Introduction to Statistics
An Applied 3-Day Hands-On Workshop with R

Lecture 8: Graphs – ggplot2 part of the tidyverse 3.2.1

November 2019
Version: October 15, 2019

Michael Schomaker
UMIT University, Institute of Public Health, MDM and HTA

What is `ggplot2`?

- It is a *R*-package

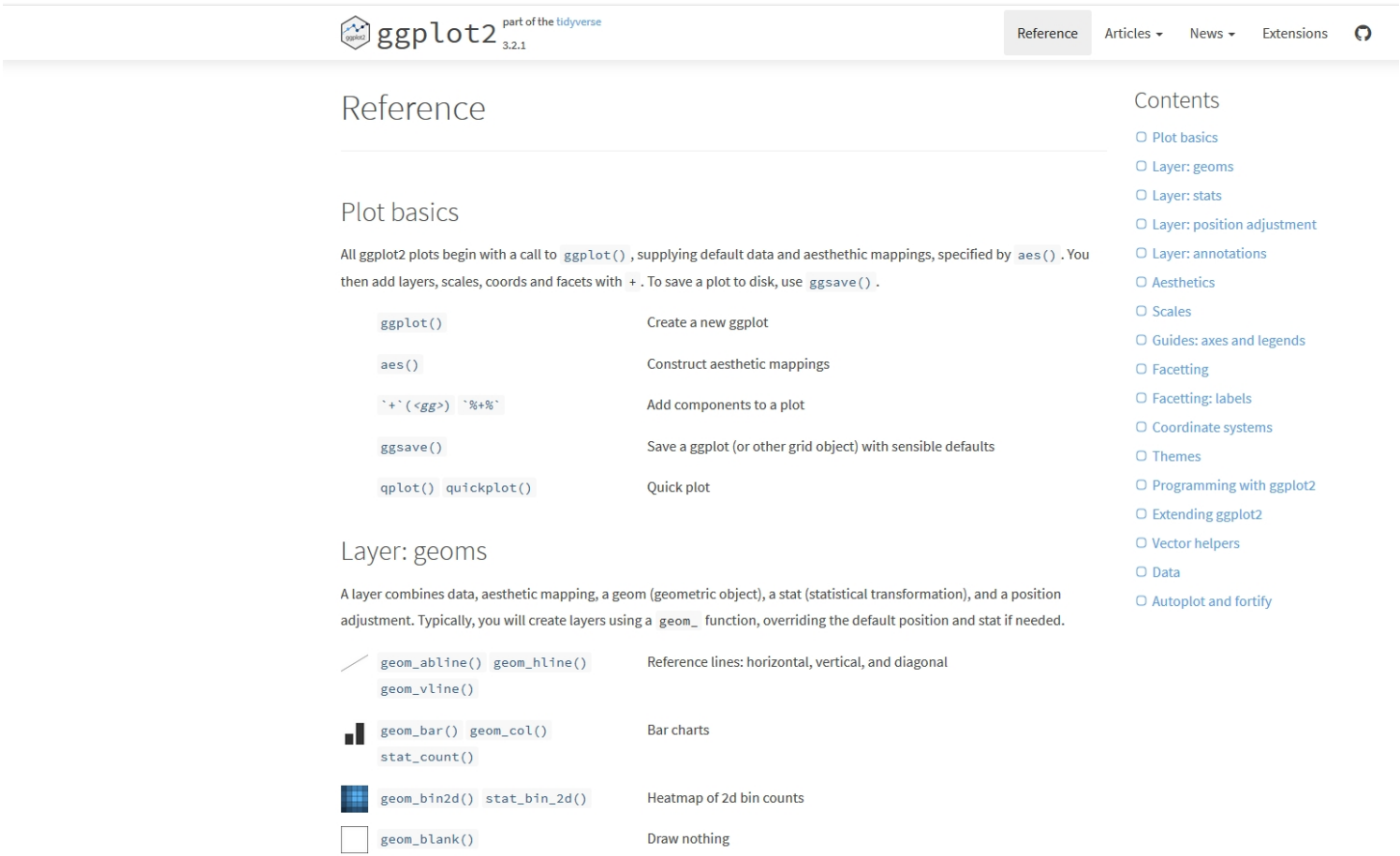
```
1 install.packages("ggplot2")
```

- The package is very popular:
  - transparency options
  - nice defaults
  - flexible and professional
- However:
  - *very* different syntax (which is not always intuitive)
  - a different “philosophy” for graphs
  - requires (initially) more time to learn and (initially) more time to solve problems

How to learn it?

Learning by doing:

`https://ggplot2.tidyverse.org/reference/index.html`



ggplot2 part of the tidyverse 3.2.1

Reference | Articles | News | Extensions

Reference

Plot basics

All ggplot2 plots begin with a call to `ggplot()`, supplying default data and aesthethic mappings, specified by `aes()`. You then add layers, scales, coords and facets with `+`. To save a plot to disk, use `ggsave()`.

`ggplot()`	Create a new ggplot
`aes()`	Construct aesthetic mappings
`` `+`(<gg>) `%+%` ``	Add components to a plot
`ggsave()`	Save a ggplot (or other grid object) with sensible defaults
`qplot()` `quickplot()`	Quick plot

Layer: geoms

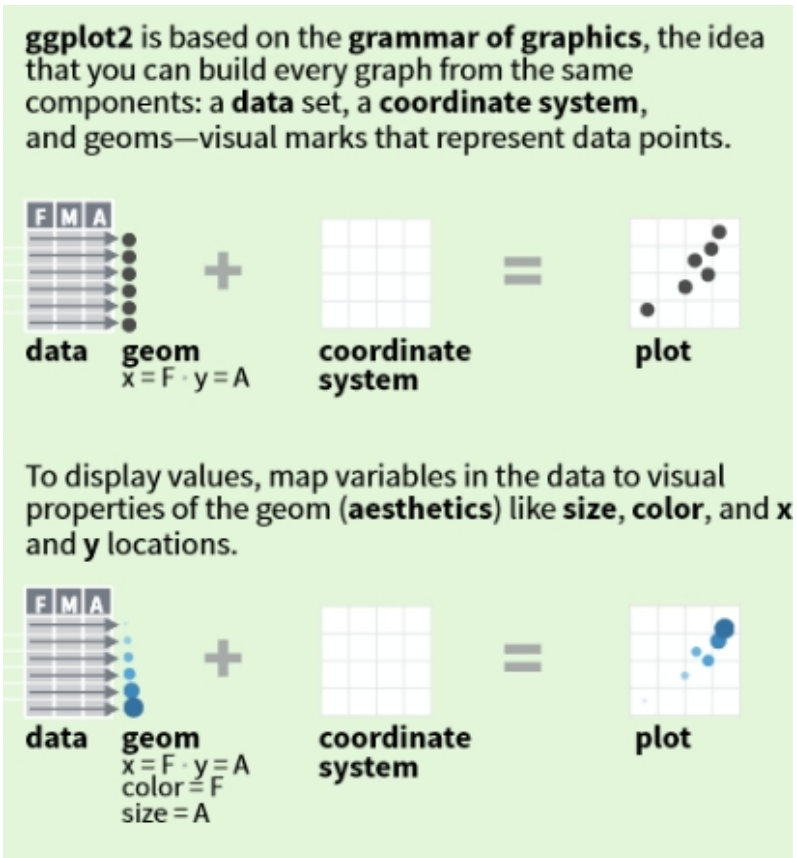
A layer combines data, aesthetic mapping, a geom (geometric object), a stat (statistical transformation), and a position adjustment. Typically, you will create layers using a `geom_` function, overriding the default position and stat if needed.

`geom_abline()` `geom_hline()` `geom_vline()`	Reference lines: horizontal, vertical, and diagonal
`geom_bar()` `geom_col()` `stat_count()`	Bar charts
`geom_bin2d()` `stat_bin_2d()`	Heatmap of 2d bin counts
`geom_blank()`	Draw nothing

Contents

- Plot basics
- Layer: geoms
- Layer: stats
- Layer: position adjustment
- Layer: annotations
- Aesthetics
- Scales
- Guides: axes and legends
- Facetting
- Facetting: labels
- Coordinate systems
- Themes
- Programming with ggplot2
- Extending ggplot2
- Vector helpers
- Data
- Autoplot and fortify

What is the principle?[1]



