ggplot2 grafiskā sistēma

Didzis Elferts

Pamatojums

- ggplot2 pakete atšķiras no citām grafiku veidošanas paketēm ar to, ka tās pamatā ir noteikta ”gramatika” (Wilkinson, 2005. Grammar of graphics)
- Sistēma sastāv no vairākām atseviškām komponentēm, kuras var dažādā veidā kombinēt
- Ir iespēja veidot jaunus grafiku veidus, kas parāda Jums aktuālo problēmu

Papildus informācija

Oficiālā dokumentācija:

http://docs.ggplot2.org/current/

R attēlu ”recepšu grāmata”:

http://www.cookbook-r.com/Graphs/

Jautājumi un atbildes:

http://stackoverflow.com/questions/tagged/ggplot2

Sistēmas komponentes

`ggplot2` attēli tiek veidoti no slāņiem:

- **data** – informācija, kuru vēlas attēlot
- **aes** jeb **aesthetics** – datu attēlošanas veids – simbolu veids, krāsa utt
- **geom** – ģeometriskie objekti (līnija, punkti, poligoni), kurus reāli redz
- **stats** – statistiskās transformācijas datiem
- **scale** – nodrošina datu vērtību attēlojumu atbilstoši izvēlētajiem datu parādīšanas veidam (aesthetics), kā arī veido leģendas un asis
- **coord** – koordinātšu sistēma, ko izmanto grafikā
- **facet** – nosaka kā sadalīt datus pa atsevišķiem grafikiem

geom veidi

- `geom_point()` – zīmē punktus, lai veidotu izkliedes grafiku
- `geom_smooth()` – zīmē līniju, kas izlīdzina datus, kā arī šīs līnijas standartkļūdu
- `geom_boxplot()` – veido box-and-whisker grafiku
- `geom_path()` un `geom_line()` – zīmē līniju, kas savieno punktus
- `geom_abline()`,`geom_hline()`,`geom_vline()` – zīmē līnijas
- `geom_histogram()` – veido histogrammu
- `geom_density()` – veido blīvuma grafiku
- `geom_bar()` – veido slokšņu jeb stabiņu grafiku

Dati un paketes

```
# Pamatpakete attēlu atveidošanai
library(ggplot2)
# Pakete nepieciešama, ja jāizmanto funkcija unit() (mērvienības)
library(grid)
# Pakete, kurā ir papildus skalas un to transformācijas
library(scales)
data(mtcars)
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Attēlu veidošana

`ggplot2` sistēmā attēlus veido ar divām funkcijām - `qplot()` un `ggplot()`. Pirmā funkcija paredzēta ātrākais attēla izveidošanai, bet vienlaicīgi tai ir daudz mazāka iespēja tikt modificētai. `ggplot()` funkcijas gadījumā ir visas iespējas veikt katra elementa modifikāciju.

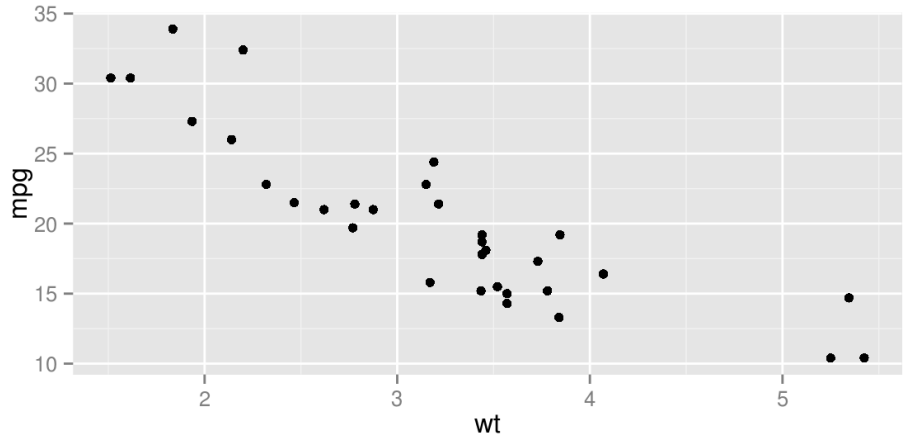
Funkcijā `ggplot()` kā pirmais arguments ir jānorāda datu tabulu (data frame), no kurienes ņemt datus, tad iekšā ir funkcija `aes()`, kur attiecīgi norāda x un y vērtības (atdalītas ar komatu).

`ggplot2` sistēmā **NAV** jāizmanto mainīgo pieraksts ar `$` zīmi.

geom_point()

Lai izveidotu punktu (izkliedes) attēlu, `ggplot()` norāda datu tabulu un x/y vērtības, un tam pieskaita nepieciešamao geom_... veidu.

