

Introduktion til datavisualisering i R med

ggplot2

Webinar 23-11-2022 - Tue Hellstern



Tue Hellstern

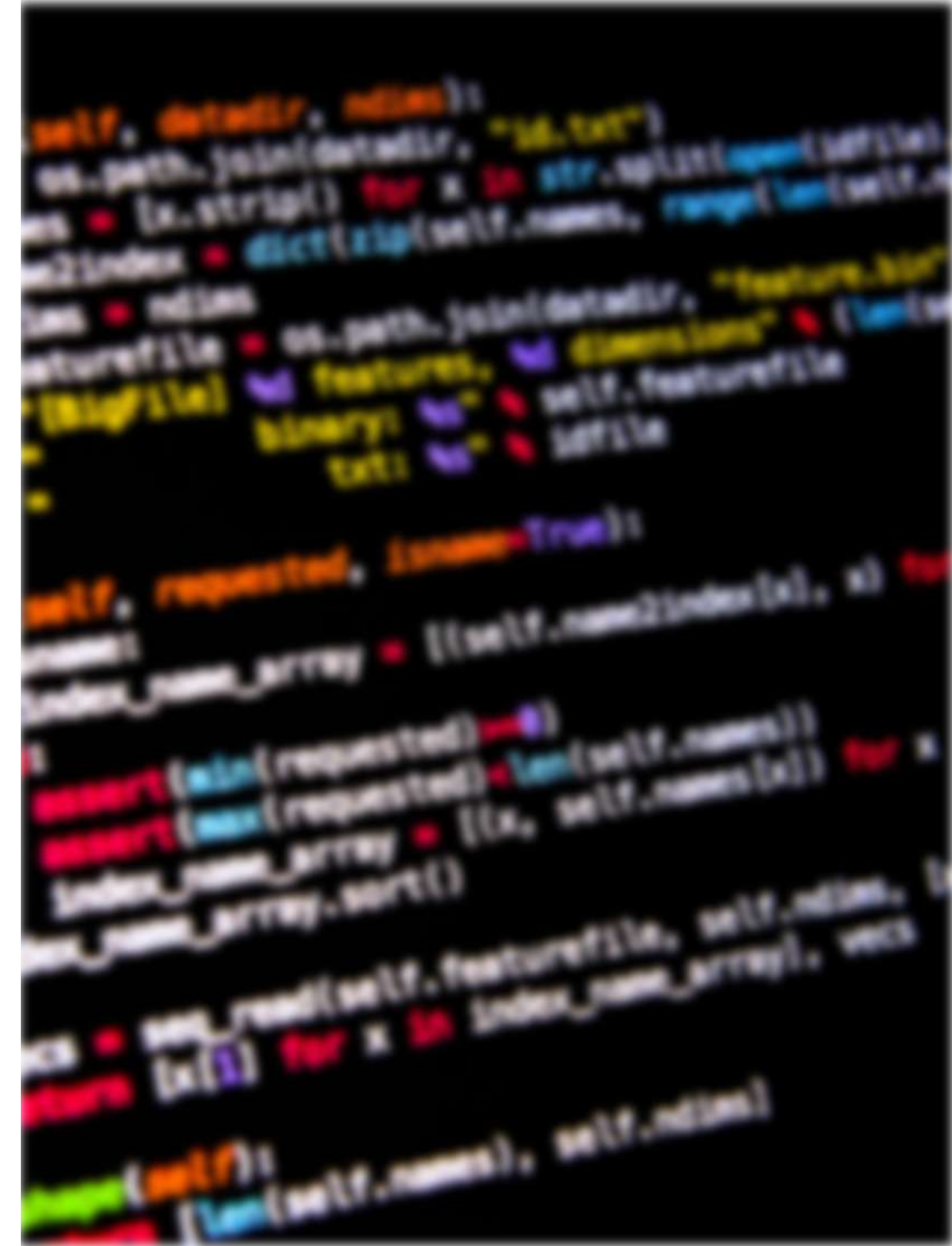
- Uddannet fra ITU
- Ekstern lektor på DTU
- Lektor på KEA
- Censor på Datamatiker, Akademiuddannelserne, PBA, Økonomi & IT og andre uddannelser

Konsulent

- Selvstændig siden 1995
- Programmering, Databaser, Integration, Projekt styring
- Typisk større virksomheder

Agenda

- Introduktion til RStudio
 - Lokalt/Online
- Introduktion til ggplot2
 - Grammar of Graphics
- Packages
- Opret et plot ud fra *Grammar of Graphics*
- Gem plot

A blurred image of R code, likely from a script or console. The code includes file path manipulations using `os.path`, dictionary creation with `dict`, and data processing logic involving `self.names`, `self.featurefile`, and `self.names`. It also shows a loop over `self.names` and a `return` statement.

```
self, detadir, ndim):  
    os.path.join(detadir, "id.txt")  
    ns = [x.strip() for x in str.split(open(detadir)  
    name2index = dict(zip(self.names, range(len(self.names))  
    ns = ndim  
    featurefile = os.path.join(detadir, "feature.bin")  
    [id] = features, dimensions = len(self.names)  
    binary = self.featurefile  
    text = idfile  
  
self, requested, isname=True):  
    names = self.names  
    index_name_array = [(self.name2index[x], x) for  
    x in requested]  
    index_name_array = [(self.name2index[x], x) for  
    x in requested]  
    index_name_array = [(x, self.names[x]) for x  
    in index_name_array.sort()  
  
    ns = seq_read(self.featurefile, self.names, len(index_name_array), vecs)  
    return [x[1] for x in index_name_array], vecs  
  
def __getitem__(self, name):  
    return self.names[name], self.names
```



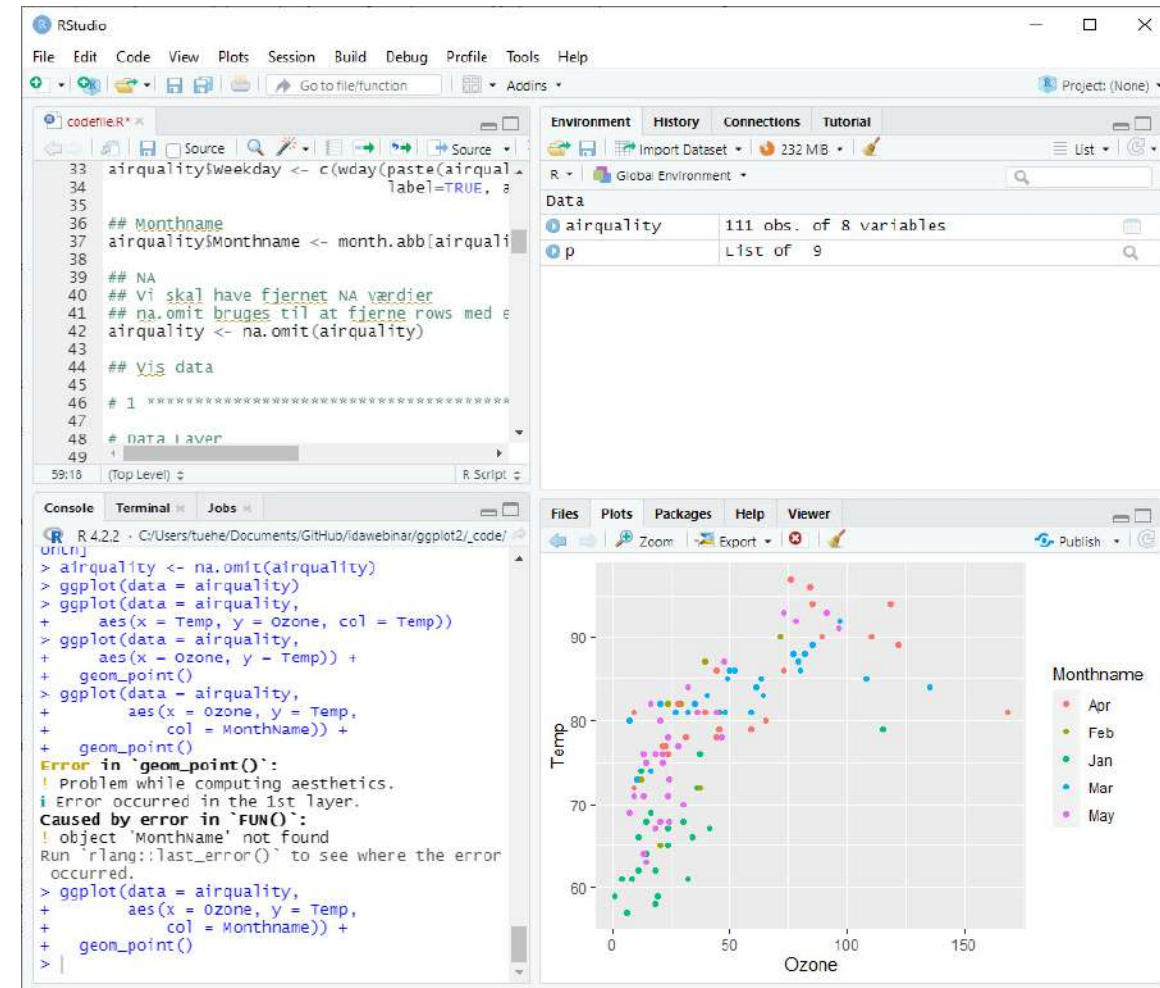
Hvem er I?

