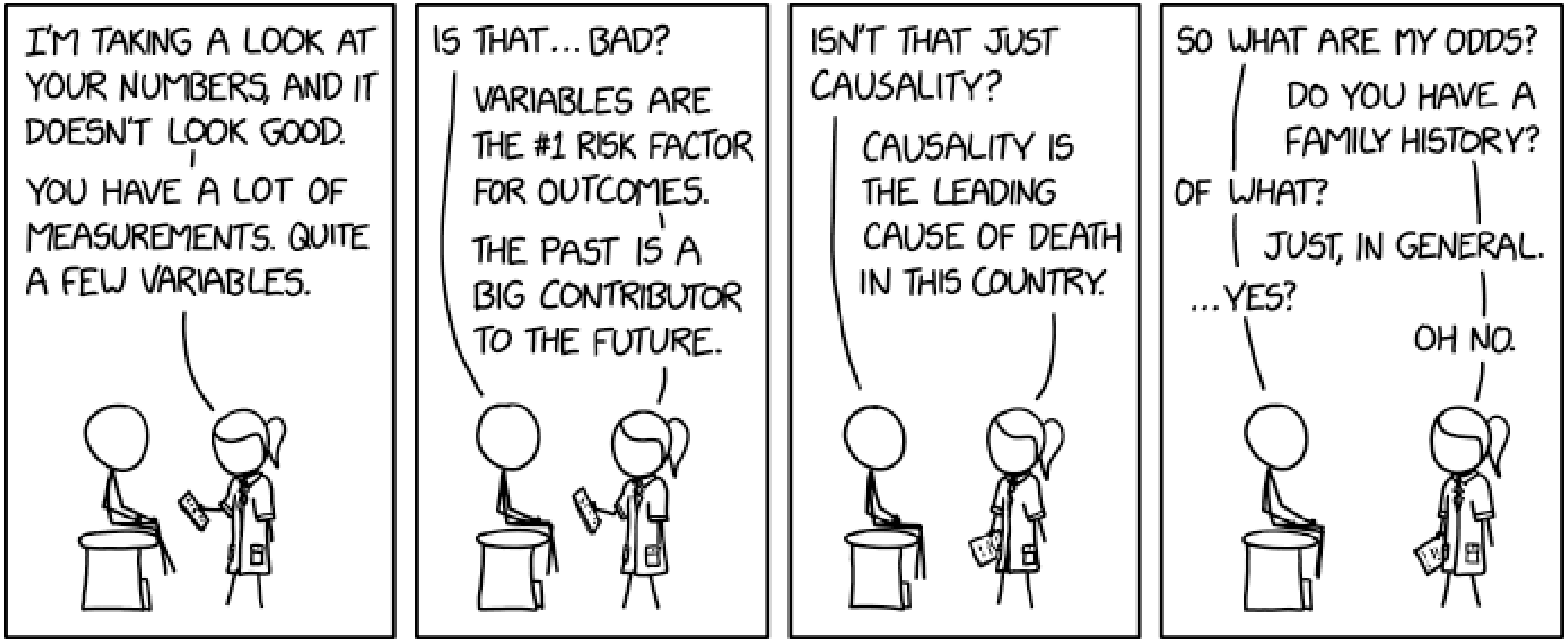


Source: xkcd.com/2620

INTRODUCTION TO FORECASTING & TIME SERIES STRUCTURE

Dr. Susan Simmons

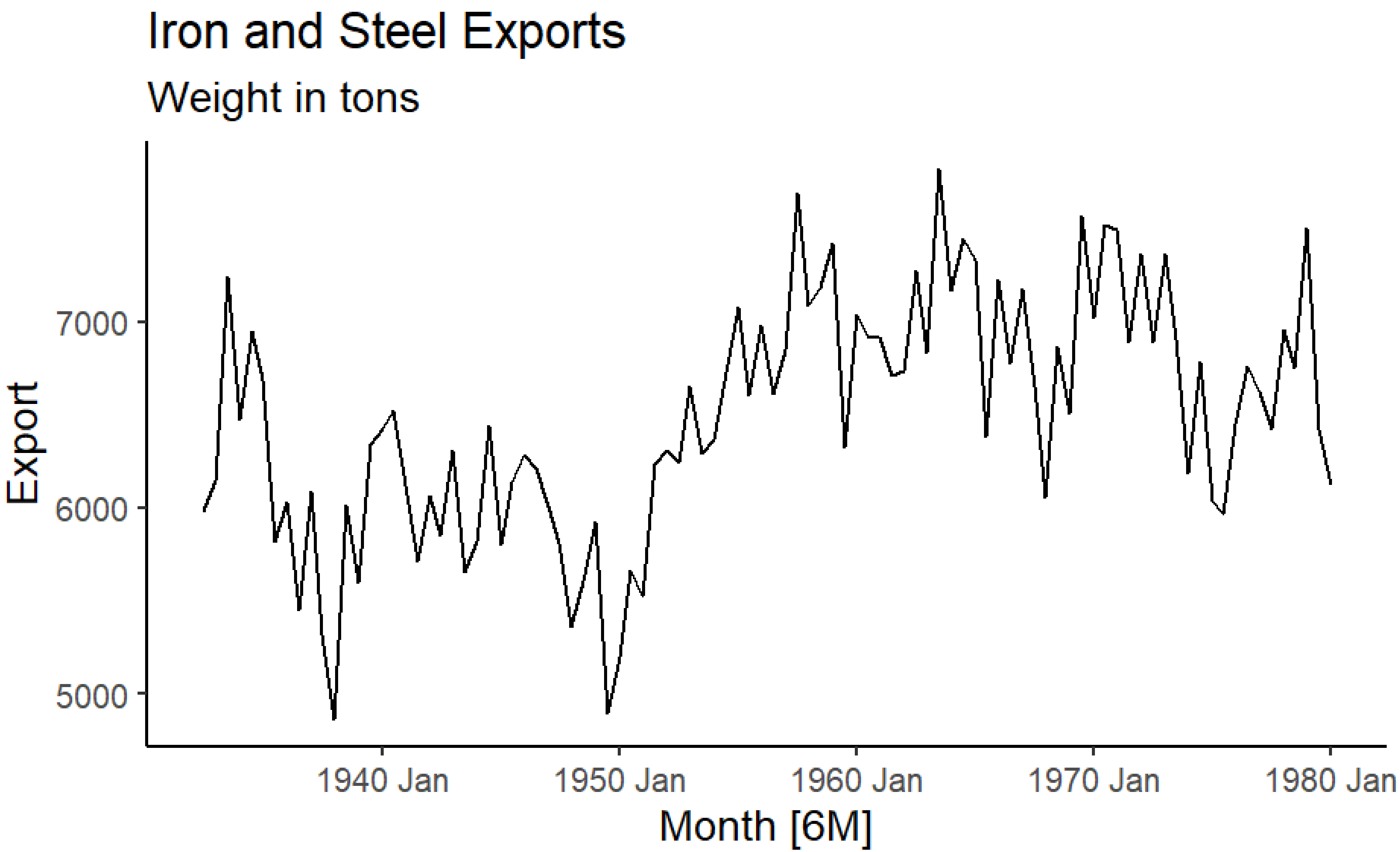
Institute for Advanced Analytics

PLOTTING DATA

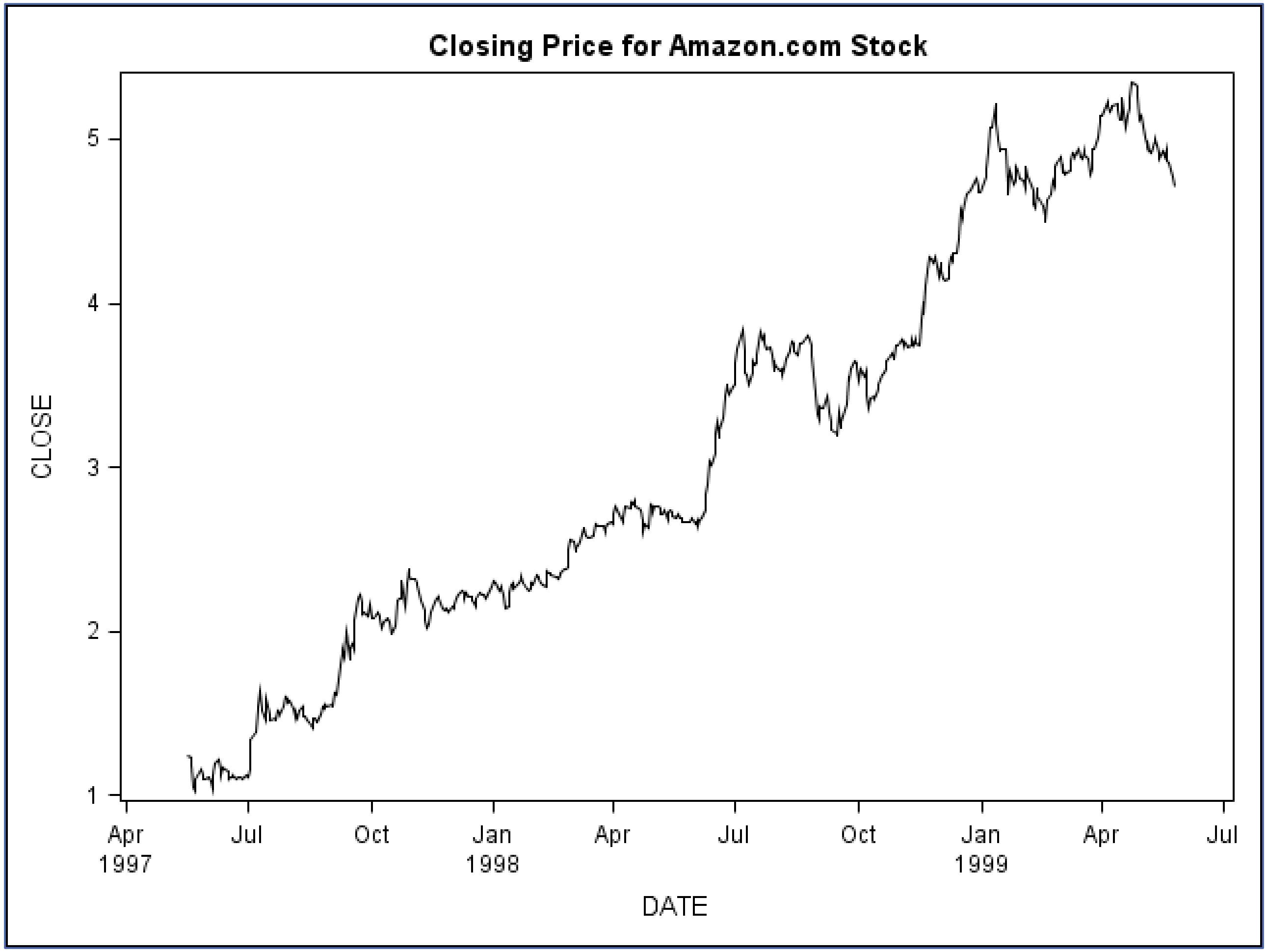
Time plots

- You should ALWAYS plot your data!!
- Best plot for time series data is the time plot (time on x-axis and variable on y-axis)
- Can observe what is happening to this variable across time

