Change

Time Series Plot From a Time Series Object (ts)

The ggfortify package allows autoplot to automatically plot directly from a time series object (ts).

## From Timeseries object (ts)

**library**(ggplot2)

**library**(ggfortify)

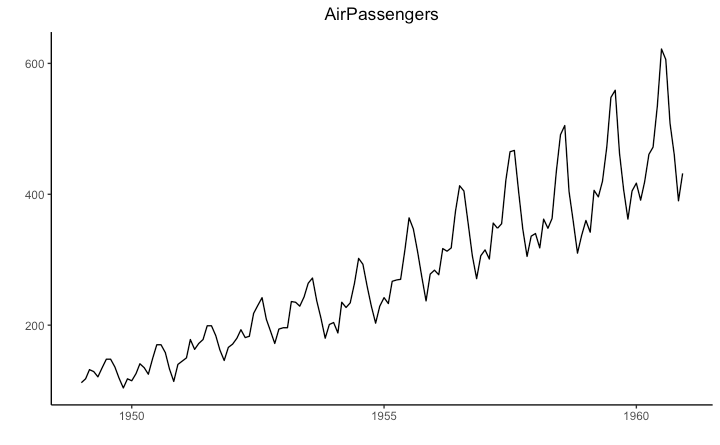
**theme\_set**(**theme\_classic**())

*# Plot*

**autoplot**(AirPassengers) +

**labs**(title="AirPassengers") +

**theme**(plot.title = **element\_text**(hjust=0.5))



Time Series Plot From a Data Frame

Using geom\_line(), a time series (or line chart) can be drawn from a data.frame as well. The X axis breaks are generated by default. In below example, the breaks are formed once every 10 years.

Default X Axis Labels

**library**(ggplot2)

**theme\_set**(**theme\_classic**())

economics$returns\_perc <- c(0, diff(economics$psavert)/economics$psavert[-length(economics$psavert)])

*# Allow Default X Axis Labels*

**ggplot**(economics, **aes**(x=date)) +

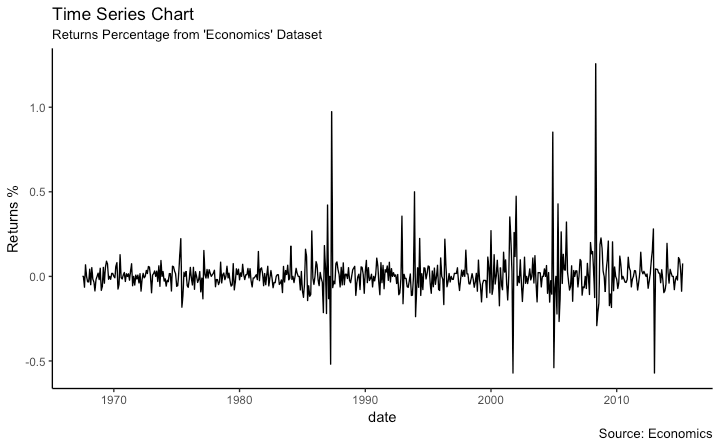
**geom\_line**(**aes**(y=returns\_perc)) +

**labs**(title="Time Series Chart",

subtitle="Returns Percentage from 'Economics' Dataset",

caption="Source: Economics",

y="Returns %")



Time Series Plot For a *Monthly* Time Series

If you want to set your own time intervals (breaks) in X axis, you need to set the breaks and labels using scale\_x\_date().