Spørgsmål

RStudio - IDE

- Lokalt
- Online
- <https://posit.co/products/cloud/cloud>

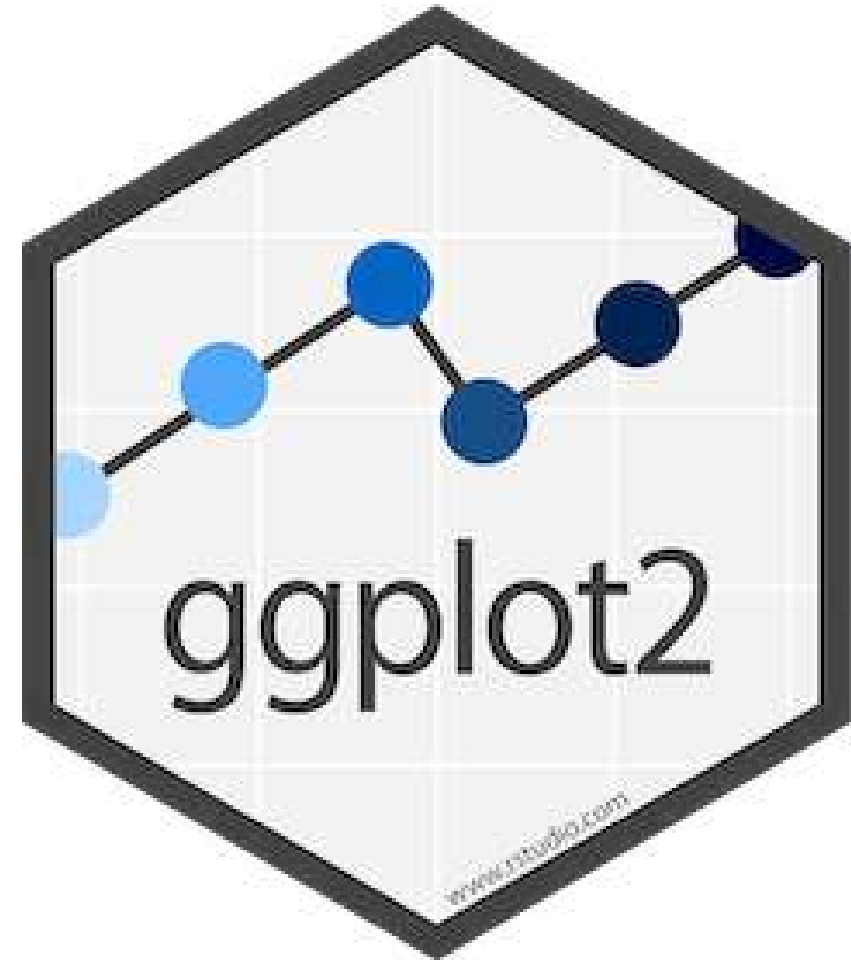
Ps. Nyt navn - "RStudio is now Posit"



ggplot2

ggplot2 er et system til oprettelse af plot/diagrammer, baseret på **The Grammar of Graphics**.

Du leverer dataene, fortæller **ggplot2**, hvordan variabler skal vises, hvilke typer plot du vil vise og **ggplot2** tager sig af detaljerne.



Pakker

Vi skal bruge nogle pakker for at tilpasse data og oprette plot.

```
install.packages('tidyverse')  
install.packages('lubridate')
```

```
library(ggplot2)  
library(lubridate)
```

Data

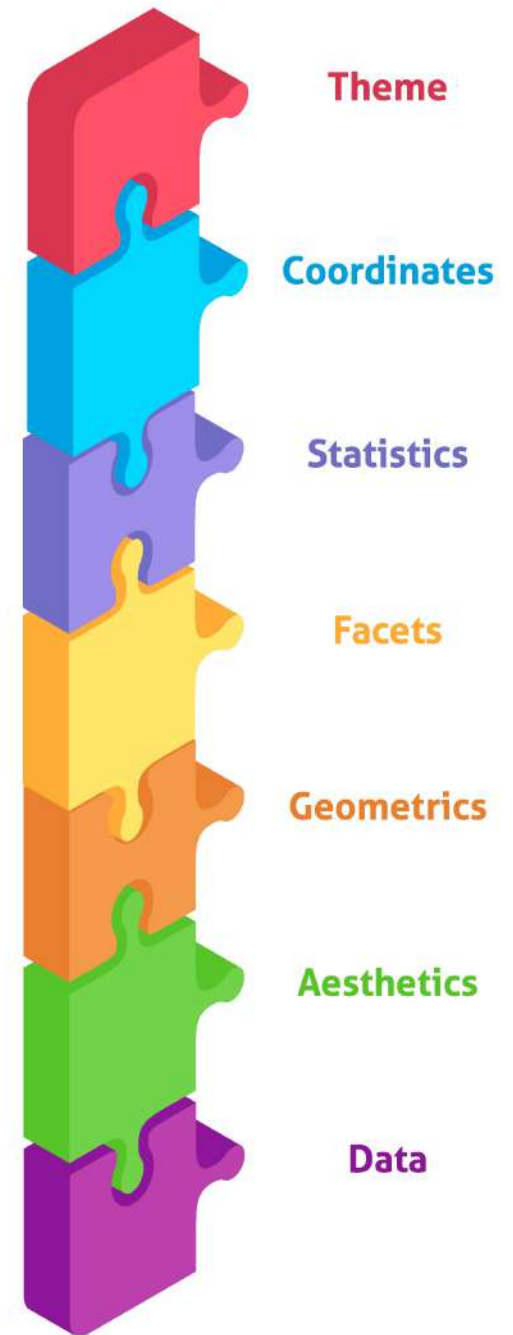
Til dette plot er det et af de *indbyggede* datasæt vi brugere - **airquality**

Dette dataset viser daglige målinger af luftkvaliteten i New York. I perioden maj til september 1973.