[1] from: https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

Plot basics

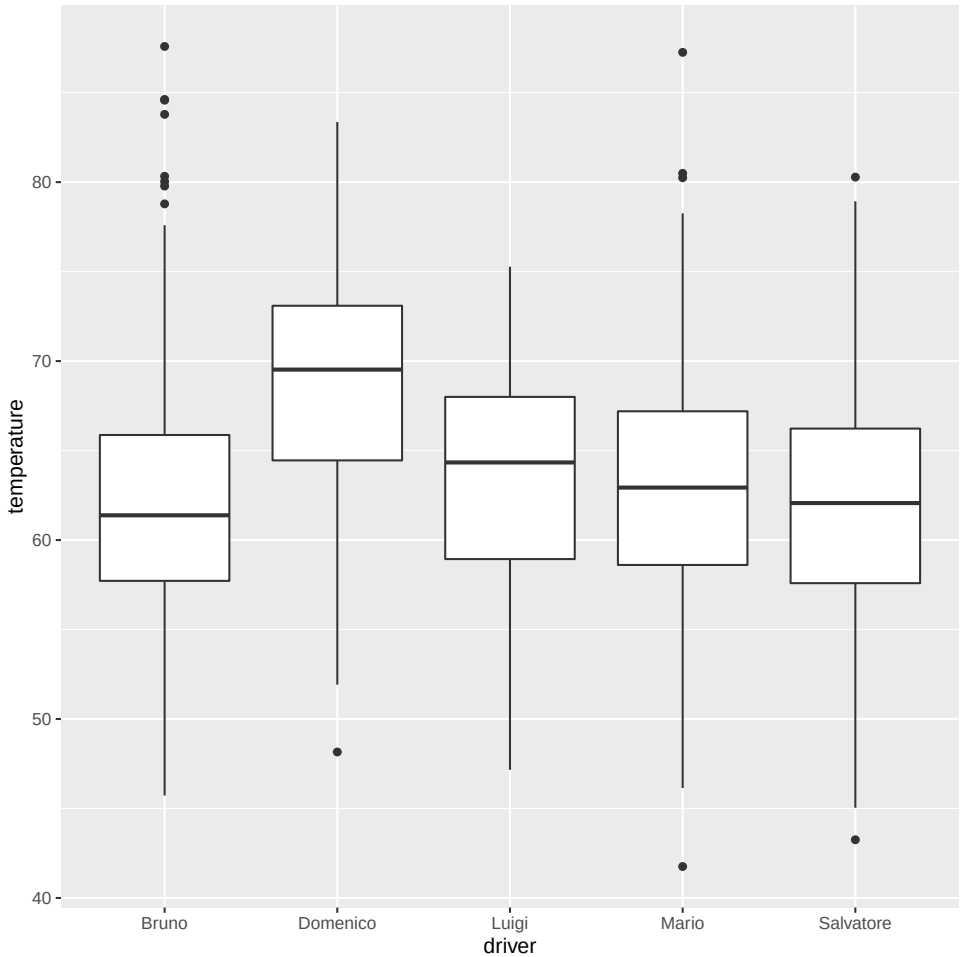

```
1 ggplot()     # Create a new ggplot
2 aes()        # Construct aesthetic mappings
3 `+`(<gg>)    # Add components to a plot
4 ggsave()     # Save a ggplot
5 qplot() quickplot()  # Quick plot
```

Syntax Example (I)

```
# Define a ggplot object
# note that we do NOT attach data
p1 <- ggplot(pizza, aes(x=driver, y=temperature))
plot(p1) # the plot is defined, now we simply need to map
    the data (or a function thereof) in here

# Option 1:
ggplot(pizza, aes(x=driver, y=temperature))
      + geom_boxplot()
# Option 2:
p2 <- p1 + geom_boxplot()
plot(p2)
```

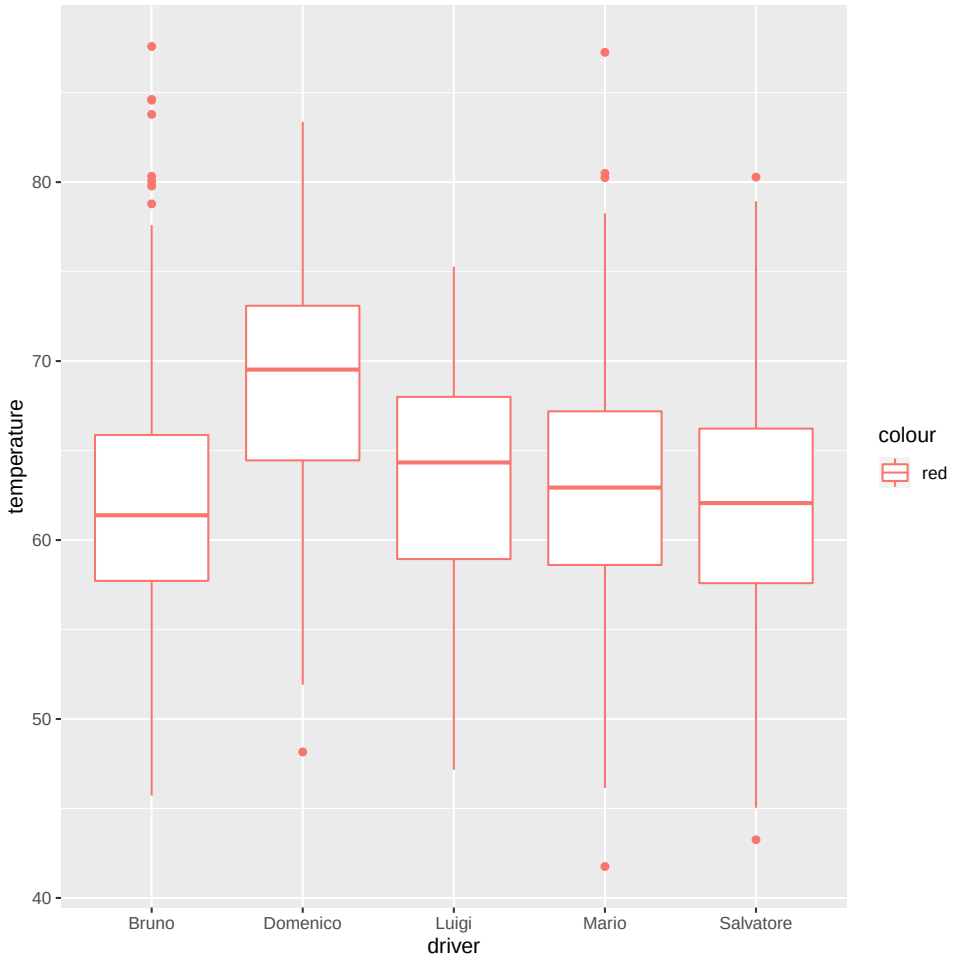
Syntax Example (II)



Syntax Example (III)

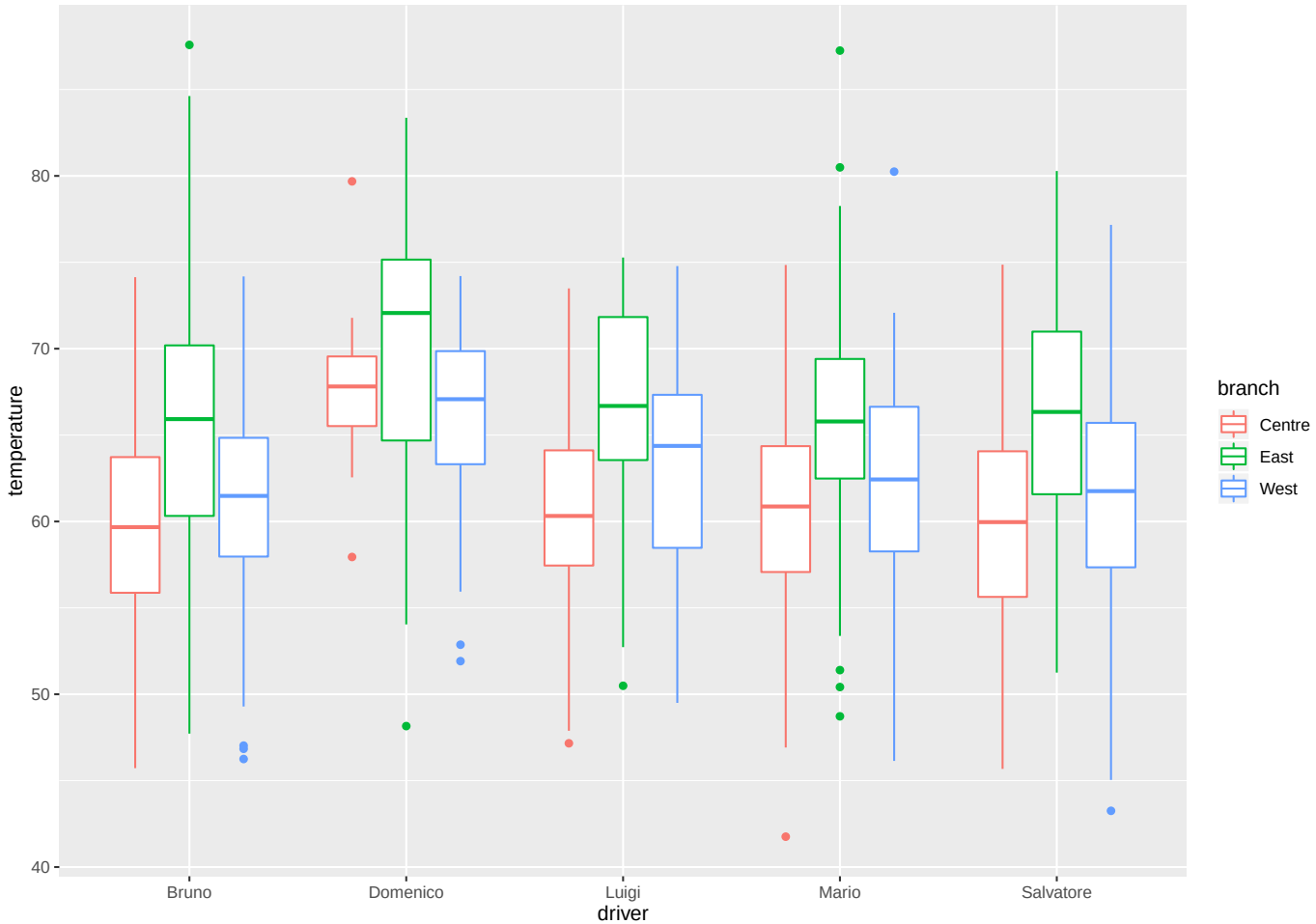

```
# aesthetics can be added...
# ...."globally"
ggplot(pizza, aes(x=driver, y=temperature, col="red")) +
    geom_boxplot()
# ...or within a layer
ggplot(pizza, aes(x=driver, y=temperature)) + geom_boxplot
    (aes(col="red"))
```