**library**(ggplot2)

**library**(lubridate)

**theme\_set**(**theme\_bw**())

economics$returns\_perc <- c(0, diff(economics$psavert)/economics$psavert[-length(economics$psavert)])

economics\_m <- economics[1:24, ]

*# labels and breaks for X axis text*

lbls <- **paste0**(month.abb[**month**(economics\_m$date)], " ", lubridate::**year**(economics\_m$date))

brks <- economics\_m$date

*# plot*

**ggplot**(economics\_m, **aes**(x=date)) +

**geom\_line**(**aes**(y=returns\_perc)) +

**labs**(title="Monthly Time Series",

subtitle="Returns Percentage from Economics Dataset",

caption="Source: Economics",

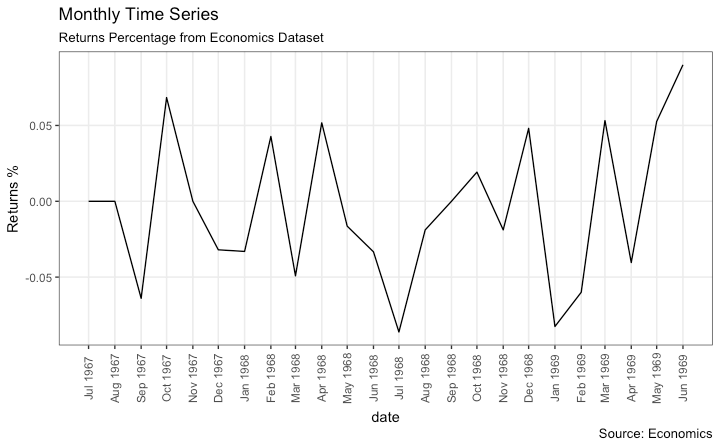
y="Returns %") + *# title and caption*

**scale\_x\_date**(labels = lbls,

breaks = brks) + *# change to monthly ticks and labels*

**theme**(axis.text.x = **element\_text**(angle = 90, vjust=0.5), *# rotate x axis text*

panel.grid.minor = **element\_blank**()) *# turn off minor grid*



Time Series Plot For a *Yearly* Time Series

**library**(ggplot2)

**library**(lubridate)

**theme\_set**(**theme\_bw**())

economics$returns\_perc <- c(0, diff(economics$psavert)/economics$psavert[-length(economics$psavert)])

economics\_y <- economics[1:90, ]

*# labels and breaks for X axis text*

brks <- economics\_y$date[**seq**(1, **length**(economics\_y$date), 12)]

lbls <- lubridate::**year**(brks)

*# plot*

**ggplot**(economics\_y, **aes**(x=date)) +

**geom\_line**(**aes**(y=returns\_perc)) +

**labs**(title="Yearly Time Series",

subtitle="Returns Percentage from Economics Dataset",

caption="Source: Economics",

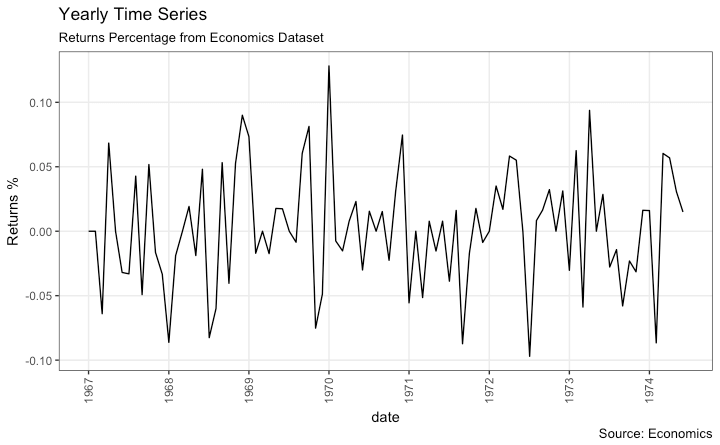
y="Returns %") + *# title and caption*

**scale\_x\_date**(labels = lbls,

breaks = brks) + *# change to monthly ticks and labels*

**theme**(axis.text.x = **element\_text**(angle = 90, vjust=0.5), *# rotate x axis text*

panel.grid.minor = **element\_blank**()) *# turn off minor grid*



Time Series Plot From Long Data Format: Multiple Time Series in Same Dataframe Column

In this example, I construct the ggplot from a long data format. That means, the column names and respective values of all the columns are stacked in just 2 variables (variable and value respectively). If you were to convert this data to wide format, it would look like the economics dataset.