	ozone	solar.R	wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10

Grammar of Graphics

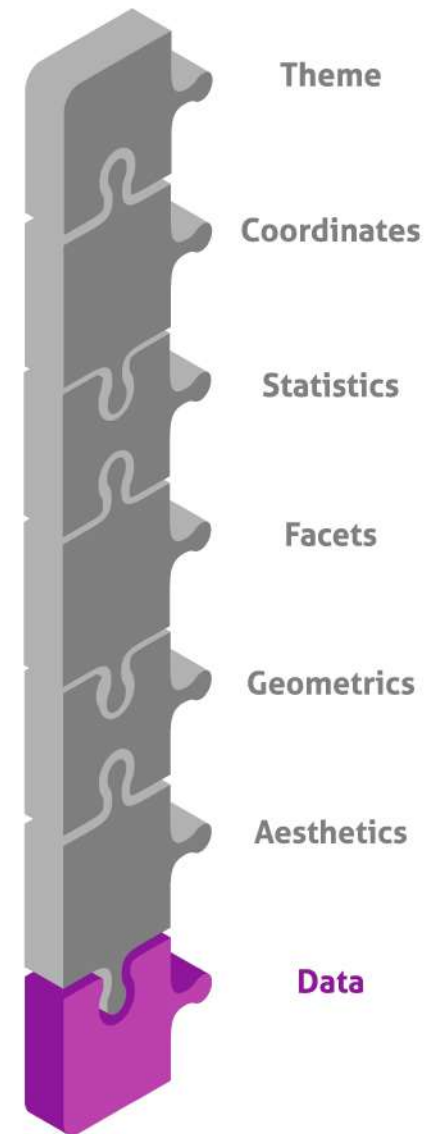
```
ggplot (data = <DATA>) +  
  <GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),  
    stat = <STAT> , position = <POSITION>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>
```



Data laget

Du indlæser datasættet **airquality** med denne R kode:

```
ggplot(data = airquality)
```



Data tilpasning

Typisk kan data ikke bruges i den form du importere dem, hvilket også gælder her. Vi skal have tilpasset følgende:

- Fjern tomme værdier (*NA*)
- Tilføj en kolonne med månedsnavn (*Monthname*)
- Tilføj en kolonne med ugedag (*Weekday*)
- Konverter *Month* til en *Factor*

Fjern NA værdier

I datasættet er der en del NA værdier, den skal vi have fjernet. Det kan du nemt gøre med denne R kommando

```
airquality <- na.omit(airquality)
```

na.omit fjerner rows der indeholder **en** eller **flere NA** værdier

Tilføj månedsnavn

```
airquality$Monthname <- month.abb[airquality$Month]
```

Tilføj ugedag

Jeg vil gerne have mulighed for at gruppere efter ugedag. Problemet er at datasættet "*kun*" indeholder **Day** og **Month**. Året kender vi - **1973**.

Det kan løses med lidt R programmering

```
airquality$Weekday <- c(wday(paste(airquality$Day, airquality$Month, '1973', sep='-'),  
                           label=TRUE, abbr=FALSE))
```


Datatyper

Du kan bruge denne R kommando til at se hvilke data typer dine data har:

```
str(airquality)
```

```
> str(airquality)
'data.frame': 153 obs. of 6 variables:
 $ Ozone   : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ wind    : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp    : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month   : Factor w/ 5 levels "5","6","7","8",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ Day     : int  1 2 3 4 5 6 7 8 9 10 ...
```

Factor - Month

Vi vil gerne have konverteret kolonnen **Month** til en *Factor*

```
airquality$Month <- factor(airquality$Month)
```

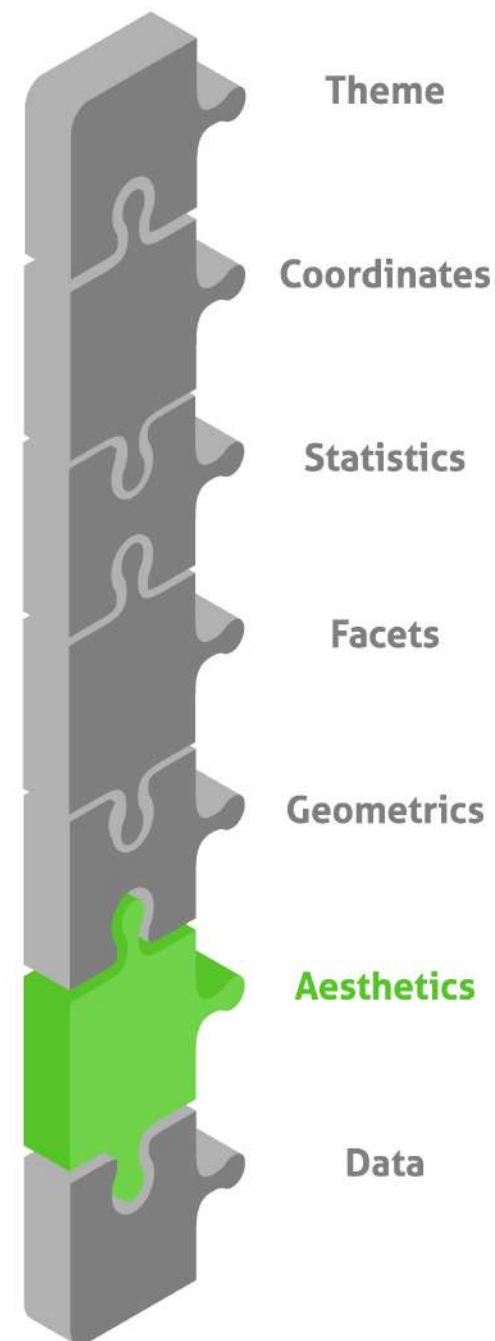
Factor er *værdier*, du kan bruge til at kategorisere dine dataene og gemme dem som niveauer. **Factor** er nyttige i de kolonner, som har et begrænset antal unikke værdier. For eksempel "*Mand*", "*Kvinde*" og *True*, *False* osv. **Factor** bruges meget i dataanalyse til statistisk modellering. En **Factor** kan være både *strenge* og *heltal*.

Aesthetics

Her betyder **Aesthetic** "*Noget du kan se*". Det er godt nok kun baggrund osv. du kan se **ikke** "*data*".

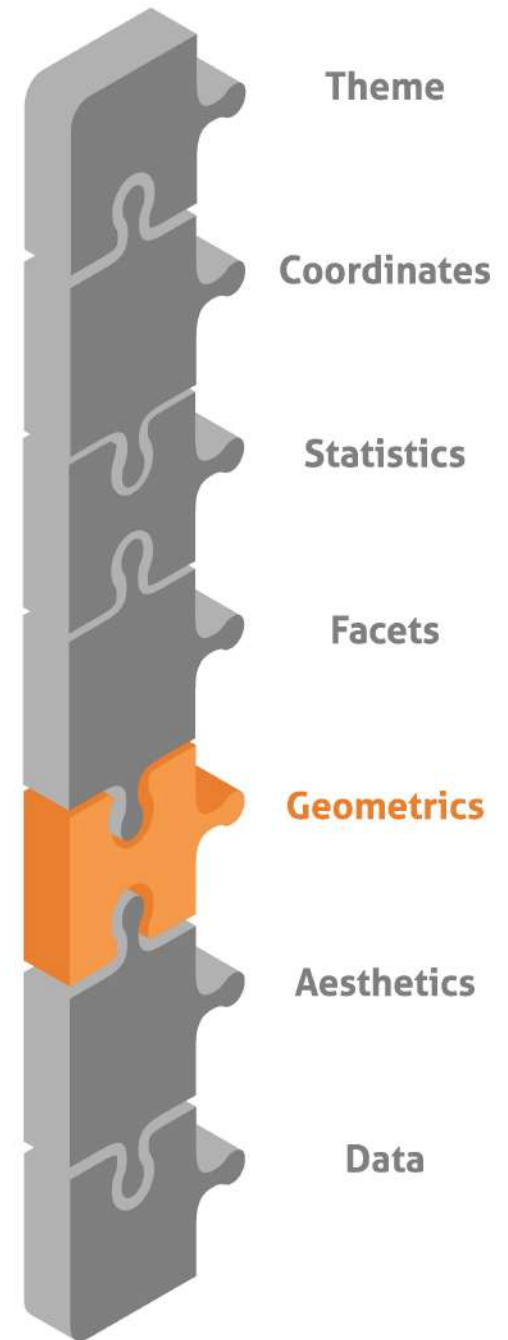
Det er fundamentet for dit plot.

```
ggplot(data = airquality,  
       aes(x = Temp, y = Ozone, col = Temp))
```