```
ggplot(mtcars, aes(wt, mpg)) + geom_point()
```



Izskata maiņa

Lai mainītu punktu, līniju, poligonu izskatu (krāsu, lielumu, formu, caurspīdīgumu) ir divas iespējas:

1. Ja izskatam ir jābūt vienādam visiem elementiem, piemēram, visi punkti sarkani, tad atbilstošais arguments ir jāievieto `geom_...` funkcijā ārpus `aes()` iekavām un jānorāda atbilstošā vērtība.

2. Ja izskatam ir jāmainās atbilstoši kādam mainīgajam (piemēram, katrai sugai cita krāsa), tad atbilstošais arguments ir jāievieto funkcijā `geom_...` vai `ggplot()`, bet OBLIGĀTI iekšā `aes()` iekavām un kā vērtība jānorāda mainīgā nosaukums.

Faktori un skaitliskas vērtības

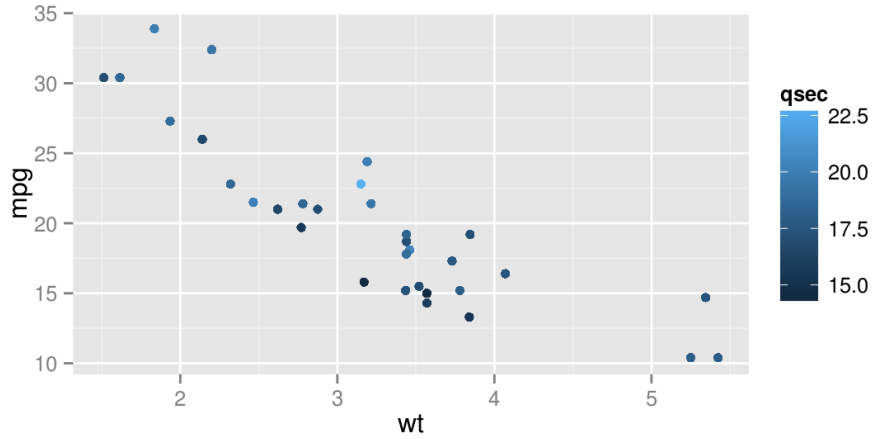
1. Ja pie izskata (krāsas, caurspīdīguma, lieluma) maiņas argumenta norāda skaitliskas vērtības, tad atbilstošie elementi mainās kā gradients (no mazākās uz lielāko vērtību).

2. Ja pie izskata maiņas argumenta norāda kā faktoru (vai skaitli, kas pārvērsts par faktoru), tad atbilstošie elementi mainās kā diskrētas (atsevišķas) vērtības.

geom_point()

Krāsa kā gradients

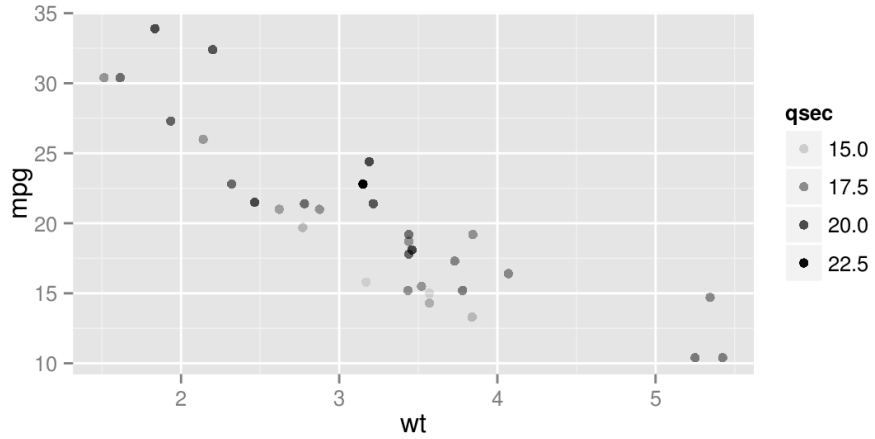
```
ggplot(mtcars, aes(wt, mpg)) + geom_point(aes(colour = qsec))
```



geom_point()

Caurspīdīgums kā gradients

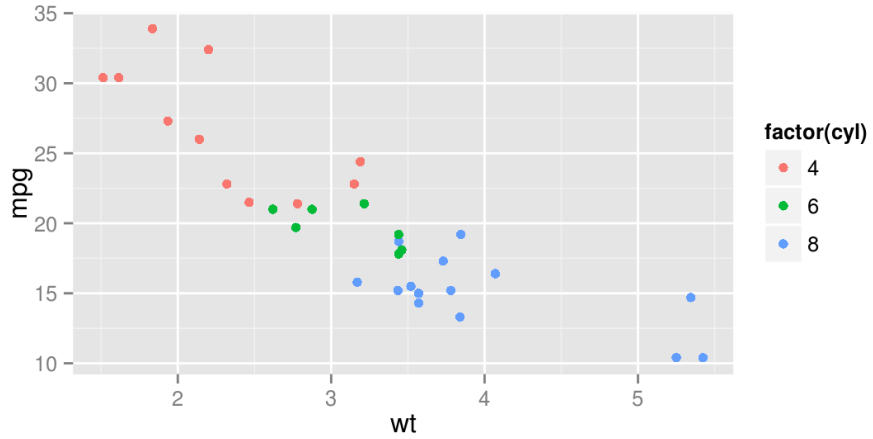
```
ggplot(mtcars, aes(wt, mpg)) + geom_point(aes(alpha = qsec))
```



geom_point()

Krāsa kā diskrēta vērtība

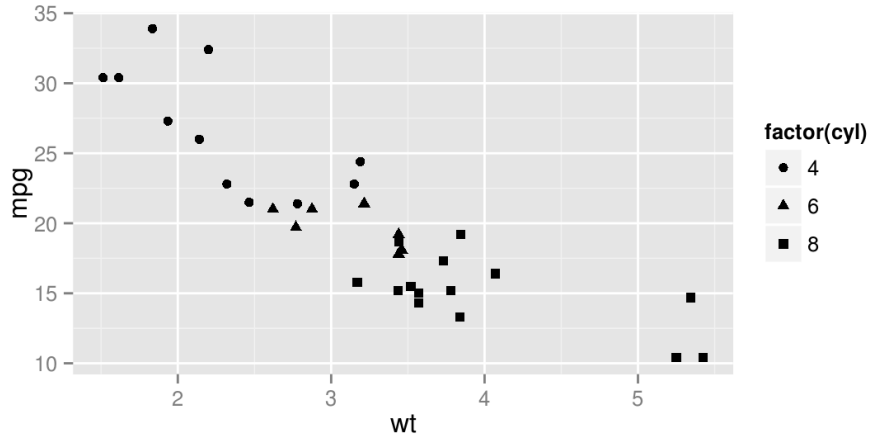
```
ggplot(mtcars, aes(wt, mpg)) + geom_point(aes(colour = factor(cyl)))
```



geom_point()

Forma kā diskrēta vērtība

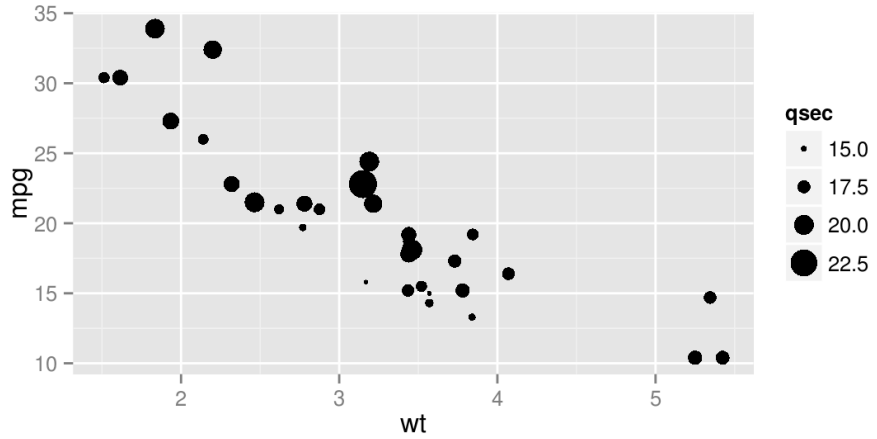
```
ggplot(mtcars, aes(wt, mpg)) + geom_point(aes(shape = factor(cyl)))
```



geom_point()

Izmērs kā gradients

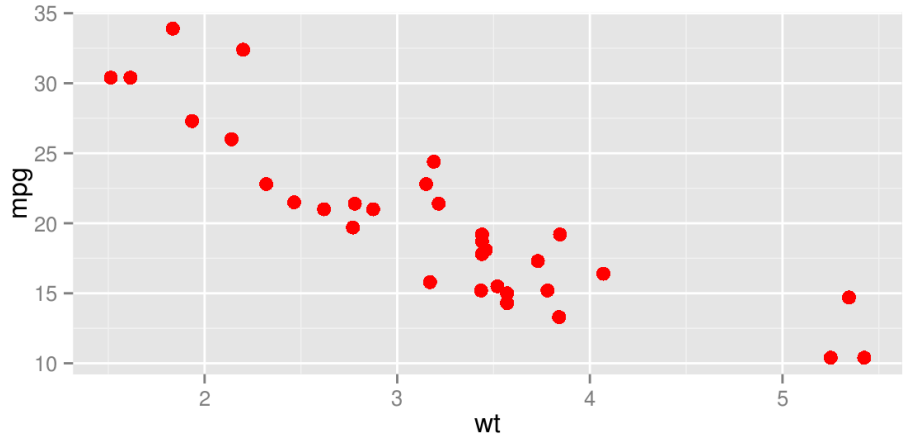
```
ggplot(mtcars, aes(wt, mpg)) + geom_point(aes(size = qsec))
```



geom_point()

Krāsa un izmērs, kas noteikts visiem punktiem vienāds

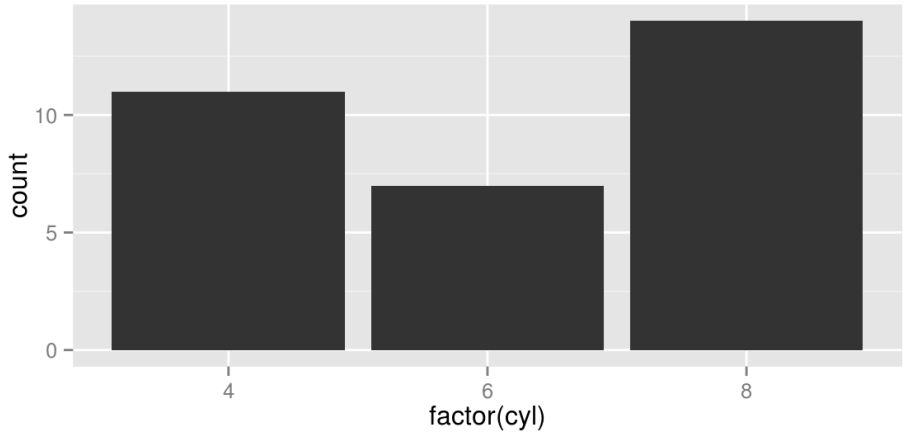
```
ggplot(mtcars, aes(wt, mpg)) + geom_point(colour = "red", size = 3)
```



geom_bar()

Veidojot stabiņu grafiku (`geom_bar()`) ir nepieciešamas tikai x vērtības (faktors), jo atkārtojumu skaitu nosaka automātiski.

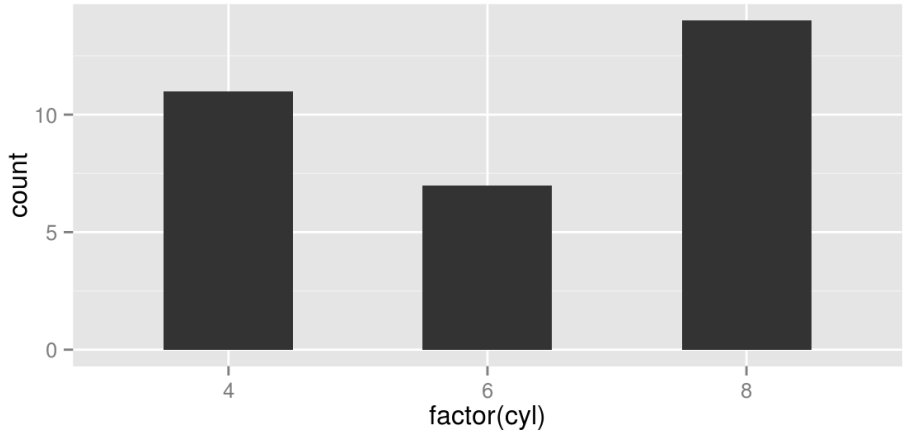
```
ggplot(mtcars, aes(factor(cyl)))+geom_bar()
```



geom_bar()

Arguments `width=` ļauj mainīt stabiņu platumu.

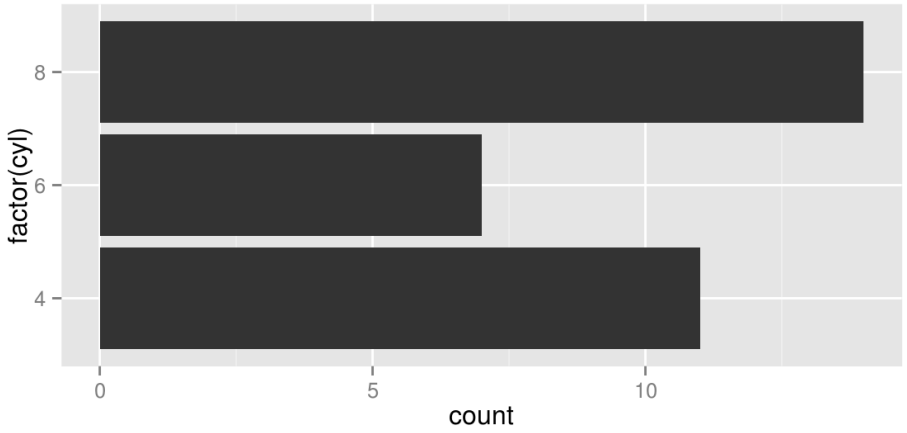
```
ggplot(mtcars, aes(factor(cyl))) + geom_bar(width=.5)
```



geom_bar()

Ar funkciju `coord_flip()` attēlu var pagriezt par 90 grādiem.

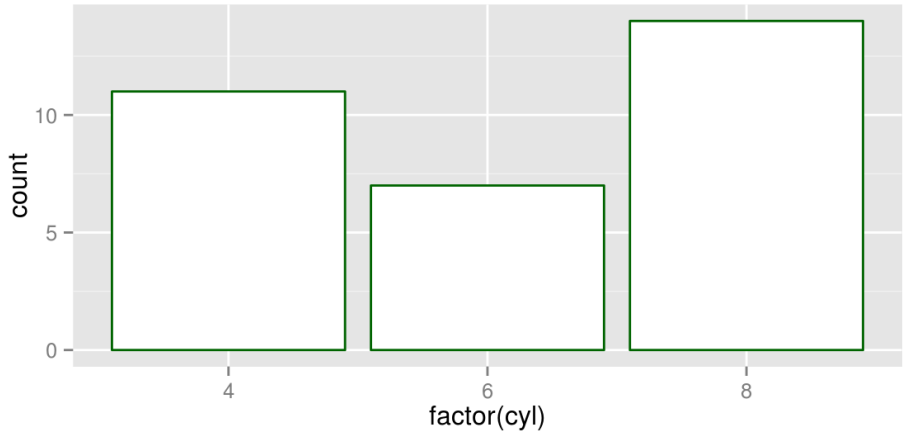
```
ggplot(mtcars, aes(factor(cyl))) + geom_bar() + coord_flip()
```



geom_bar()

Stabiņam ir iespējams noteik krāsu (līniju, kas ir apkārt) un aizpildījumu (`fill=`).

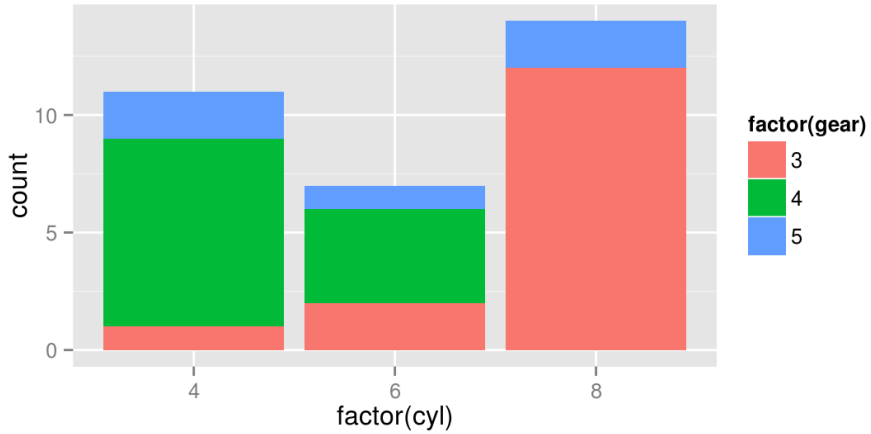
```
ggplot(mtcars, aes(factor(cyl))) + geom_bar(fill="white", colour="darkgreen")
```



geom_bar()

Ja aizpildījumu norāda kā mainīgo `aes()`, pēc noklusējuma stabiņi tiek sadalīti pa daļām atbilstoši mainīgajam.

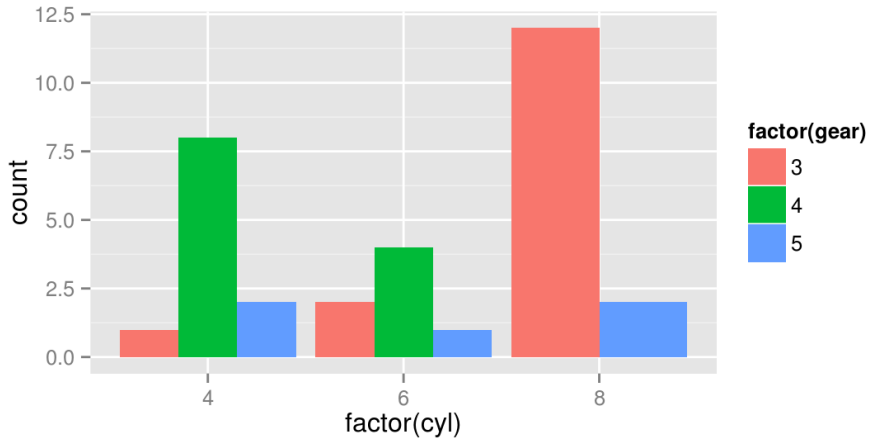
```
ggplot(mtcars, aes(factor(cyl),fill=factor(gear))) + geom_bar()
```



geom_bar()

Lai iegūtu stabiņus, kas sadalīti pēc mainīgā un būtu novietoti blakus, papildus jānorāda arguments `position="dodge"`.

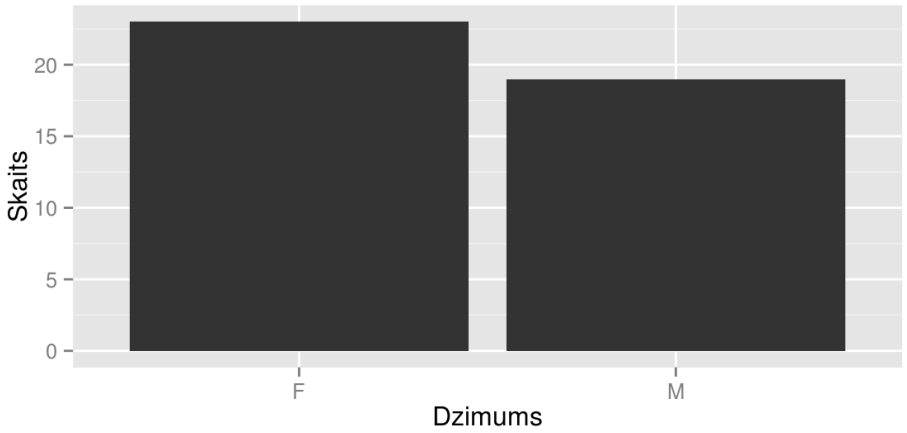
```
ggplot(mtcars, aes(factor(cyl),fill=factor(gear))) + geom_bar(position="dodge")
```



geom_bar()

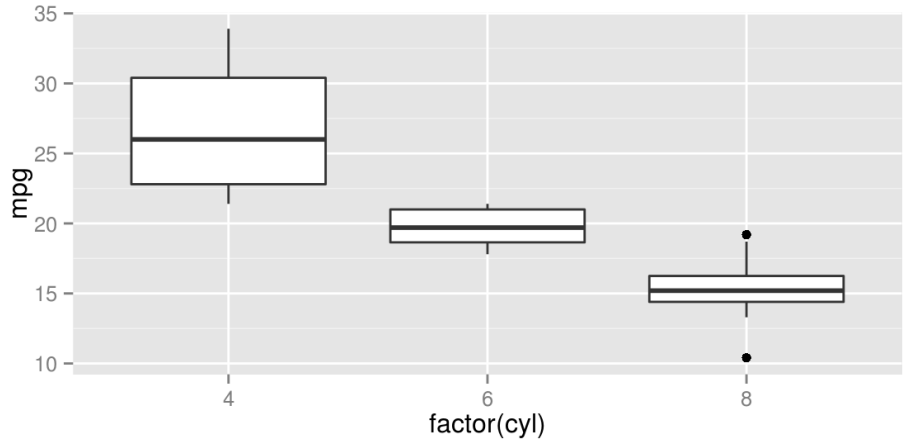
Ja y vērtības stabiņu attēlam jau ir zināmas, tad kā papildus arguments jānorāda `stat="identity"`.