In below example, the geom\_line is drawn for value column and the aes(col) is set to variable. This way, with just one call to geom\_line, multiple colored lines are drawn, one each for each unique value in variable column. The scale\_x\_date() changes the X axis breaks and labels, and scale\_color\_manualchanges the color of the lines.

**data**(economics\_long, package = "ggplot2")

**head**(economics\_long)

*#> date variable value value01*

*#> <date> <fctr> <dbl> <dbl>*

*#> 1 1967-07-01 pce 507.4 0.0000000000*

*#> 2 1967-08-01 pce 510.5 0.0002660008*

*#> 3 1967-09-01 pce 516.3 0.0007636797*

*#> 4 1967-10-01 pce 512.9 0.0004719369*

*#> 5 1967-11-01 pce 518.1 0.0009181318*

*#> 6 1967-12-01 pce 525.8 0.0015788435*

**library**(ggplot2)

**library**(lubridate)

**theme\_set**(**theme\_bw**())

df <- economics\_long[economics\_long$variable %in% **c**("psavert", "uempmed"), ]

df <- df[lubridate::**year**(df$date) %in% **c**(1967:1981), ]

*# labels and breaks for X axis text*

brks <- df$date[**seq**(1, **length**(df$date), 12)]

lbls <- lubridate::**year**(brks)

*# plot*

**ggplot**(df, **aes**(x=date)) +

**geom\_line**(**aes**(y=value, col=variable)) +

**labs**(title="Time Series of Returns Percentage",

subtitle="Drawn from Long Data format",

caption="Source: Economics",

y="Returns %",

color=NULL) + *# title and caption*

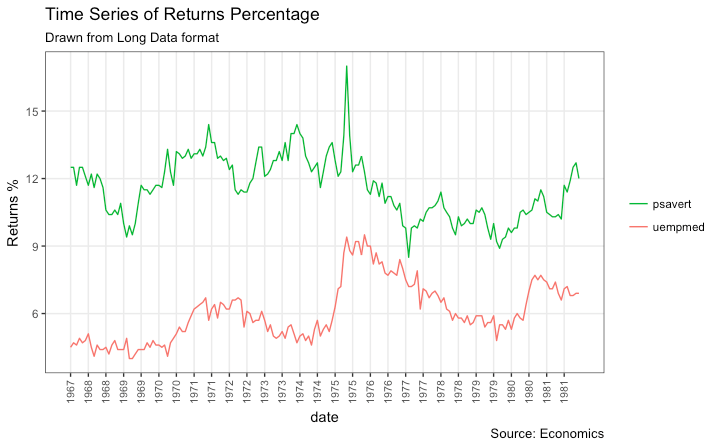
**scale\_x\_date**(labels = lbls, breaks = brks) + *# change to monthly ticks and labels*

**scale\_color\_manual**(labels = **c**("psavert", "uempmed"),

values = **c**("psavert"="#00ba38", "uempmed"="#f8766d")) + *# line color*

**theme**(axis.text.x = **element\_text**(angle = 90, vjust=0.5, size = 8), *# rotate x axis text*

panel.grid.minor = **element\_blank**()) *# turn off minor grid*



Time Series Plot From Wide Data Format: Data in Multiple Columns of Dataframe

As noted in the [part 2](http://r-statistics.co/Complete-Ggplot2-Tutorial-Part2-Customizing-Theme-With-R-Code.html#2.%20Modifying%20Legend) of this tutorial, whenever your plot’s geom (like points, lines, bars, etc) changes the fill, size, col, shape or stroke based on another column, a legend is automatically drawn.

But if you are creating a time series (or even other types of plots) from a wide data format, you have to draw each line manually by calling geom\_line() once for every line. So, a legend will not be drawn by default.