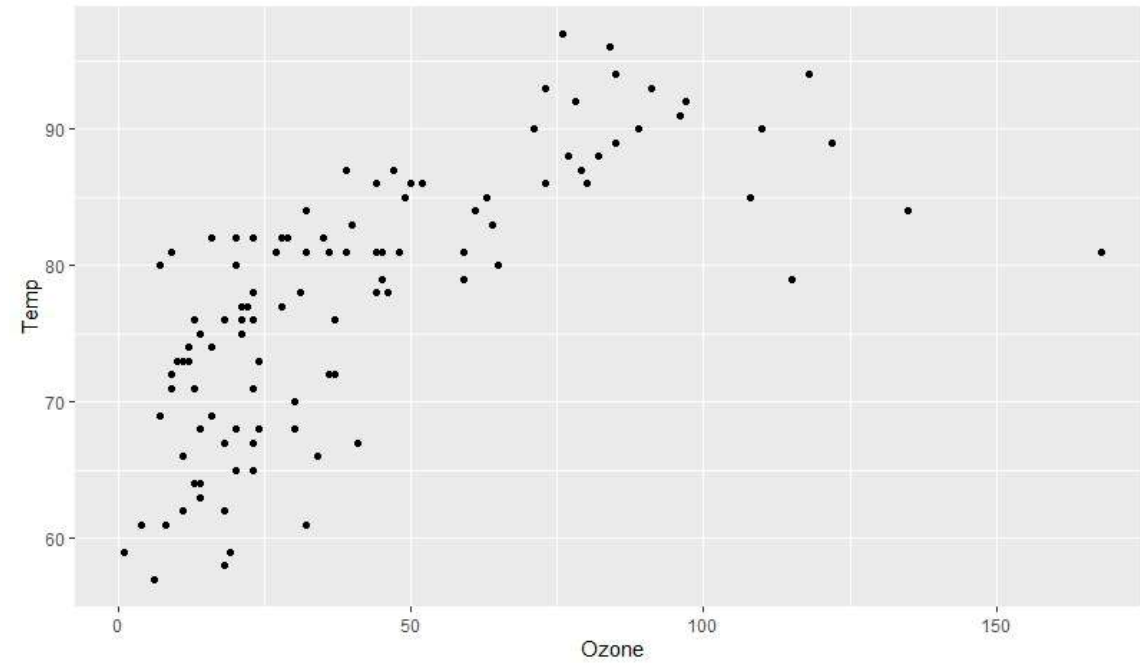


Geometris

Nu sker der noget, her vælger du hvilken plot "type" du vil vise.

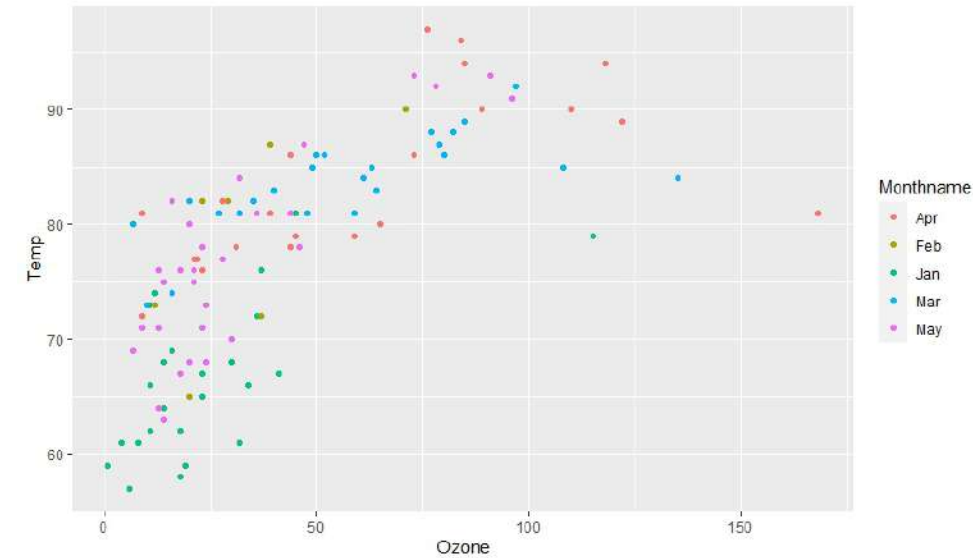


```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp)) +  
  geom_point()
```



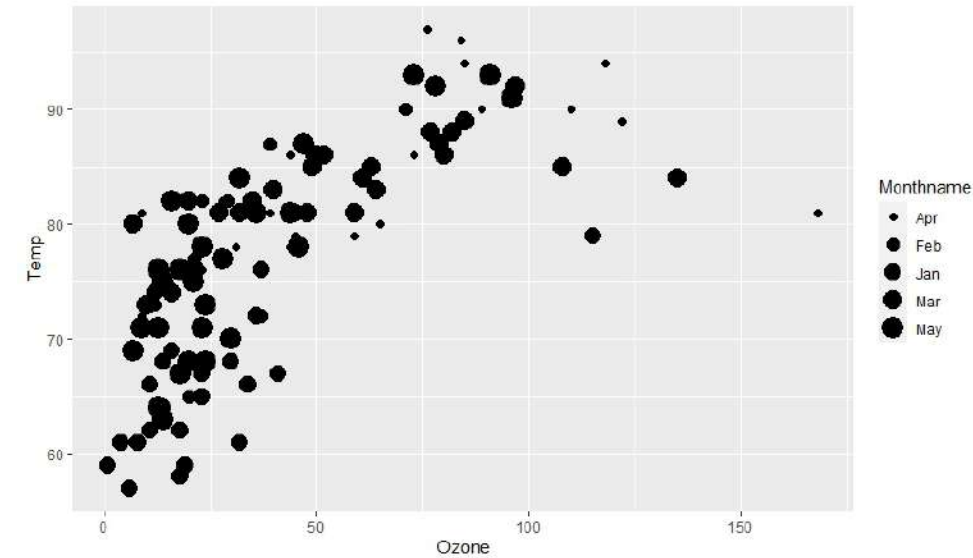
Tilføj Color til Geometric layer

```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           col = Month)) +  
  geom_point()
```



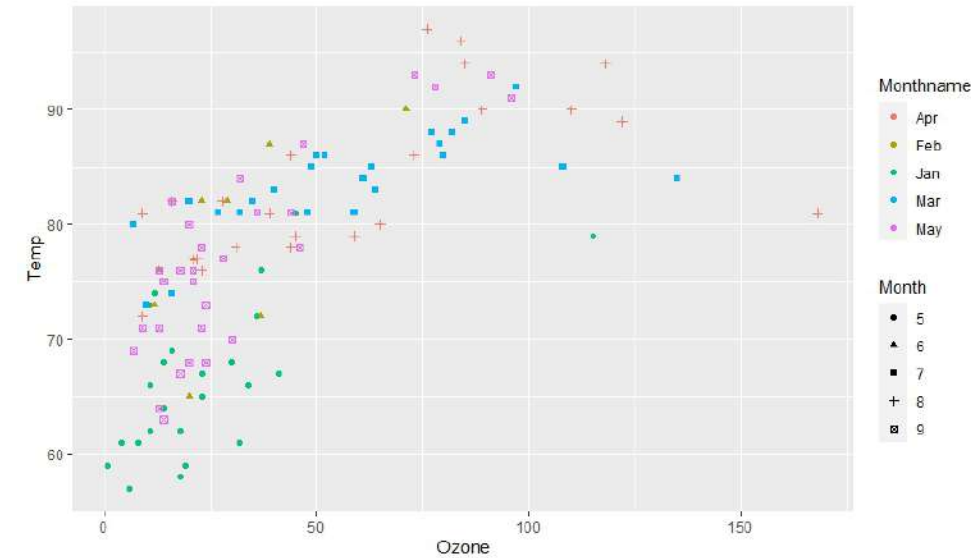
Tilføj Size til Geometric layer

```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           size = Month)) +  
  geom_point()
```