```
df<-data.frame(Dzimums=c("F","M"),Skaits=c(23,19))
ggplot(df,aes(Dzimums,Skaits))+geom_bar(stat="identity")
```



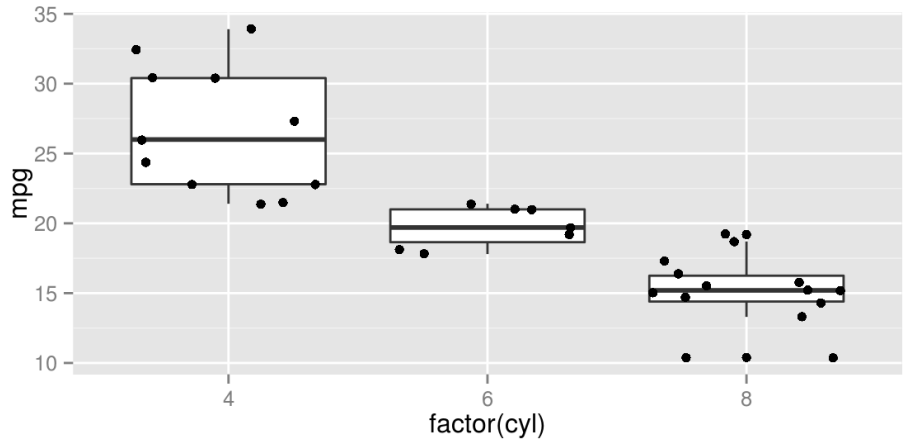
geom_boxplot()