However, having a legend would still be nice. This can be done using the scale\_aesthetic\_manual()format of functions (like, scale\_color\_manual() if only the color of your lines change). Using this function, you can give a legend title with the name argument, tell what color the legend should take with the valuesargument and also set the legend labels.

Even though the below plot looks exactly like the previous one, the approach to construct this is different.

You might wonder why I used this function in previous example for long data format as well. Note that, in previous example, it was used to change the color of the line only. Without scale\_color\_manual(), you would still have got a legend, but the lines would be of a different (default) color. But in current example, without scale\_color\_manual(), you wouldn’t even have a legend. Try it out!

**library**(ggplot2)

**library**(lubridate)

**theme\_set**(**theme\_bw**())

df <- economics[, **c**("date", "psavert", "uempmed")]

df <- df[lubridate::**year**(df$date) %in% **c**(1967:1981), ]

*# labels and breaks for X axis text*

brks <- df$date[**seq**(1, **length**(df$date), 12)]

lbls <- lubridate::**year**(brks)

*# plot*

**ggplot**(df, **aes**(x=date)) +

**geom\_line**(**aes**(y=psavert, col="psavert")) +

**geom\_line**(**aes**(y=uempmed, col="uempmed")) +

**labs**(title="Time Series of Returns Percentage",

subtitle="Drawn From Wide Data format",

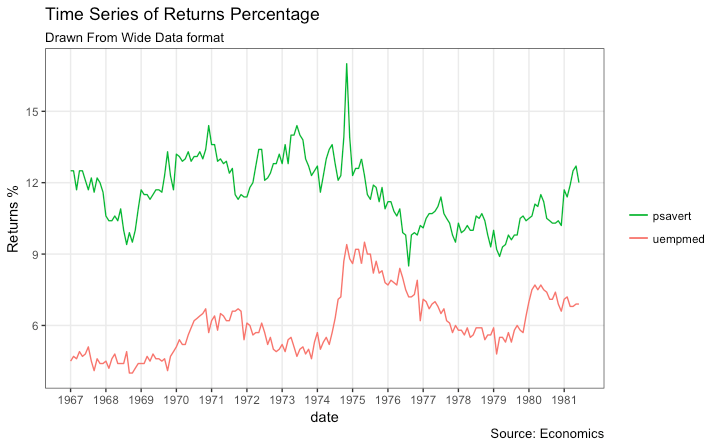
caption="Source: Economics", y="Returns %") + *# title and caption*

**scale\_x\_date**(labels = lbls, breaks = brks) + *# change to monthly ticks and labels*

**scale\_color\_manual**(name="",

values = **c**("psavert"="#00ba38", "uempmed"="#f8766d")) + *# line color*

**theme**(panel.grid.minor = **element\_blank**()) *# turn off minor grid*



Stacked Area Chart

Stacked area chart is just like a line chart, except that the region below the plot is all colored. This is typically used when:

1. You want to describe how a quantity or volume (rather than something like price) changed over time
2. You have many data points. For very few data points, consider plotting a bar chart.
3. You want to show the contribution from individual components.

This can be plotted using geom\_area which works very much like geom\_line. But there is an important point to note. By default, each geom\_area() starts from the bottom of Y axis (which is typically 0), but, if you want to show the contribution from individual components, you want the geom\_area to be stacked over the top of previous component, rather than the floor of the plot itself. So, you have to add all the bottom layers while setting the y of geom\_area.

In below example, I have set it as y=psavert+uempmed for the topmost geom\_area().