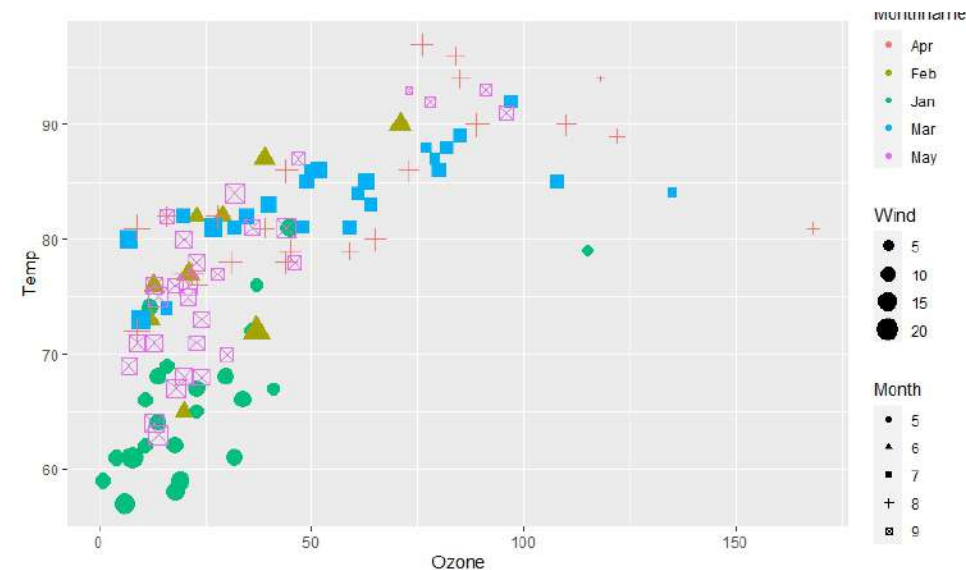
Tilføj Shape og Color til Geometric layer

```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           col = Month,  
           shape = Month)) +  
  geom_point()
```



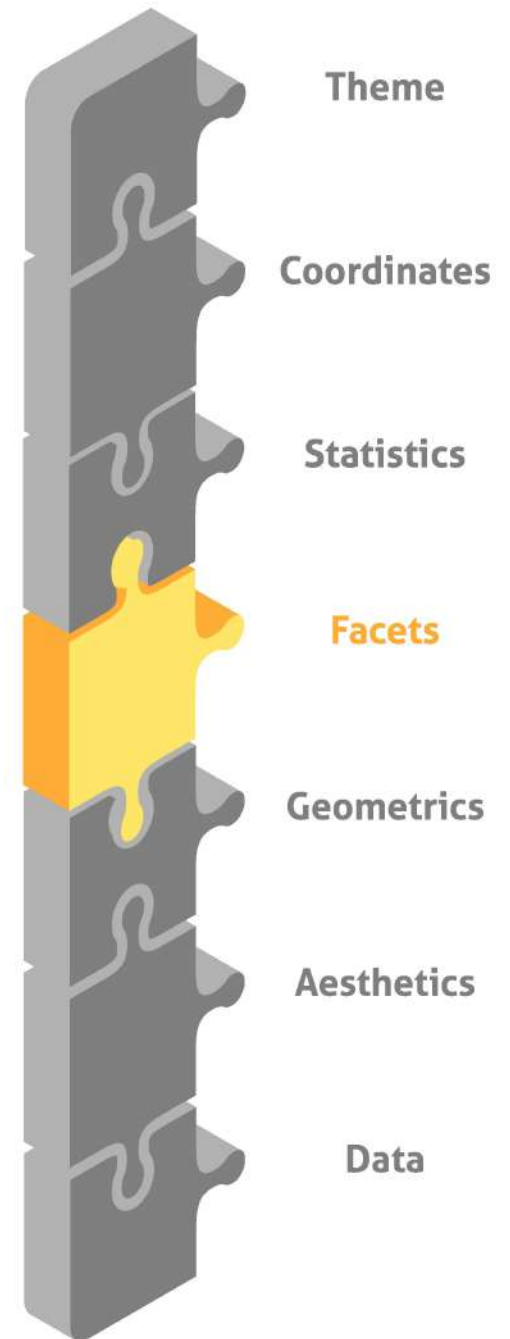
Shape, Color og Size til Geometric layer

```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           col = Month,  
           size = Wind,  
           shape = Month)) +  
  geom_point()
```



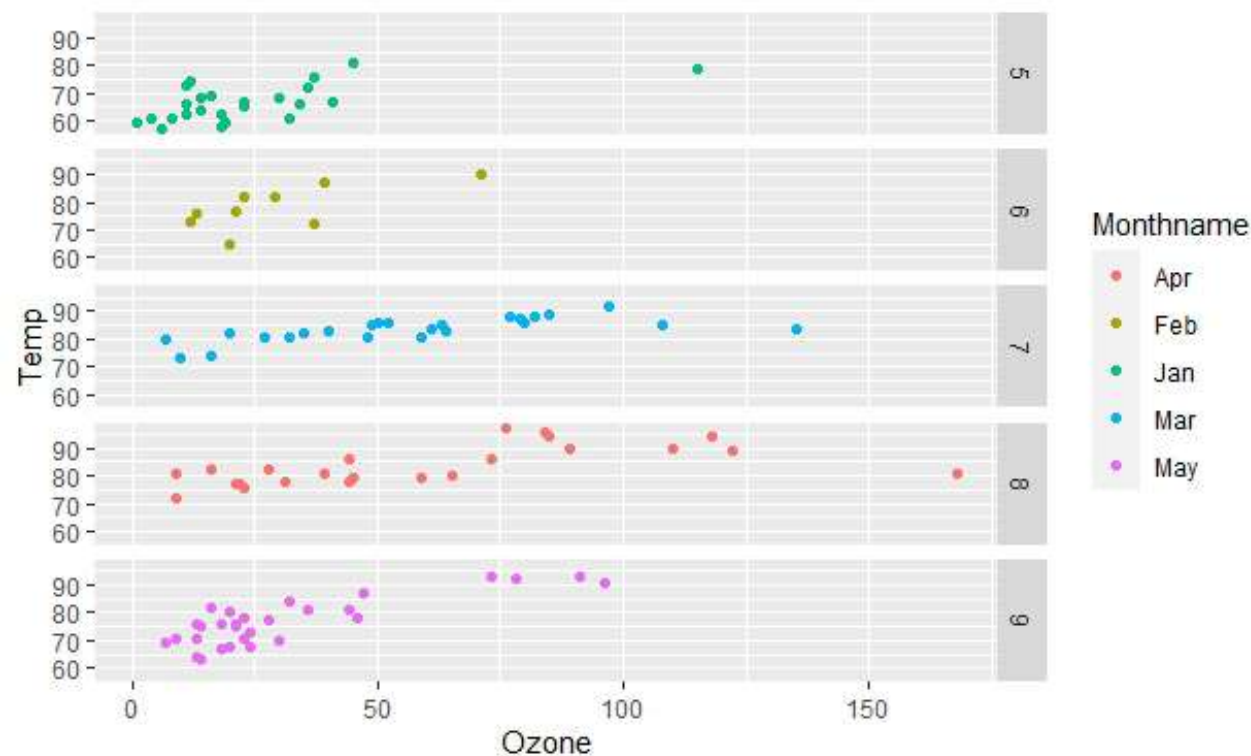
Facets

```
p <- ggplot(data = airquality,  
            aes(x = Ozone, y = Temp,  
                col = Monthname)) +  
  geom_point()
```