```
ggplot(mtcars, aes(factor(cyl), mpg)) + geom_boxplot()
```



geom_boxplot()

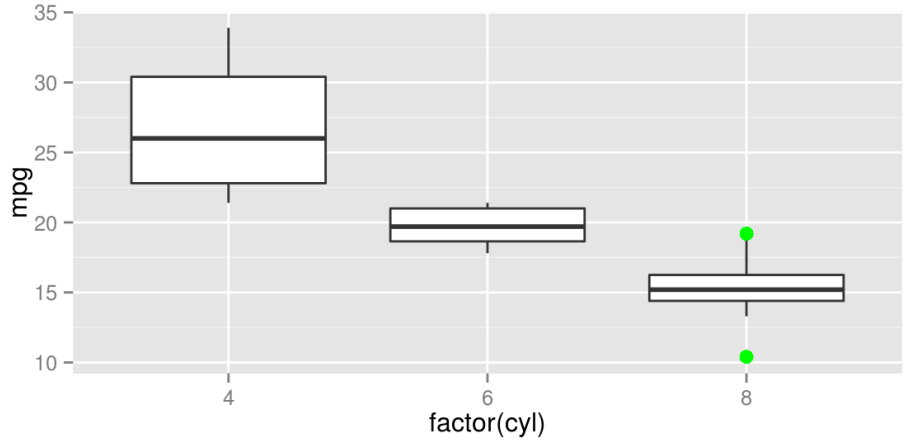
```
ggplot(mtcars, aes(factor(cyl), mpg)) + geom_boxplot() + geom_jitter()
```



geom_boxplot()

Ar argumentiem `outlier.colour=` un `outlier.size=` boxplot attēlos var atsevišķi ietekmē tikai izlēcošo vērtību izskatu.

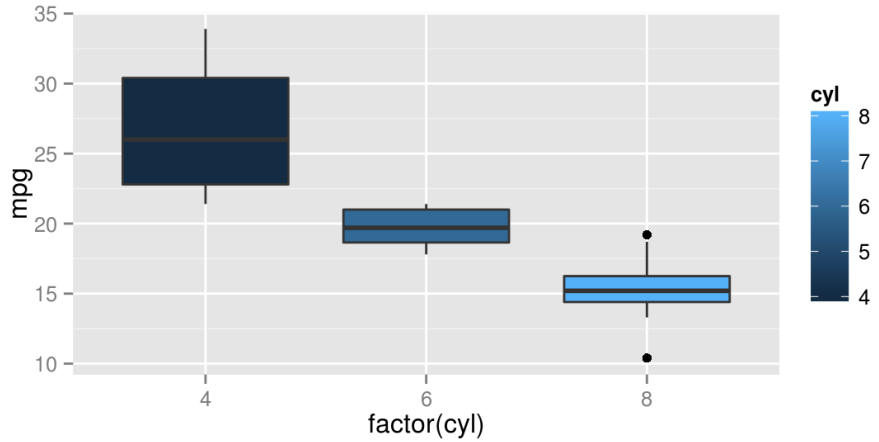
```
ggplot(mtcars, aes(factor(cyl), mpg)) +
      geom_boxplot(outlier.colour = "green", outlier.size = 3)
```



geom_boxplot

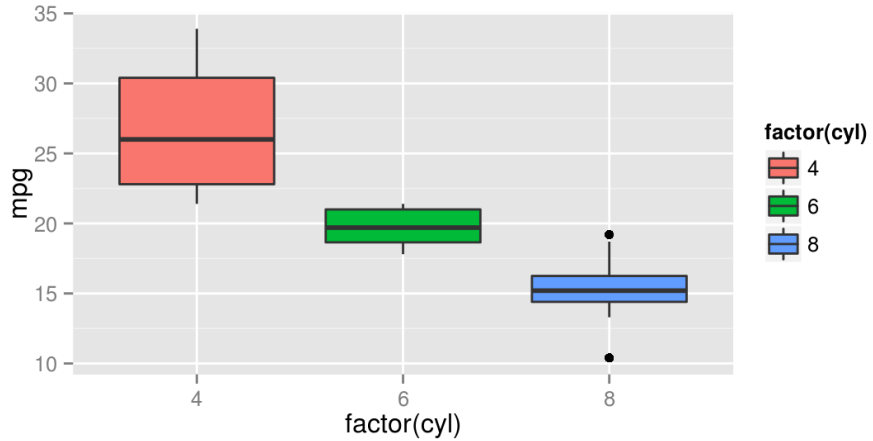
Boxplot attēliem līdzīgi kā stabiņu attēliem krāsa (`colour=`) attiecas uz ārējo malu un aizpildījums (`fill=`) uz iekšējo krāsojumu.

```
ggplot(mtcars, aes(factor(cyl), mpg)) + geom_boxplot(aes(fill = cyl))
```



geom_boxplot()

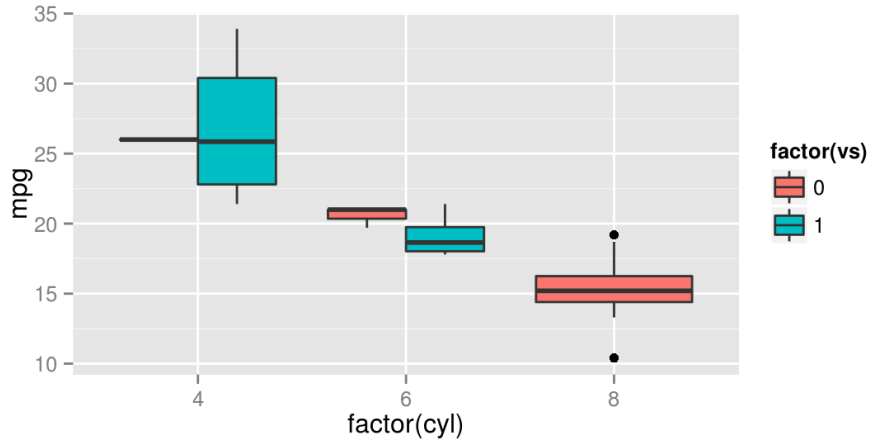
```
ggplot(mtcars, aes(factor(cyl), mpg))+ geom_boxplot(aes(fill = factor(cyl)))
```



geom_boxplot()

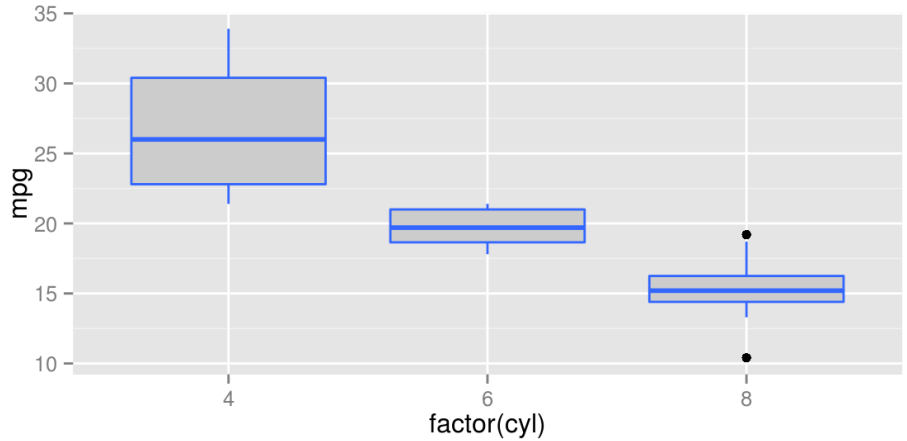
Ja kā aizpildījumu norāda mainīgo, kas nav y vērtība, pie katras y vērtības boxplot attēli tiek sadalīti atbilstoši šim mainīgajam.

```
ggplot(mtcars, aes(factor(cyl), mpg)) + geom_boxplot(aes(fill = factor(vs)))
```



geom_boxplot()

```
ggplot(mtcars, aes(factor(cyl), mpg)) + geom_boxplot(fill = "grey80", colour = "#3366FF")
```

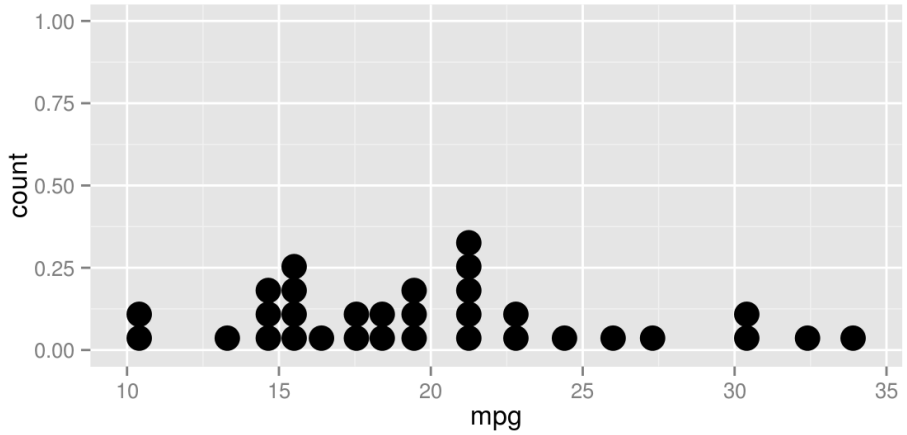


geom_dotplot()

`geom_dotplot()` pārāda vērtību sadalījumu datos relatīvās vienībās (līdzīgi kā histogramma).

```
ggplot(mtcars, aes(x=mpg)) + geom_dotplot()
```

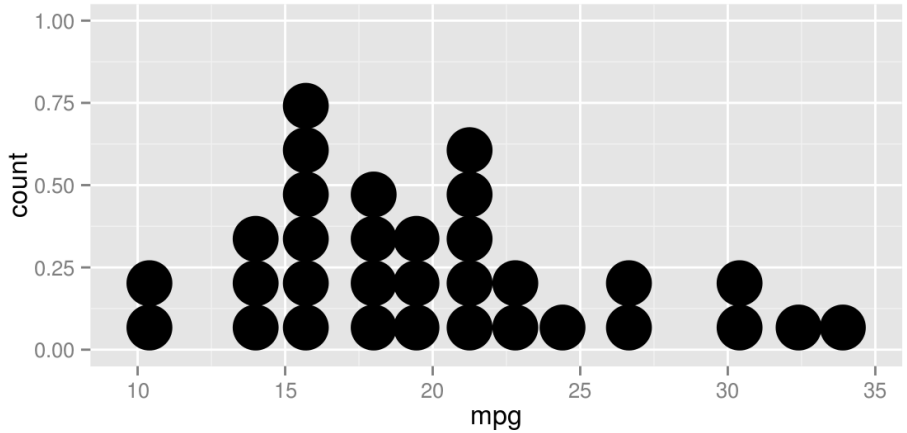
```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



geom_dotplot()

Arguments `binwidth=` nosaka dalījumu intervālu.

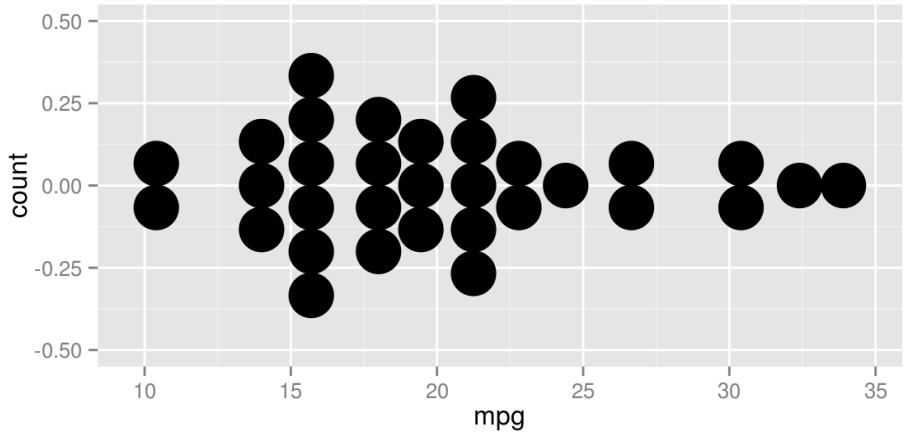
```
ggplot(mtcars, aes(x=mpg)) + geom_dotplot(binwidth = 1.5)
```



geom_dotplot()

Ar argumentu `stackdir="center"` var mainīt punktu novietojumu (centrēt).