Syntax Example (IV)



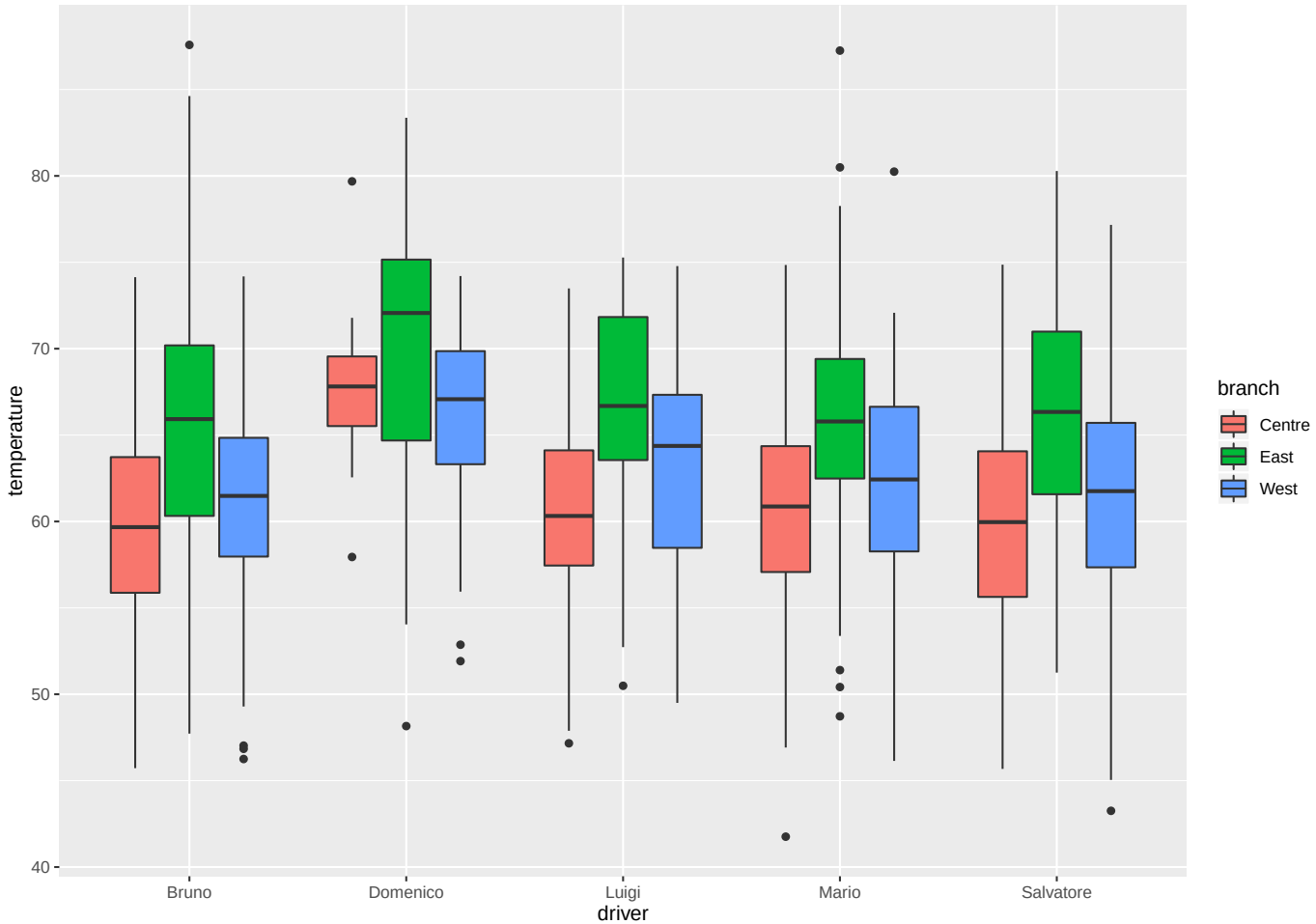
Syntax Example (V)

```
# defining groups as part of aesthetics
ggplot(pizza, aes(x=driver, y=temperature, col=branch)) +
    geom_boxplot()
```



Syntax Example (VI)

```
1 # defining groups as part of aesthetics
2 ggplot(pizza, aes(x=driver, y=temperature, fill=branch)) +
      geom_boxplot()
```



Layers

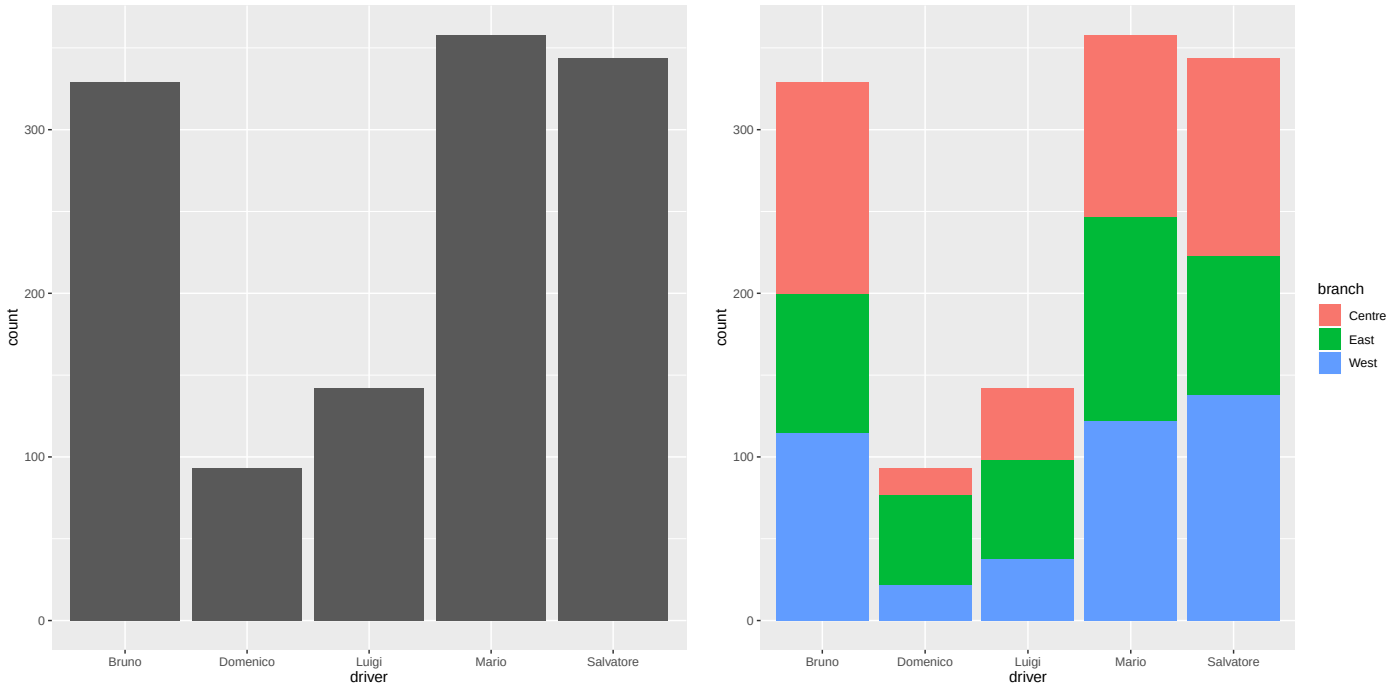
A layer combines data, aesthetic mapping, a geom, a stat, and a position adjustment. Examples are:

- box plots
- bar charts
- kernel density plots
- scatter plots
- confidence interval plots
- contour plots
- ...

Bar Charts (I)


```
> # bar chart
> ggplot(pizza, aes(x=driver)) + geom_bar()
> # stratified bar chart
> ggplot(pizza, aes(x=driver, y=branch)) + geom_bar() #
    does not work
Fehler: stat_count() must not be used with a y aesthetic.
> ggplot(pizza, aes(x=driver)) +
>  geom_bar(aes(fill=branch))  # Option 1
> ggplot(pizza, aes(x=driver, fill=branch)) +
>  geom_bar()      # Option 2
```

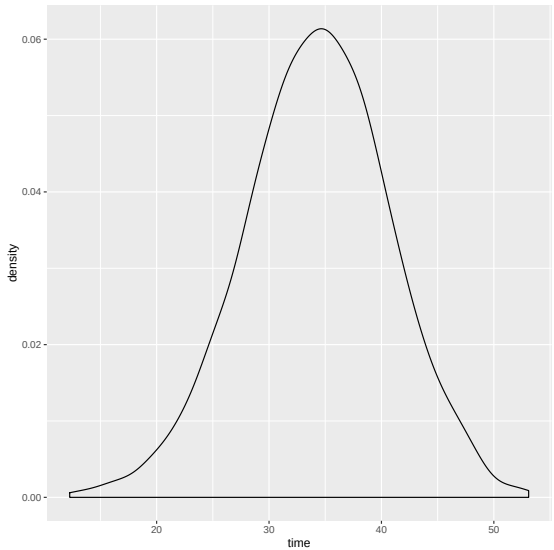
Bar Charts (II)