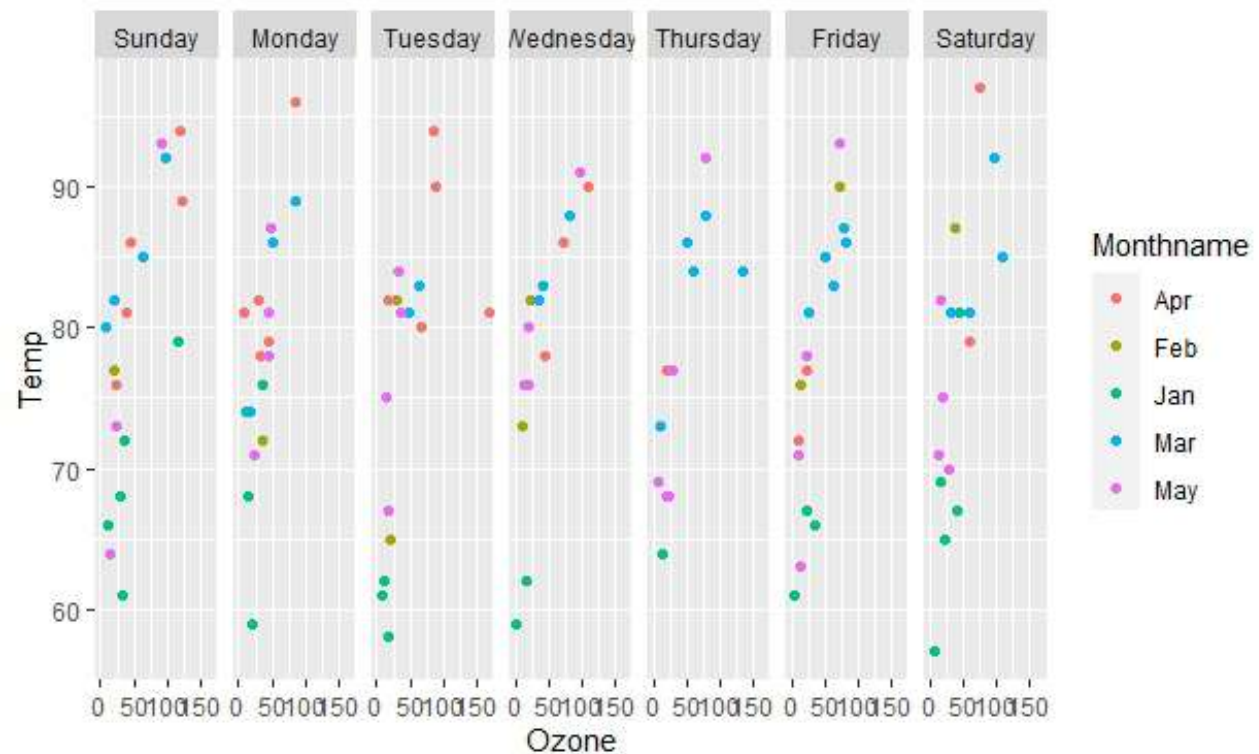
Opdel i rækker efter Måned (Month)

```
p + facet_grid(Month ~ .)
```

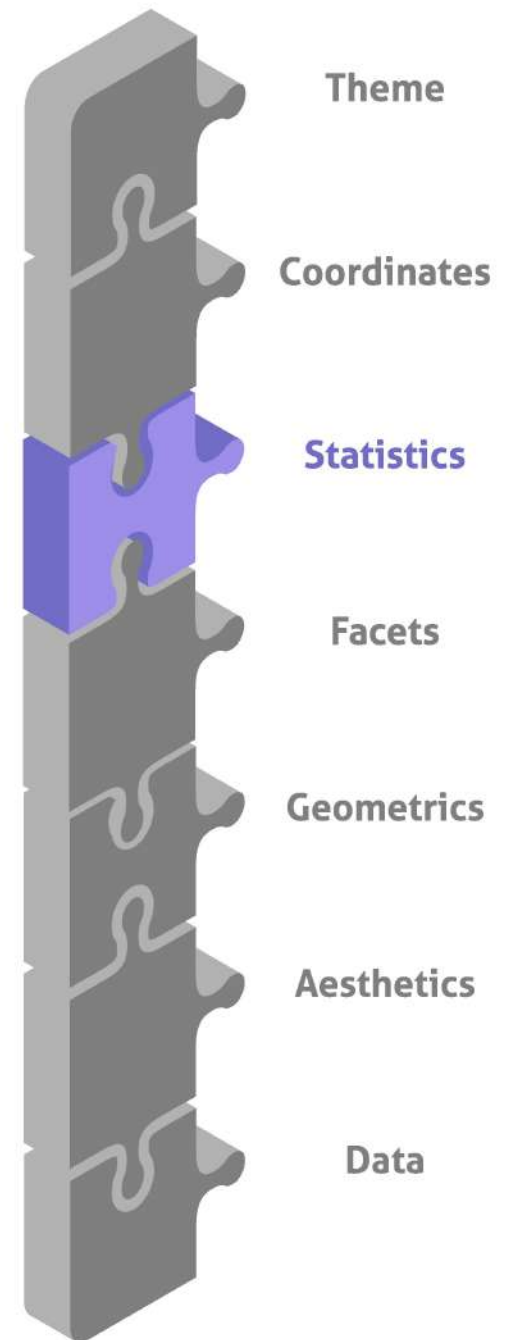


Opdel i koloner efter Ugedag (Weekday)