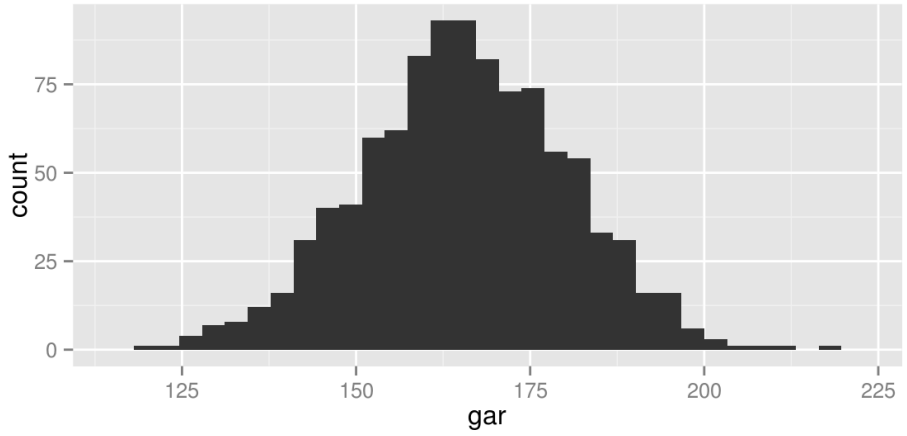
```
ggplot(mtcars, aes(x=mpg)) + geom_dotplot(binwidth = 1.5, stackdir = "center")
```



geom_histogram()

```
df<-data.frame(gar=rnorm(1000,165,15))
ggplot(df, aes(x=gar)) + geom_histogram()
```

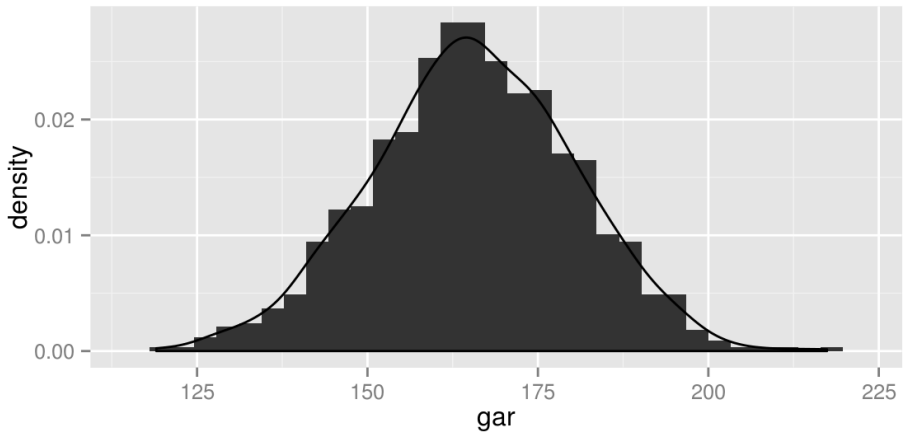
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



geom_histogram()

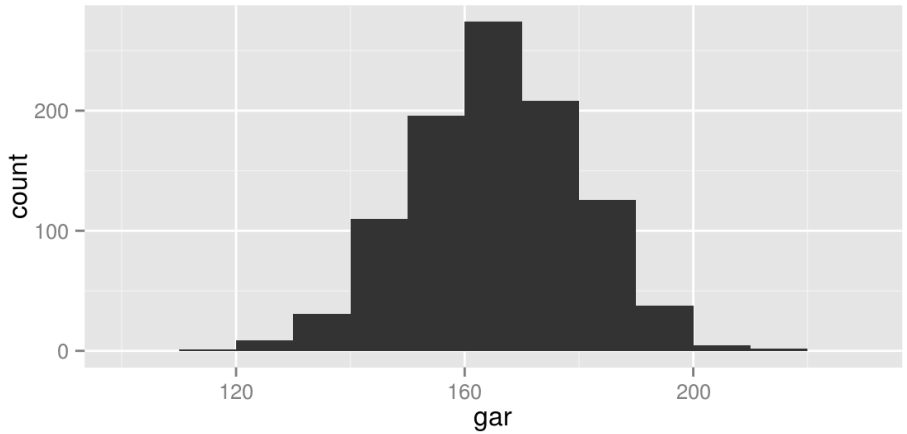
```
ggplot(df, aes(x=gar)) + geom_histogram(aes(y = ..density..)) + geom_density()
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



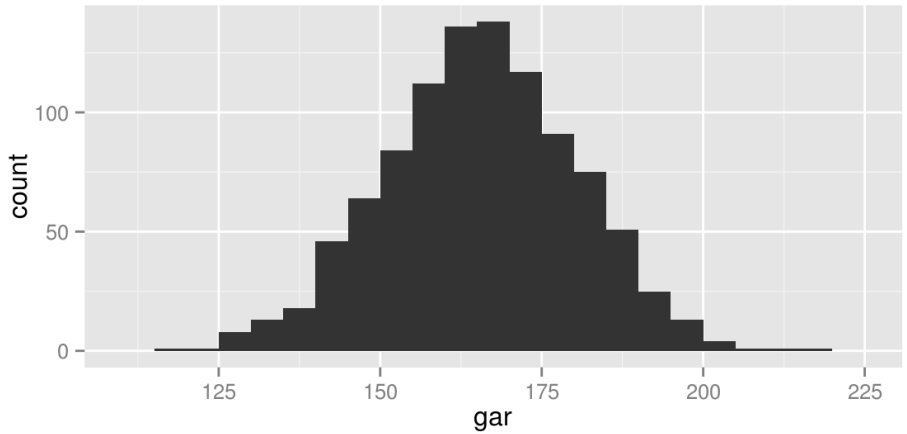
geom_histogram()

```
ggplot(df, aes(x=gar))+ geom_histogram(binwidth = 10)
```



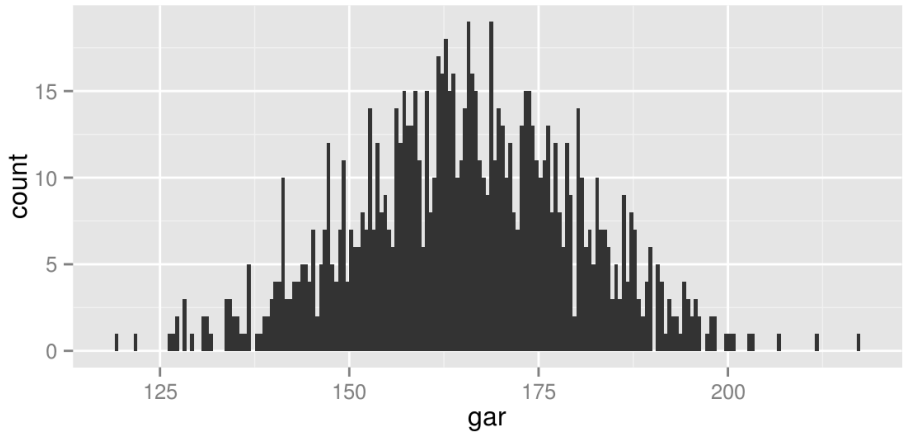
geom_histogram()

```
ggplot(df, aes(x=gar)) + geom_histogram(binwidth = 5)
```



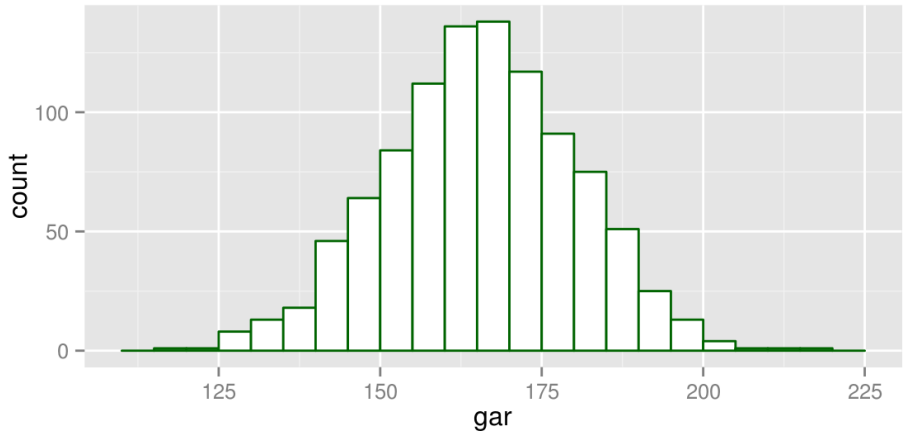
geom_histogram()