Example 1: Iron and Steel Exports

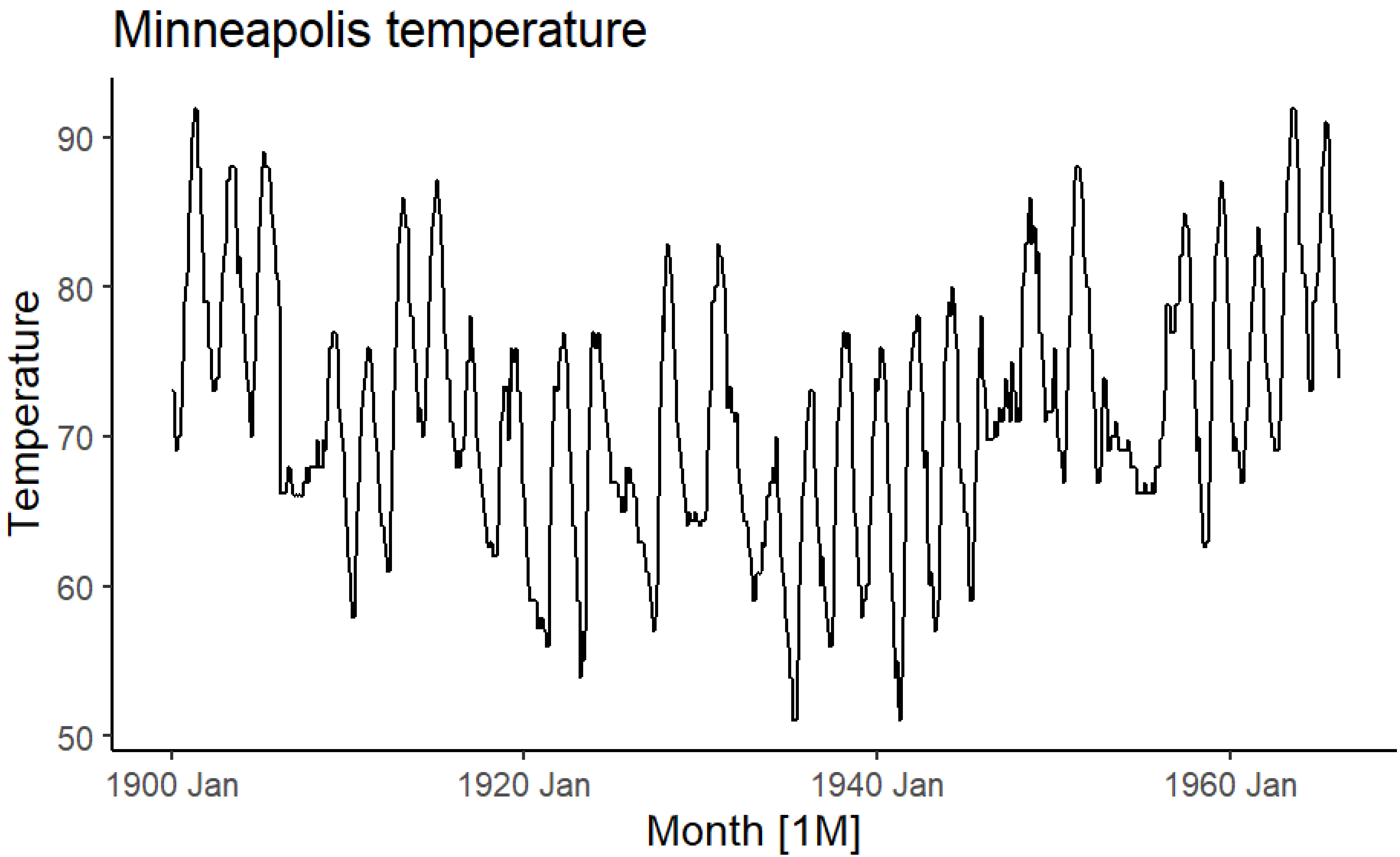


Example 2: Amazon.com Stock

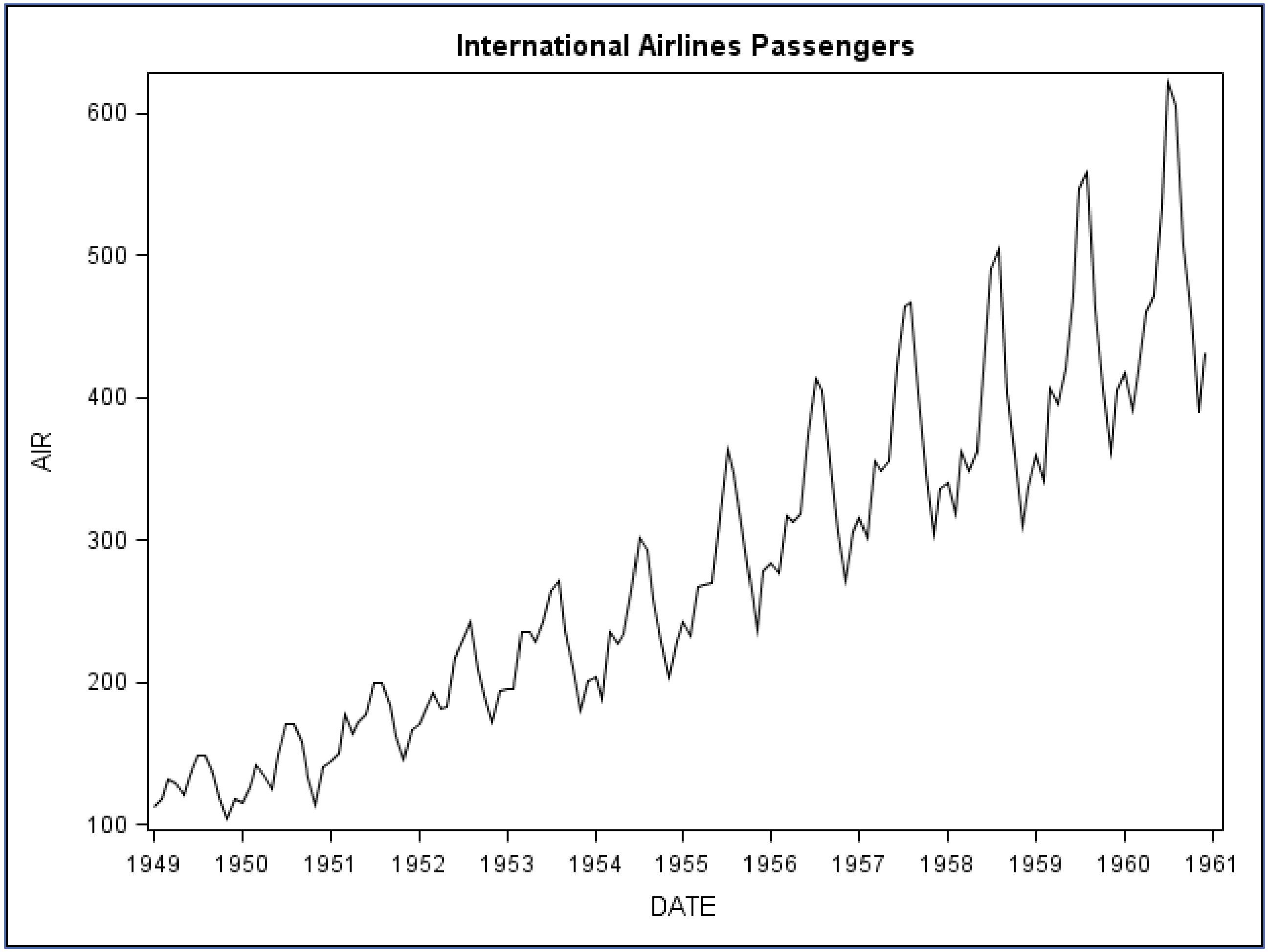


Example 2: Amazon.com Stock

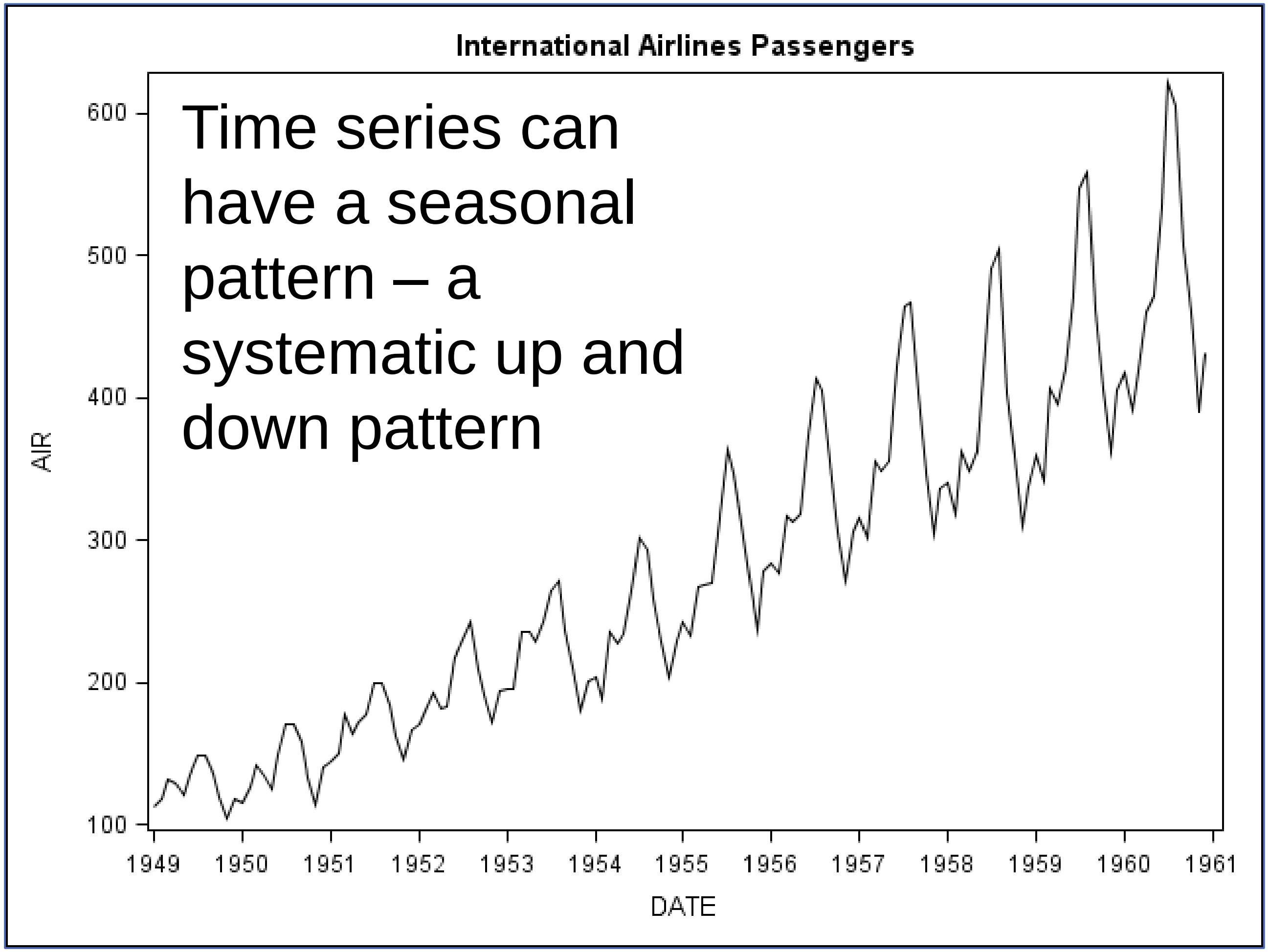
Quadratic trend



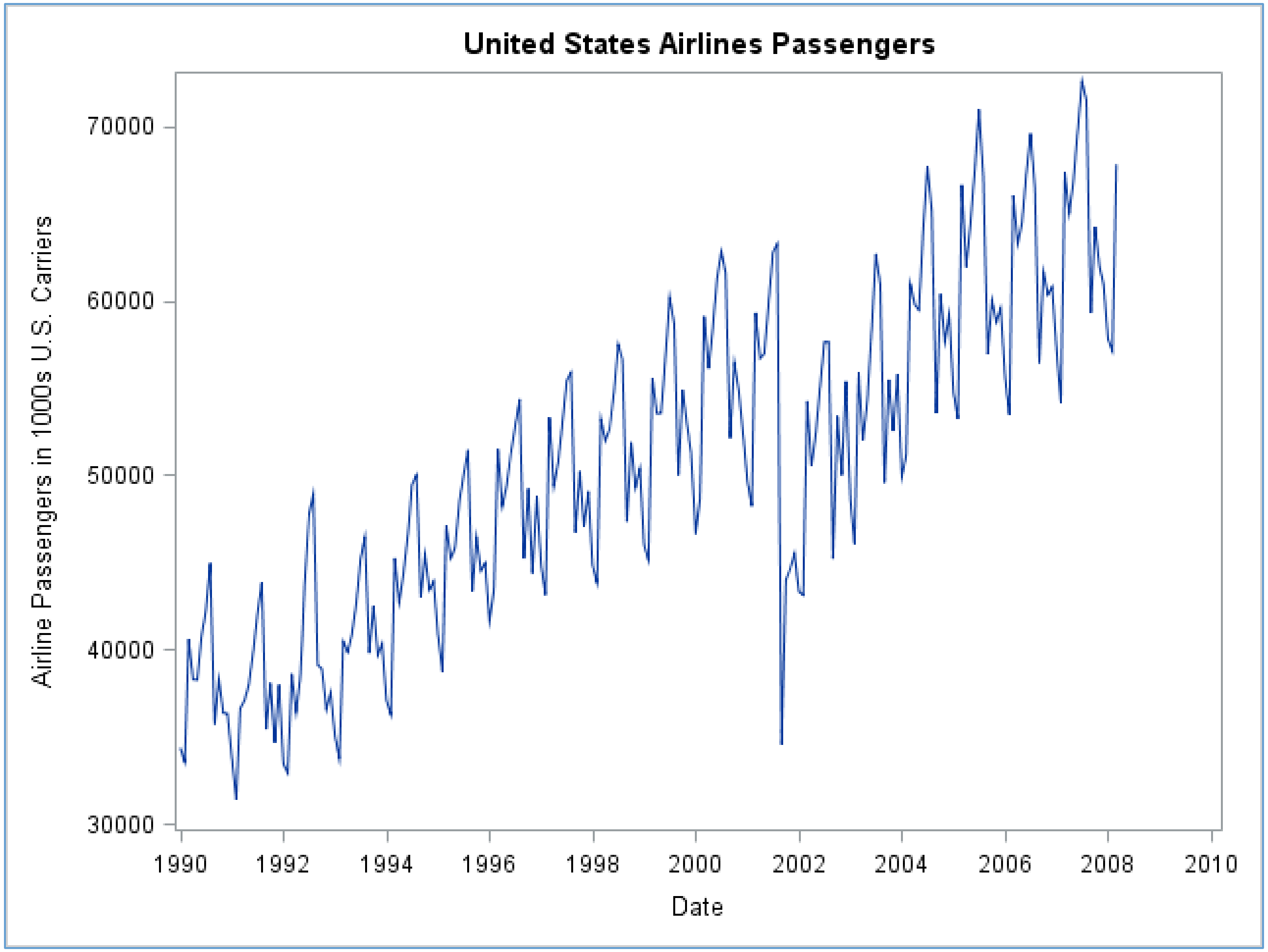
Example 3: Airlines Passengers



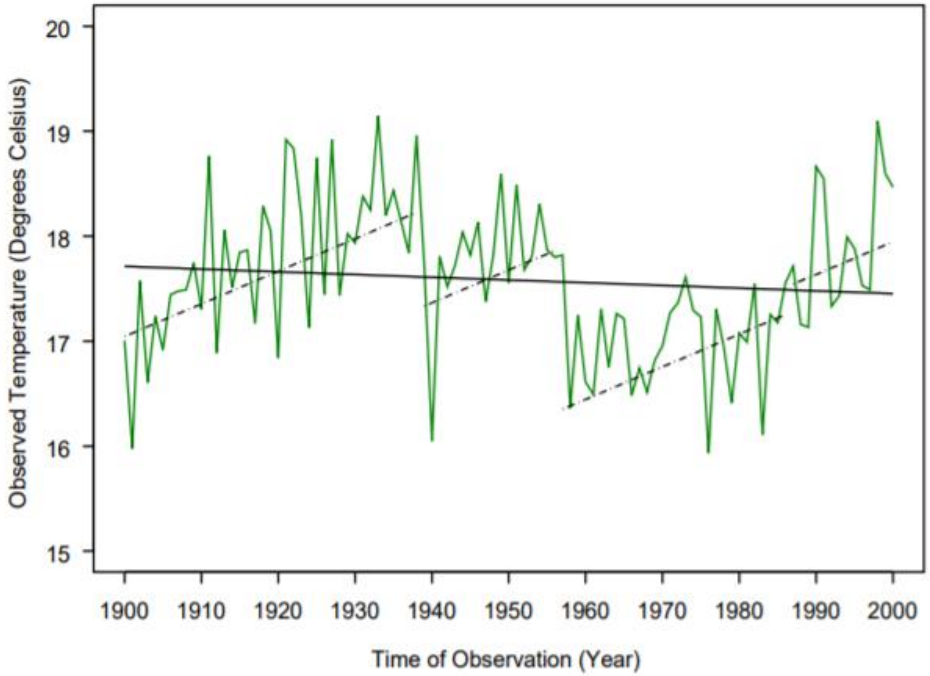
Example 3: Airlines Passengers



Example 5: Airline Passengers Again



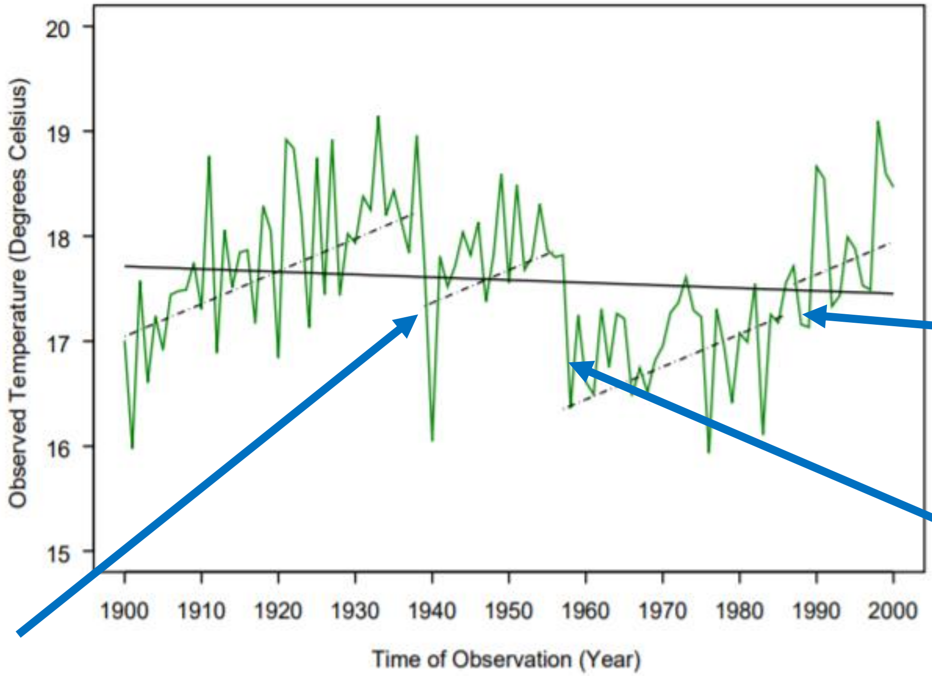
Temperature over the past century for Tuscaloosa, Alabama