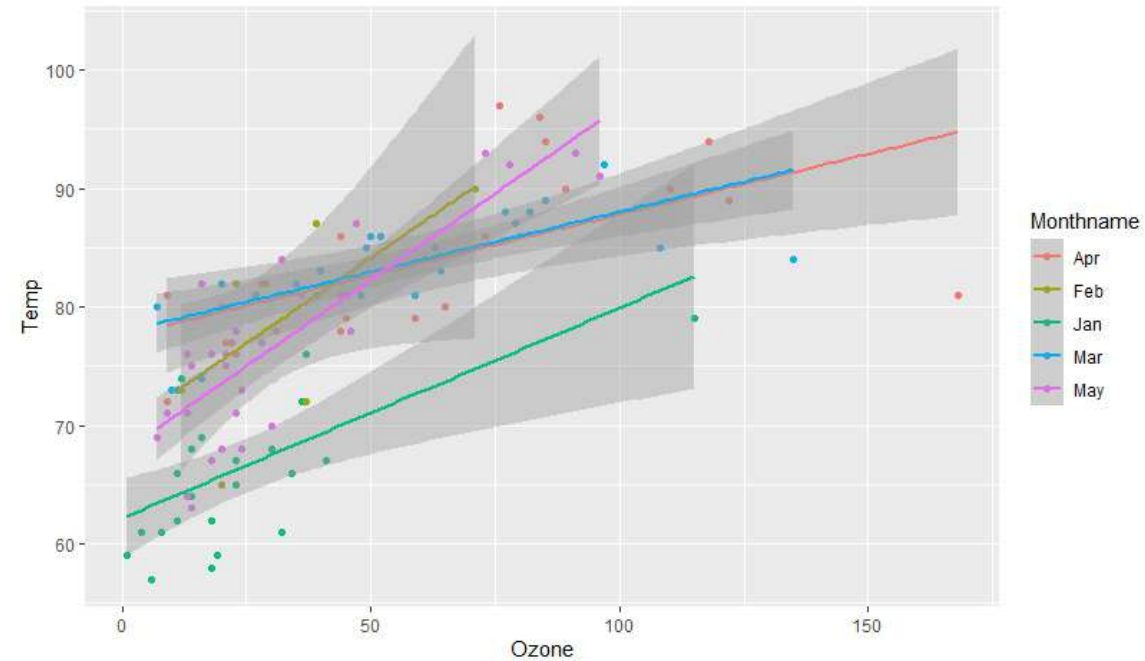
```
p + facet_grid(. ~ Weekday)
```



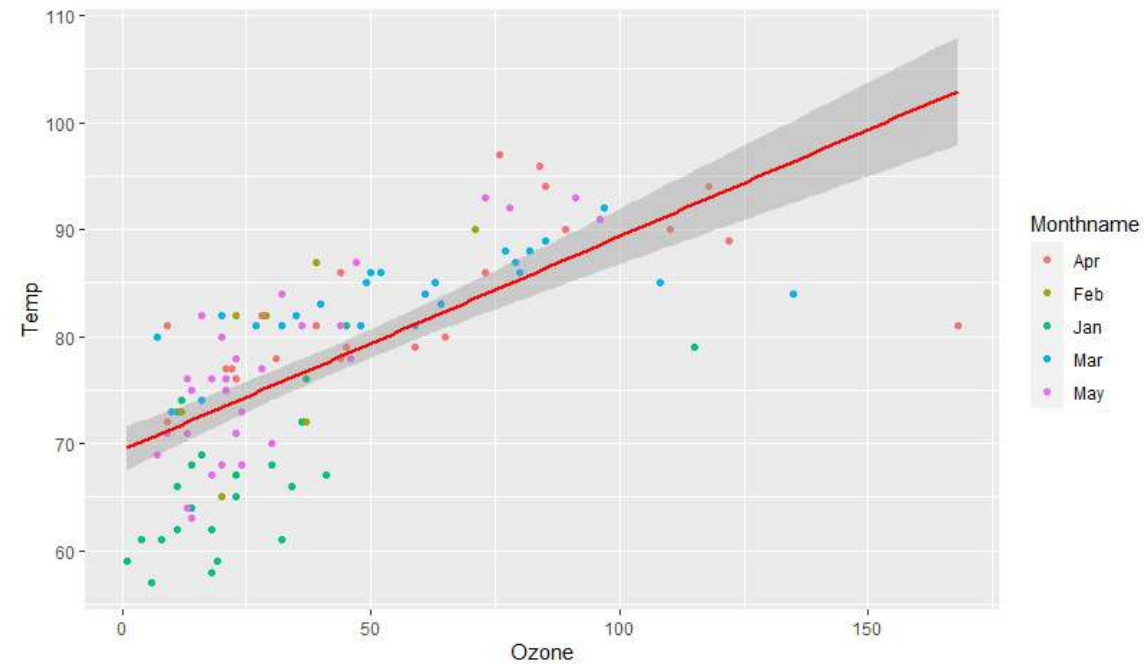
Statistics



```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           col = Monthname)) +  
  geom_point() +  
  geom_smooth(se = T, method = lm)
```

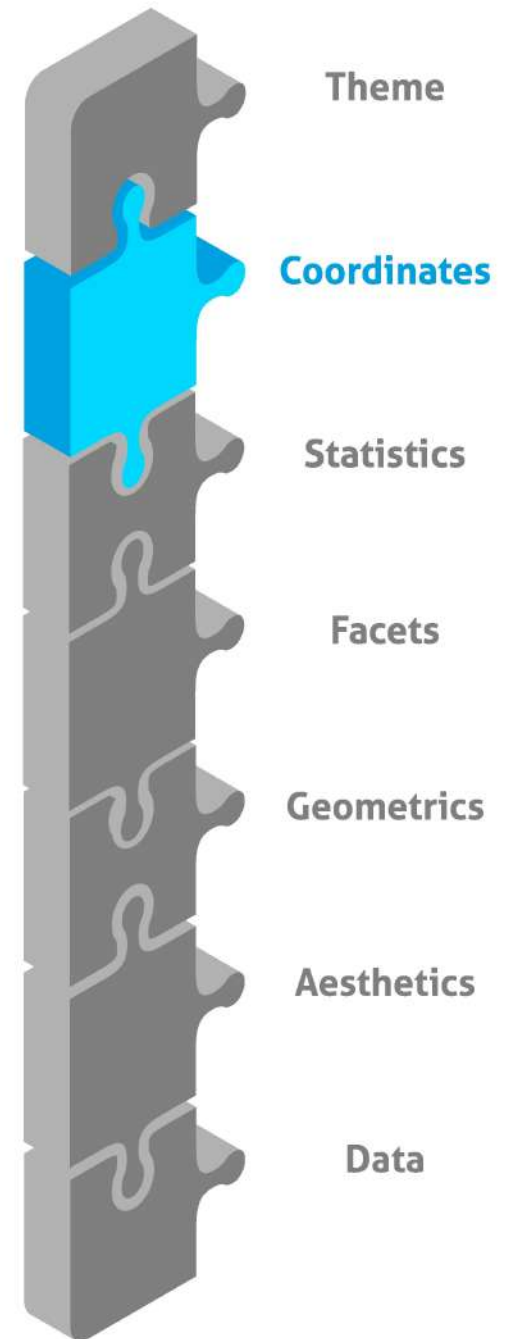


```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           col = Monthname)) +  
  geom_point() +  
  geom_smooth(se = T, method = lm, col = 'red')
```



Coordinates

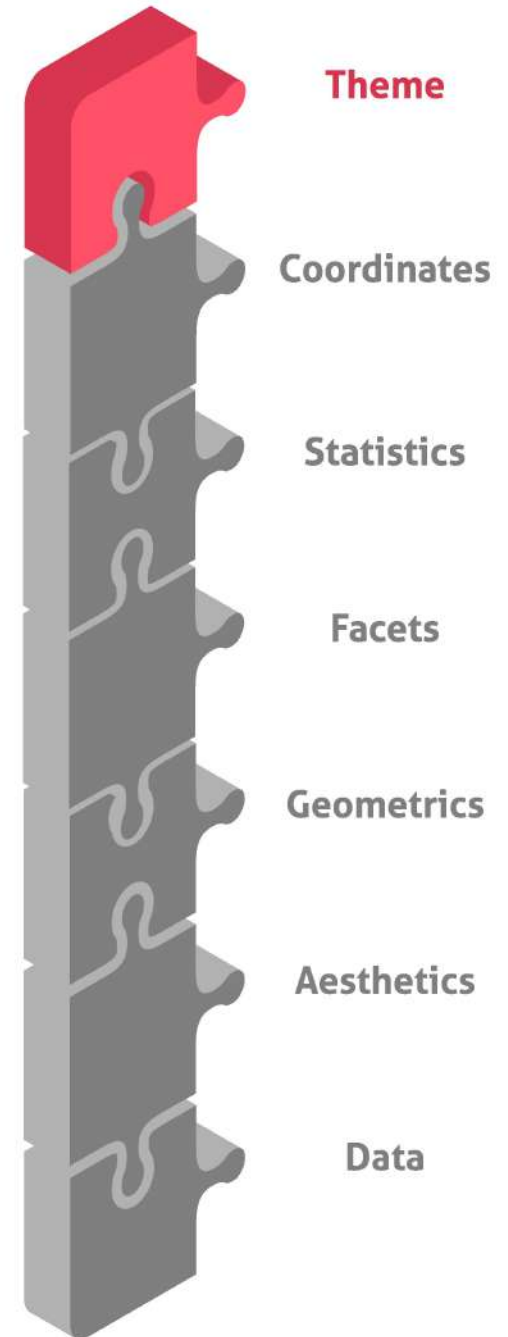
Normalt vil du bruge standard
koordinatsystemet - *The Cartesian system*