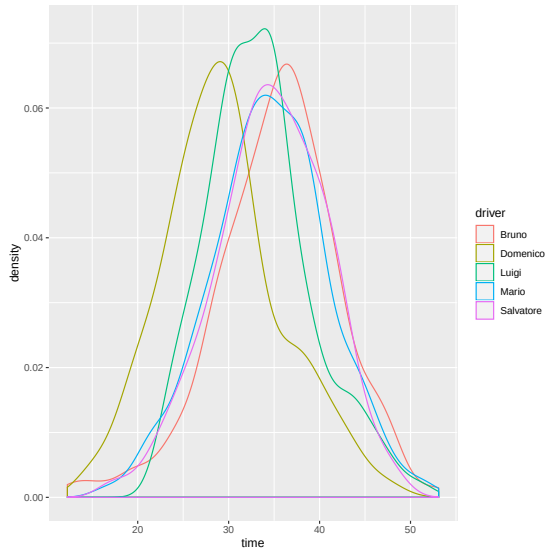


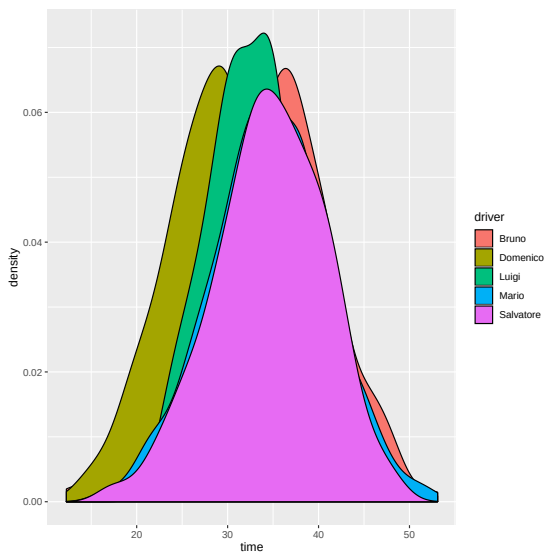
Kernel Density Plots (I)


```
> # kernel density plot
> ggplot(pizza, aes(x=time)) + geom_density()
> # stratified kernel density plots
> ggplot(pizza, aes(x=time, col=driver))  + geom_density()
> ggplot(pizza, aes(x=time, fill=driver)) + geom_density()
```

Kernel Density Plots (II)

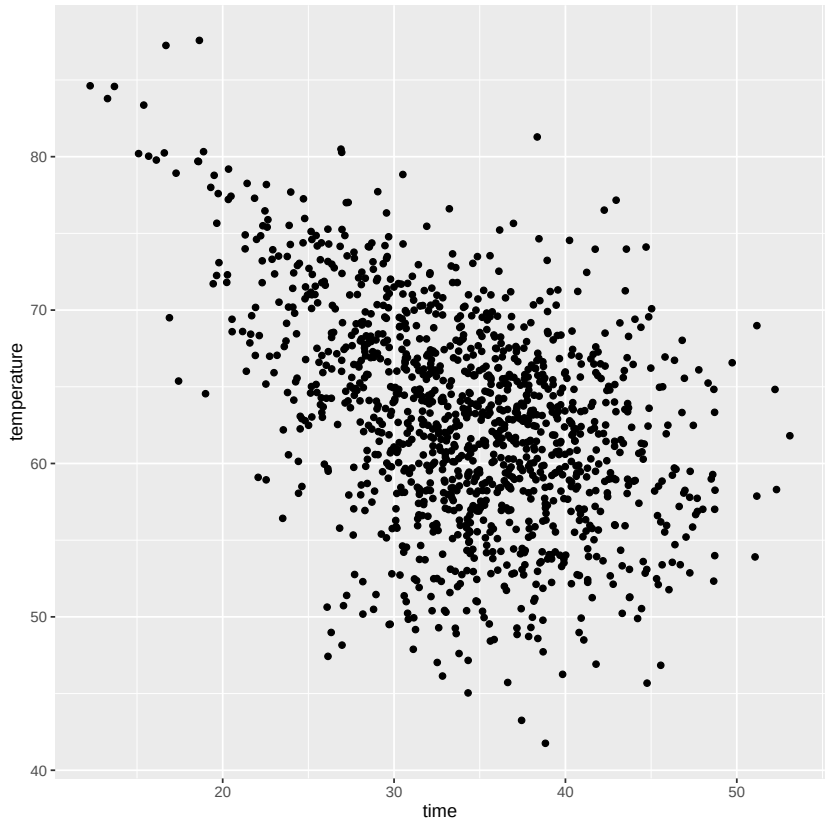






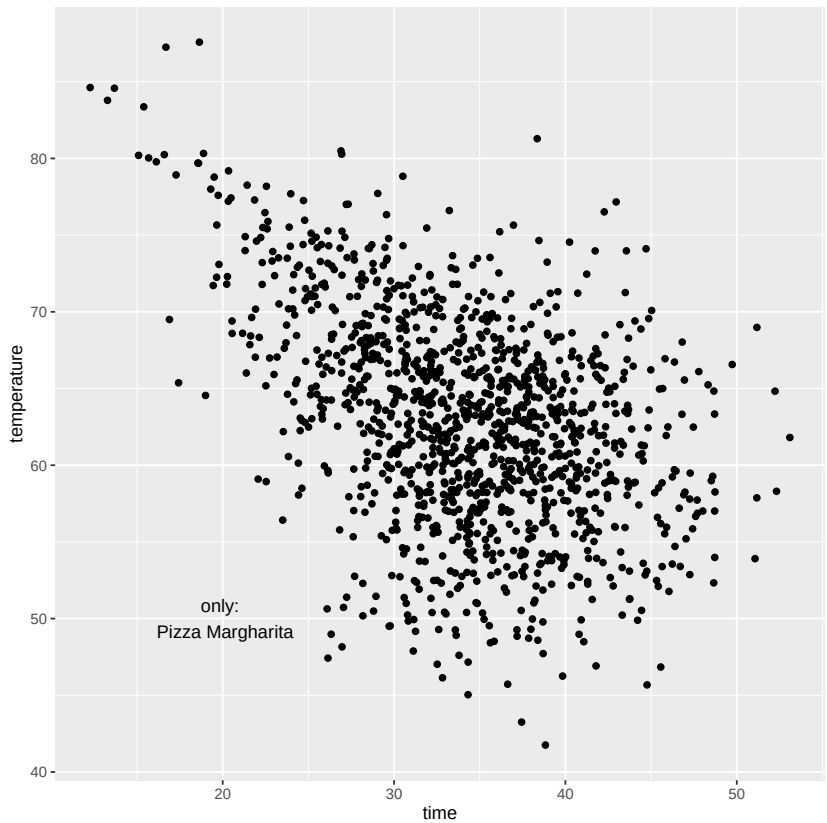
Scatter Plots

```
> # scatter plots
> ggplot(pizza, aes(time, temperature))+ geom_point()
```



Text Layer

```
> # untypical layer/geom: text
> p4 <- ggplot(pizza, aes(time, temperature))+geom_point()
> p4 + annotate("text", x = 20, y = 50, label = "only: \n
    Pizza Margharita")
```



What if we don't deal with raw data?

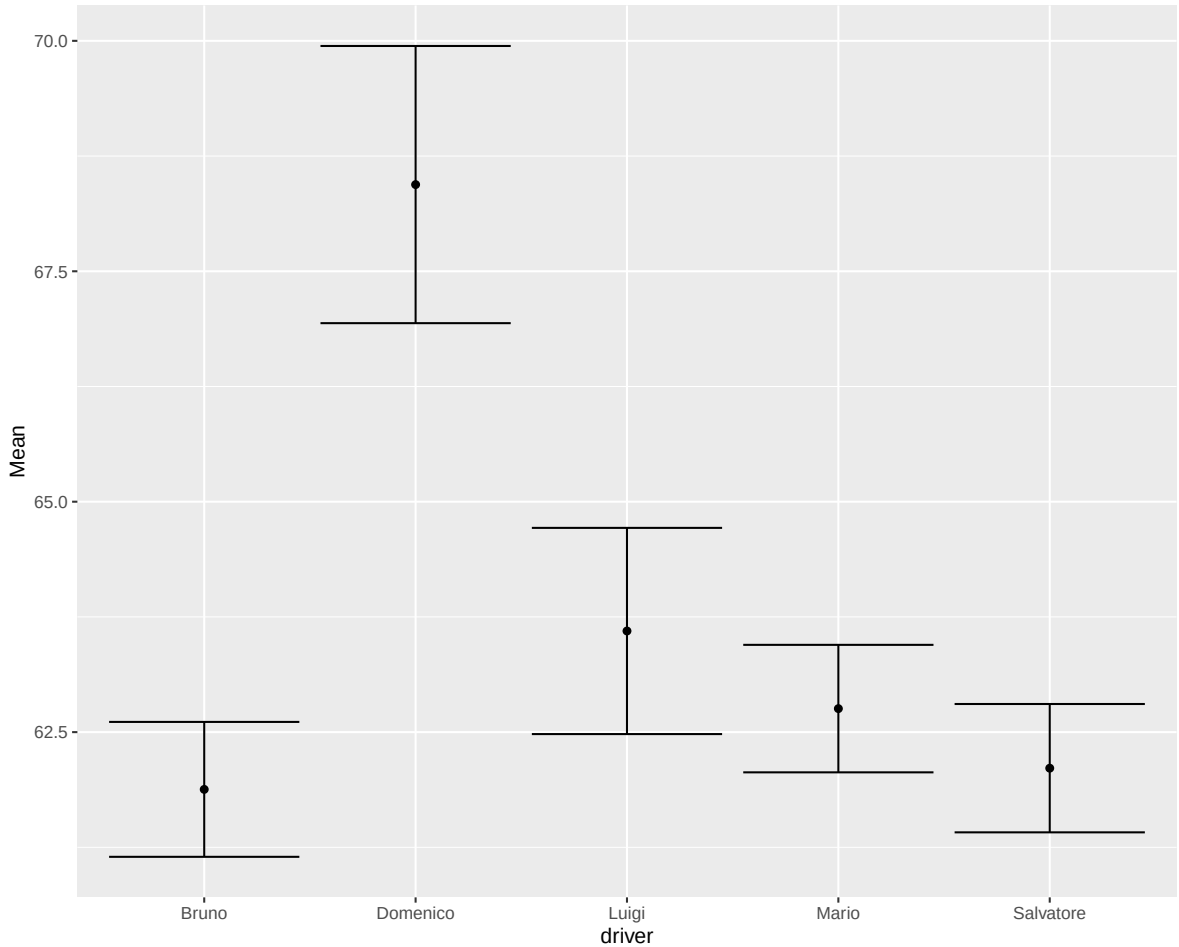
- Plots are not always based on a classical data analysis
- Graphs can be individual, and summarize various aspects of data
- For `ggplot2` one *has to* arrange the number one needs in a data frame
- This is relevant for confidence intervals plots, confidence bands, visualized regression coefficients, very individual graphs, etc.