```
ggplot(df, aes(x=gar)) + geom_histogram(binwidth = 0.5)
```



geom_histogram()

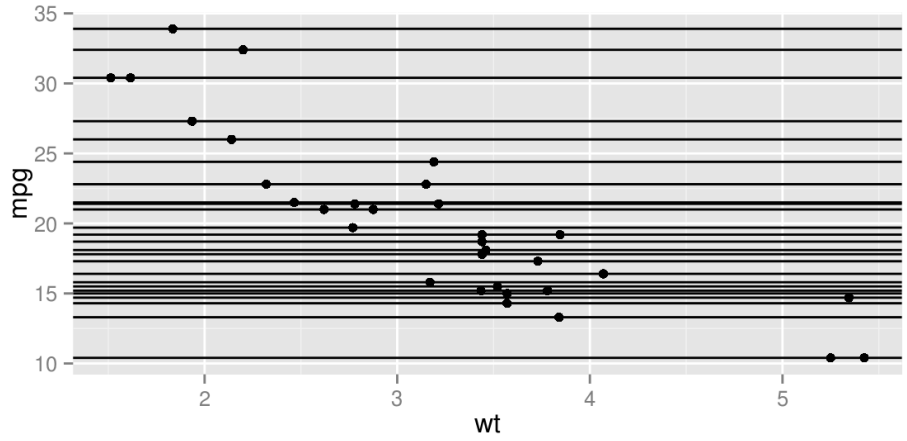
```
ggplot(df, aes(x=gar)) +
      geom_histogram(colour = "darkgreen", fill = "white", binwidth = 5)
```



geom_hline()

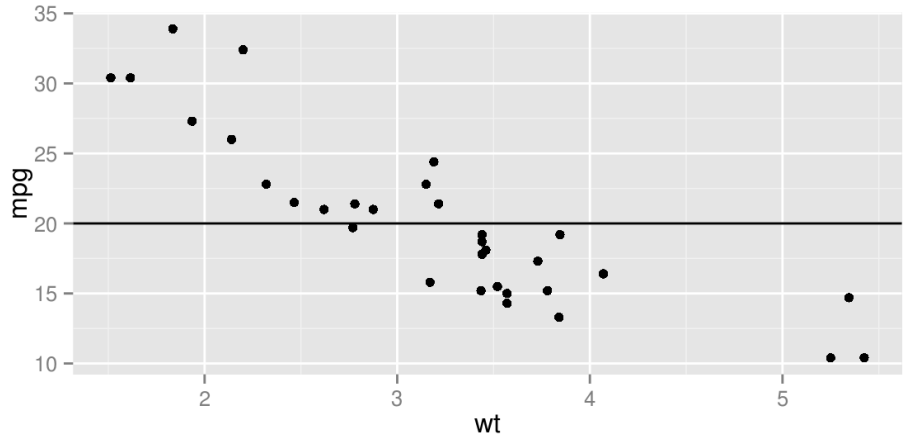
Horizontālu līniju pievienošanai izmanto funkciju `geom_hline()`, kurai kā argumentu norāda yintercept=, kas var būt gan iekļauts `aes()`, gan ārpus tā.

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_point()+geom_hline(aes(yintercept=mpg))
```



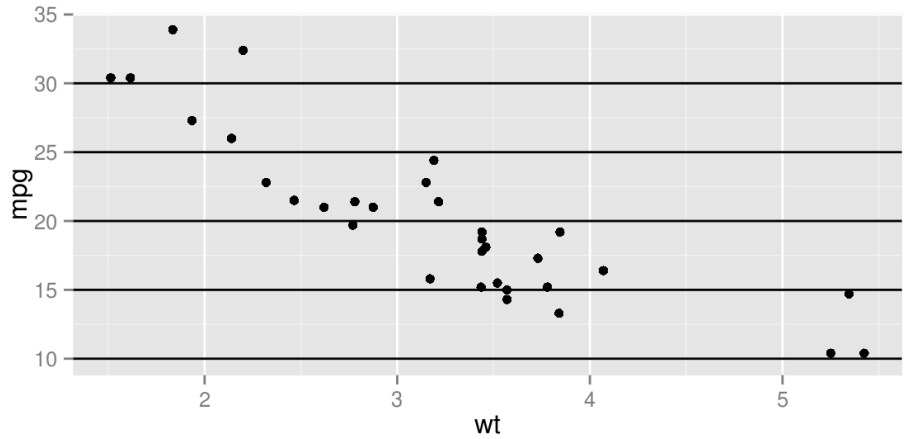
geom_hline()

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_point()+geom_hline(yintercept=20)
```



geom_hline()

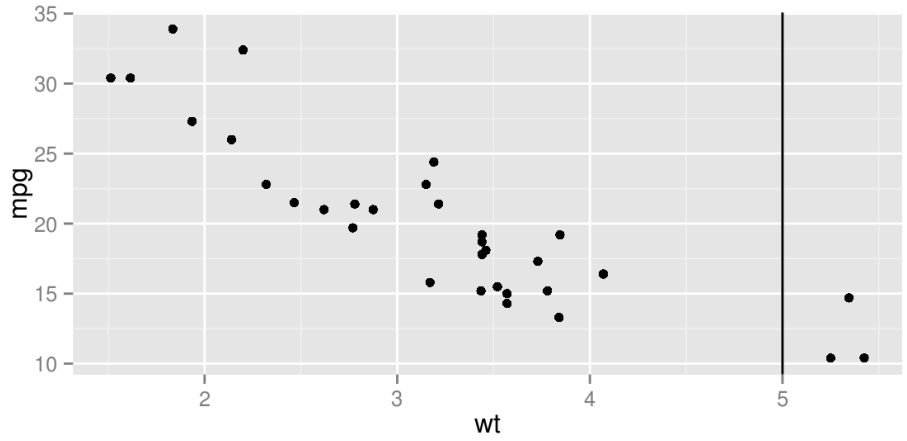
```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_point() +
      geom_hline(yintercept=seq(10, 30, by=5))
```



geom_vline

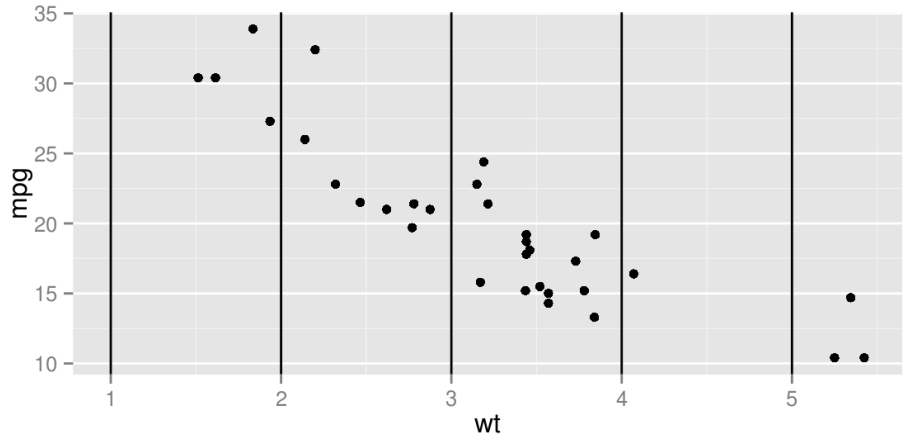
Vertikālu līniju pievienošanai izmanto `geom_vline()` un argumentu `xintercept=`.

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_point() + geom_vline(xintercept = 5)
```



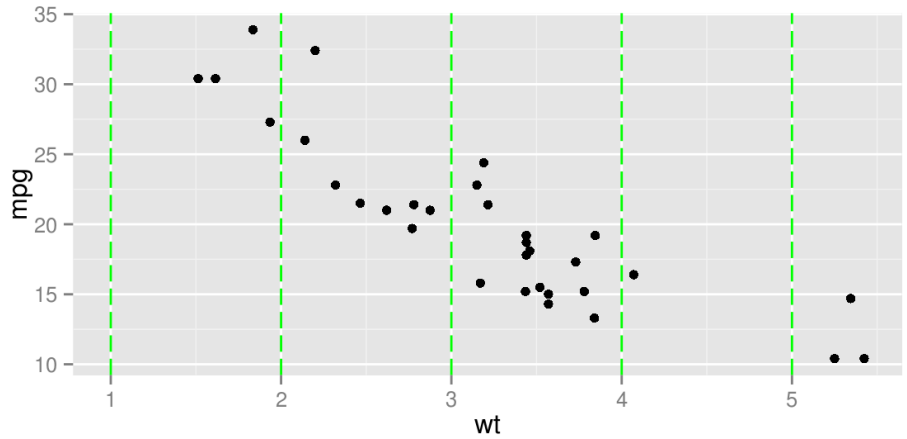
geom_vline()

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_point() + geom_vline(xintercept = 1:5)
```



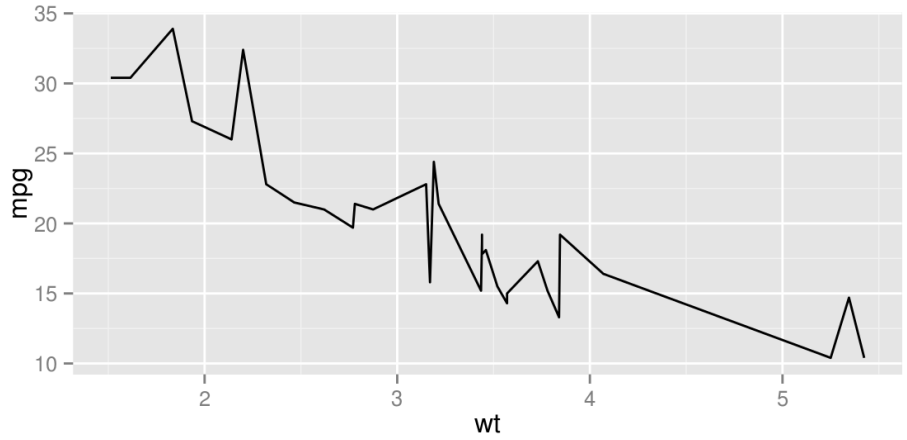
geom_vline()

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_point() +
      geom_vline(xintercept = 1:5, colour="green", linetype = "longdash")
```