Theme

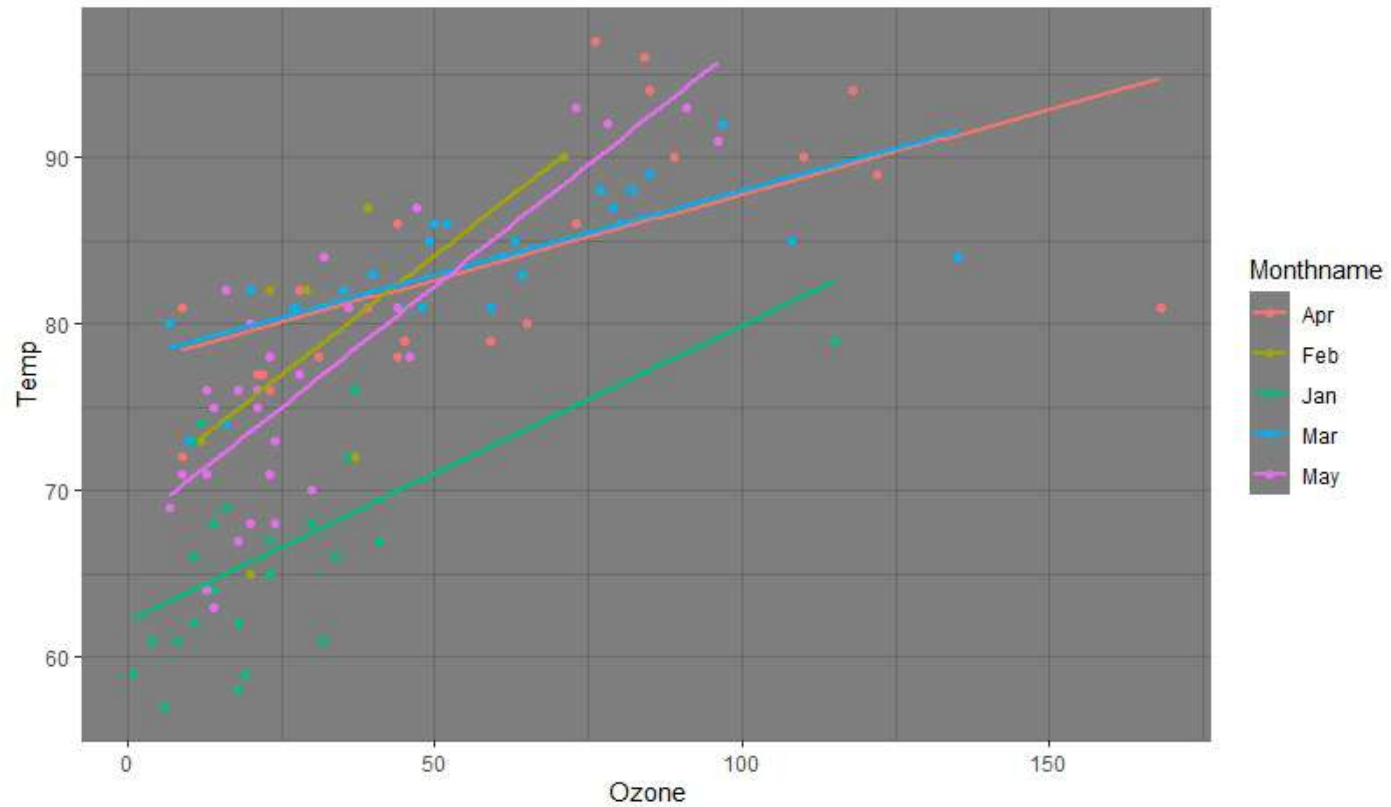
ggplot2 kommer med 8 indbyggede *Themes*:

- `theme_gray()`
- `theme_bw()`
- `theme_linedraw()`
- `theme_light()`
- `theme_dark()`
- `theme_minimal()`
- `theme_classic()`
- `theme_void()`



theme_dark()

```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           col = Monthname)) +  
  geom_point() +  
  geom_smooth(se = F, method = lm) +  
  theme_dark()
```

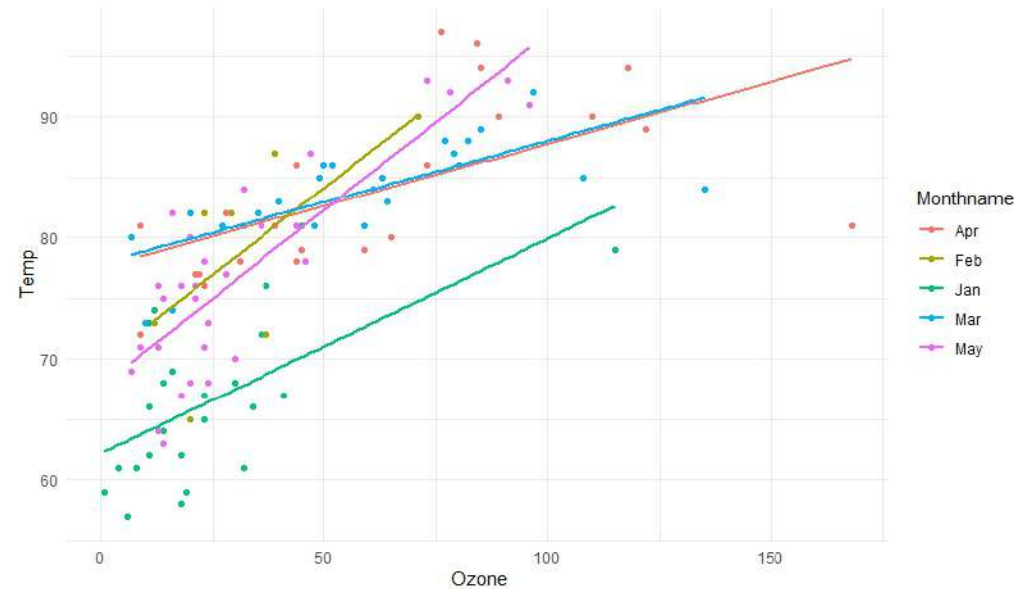


Default Theme

Default er `theme_gray()`, så hvis du vil bruge det behøver du ikke at specificere noget.

Du kan ændre default *Theme* på denne måde - her til `theme_minimal()`

```
theme_set(theme_minimal())
```



Theme pakker

Ud over de indbyggede *Themes* er det muligt at anvende forskellige 3-parts *Themes*:

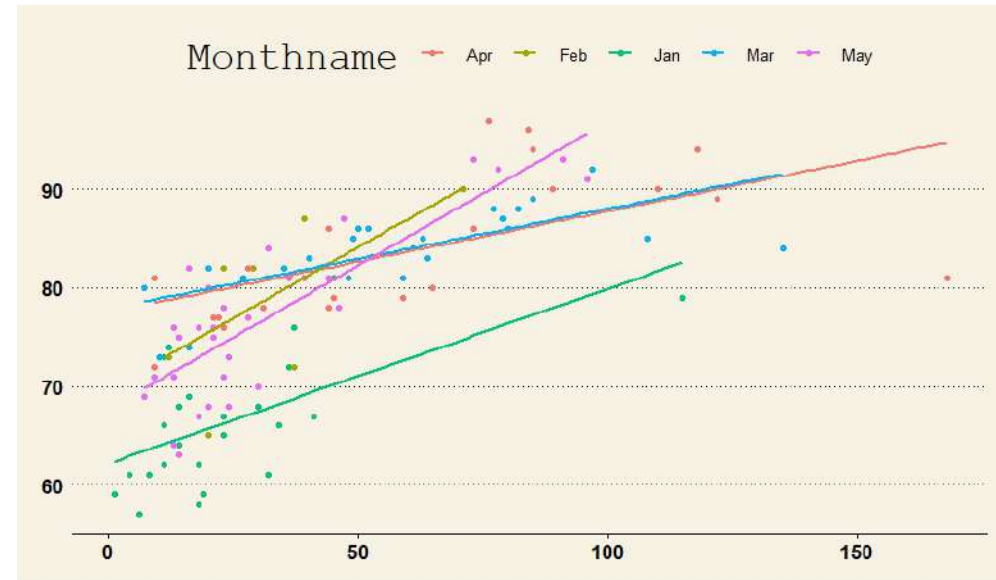
- ggthemes
- hrbrthemes
- ggthemr
- ggtech
- ggdark

Du skal installere den/de Themes pakker du vil anvende - her **ggthemes**

```
install.packages("ggthemes")  
library(ggthemes)
```

Wall Street Journal theme

```
ggplot(data = airquality,  
       aes(x = Ozone, y = Temp,  
           col = Monthname)) +  
  geom_point() +  
  geom_smooth(se = F, method = lm) +  
  theme_wsj()
```



Gem plot

Du kan gemme det sidste plot du har oprettet med.
plot.png bliver gemt i dit **Working Directory**

```
ggsave('plot.png', width = 5, height = 5)
```

ggsave understøtter følgende filetyper:

- jpeg
- png
- tiff
- eps/ps
- pdf
- bmp
- svg
- wmf - *kun Windows*



Links

- ggplot2.tidyverse.org
- ggplot2-book.org
- www.rdocumentation.org/packages/ggplot2
- [R Graphics Cookbook](#)
- [Cheat Sheet](#)
- posit.co