Confidence Intervals (I)

```
> # for some geoms we can't use the raw data, but need to
    create summaries
> pdrivers    <- names(table(pizza$driver))
> tempdriver  <- matrix(NA,ncol=3,nrow=5,
>             dimnames=list(pdrivers,c("Mean","LCI","UCI")))
> for(i in 1:5){tempdriver[i,] <-
    c(mean(pizza$temperature[pizza$driver==pdrivers[i]]),
      t.test(pizza$temperature[pizza$driver==pdrivers[i]])
          $conf.int)
       }
> tempdriver  <- as.data.frame(tempdriver)
> tempdriver$driver <- pdrivers
> tempdriver
                Mean      LCI      UCI    driver
Bruno      61.87884 61.14706 62.61062     Bruno
Domenico   68.44068 66.93705 69.94431  Domenico
Luigi      63.59733 62.47871 64.71596     Luigi
Mario      62.75523 62.06364 63.44681     Mario
Salvatore 62.10885 61.41257 62.80514 Salvatore

> ggplot(tempdriver, aes(x=driver,ymin=LCI,ymax=UCI)) +
    geom_errorbar()
> ggplot(tempdriver, aes(x=driver,ymin=LCI,ymax=UCI,
           y=Mean)) + geom_errorbar() + geom_point()
```

Confidence Intervals (II)

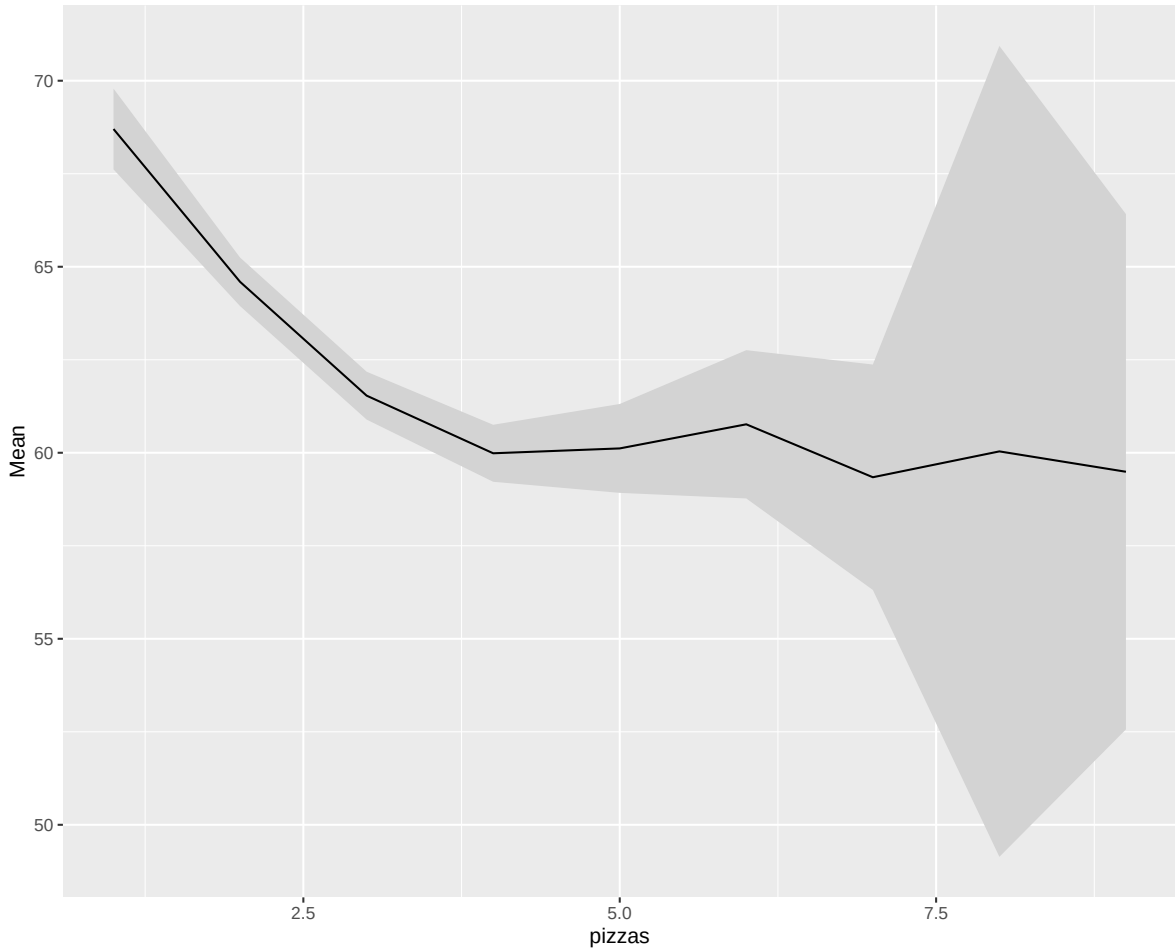


Confidence Intervals (III)

```
> #  Alternative to "errorbar": ribbon/confidence bands
> t2 <- as.data.frame(matrix(NA,ncol=3,nrow=9,
                dimnames=list(NULL,c("Mean","LCI","UCI"))))
> for(i in 1:9){t2[i,]<- c(mean(pizza$temperature[pizza$
    pizzas==i]),t.test(pizza$temperature[pizza$pizzas==i
    ])$conf.int)}
> t2$pizzas <- c(1:9)
> t2
      Mean       LCI       UCI pizzas
1 68.70181 67.61728 69.78633      1
2 64.59424 63.94210 65.24638      2
3 61.53537 60.89231 62.17843      3
4 59.98639 59.22094 60.75184      4
5 60.11682 58.92250 61.31113      5
6 60.76490 58.77232 62.75747      6
7 59.34240 56.31328 62.37152      7
8 60.03677 49.13937 70.93417      8
9 59.48968 52.56421 66.41514      9

> ggplot(t2, aes(x=pizzas,ymin=LCI,ymax=UCI,y=Mean)) +
    geom_ribbon(fill="lightgrey") + geom_line()
```

Confidence Intervals (IV)



Aesthetics (I)