Yearly Temperatures at Tuscaloosa AL With Least Squares Trends

Source: Dr. Robert Lund

Temperature over the past century for Tuscaloosa, Alabama



Yearly Temperatures at Tuscaloosa AL With Least Squares Trends

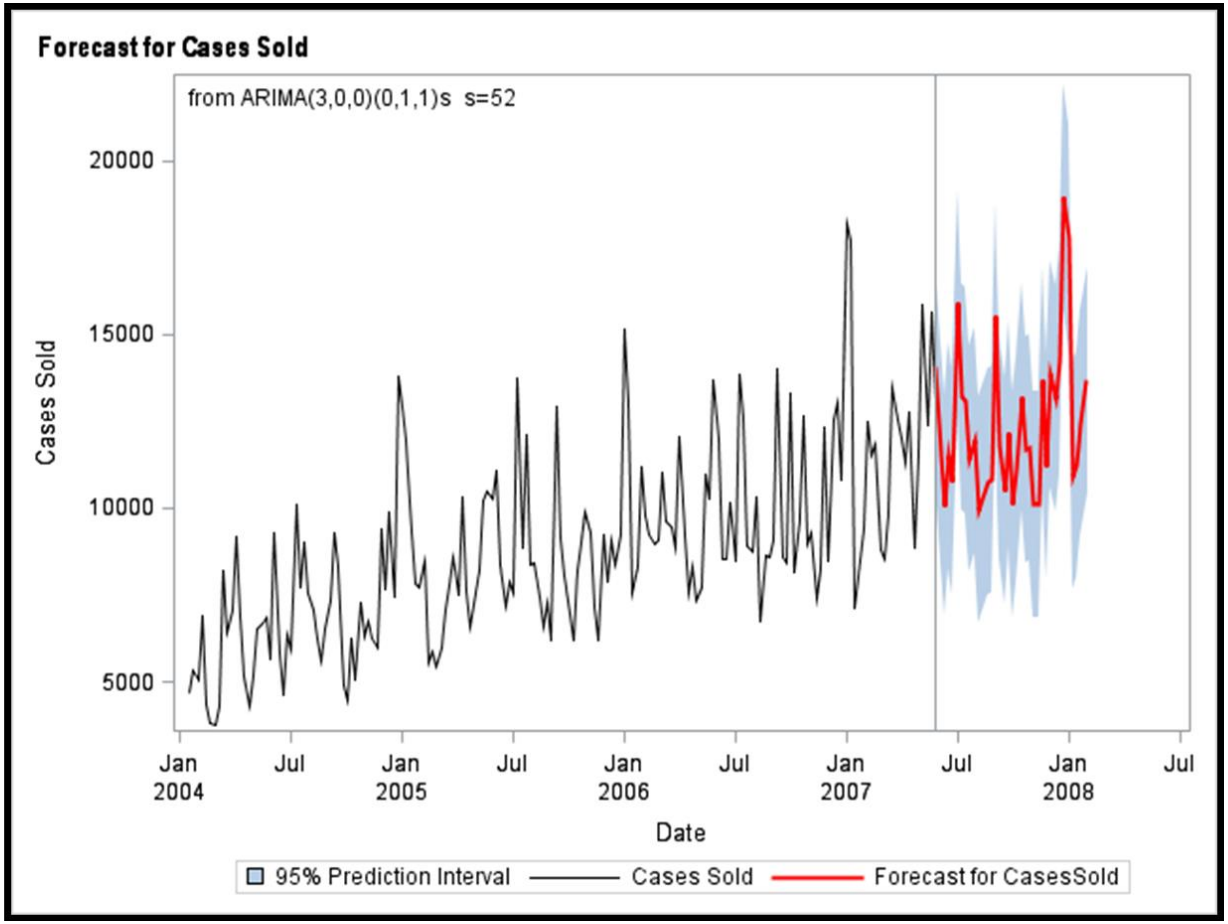
Station was relocated

Station relocated and instrumentation was changed

Thermometer was changed

Source: Dr. Robert Lund

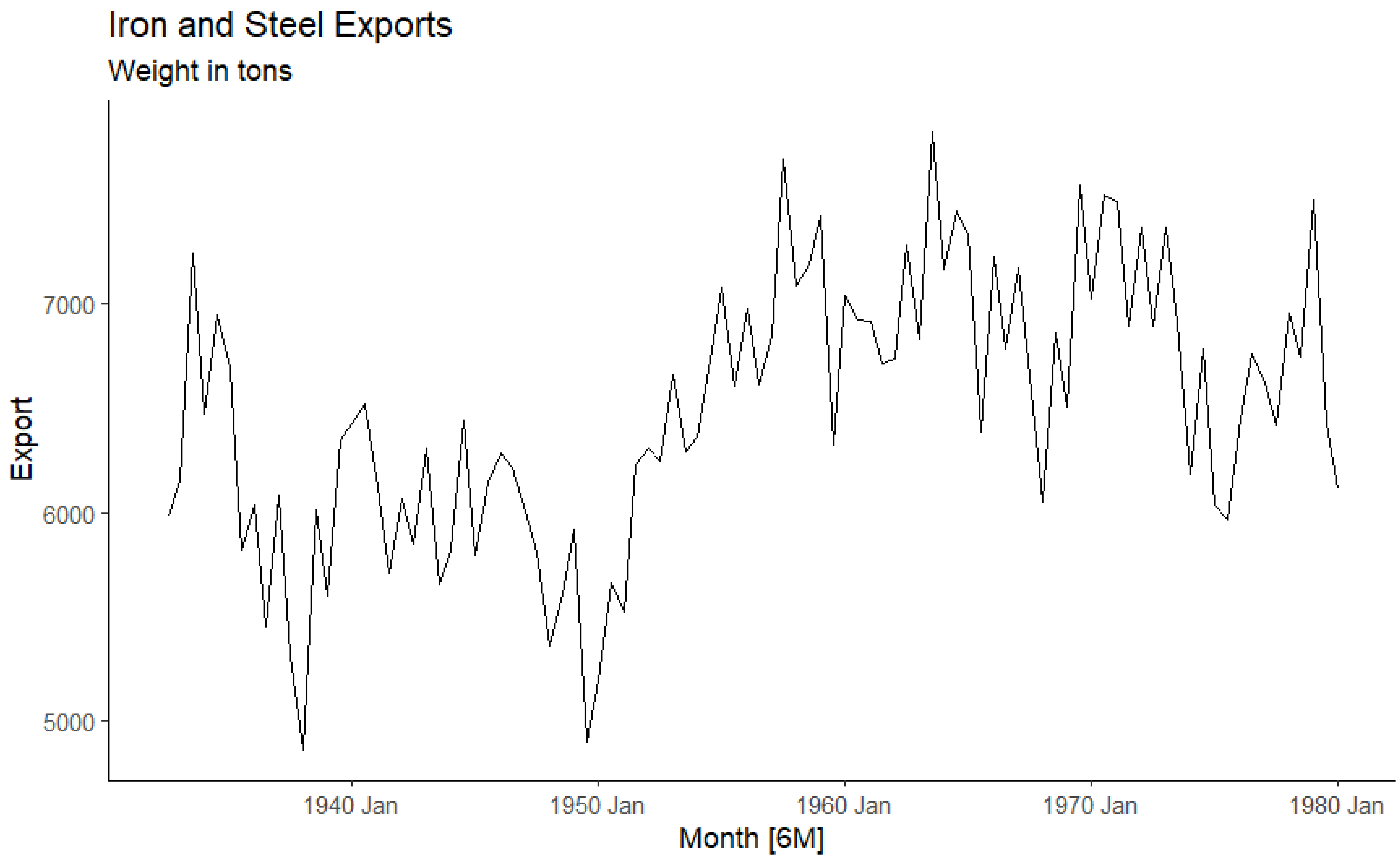
Time Series to Forecast



Code to create time plots

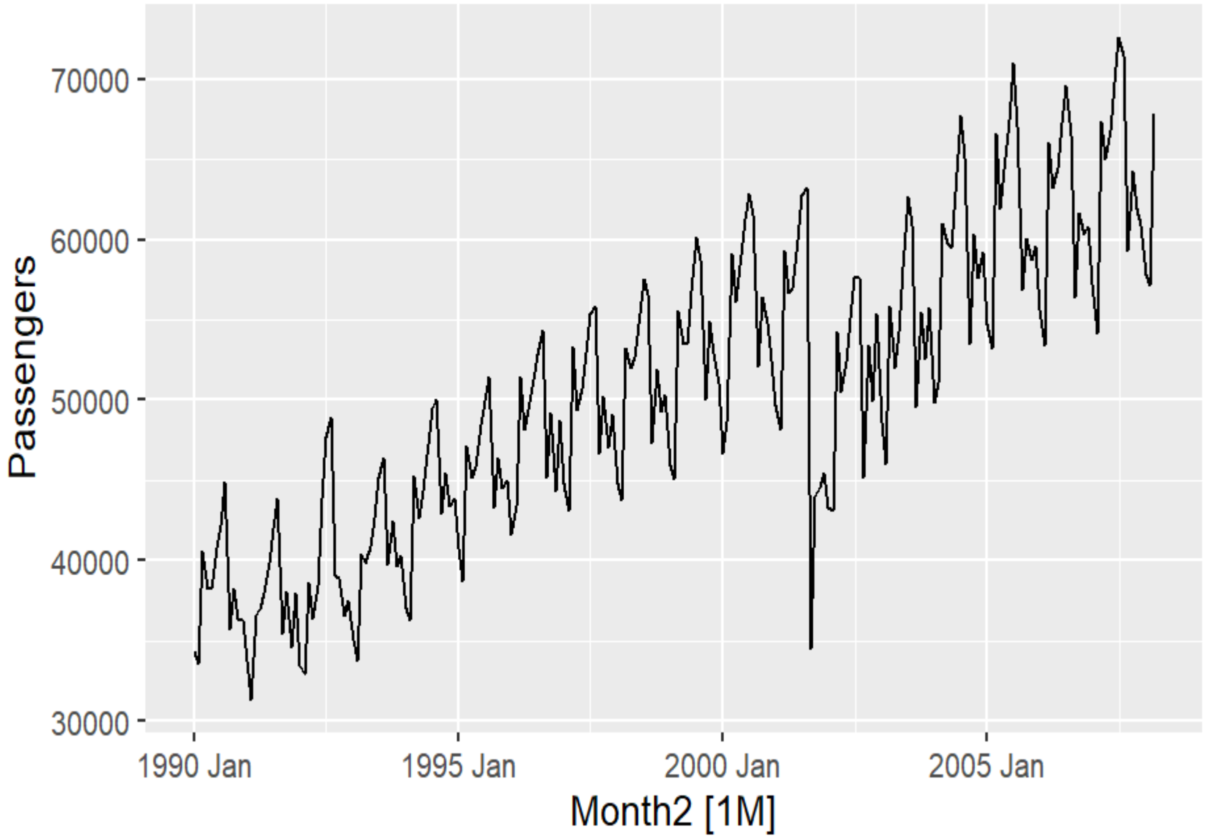
- If data is in tsibble format, creating time plots are very easy
- The autoplot function can be used (and has options similar to what you saw in ggplots)
- For example, to create the Steel Imports graph:

```
Steel <- Steel |> mutate(date = seq(ymd('1932-07-01'),ymd('1980-01-01'),by='6 months'))

steel_ts<-Steel |> mutate(Month=yearmonth(date)) |> as_tsibble(index=Month)

autoplot(steel_ts,steelshp) + labs(title= "Iron and Steel Exports", subtitle = "Weight in tons", y= "Export") + theme_classic()
```



Iron and Steel Exports
Weight in tons
Export
7000
6000
5000
1940 Jan
1950 Jan
1960 Jan
1970 Jan
1980 Jan
Month [6M]

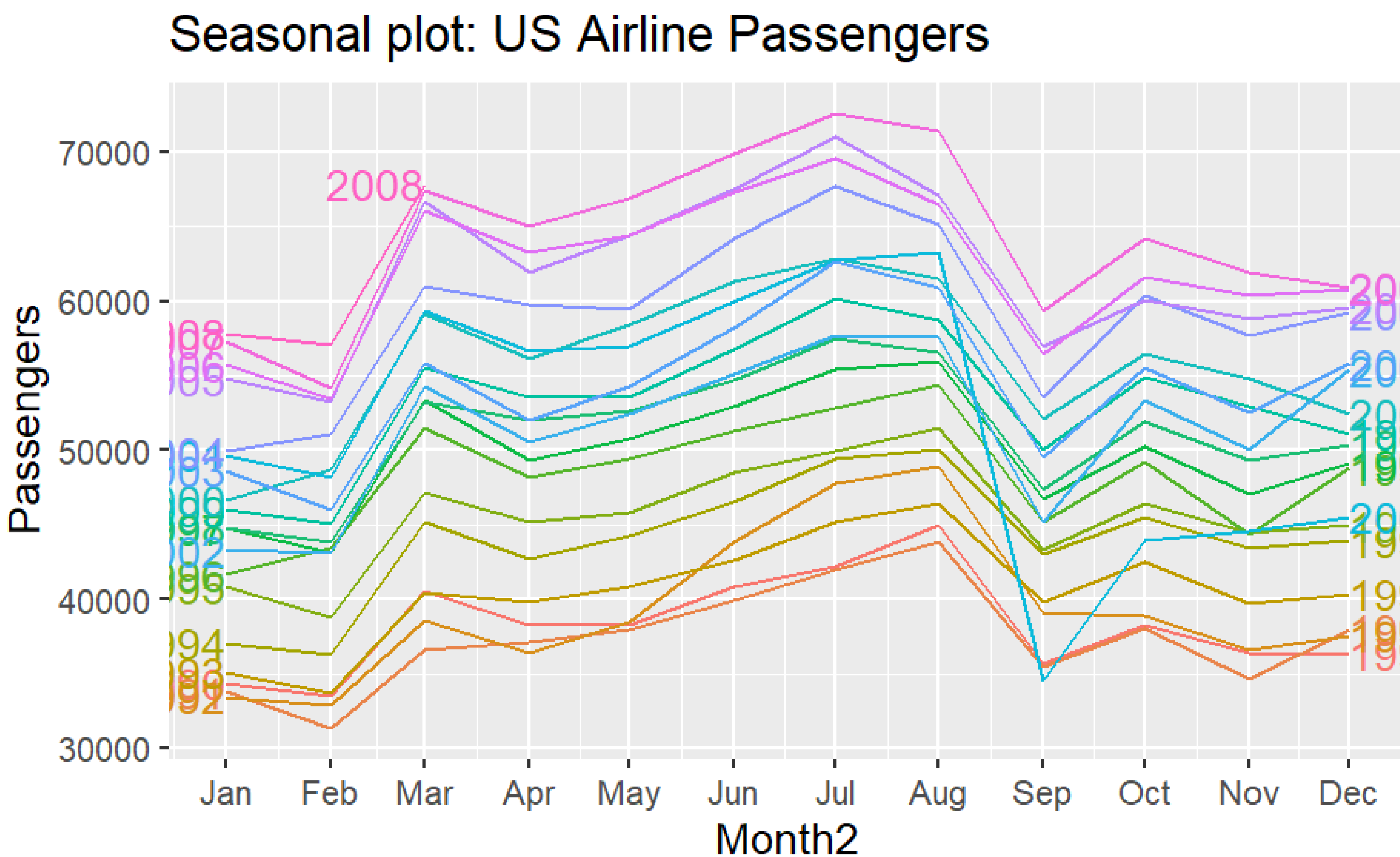
```
USAirlines_ts <- USAirlines |> mutate(date= myd(paste(Month, Year,
"1"))) |> mutate(Month2= yearmonth(date)) |> as_tsibble(index= Month2)

autoplot(USAirlines_ts, Passengers)
```



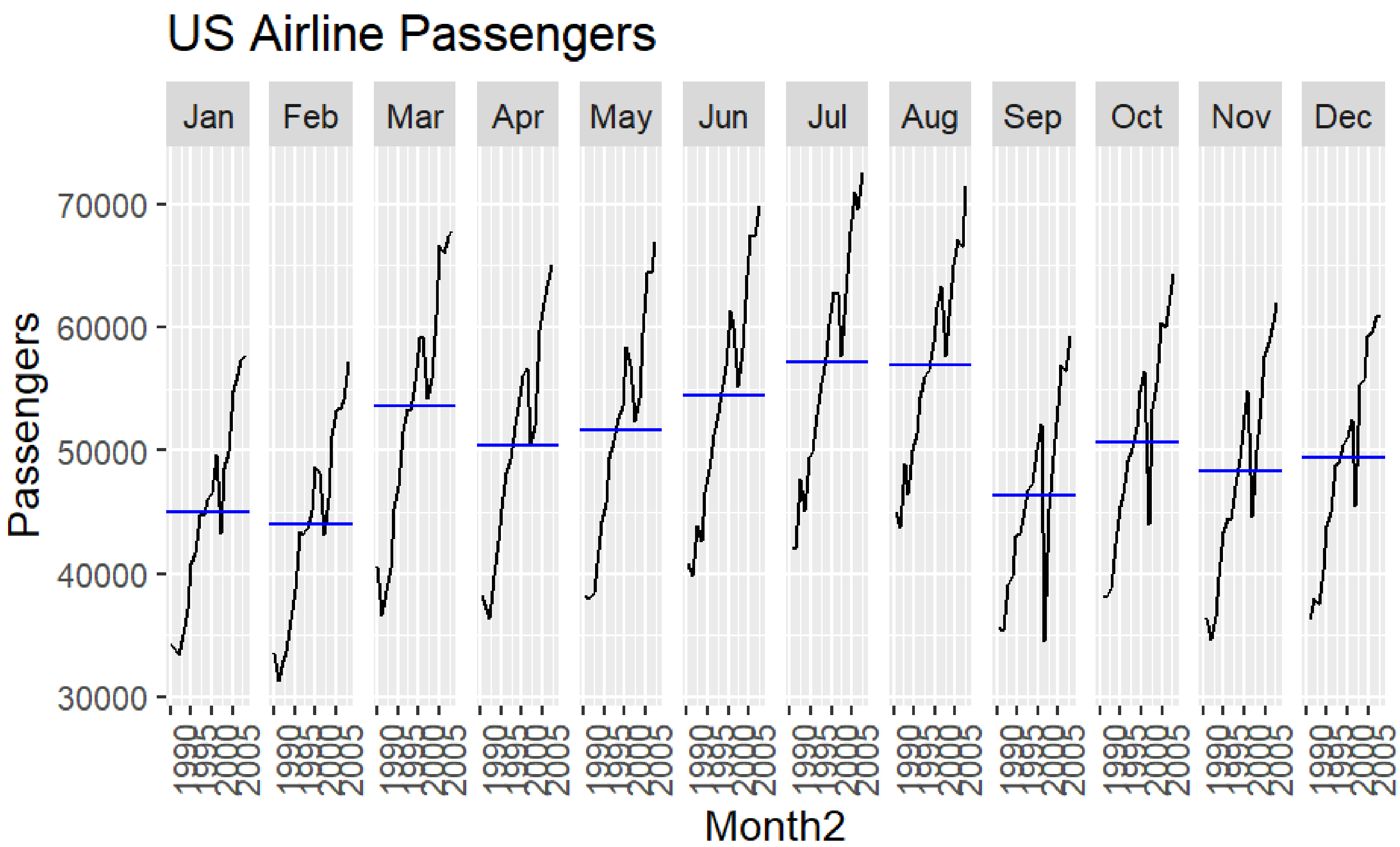
Other graphs to understand signals:

- As seen in time plots, a time series could have
  - Trend
  - Season
  - Cycle (note: cycle is different than season!!)
- We can use some plots to help us better understand these potential signals (if these signals exist, we want to be able to capture them in the model)
- For example, seasonality can be explored by:
  - Seasonal plot
  - Seasonal Subplots

```
USAirlines_ts |>  gg_season(Passengers, labels = "both")
+  labs(y = "Passengers",       title = "Seasonal plot: US
Airline Passengers")
```



```
USAirlines_ts |>  gg_subseries(Passengers) +  labs(    y = 
"Passengers",    title = "US Airline Passengers"  )
```