However nice the plot looks, the caveat is that, it can easily become complicated and uninterprettable if there are too many components

**library**(ggplot2)

**library**(lubridate)

**theme\_set**(**theme\_bw**())

df <- economics[, **c**("date", "psavert", "uempmed")]

df <- df[lubridate::**year**(df$date) %in% **c**(1967:1981), ]

*# labels and breaks for X axis text*

brks <- df$date[**seq**(1, **length**(df$date), 12)]

lbls <- lubridate::**year**(brks)

*# plot*

**ggplot**(df, **aes**(x=date)) +

**geom\_area**(**aes**(y=psavert+uempmed, fill="psavert")) +

**geom\_area**(**aes**(y=uempmed, fill="uempmed")) +

**labs**(title="Area Chart of Returns Percentage",

subtitle="From Wide Data format",

caption="Source: Economics",

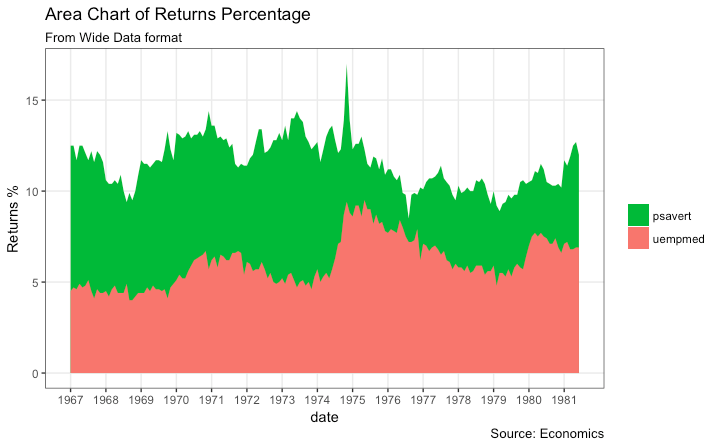
y="Returns %") + *# title and caption*

**scale\_x\_date**(labels = lbls, breaks = brks) + *# change to monthly ticks and labels*

**scale\_fill\_manual**(name="",

values = **c**("psavert"="#00ba38", "uempmed"="#f8766d")) + *# line color*

**theme**(panel.grid.minor = **element\_blank**()) *# turn off minor grid*



### Slope Chart

Slope chart is a great tool of you want to visualize change in value and ranking between categories. This is more suitable over a time series when there are very few time points.

**library**(dplyr)

**theme\_set**(**theme\_classic**())

source\_df <- **read.csv**("https://raw.githubusercontent.com/jkeirstead/r-slopegraph/master/cancer\_survival\_rates.csv")

*# Define functions. Source: https://github.com/jkeirstead/r-slopegraph*

tufte\_sort <- function(df, x="year", y="value", group="group", method="tufte", min.space=0.05) {

## First rename the columns for consistency

ids <- **match**(**c**(x, y, group), **names**(df))

df <- df[,ids]

**names**(df) <- **c**("x", "y", "group")

## Expand grid to ensure every combination has a defined value

tmp <- **expand.grid**(x=**unique**(df$x), group=**unique**(df$group))

tmp <- **merge**(df, tmp, all.y=TRUE)

df <- **mutate**(tmp, y=**ifelse**(**is.na**(y), 0, y))

## Cast into a matrix shape and arrange by first column

**require**(reshape2)

tmp <- **dcast**(df, group ~ x, value.var="y")

ord <- **order**(tmp[,2])

tmp <- tmp[ord,]

min.space <- min.space\***diff**(**range**(tmp[,-1]))

yshift <- **numeric**(**nrow**(tmp))

## Start at "bottom" row

## Repeat for rest of the rows until you hit the top

for (i in 2:**nrow**(tmp)) {

## Shift subsequent row up by equal space so gap between

## two entries is >= minimum

mat <- **as.matrix**(tmp[(i-1):i, -1])

d.min <- **min**(**diff**(mat))

yshift[i] <- **ifelse**(d.min < min.space, min.space - d.min, 0)

}

tmp <- **cbind**(tmp, yshift=**cumsum**(yshift))

scale <- 1

tmp <- **melt**(tmp, id=**c**("group", "yshift"), variable.name="x", value.name="y")

## Store these gaps in a separate variable so that they can be scaled ypos = a\*yshift + y

tmp <- **transform**(tmp, ypos=y + scale\*yshift)