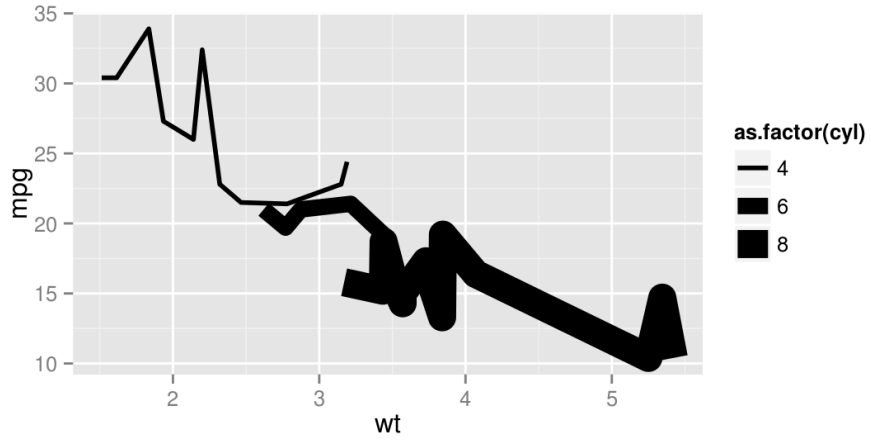
geom_line()

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_line()
```



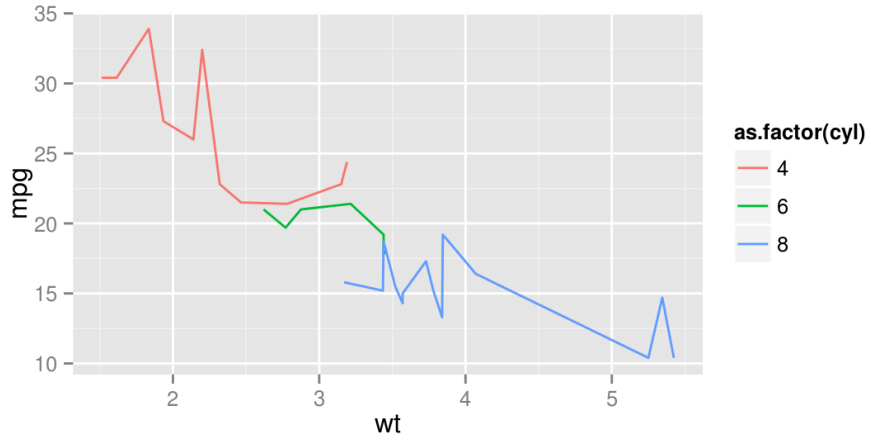
geom_line()

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_line(aes(size = as.factor(cyl)))
```



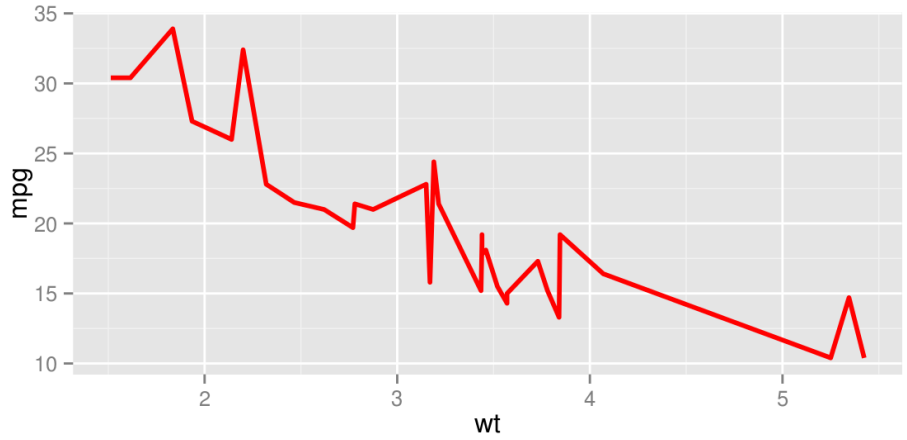
geom_line()

```
ggplot(mtcars, aes(x = wt, y=mpg)) + geom_line(aes(colour = as.factor(cyl)))
```



geom_line()

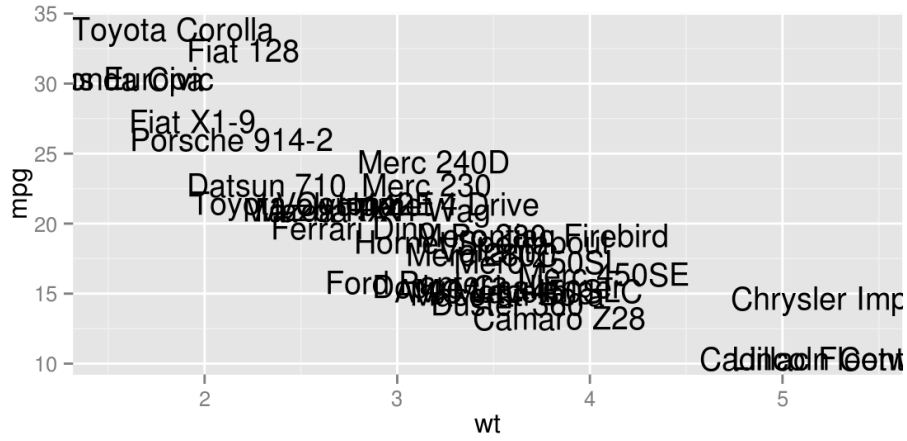
```
ggplot(mtcars, aes(x = wt, y=mpg))  + geom_line(colour = "red", size = 1)
```



geom_text()

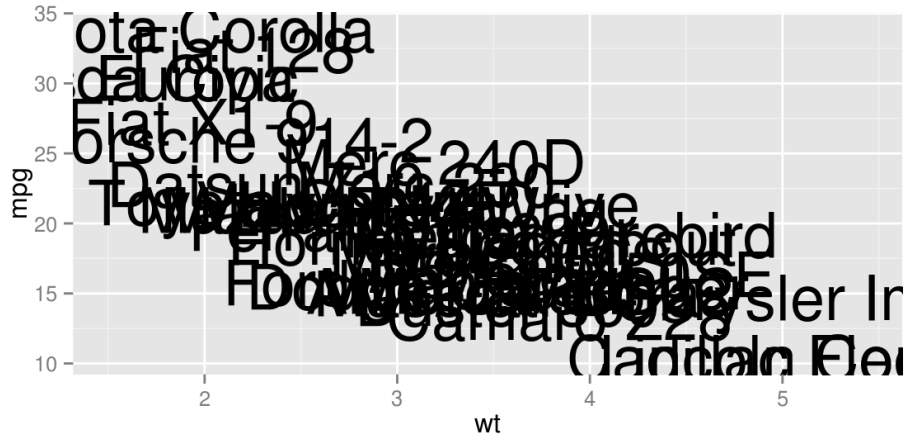
`geom_text()` attēlā izvieto tekstu norādītajās x un y koordinātēs. Kā papildus arguments jānorāda `label=`, kas parāda attēlojamo tekstu.

```
ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) + geom_text()
```



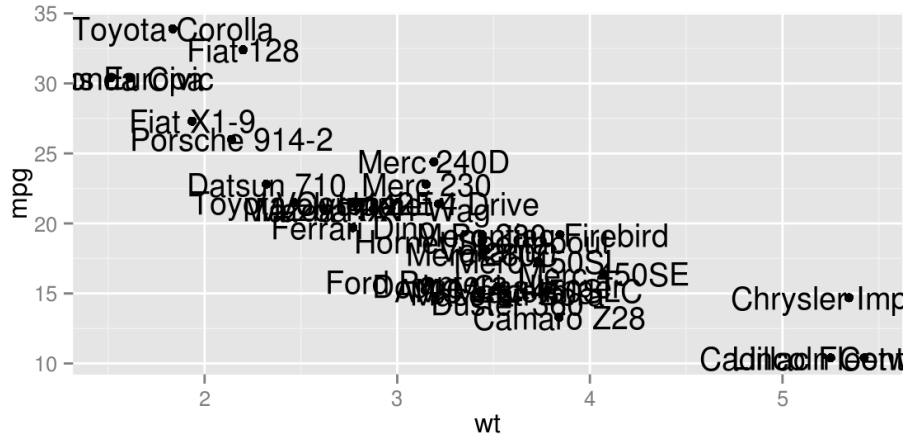
geom_text()

```
ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) + geom_text(size=10)
```



geom_text()

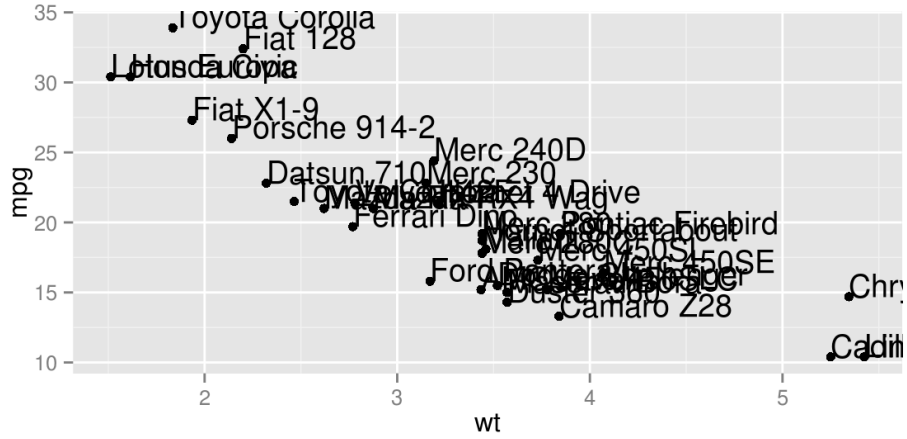
```
ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) + geom_point() + geom_text()
```



geom_text()

Ar argumentiem `hjust=` un `vjust=` var mainīt teksta novietojumu attiecībā pret norādīto koordināti.