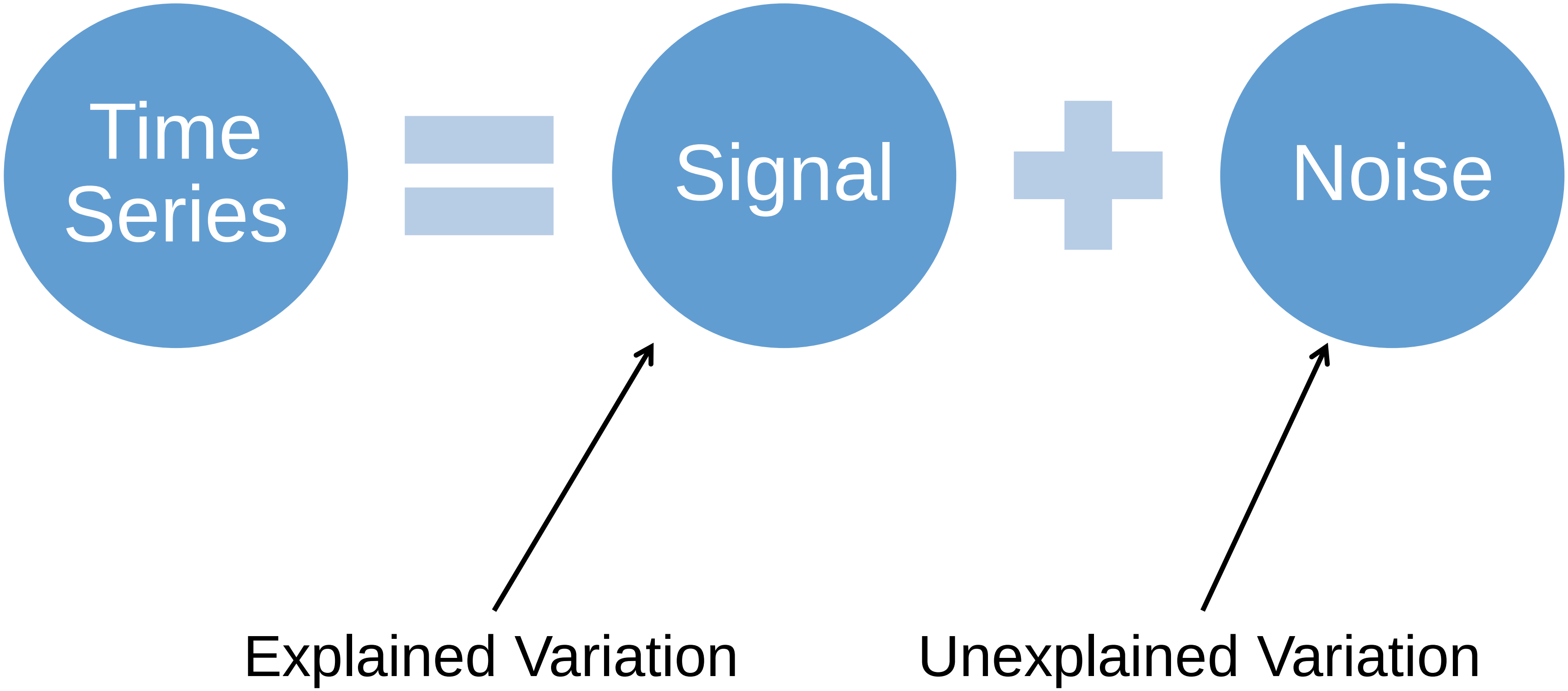


SIGNAL AND NOISE

Statistical Forecasting

Time Series = Signal + Noise

Statistical Forecasting



Statistical Forecasting

Time Series = Signal + Noise

Explained variation:
Trend/Cycle
Seasonality

Error

Statistical Forecasting

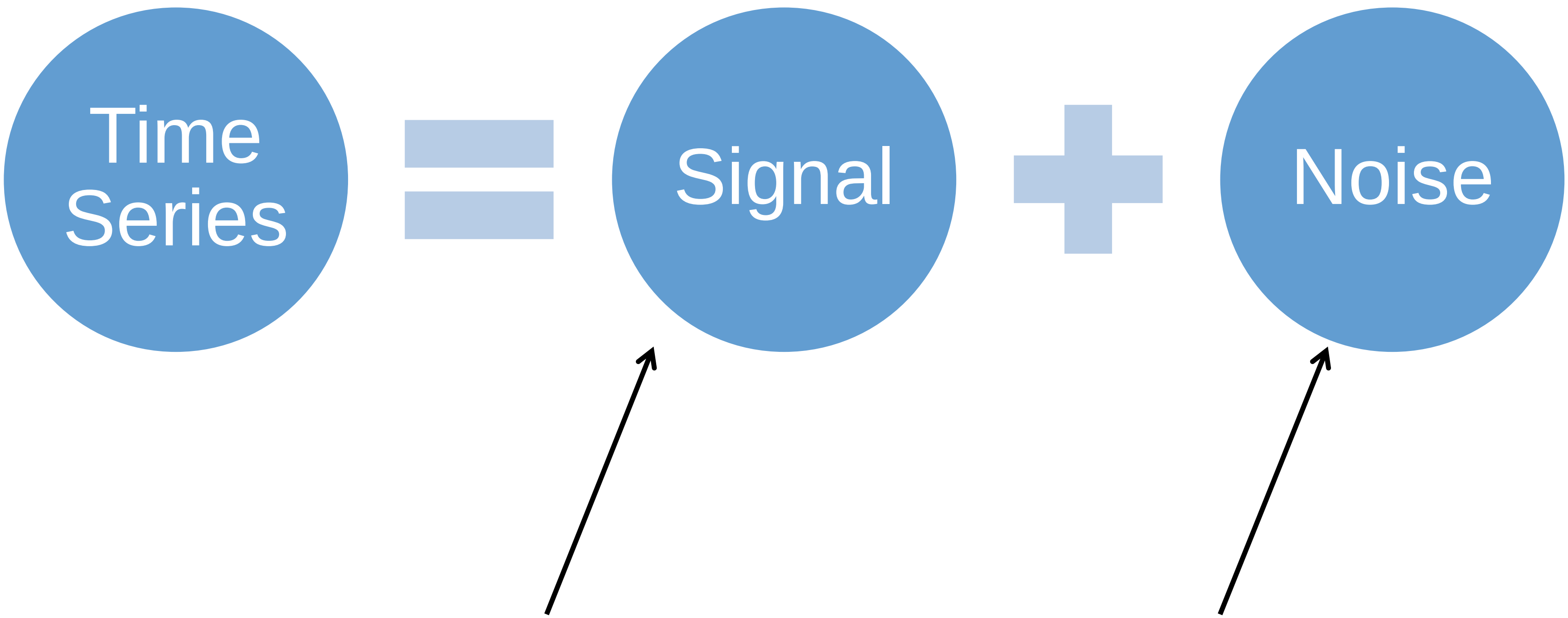
Time Series = Signal + Noise

Explained variation:
Trend/Cycle
Seasonality
Dependence Structure

Error

This will be discussed in ARIMA

Statistical Forecasting



Forecasts extrapolate signal portion of model.

Confidence intervals account for uncertainty.

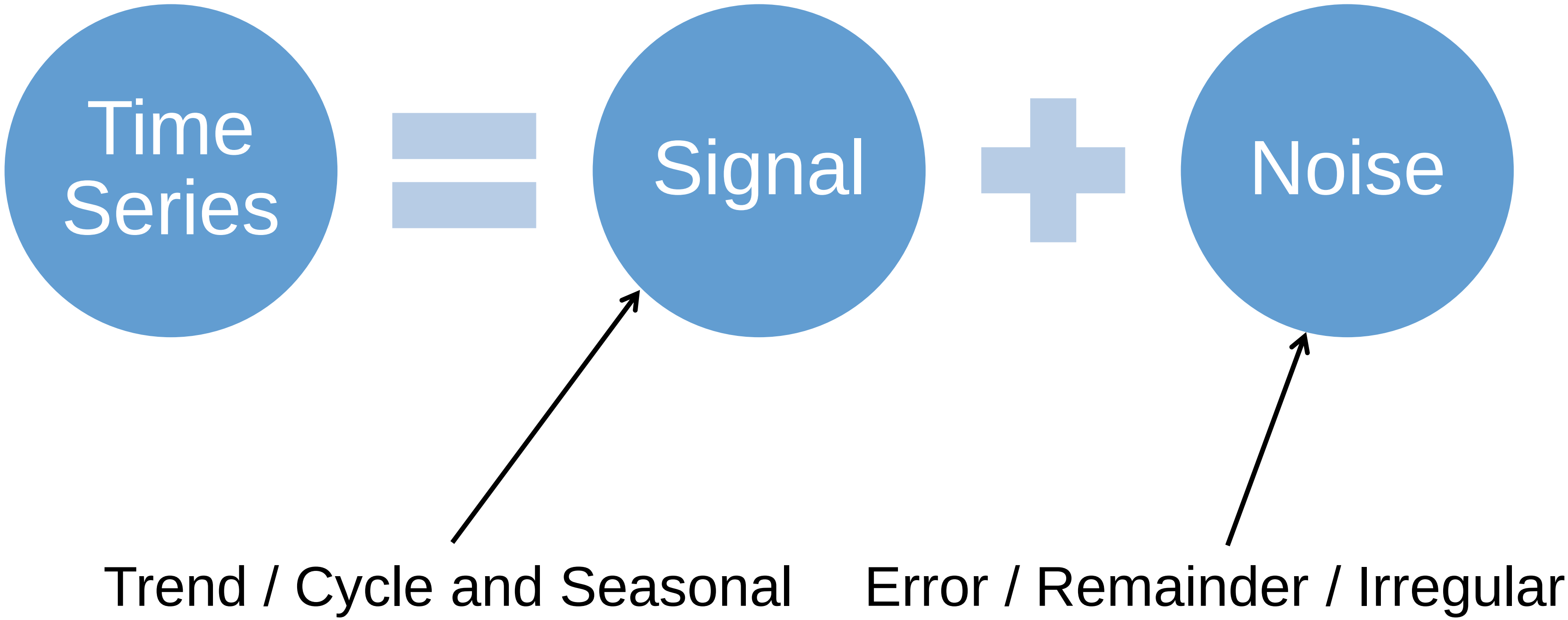
DECOMPOSITION

Time Series Decomposition

- If a time series only has trend/cycle patterns, there is no need to decompose
- If a time series has both trend/cycle patterns AND seasonal variation, we can decompose series into these individual parts:
  - Trend/Cycle patterns
  - Seasonal variation
  - Error

Time Series Decomposition

- The signal part of the time series can typically be broken down into two components:



Time Series Decomposition

- The whole time series can now be thought of like the equations below.
  - Additive:

$$Y_t = T_t + S_t + R_t$$

  - Multiplicative:

$$Y_t = T_t \times S_t \times R_t$$

Time Series Decomposition

- The whole time series can now be thought of like the equations below.
  - Additive:

$$Y_t = T_t + S_t + R_t$$

Trend / Cycle

  - Multiplicative:

$$Y_t = T_t \times S_t \times R$$

Time Series Decomposition

- The whole time series can now be thought of like the equations below.
  - Additive:

$$Y_t = T_t + S_t + R_t$$

Seasonal

  - Multiplicative:

$$Y_t = T_t \times S_t \times R_t$$

Time Series Decomposition

- The whole time series can now be thought of like the equations below.
  - Additive:

$$Y_t = T_t + S_t + R_t$$

Error

  - Multiplicative:

$$Y_t = T_t \times S_t \times R_t$$

Time Series Decomposition

- The whole time series can now be thought of like the equations below.
  - Additive:

$$Y_t = T_t + S_t + R_t$$

  - Multiplicative:

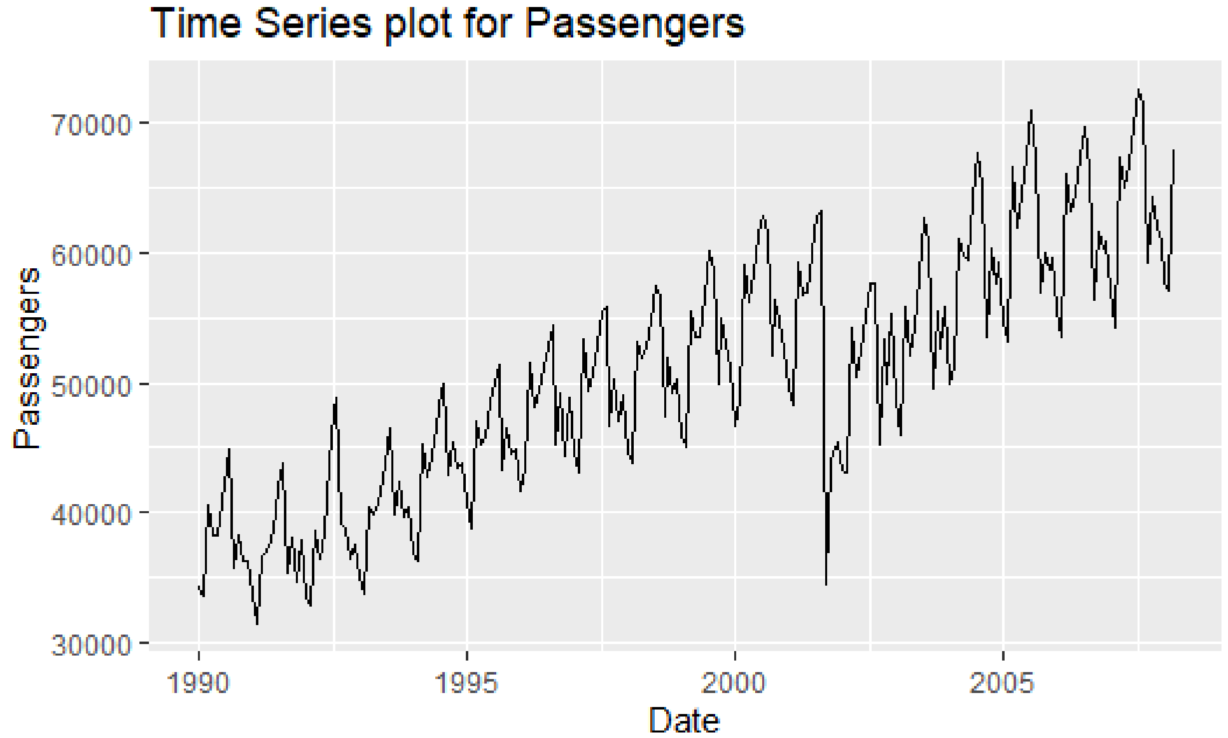
$$Y_t = T_t \times S_t \times R_t$$

OR

$$\log(Y_t) = \log(T_t) + \log(S_t) + \log(R_t)$$

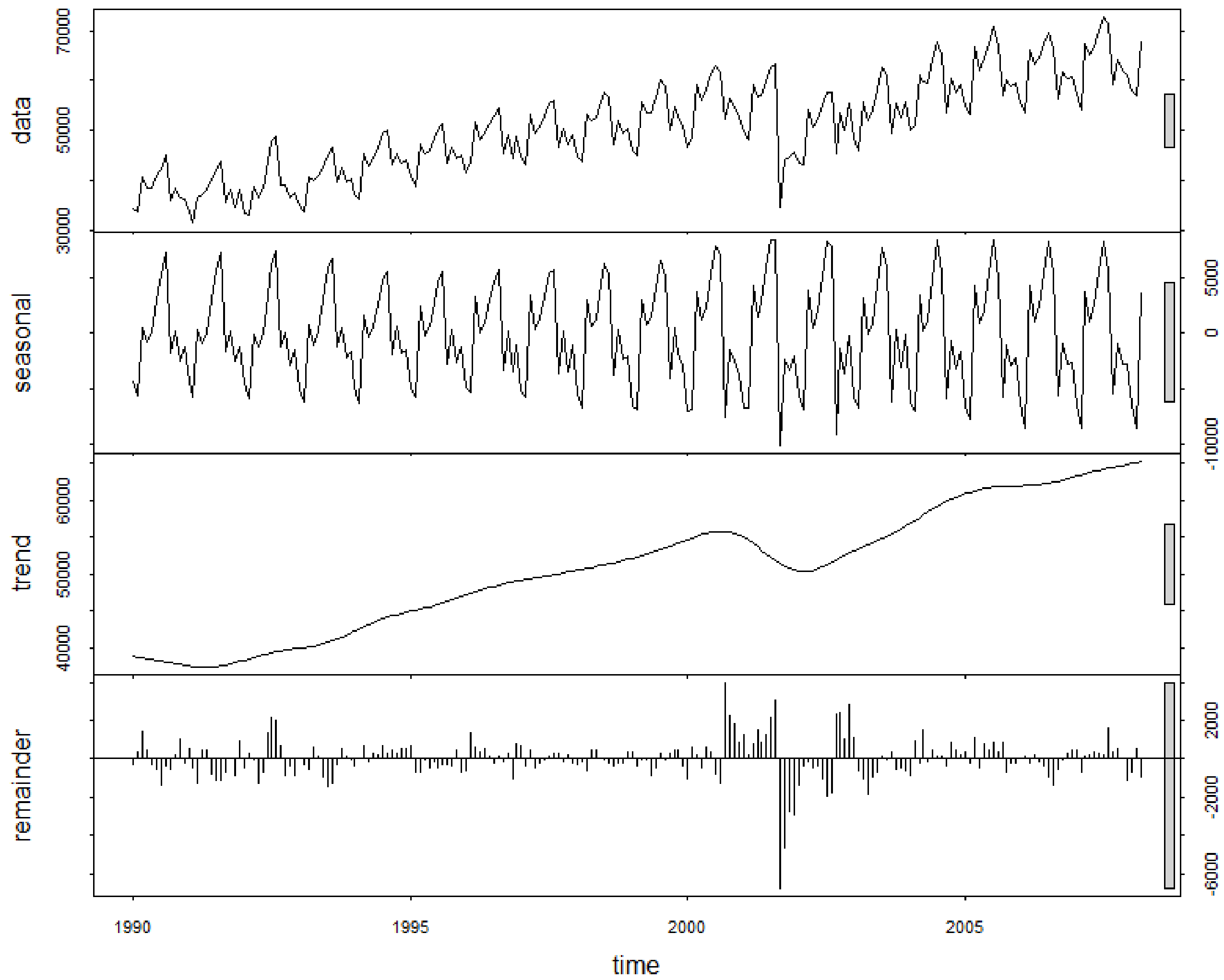
Airline data set

- Data contains number of US airline passengers from January 1990 – March 2008
- Data is monthly (length of season is 12…repeats pattern every 12 observations)