**return**(tmp)

}

plot\_slopegraph <- function(df) {

ylabs <- **subset**(df, x==**head**(x,1))$group

yvals <- **subset**(df, x==**head**(x,1))$ypos

fontSize <- 3

gg <- **ggplot**(df,**aes**(x=x,y=ypos)) +

**geom\_line**(**aes**(group=group),colour="grey80") +

**geom\_point**(colour="white",size=8) +

**geom\_text**(**aes**(label=y), size=fontSize, family="American Typewriter") +

**scale\_y\_continuous**(name="", breaks=yvals, labels=ylabs)

**return**(gg)

}

## Prepare data

df <- **tufte\_sort**(source\_df,

x="year",

y="value",

group="group",

method="tufte",

min.space=0.05)

df <- **transform**(df,

x=**factor**(x, levels=**c**(5,10,15,20),

labels=**c**("5 years","10 years","15 years","20 years")),

y=**round**(y))

## Plot

**plot\_slopegraph**(df) + **labs**(title="Estimates of % survival rates") +

**theme**(axis.title=**element\_blank**(),

axis.ticks = **element\_blank**(),

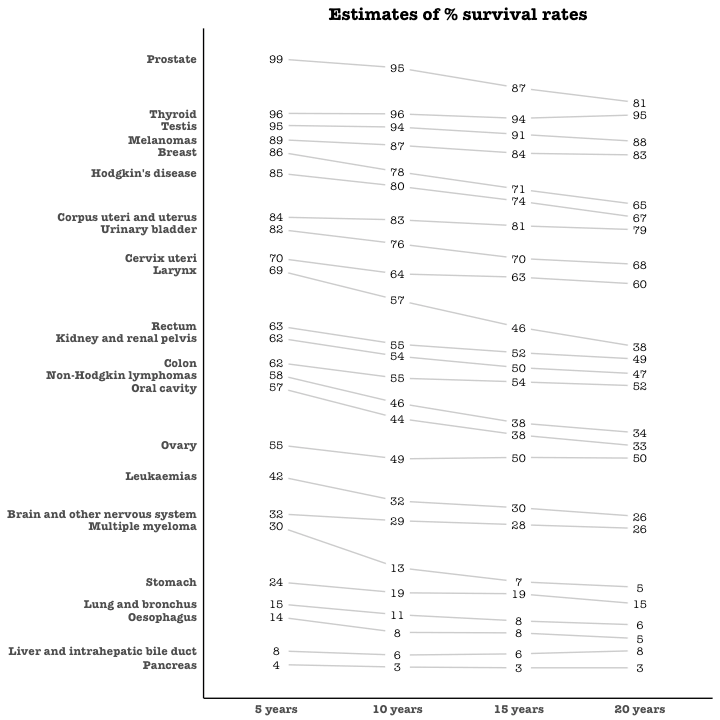
plot.title = **element\_text**(hjust=0.5,

family = "American Typewriter",

face="bold"),

axis.text = **element\_text**(family = "American Typewriter",

face="bold"))



### Seasonal Plot

If you are working with a time series object of class ts or xts, you can view the seasonal fluctuations through a seasonal plot drawn using forecast::ggseasonplot. Below is an example using the native AirPassengers and nottem time series.

You can see the traffic increase in air passengers over the years along with the repetitive seasonal patterns in traffic. Whereas Nottingham does not show an increase in overal temperatures over the years, but they definitely follow a seasonal pattern.

**library**(ggplot2)

**library**(forecast)

**theme\_set**(**theme\_classic**())

*# Subset data*

nottem\_small <- **window**(nottem, start=**c**(1920, 1), end=**c**(1925, 12)) *# subset a smaller timewindow*

*# Plot*

**ggseasonplot**(AirPassengers) + **labs**(title="Seasonal plot: International Airline Passengers")

**ggseasonplot**(nottem\_small) + **labs**(title="Seasonal plot: Air temperatures at Nottingham Castle")