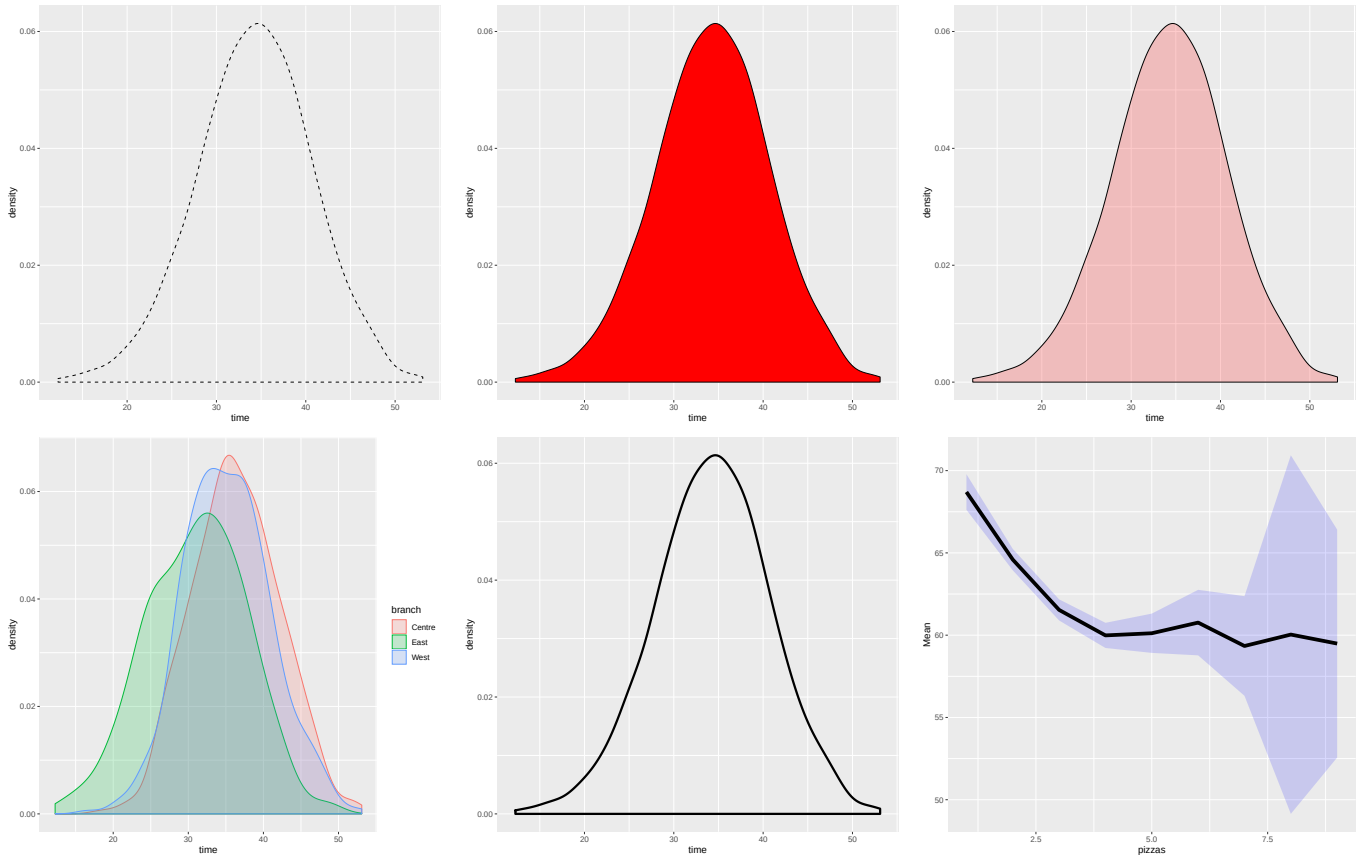
Aesthetics can be used to change

- colour of lines
- coour of areas
- line width
- transparency
- ...

Aesthetics (II)


```
> # linetype
> ggplot(pizza, aes(x=time)) + geom_density(linetype=2)
> # colour
> ggplot(pizza, aes(x=time)) + geom_density(fill="red")
> # transparency
> ggplot(pizza, aes(x=time)) + geom_density(fill="red",
    alpha=0.2)
> # stratified and transparent
> ggplot(pizza, aes(x=time, colour=branch, fill=branch)) +
    geom_density(alpha=0.2)
> # thickness
> ggplot(pizza, aes(x=time)) + geom_density(size=1.2)
> ggplot(t2, aes(x=pizzas,ymin=LCI,ymax=UCI,y=Mean)) +
>      geom_ribbon(fill="blue", alpha=0.15) +
>      geom_line(size=2)
```

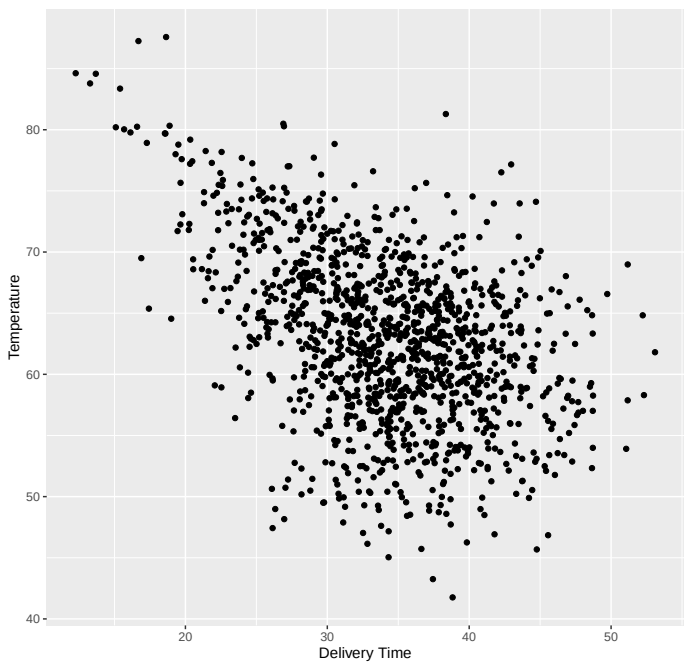
Aesthetics (III)



Scales (I)

Scales control the details of how data values are translated to visual properties.

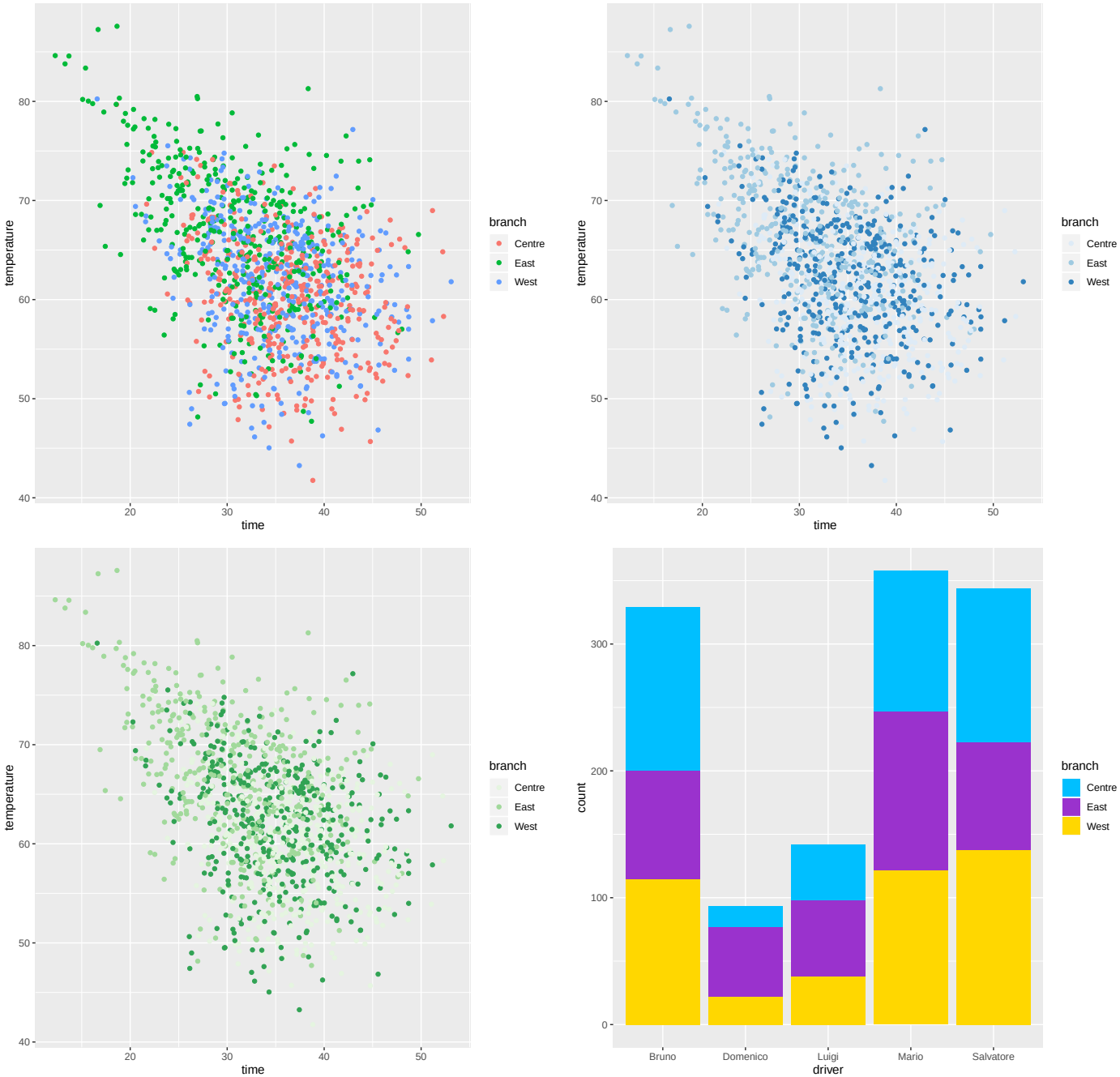
```
1 # Change axes names and properties (tick marks)
2 ggplot(pizza, aes(time, temperature))+ geom_point() +
3    scale_x_continuous("Delivery Time",
4                       breaks=seq(10,60,10)) +
5    scale_y_continuous("Temperature")
```



Scales (II)


```
> # Change colour scheme
> ggplot(pizza, aes(time, temperature, colour=branch))+
    geom_point()
> ggplot(pizza, aes(time, temperature, colour=branch))+
    geom_point()+ scale_colour_brewer()
> ggplot(pizza, aes(time, temperature, colour=branch))+
    geom_point()+ scale_colour_brewer(palette = "Greens")
> ggplot(pizza, aes(x=driver))+ geom_bar(aes(fill=branch))
    + scale_fill_manual(values = c("deepskyblue", "
    darkorchid3", "gold"))
```

Scales (III)