Time Series Decomposition

$$Y_t = S_t + T_t + R_t$$



Time Series Decomposition

Y_t

$=$

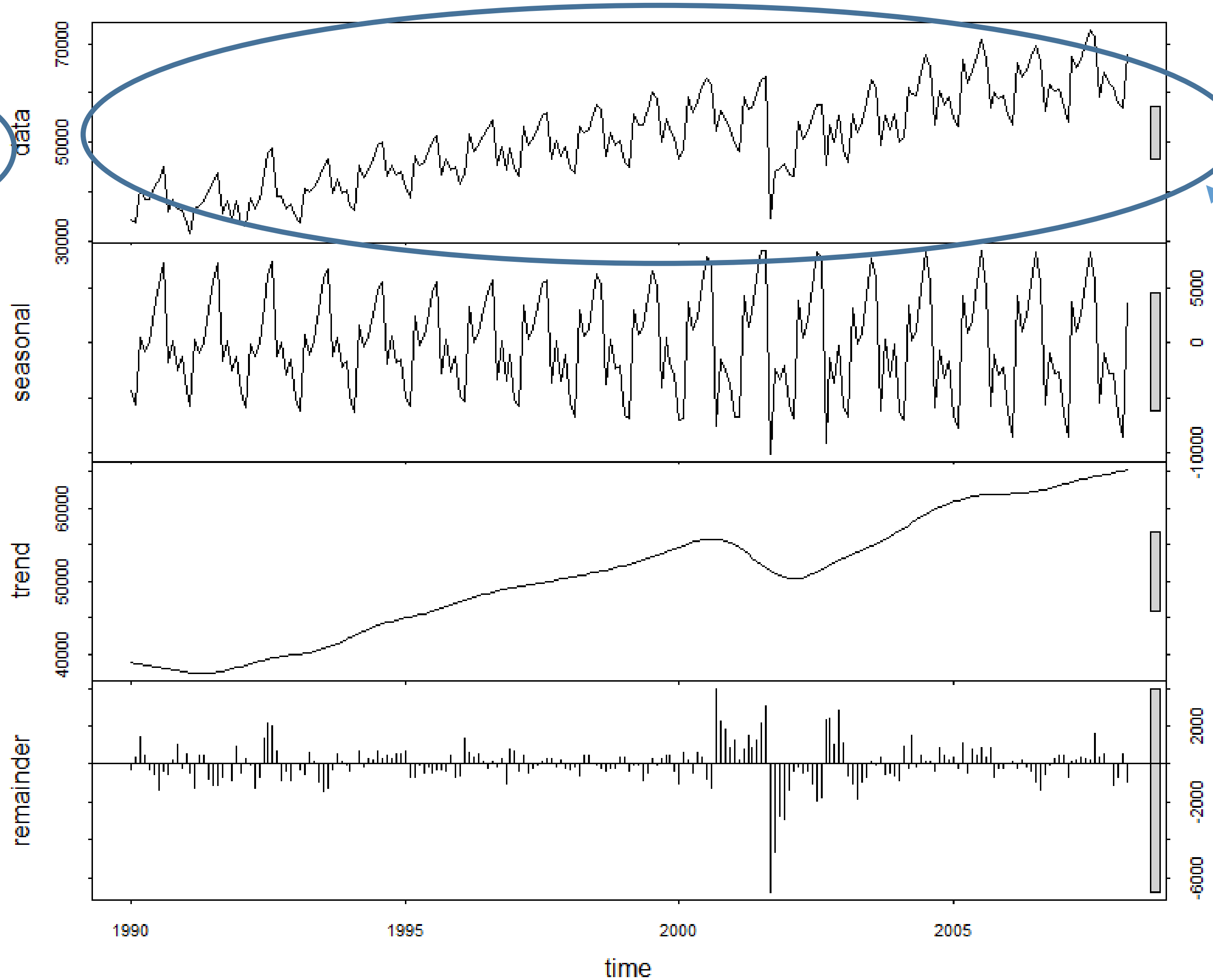
S_t

$+$

T_t

$+$

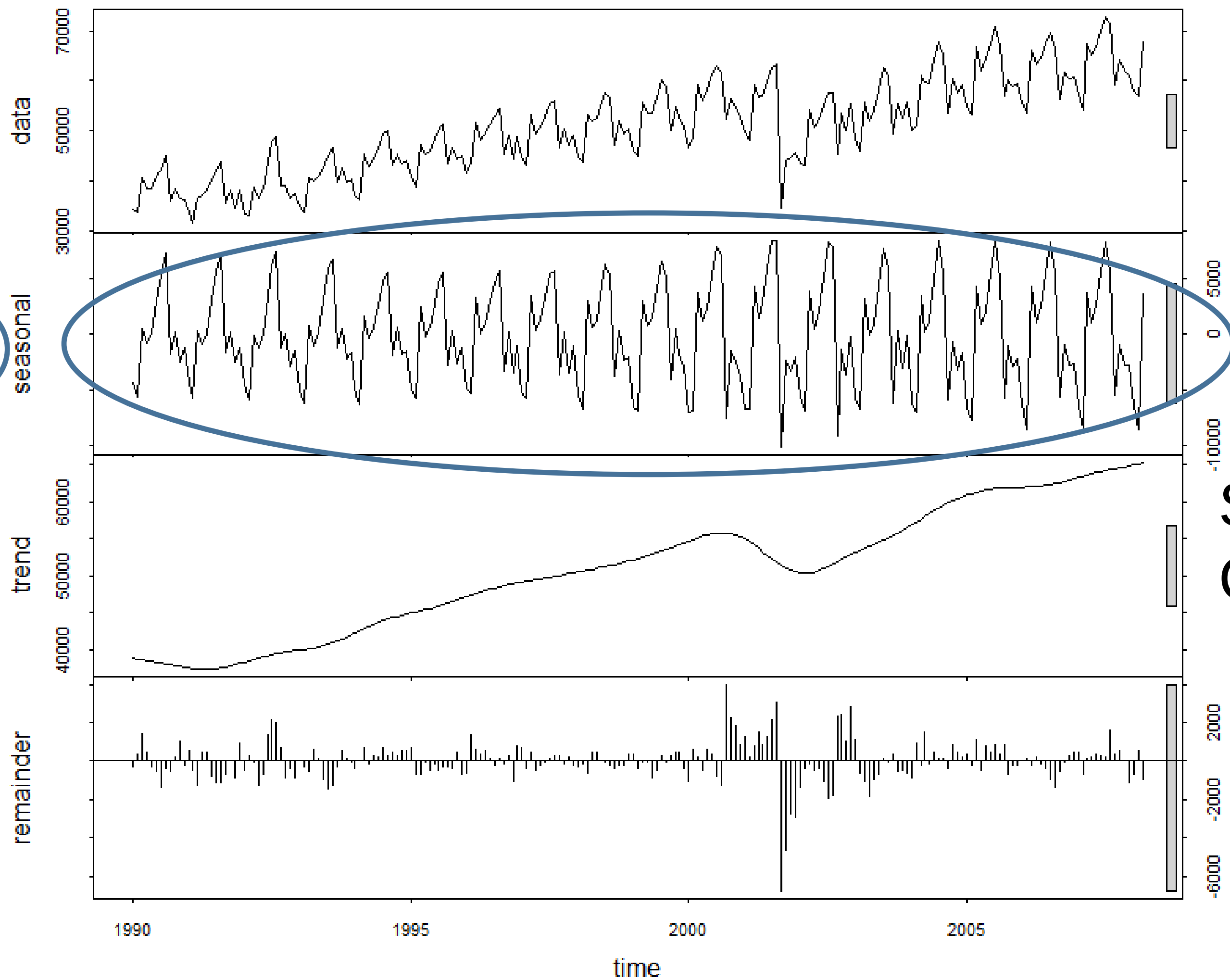
R_t



Original data

Time Series Decomposition

$$Y_t = S_t + T_t + R_t$$



Seasonal Component

Time Series Decomposition

Y_t

$=$

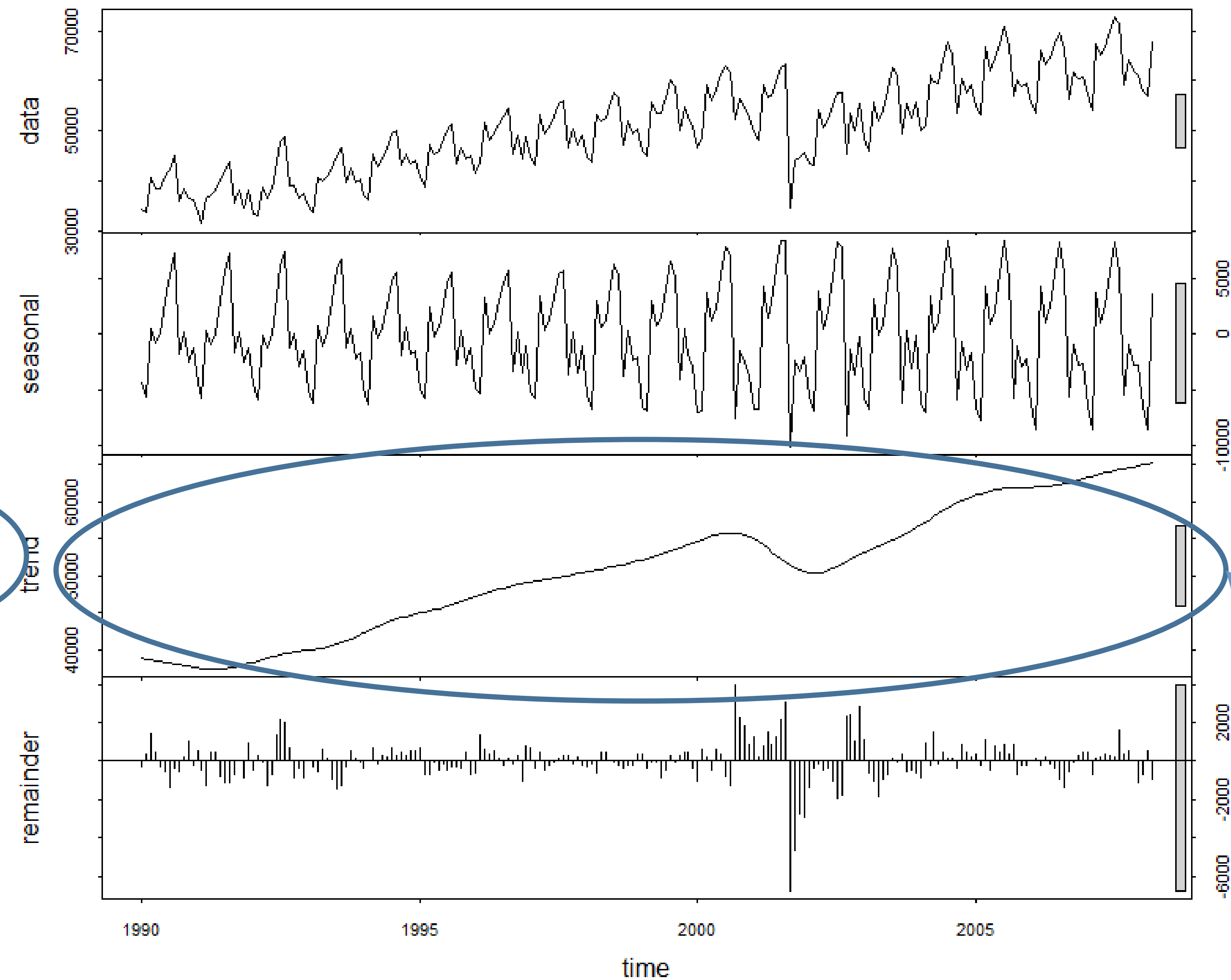
S_t

$+$

T_t

$+$

R_t



Trend Component

Time Series Decomposition

Y_t

$=$

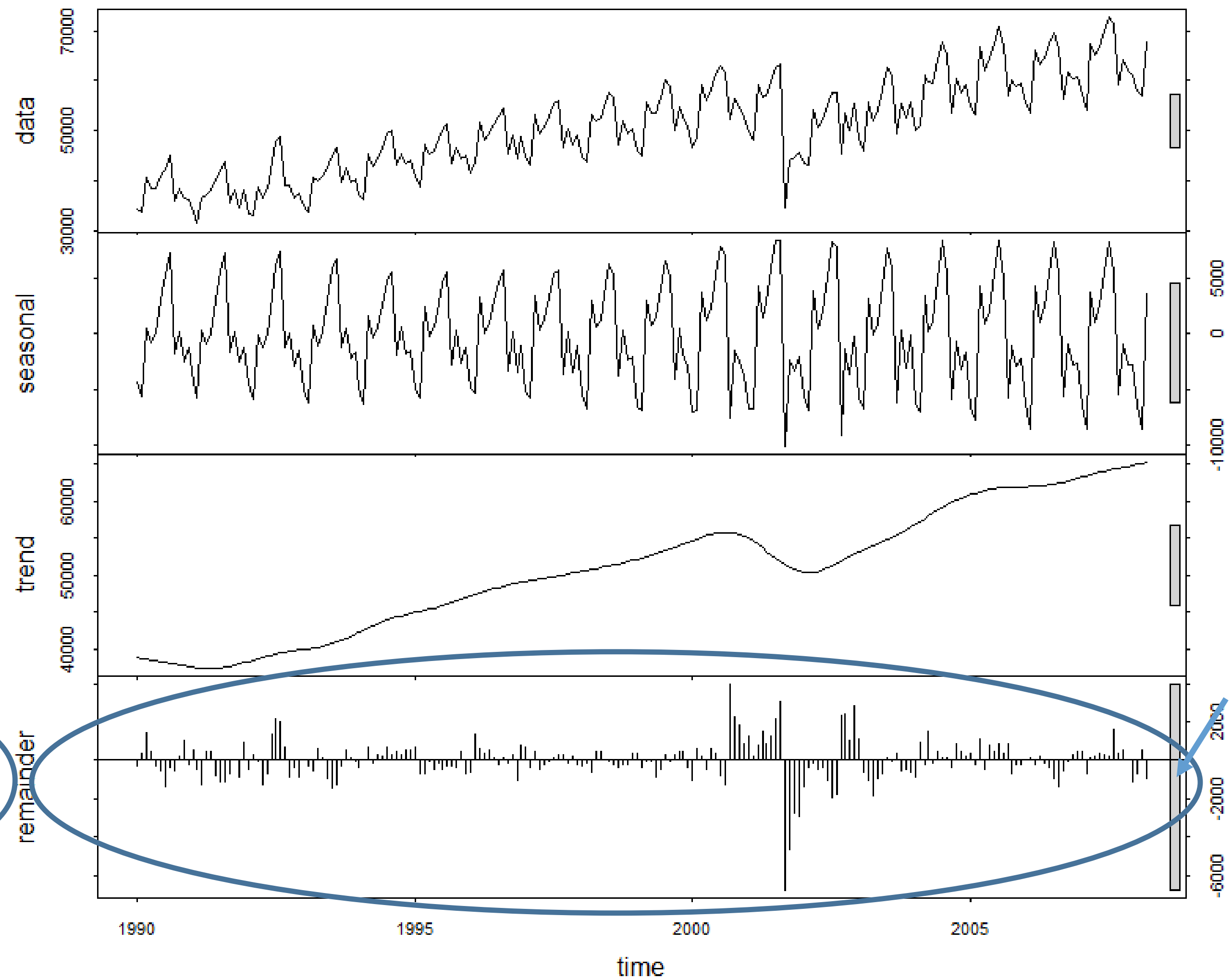
S_t

$+$

T_t

$+$

R_t



Remainder (What's left over)

Components of decomposition

Using the STL method (to be discussed shortly), the components of the decomposition are stored in a dable

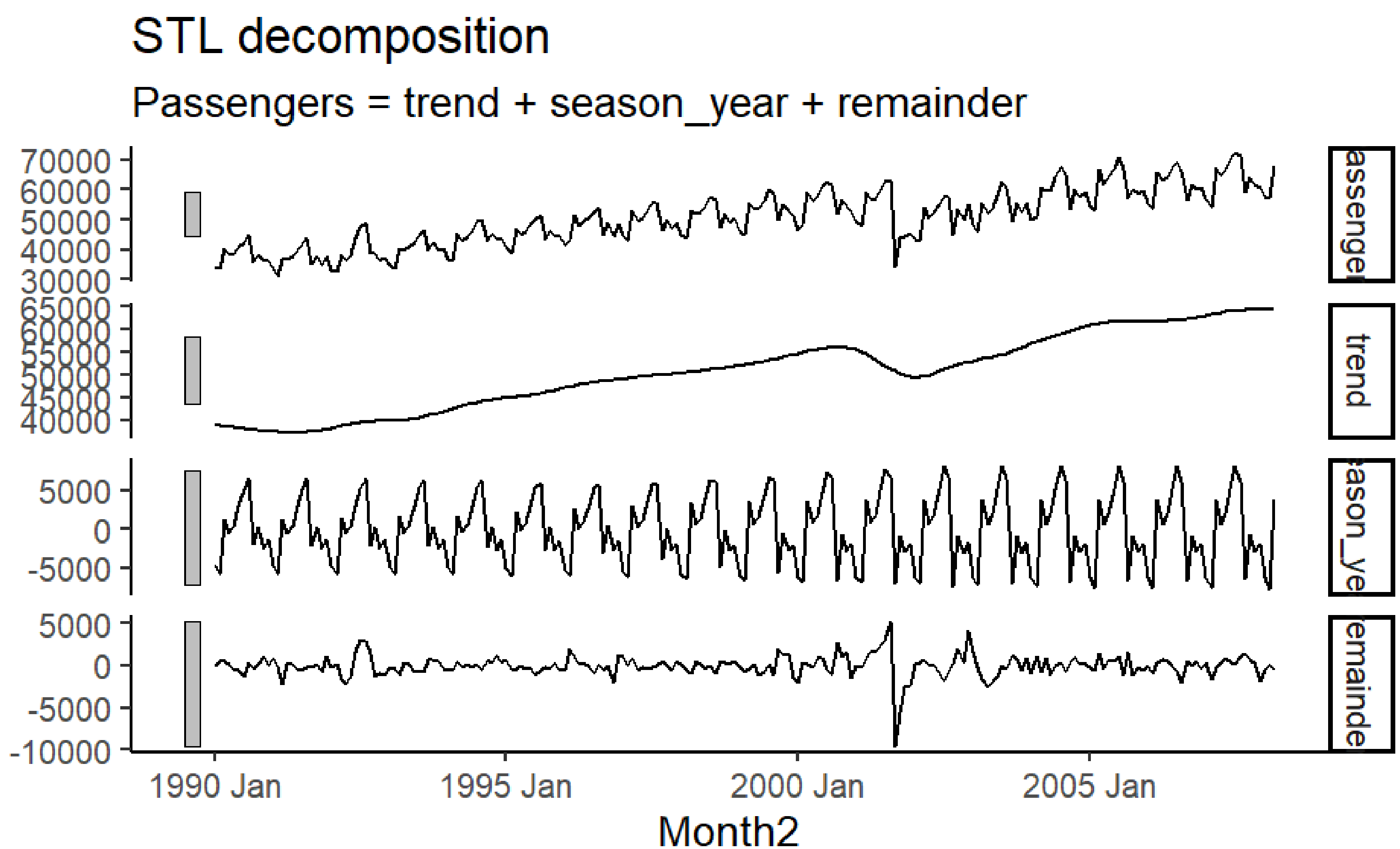
```
dcmp <- USAirlines_ts |>  model(stl = STL(Passengers))
components(dcmp)
```

34348=39033+(-4676)+(-8.97)

```
# A dable: 219 x 7 [1M]
# Key:     .model [1]
# :        Passengers = trend + season_year + remainder
  .model    Month2 Passengers  trend season_year remainder
  <chr>      <mth>      <dbl>  <dbl>       <dbl>     <dbl>
1 stl     1990 Jan      34348 39033.      -4676.     -8.97
2 stl     1990 Feb      33536 38897.      -5842.    481.
3 stl     1990 Mar      40578 38762        1238     578
```

Plot decomposition

```
components(dcmp) |> autoplot() + theme_classic()
```



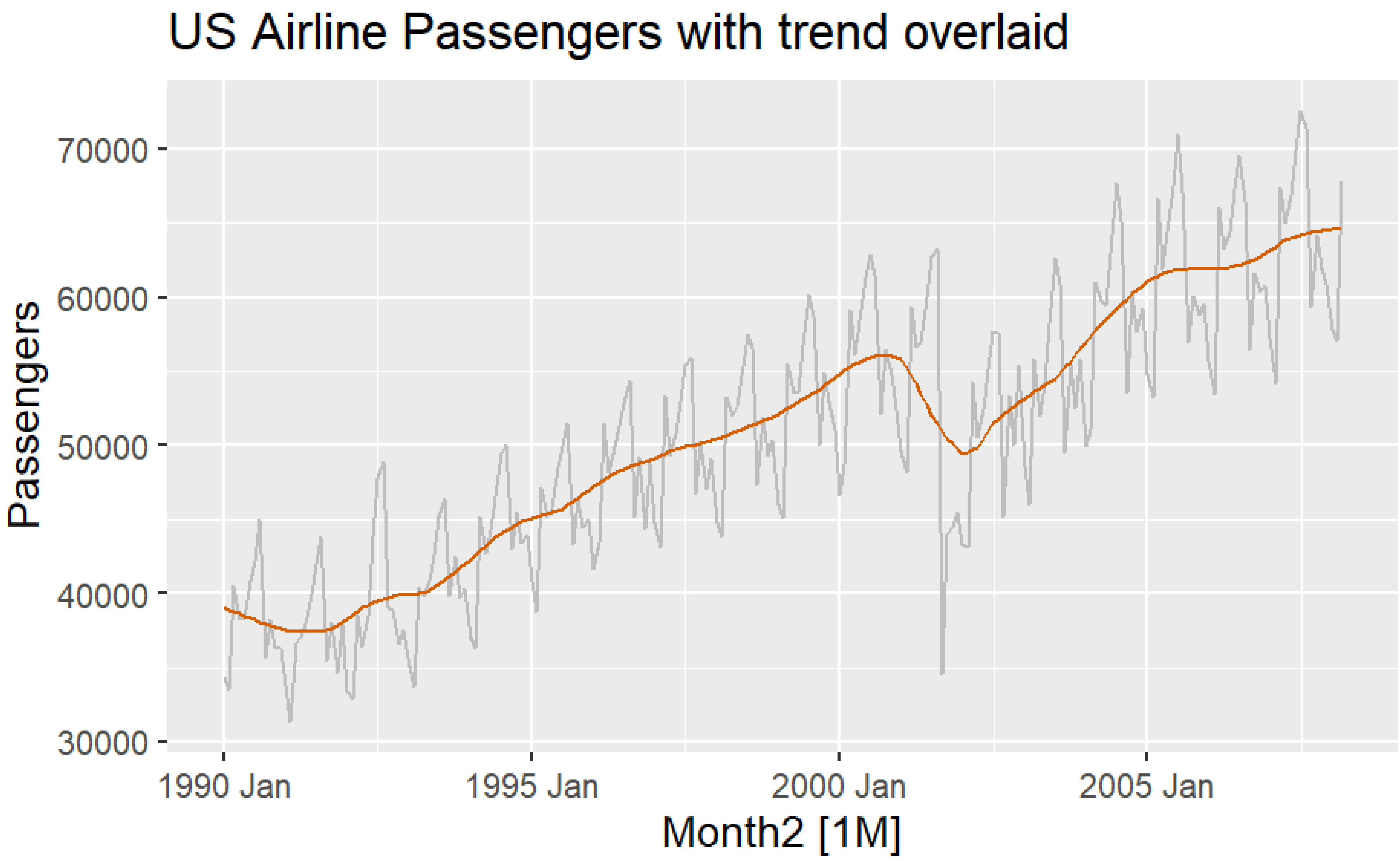
Information from decomposition

- We can pull off information from decomposition to better understand our time series
- For example, we can see the overall trend of the series (we can even overlay the trend component with the original series)

Time Series Decomposition-trend overlay

```
components(dcmp) |>  as_tsibble() |>
autoplot(Passengers, colour="gray") +
geom_line(aes(y=trend), colour = "#D55E00") +  labs( y =
"Passengers",    title = "US Airline Passengers with trend
overlaid"  )
```

Overlay the trend component

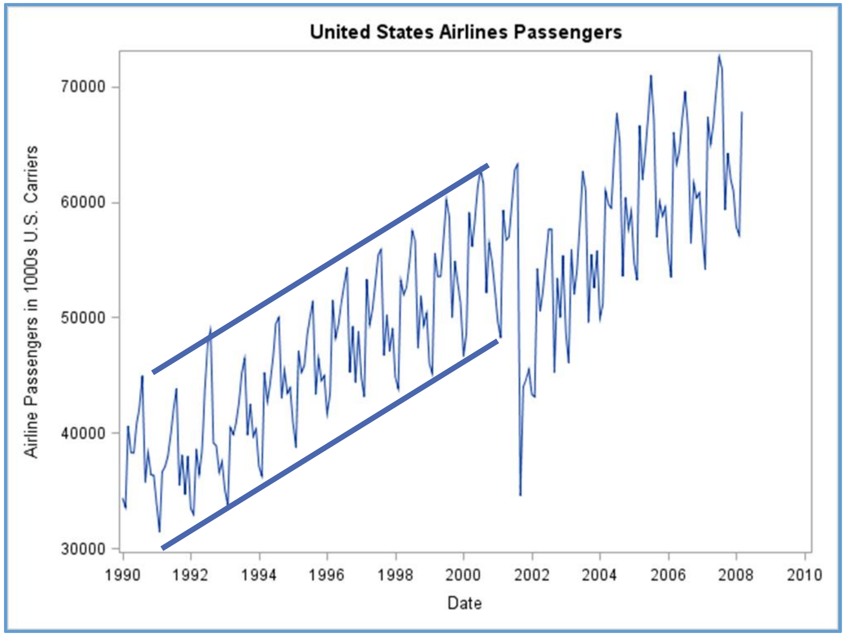

US Airline Passengers with trend overlaid
Passengers
70000
60000
50000
40000
30000
1990 Jan
1995 Jan
2000 Jan
2005 Jan
Month2 [1M]

Information from decomposition

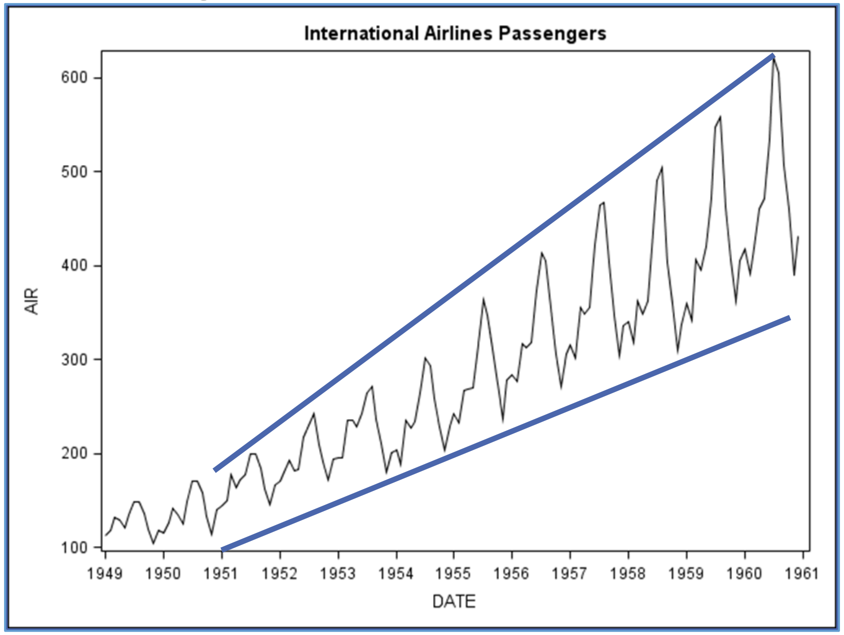
- We can get seasonally adjusted data

Additive vs. Multiplicative

- Additive – magnitude of variation around trend / cycle remains constant.



- Multiplicative – magnitude of the variation around trend / cycle proportionally changes.



Seasonally Adjusted Data

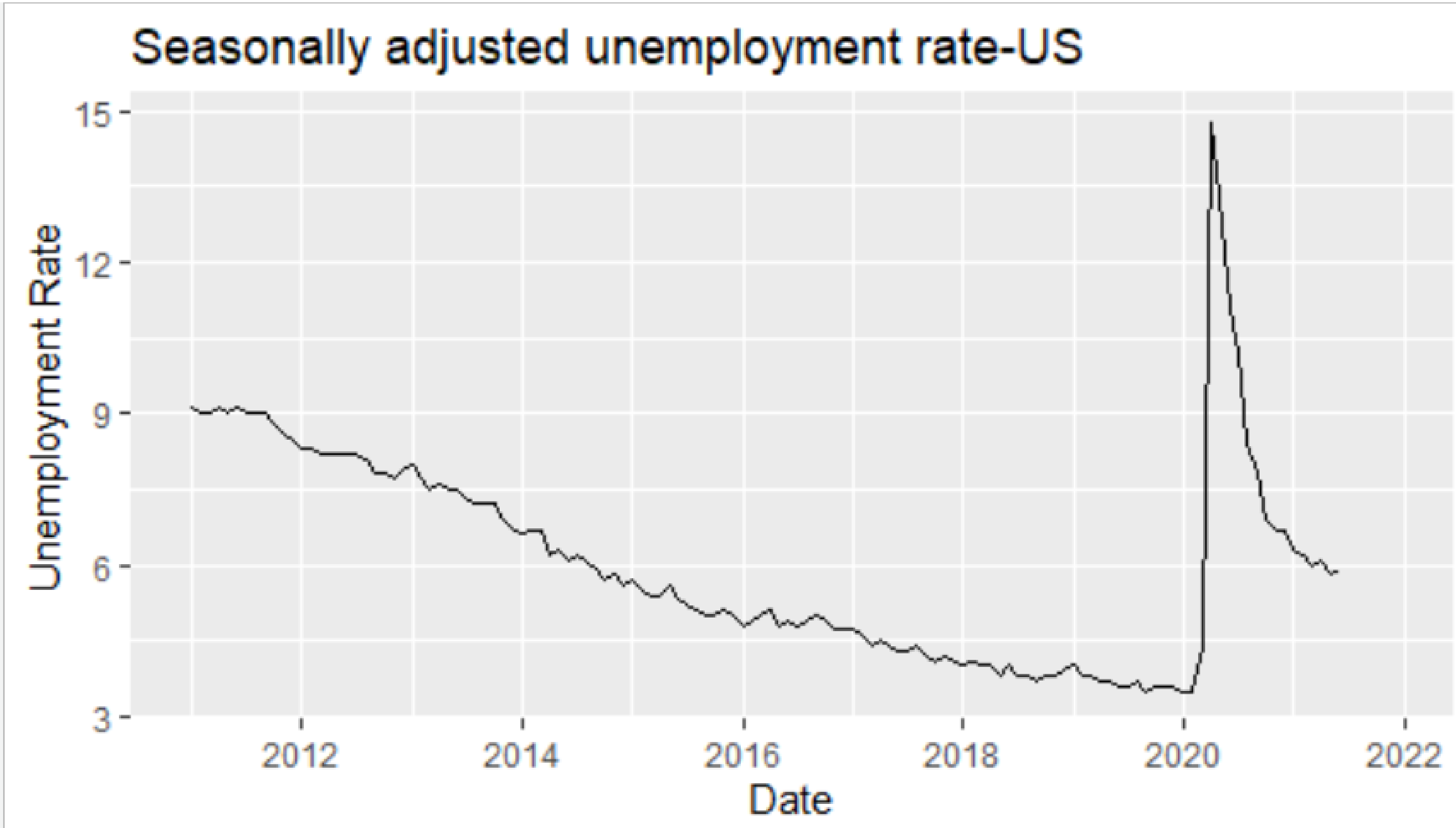
One advantage of time series decomposition is that we are able to create seasonally adjusted data (i.e. remove the "effect of Seasonality")