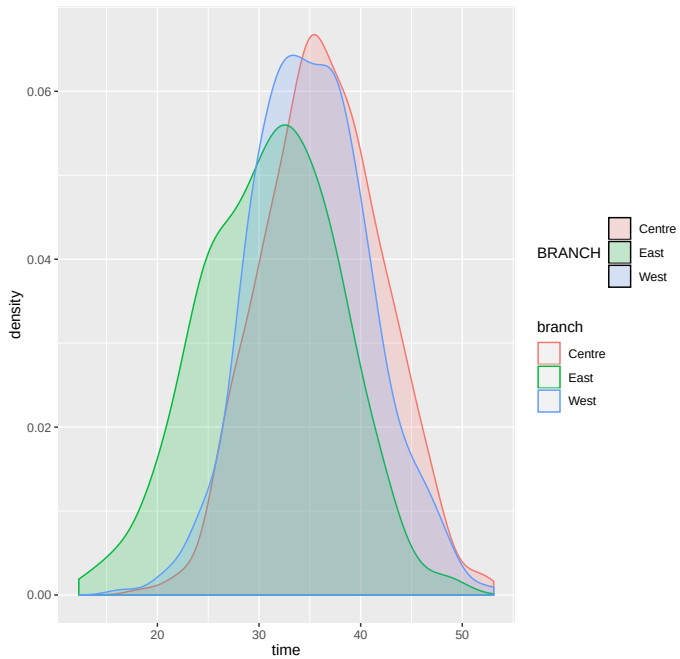


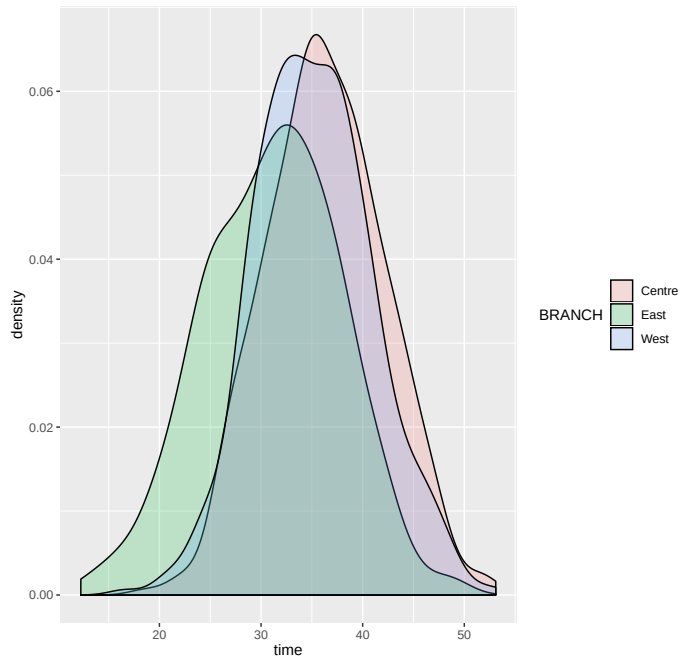
Guides (I)

The guides (the axes and legends) help readers interpreting plots. Guides are mostly controlled via the scale, but sometimes additional control over guide appearance is required.

```
> # Guides: override legends etc.
> ggplot(pizza, aes(x=time, colour=branch, fill=branch)) +
    geom_density(alpha=0.2) +
>     guides(fill = guide_legend(title = "BRANCH", title.
    position = "left"))
>
> # to avoid duplicate legends: move aesthetics to "lower
    level"
> ggplot(pizza, aes(x=time)) + geom_density(alpha=0.2, aes
    (fill=branch)) +
>     guides(fill = guide_legend(title = "BRANCH", title.
    position = "left"))
```

Guides (II)



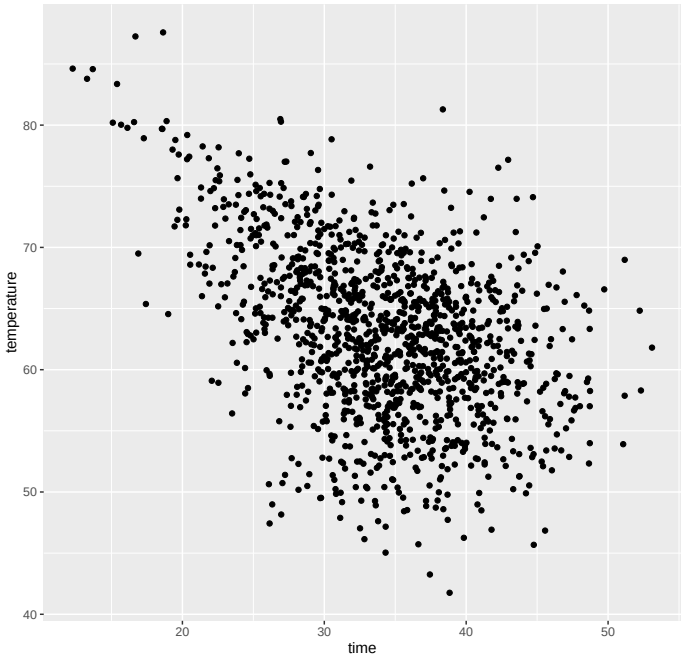


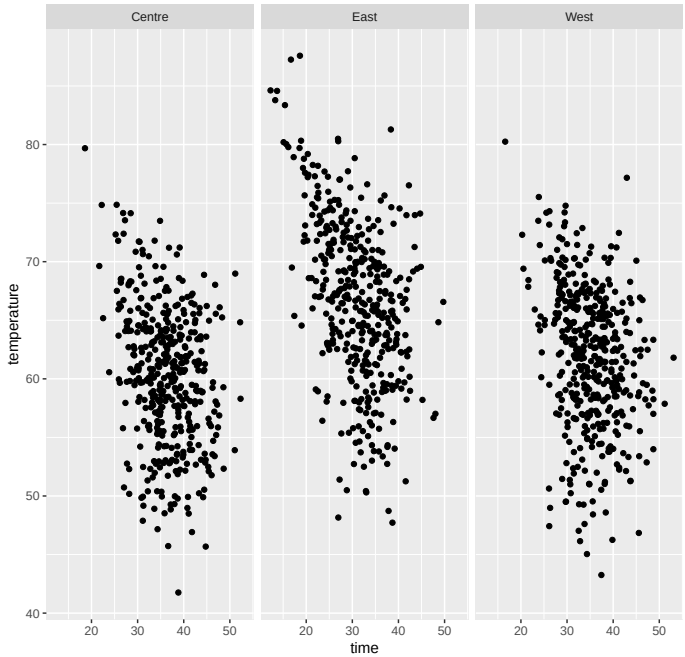
Facetting (I)

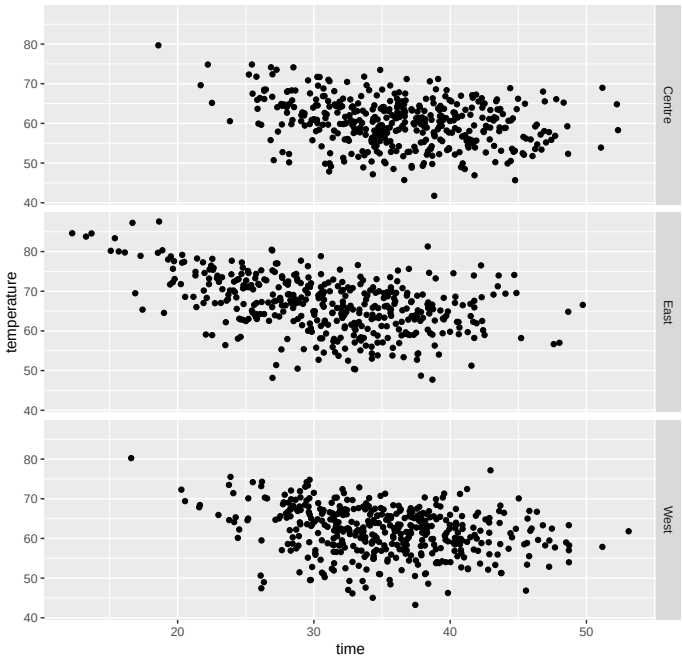
Facetting generates small multiples, each displaying a different subset of the data. Facets are an alternative to aesthetics for displaying additional discrete variables.