This allows analysts to understand the trend of the series

$$Y_t = T_t + S_t + R_t$$

$Y_t - S_t$ $(T_t + R_t)$

Seasonally adjusted (additive)

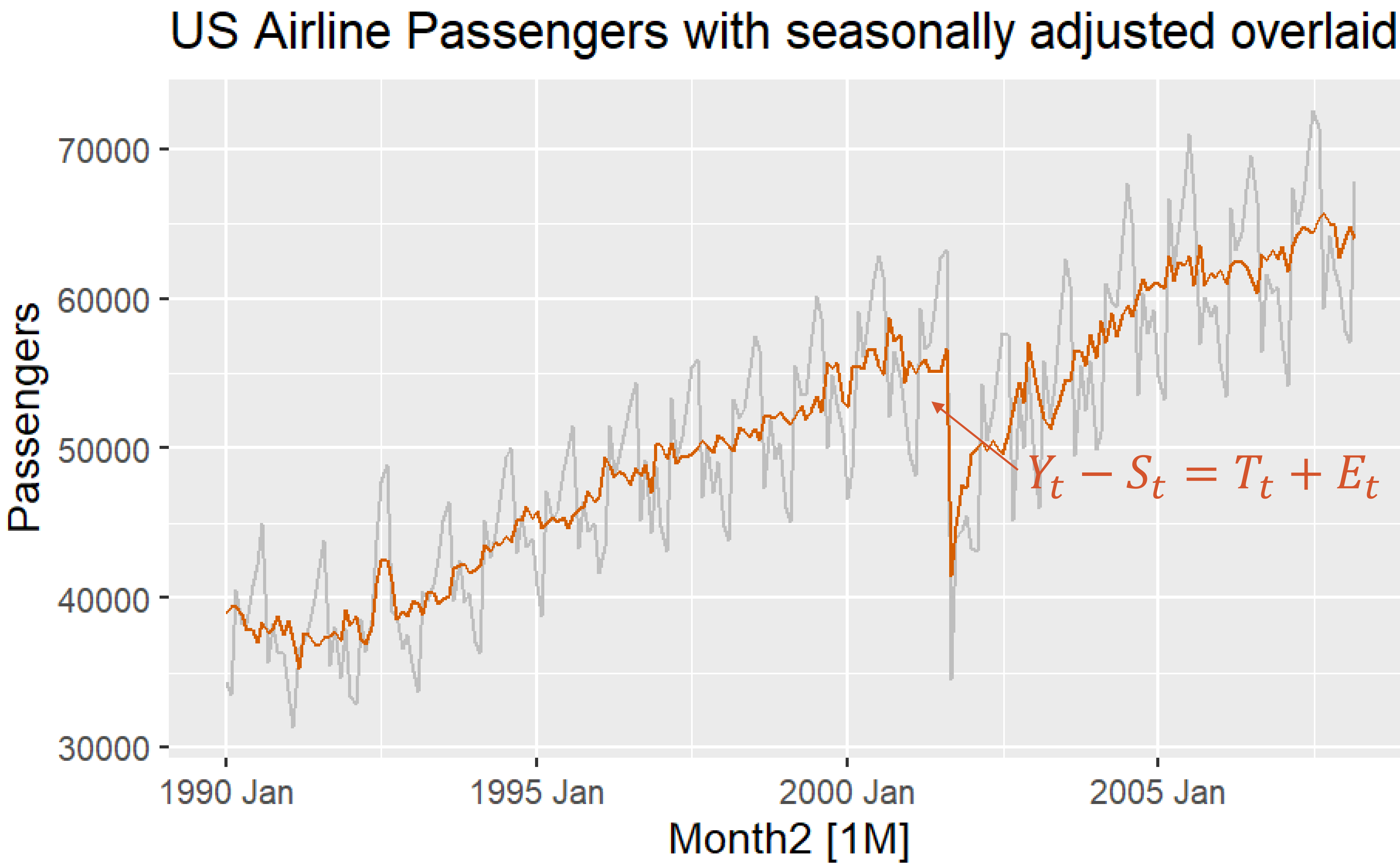

Seasonally adjusted unemployment rate-US
Unemployment Rate
15
12
9
6
3
2012
2014
2016
2018
2020
2022
Date

Time Series Decomposition

```
components(dcmp) |>  as_tsibble() |>  autoplot(Passengers,
colour="gray") +  geom_line(aes(y=season_adjust), colour =
"#D55E00") +  labs( y = "Passengers",   title = "US Airline
Passengers with seasonally adjusted overlaid"  )
```

Overlay seasonally adjusted

Time Series Decomposition



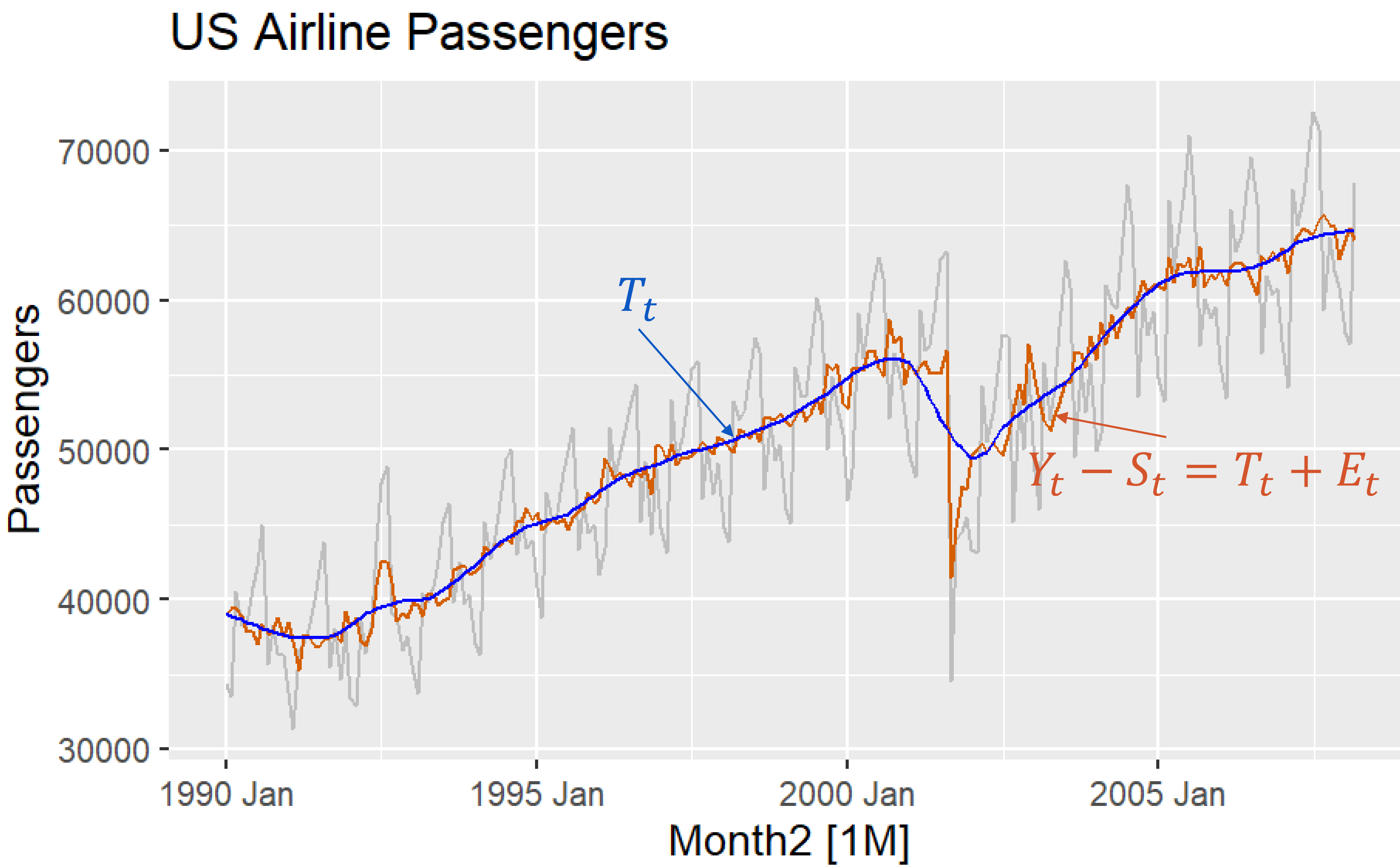
Time Series Decomposition

```
components(dcmp) |>  as_tsibble() |>  autoplot(Passengers,
colour="gray") +  geom_line(aes(y=season_adjust), colour =
"#D55E00") +  geom_line(aes(y=trend),colour="blue")+  labs(
y = "Passengers",    title = "US Airline Passengers "  )
```

Overlay the trend component

Overlay seasonally adjusted

Time Series Decomposition



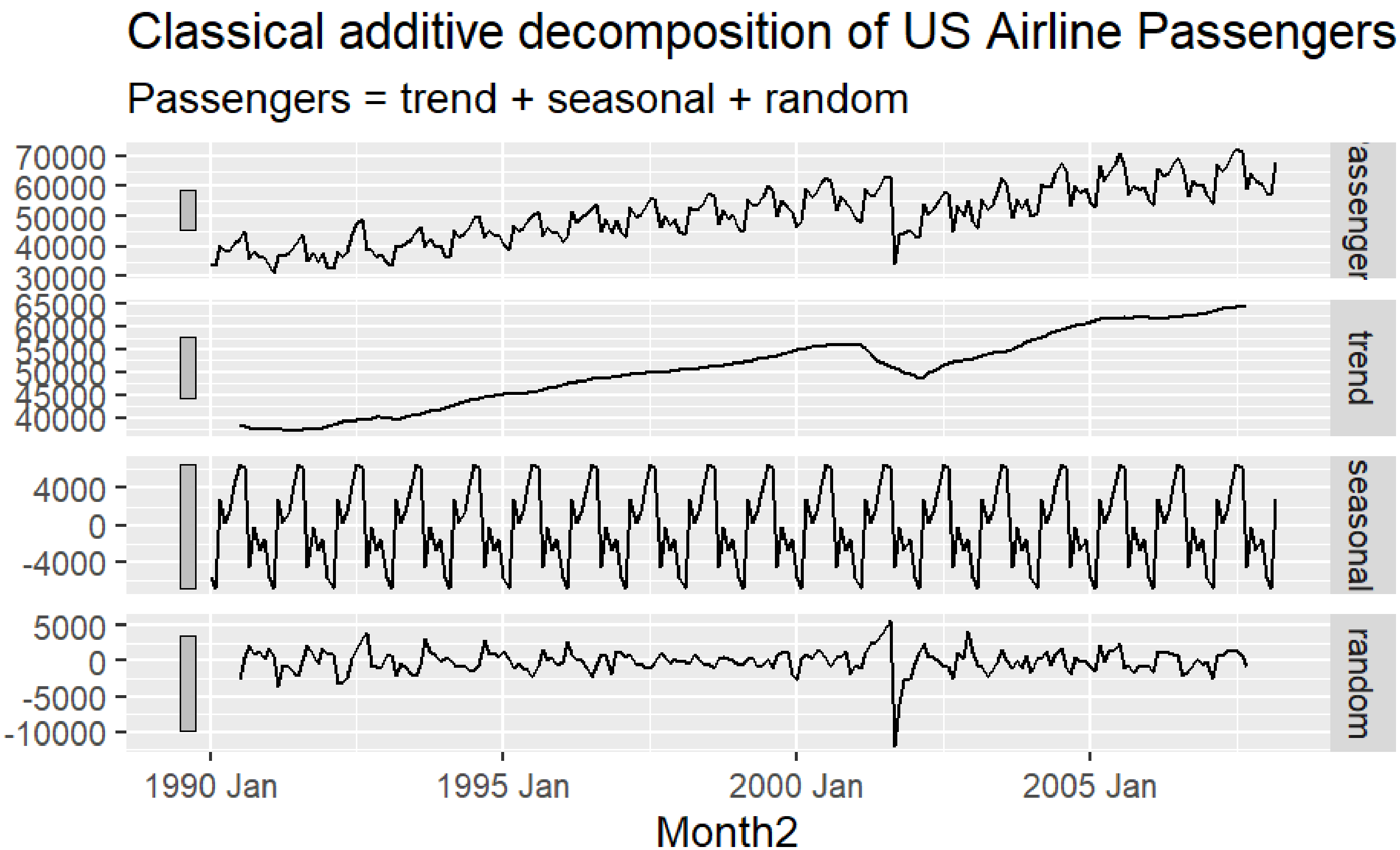
Decomposition Techniques

- There are different ways to decompose time series data.
- Here are 3 common techniques:
  1. Classical Decomposition

Decomposition Techniques

- There are different ways to decompose time series data.
- Here are 3 common techniques:
  1. Classical Decomposition
     a. Trend – Uses Moving / Rolling Average Smoothing
     b. Seasonal – Average De-trended Values Across Seasons (assumed to CONSTANT throughout series)

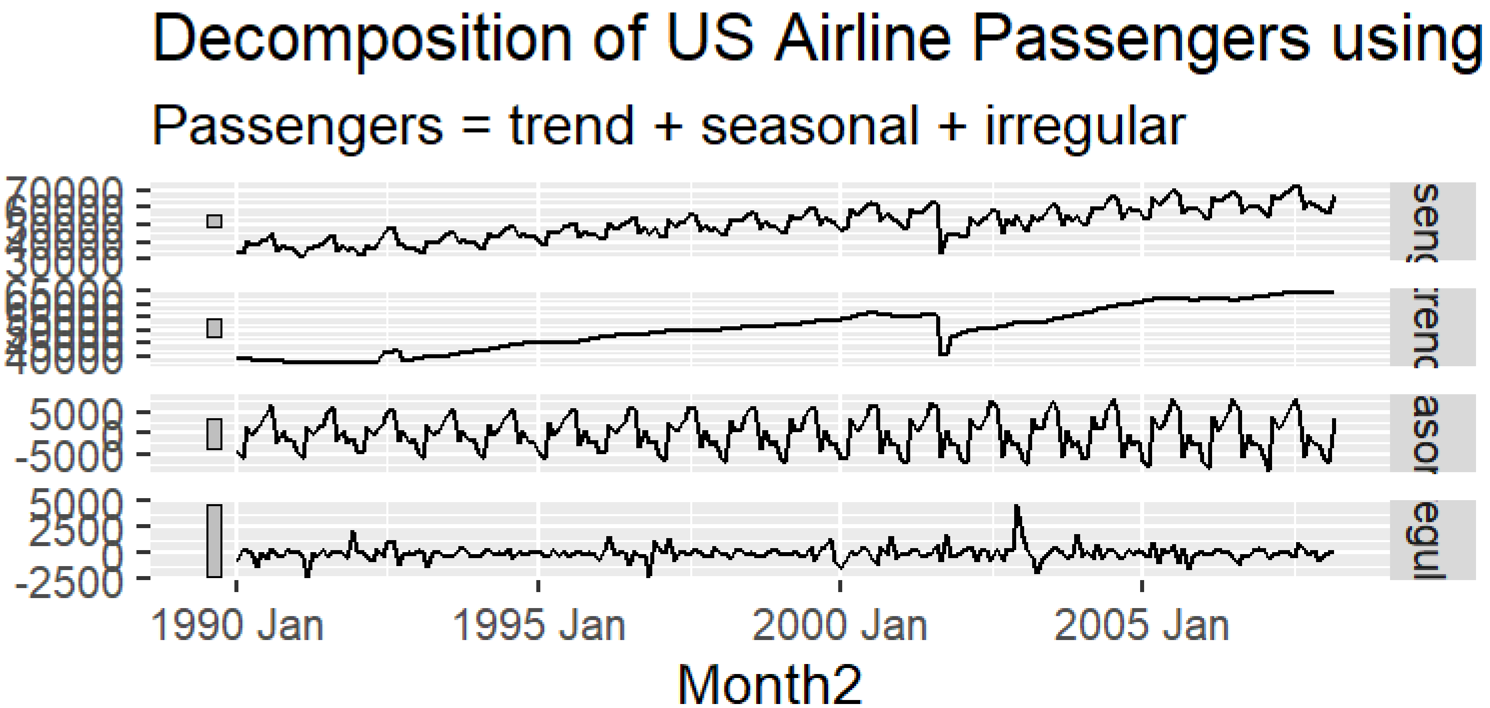
```
USAirlines_ts |>  model(classical_decomposition(Passengers,
type = "additive")  ) |>
components() |>  autoplot() +  labs(title = "Classical additive
decomposition of US Airline Passengers")
```



Decomposition Techniques

- There are different ways to decompose time series data.
- Here are 3 common techniques:
  1. Classical Decomposition
  2. X-11 ARIMA Decomposition
     a. Trend – Uses Moving / Rolling Average Smoothing
     b. Seasonal – Uses Moving / Rolling Average Smoothing
     c. Iteratively Repeats Above Methods and ARIMA Modeling
     d. Can handle outliers
     e. Automatic (will choose best…either additive or multiplicative, etc).

```
x11_dcmp <- USAirlines_ts |>  model(x11 =
X_13ARIMA_SEATS(Passengers ~ x11())) |>  components()

autoplot(x11_dcmp) +  labs(title = "Decomposition of US Airline
Passengers using X-11.")
```