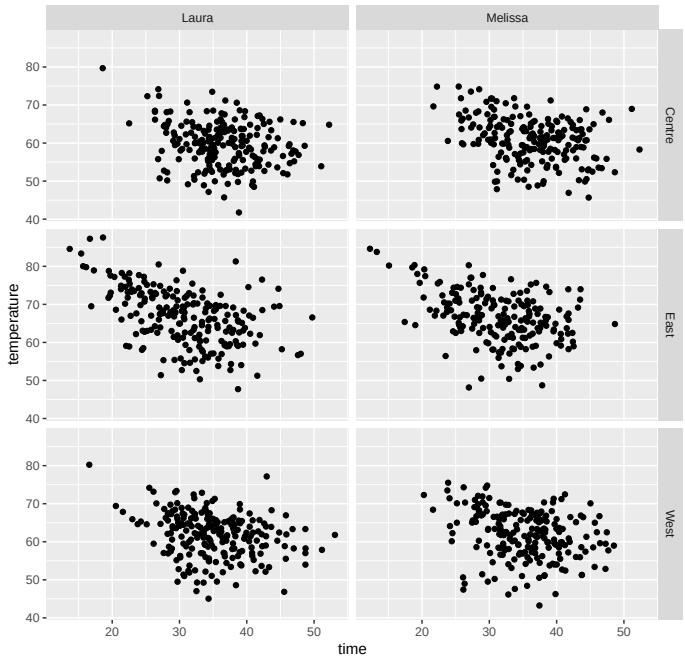
```
> ### Facetting: stratify by using different plots
> ggplot(pizza, aes(time, temperature))+ geom_point()
> ggplot(pizza, aes(time, temperature))+ geom_point() +
    facet_grid(. ~ branch)
> ggplot(pizza, aes(time, temperature))+ geom_point() +
    facet_grid(branch ~ .)
> ggplot(pizza, aes(time, temperature))+ geom_point() +
    facet_grid(branch ~ operator)
```

Facetting (II)





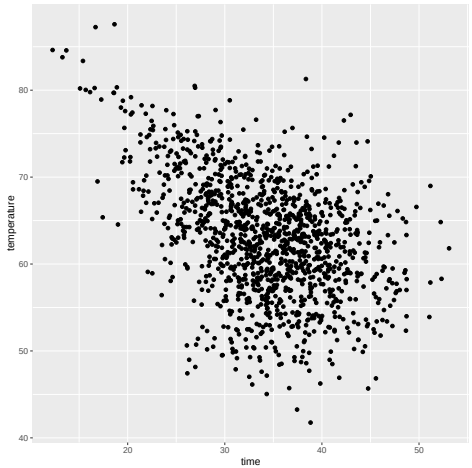


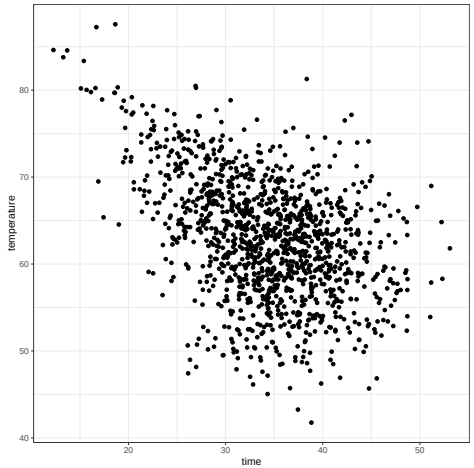


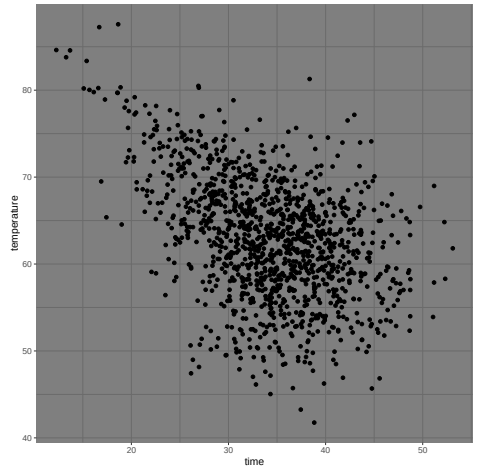
Themes (I)

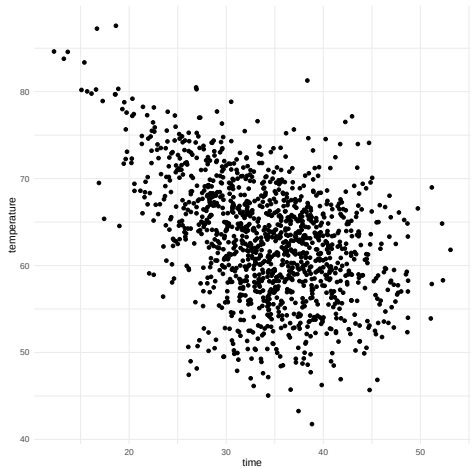

```
> # general themes
> ggplot(pizza, aes(time, temperature))+ geom_point()
> ggplot(pizza, aes(time, temperature))+ geom_point()+
    theme_bw()
> ggplot(pizza, aes(time, temperature))+ geom_point()+
    theme_dark()
> ggplot(pizza, aes(time, temperature))+ geom_point()+
    theme_minimal()
> ggplot(pizza, aes(time, temperature))+ geom_point()+
    theme_void()
```

Themes (II)





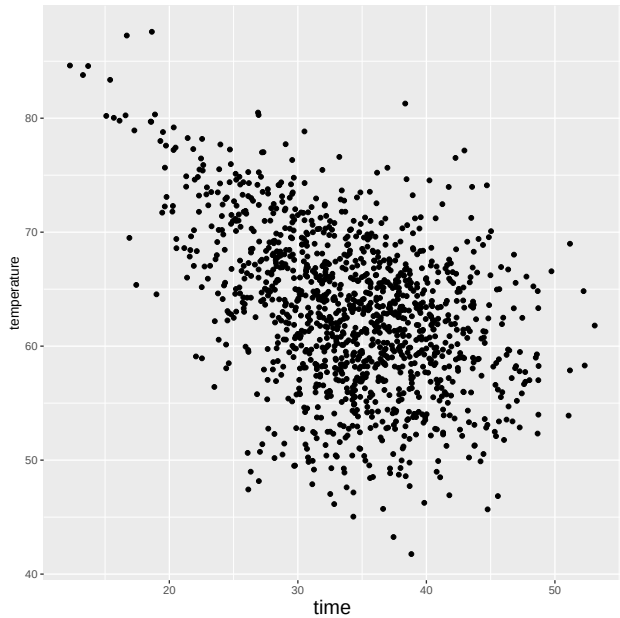


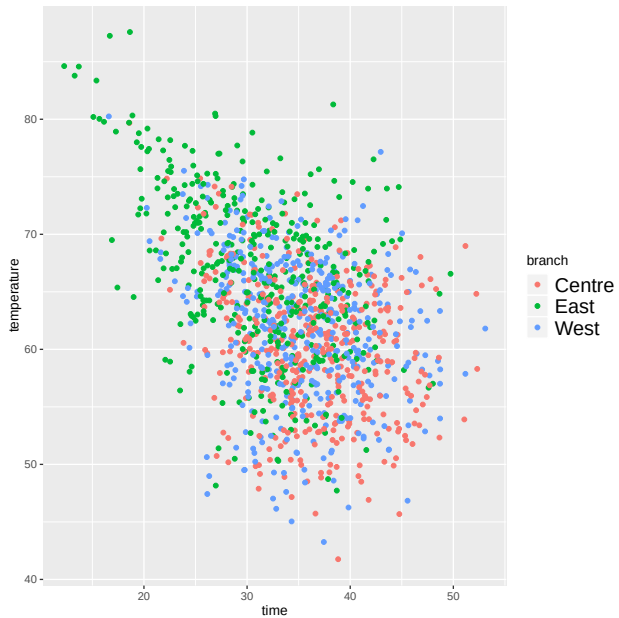


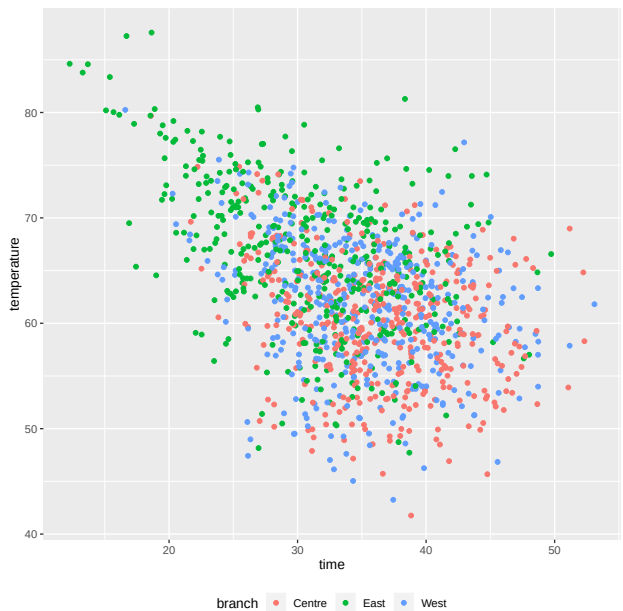
Themes (III)


```
> # change details of a theme
> ggplot(pizza, aes(time, temperature))+ geom_point()+
    theme(axis.title.x = element_text(size=16))
> ggplot(pizza, aes(time, temperature,colour=branch))+
    geom_point()+ theme(legend.text =  element_text(size
    =16))
> ggplot(pizza, aes(time, temperature,colour=branch))+
    geom_point()+ theme(legend.position =   "bottom")
```

Themes (IV)







How to combine these options?

```
ggplot(pizza, aes(x=driver, y=temperature, fill=branch)) +
    geom_boxplot(alpha=0.2) +
      facet_grid(operator ~ .) +
      theme_bw() +
      scale_fill_manual(values = c("deepskyblue","
          darkorchid3","gold")) +
      scale_x_discrete("Scooter Driver") +
      scale_y_continuous("Temperature", breaks=seq
          (30,90,15)) +
      guides(fill = guide_legend(keywidth = 2, keyheight
          = 2, title="BRANCH")) +
      ggtitle("Temperature, stratified by branch and
          driver")  +
      theme(axis.title.x = element_text(size=13), axis.
          text.x = element_text(size=13),axis.title.y =
          element_text(size=13, angle = 90),
           axis.text.y = element_text(size=13), legend.
              text =  element_text(size=13), legend.
              title = element_text(size=13, face = "
              bold", hjust = 0),legend.position =   "
              right") +
      annotate("text", x = 5, y = 85, label = "May 2014")
```

A nice graph