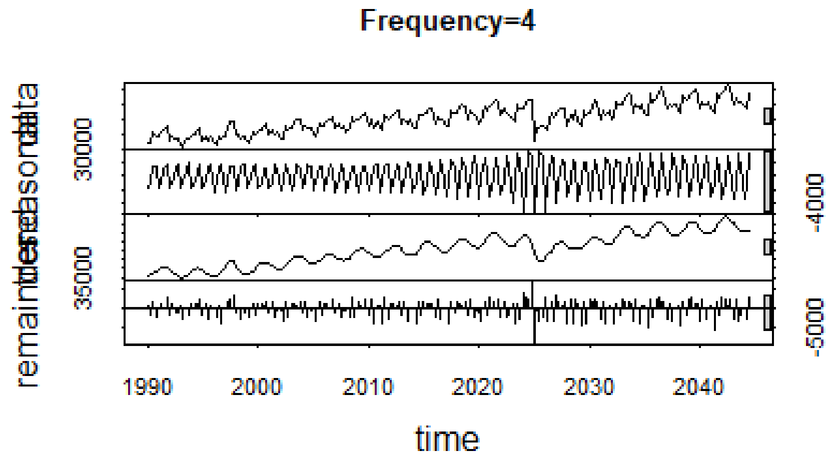


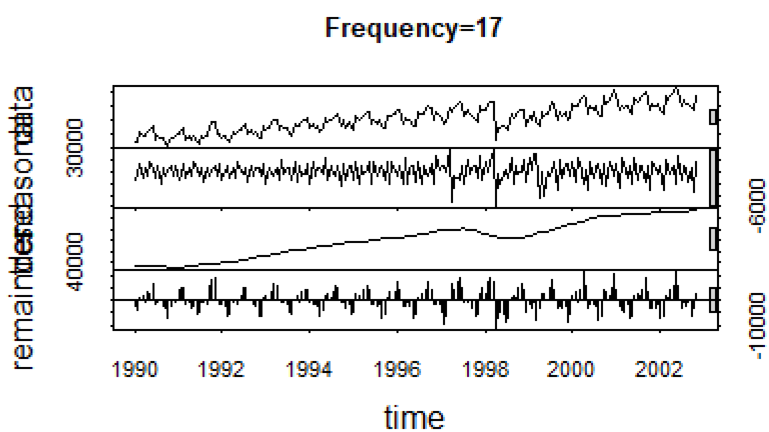
Decomposition Techniques

- There are different ways to decompose time series data.
- Here are 3 common techniques:
  1. Classical Decomposition
  2. X-11 ARIMA Decomposition
  3. STL (Seasonal and Trend using LOESS estimation) Decomposition
     a. Default of stl Function in R (Not available in SAS)
     b. Uses **LO**cal regr**ESS**ion Techniques to Estimate Trend and Seasonality
     c. Allows Changing Effects for Trend and Season
     d. Adapted to Handle Outliers

Cautions on decomposition

- You MUST have more than one observation per year in order to “decompose” a data set
- Decomposition will NOT tell you if you have seasonal data (nor the length of seasonality)
- See Airline data to the right (with a specifying “quarterly” and “17 observations per year”





Measures for “strength” of trend and/or seasonality

- Measures provided by Hyndman and Athanasopoulos
- Values of F close to 0 indicate little strength and values close to 1 indicate high strength

$$F_T = \max\left(0, 1 - \frac{\mathrm{Var}(R_t)}{\mathrm{Var}(T_t + R_t)}\right).$$

$$F_S = \max\left(0, 1 - \frac{\mathrm{Var}(R_t)}{\mathrm{Var}(S_t + R_t)}\right).$$

```
USAirlines_ts |>
features(Passengers, feat_stl)
```

A tibble: 1 × 9

trend_strength <dbl>	seasonal_strength_year <dbl>
0.9760015	0.9186362

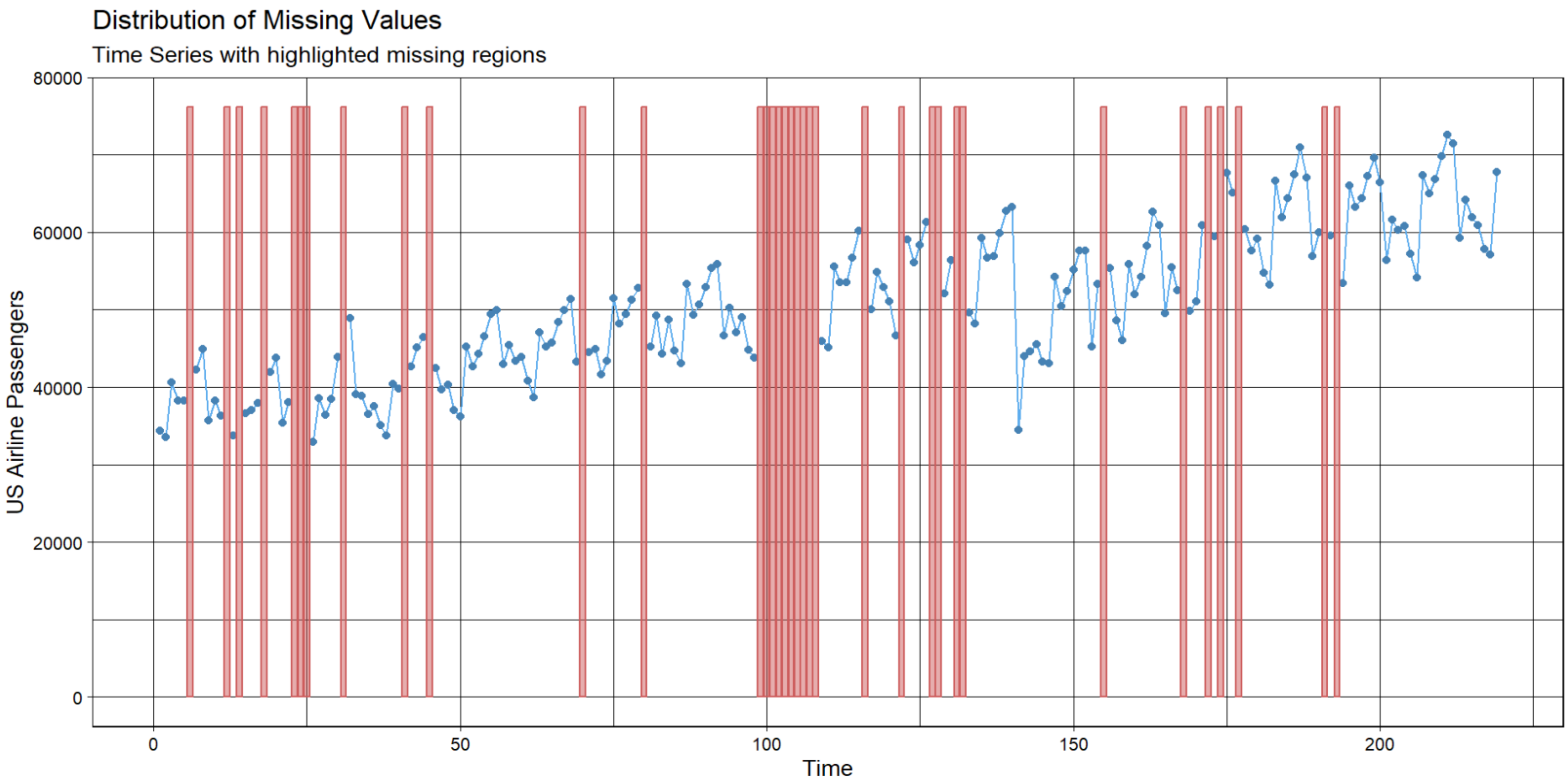
1 row | 1-2 of 9 columns

IMPUTING MISSING DATA

Imputets

- Package imputets for imputing time series data
- Allows for MANY different ways to impute!!
- However, need to use OLD formatting for time series data (from forecast package):

```
library(imputeTS)
library(forecast)
### NOTE: I made some of these values missing
Pass_miss<-ts(US_temp$Passengers,start = c(1990,1),frequency = 12)
ggplot_na_distribution(Pass_miss)+labs(y="US Airline Passengers")
```

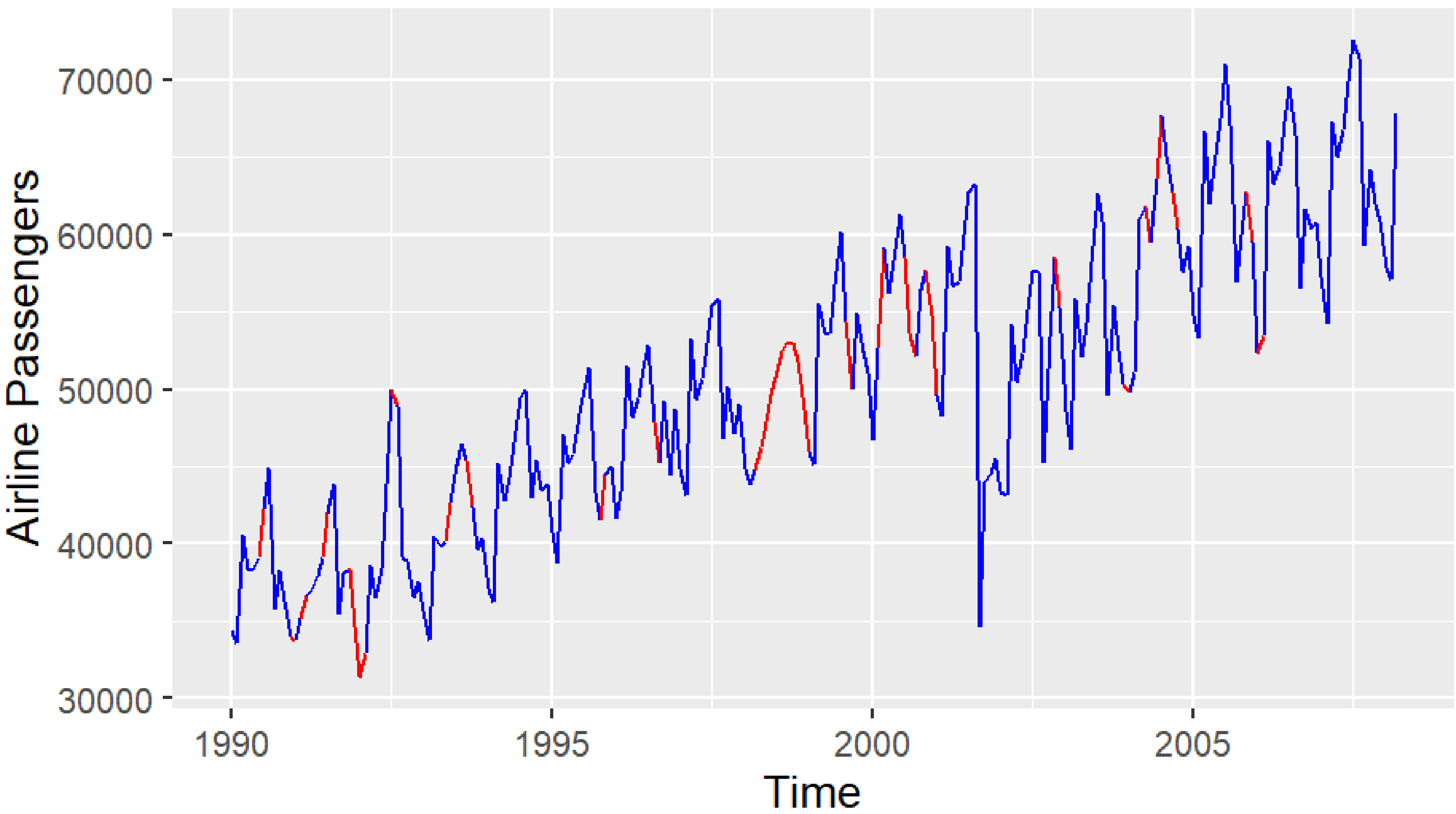


Distribution of Missing Values
Time Series with highlighted missing regions
80000
60000
40000
20000
0
US Airline Passengers
0
50
100
150
200
Time

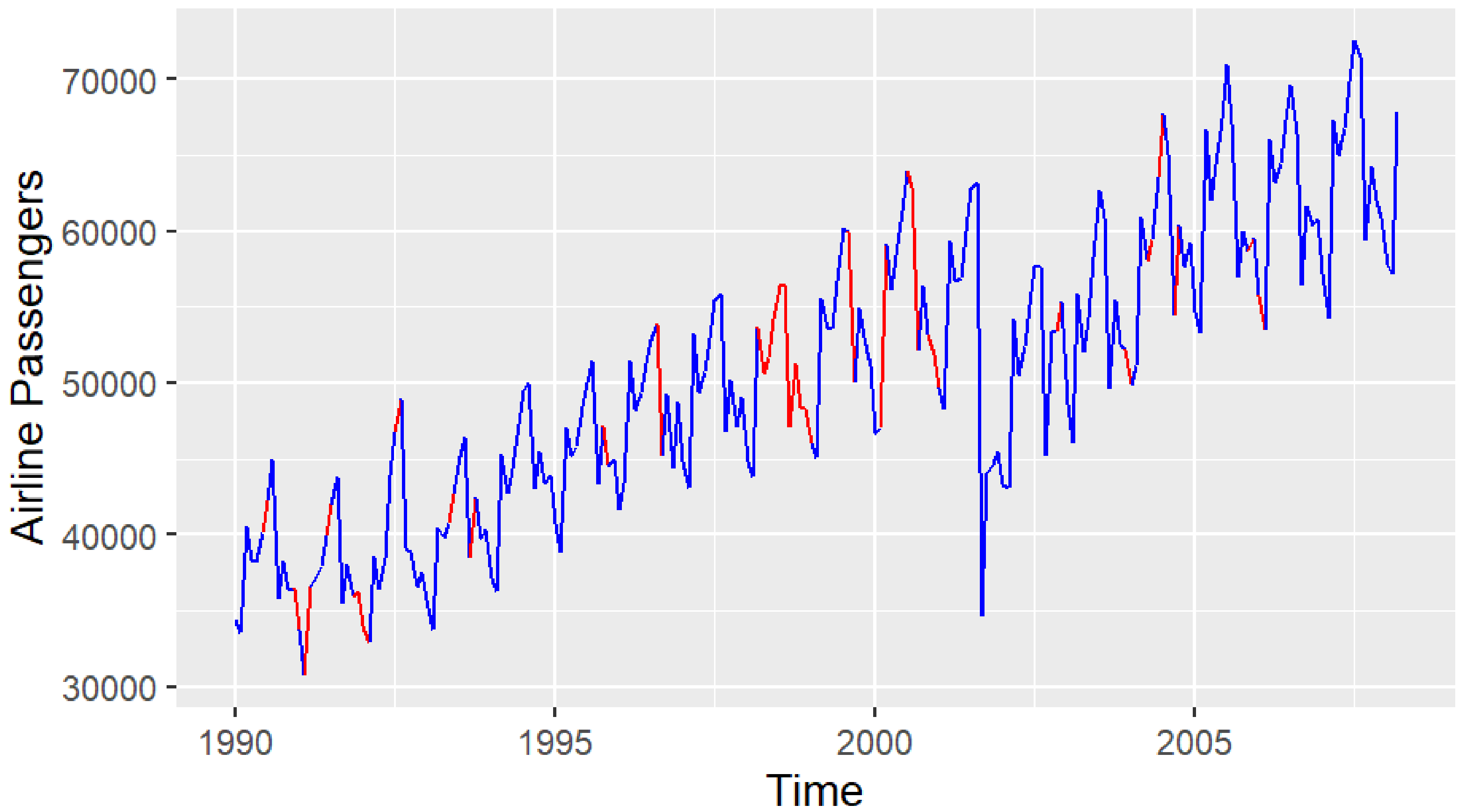
Some types of imputing:

- Mean – uses the mean of the series (not a good imputation method for time series data)
- Locf – last observation carried forward (could also do Next Observation carried backward…NOCB)
- Spline – uses splines to impute missing values (could also use linear, quadratic, etc)
- Seadec – Seasonally decomposed imputation (does a decomposition and removes the seasonal component, imputes data then puts seasonal component back in)
- And many more…. See https://cran.r-project.org/web/packages/imputeTS/imputeTS.pdf for more information!!

```
Pass_impute<-Pass_miss %>% na_interpolation(option = "spline")
autoplot(Pass_impute,color=col_vector) + labs(y="Airline Passengers")
```



```
Pass_impute<-Pass_miss %>% na_seadec(algorithm = "interpolation")
autoplot(Pass_impute,color=col_vector) + labs(y="Airline Passengers")
```



Questions?