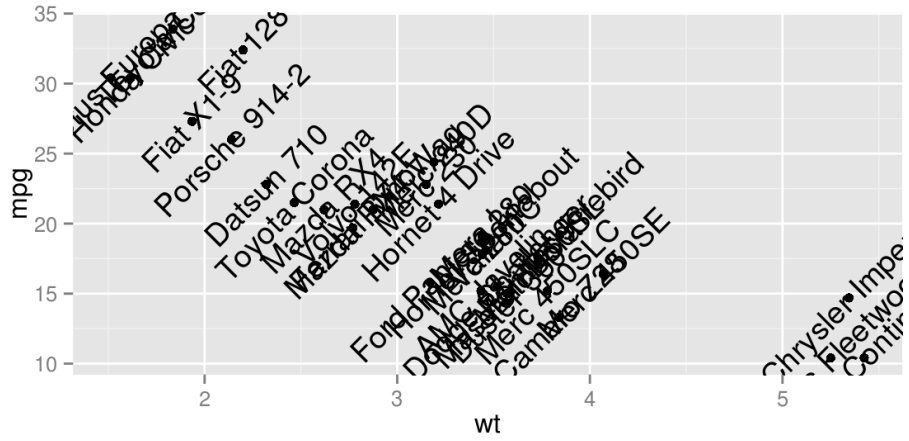
```
ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) +
      geom_point() + geom_text(hjust=0, vjust=0)
```



geom_text()

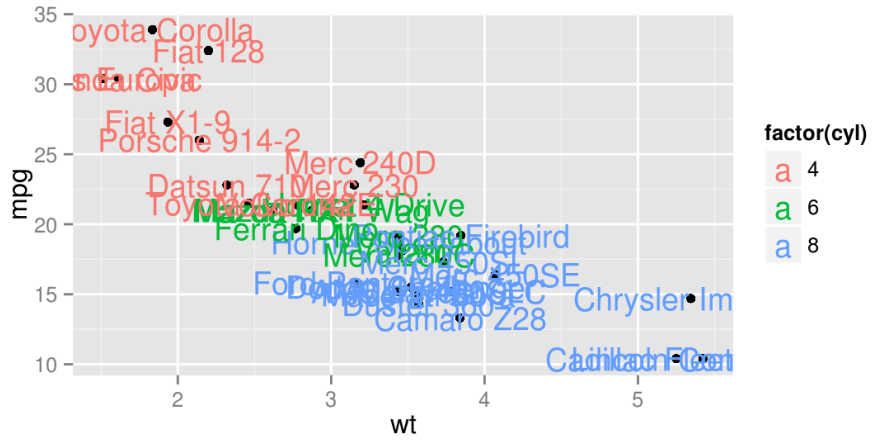
Arguments `angle=` maina teksta leņķi.

```
ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) + geom_point()  +
      geom_text(angle = 45)
```



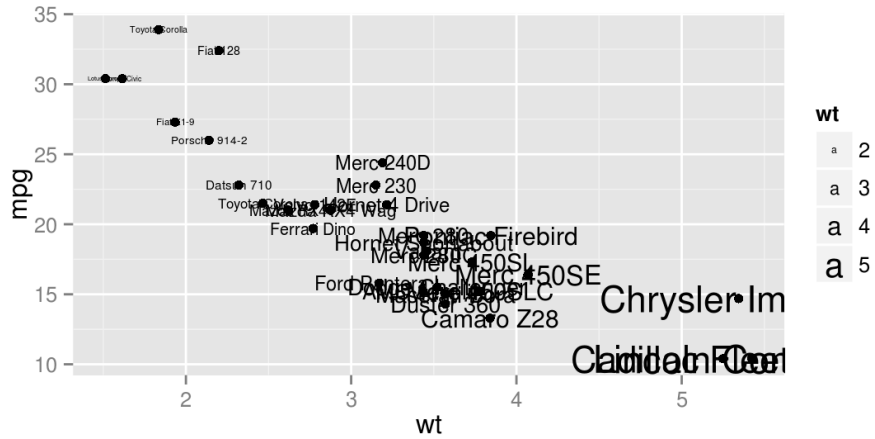
geom_text()

```
ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) + geom_point()+
      geom_text(aes(colour=factor(cyl)))
```



geom_text()

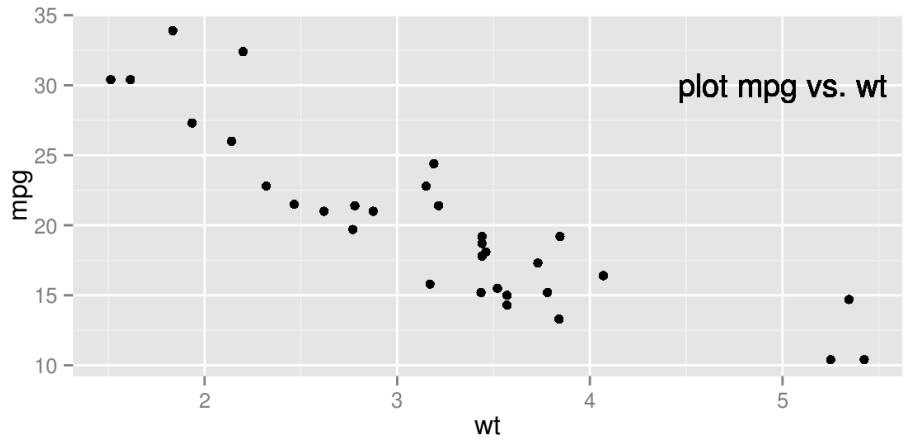
```
ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) + geom_point() +
      geom_text(aes(size=wt))
```



geom_text()

Tekstu var izvietot arī neizmantojot gatavu datu tabulu. Šādā gadījumā jāpievienos arguments `data=NULL` un visi pārējie argumenti jāraksta ārpus `aes()`.

```
ggplot(mtcars, aes(wt, mpg)) + geom_point() +
      geom_text(data = NULL, x = 5, y = 30, label = "plot mpg vs. wt")
```

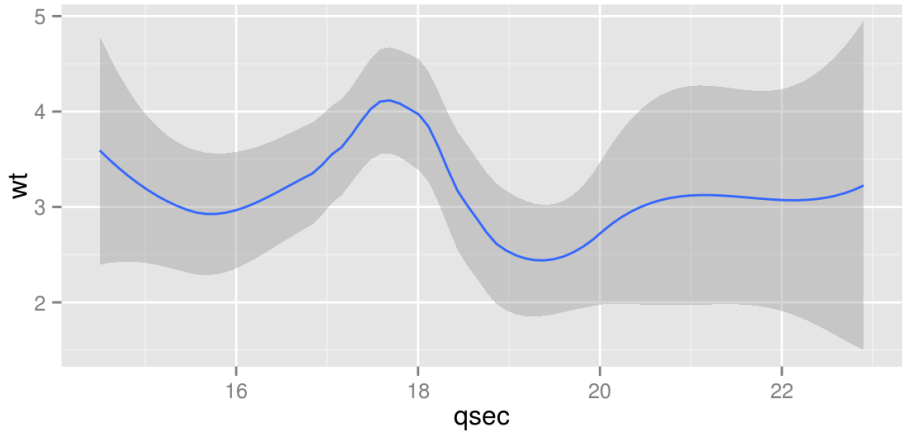


stat_smooth()

Ar funkciju `stat_smooth()` var pievienot "trenda" līniju kopā ar ticamības intervālu. Izlīdzinātās līnijas veids tiek noteikts automātiski.

```
ggplot(mtcars, aes(qsec, wt)) + stat_smooth()
```

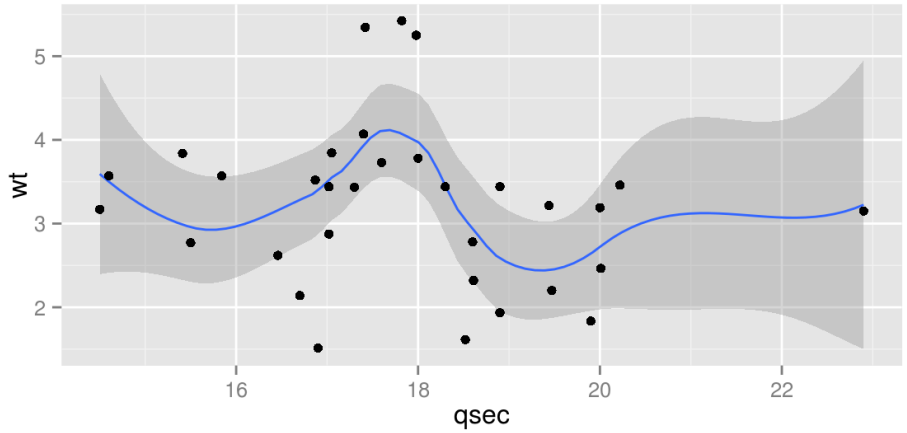
```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method =
```



stat_smooth()

```
ggplot(mtcars, aes(qsec, wt)) + stat_smooth() + geom_point()
```

```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method =
```

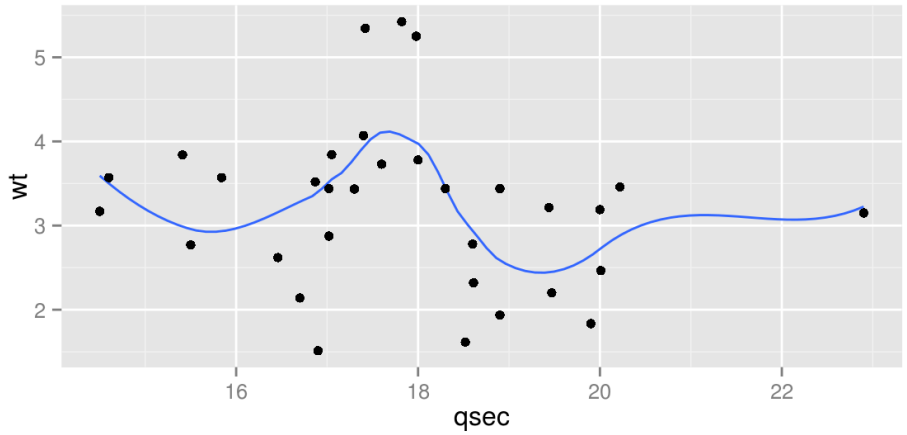


stat_smooth()

Ar argumentu se=FALSE var noņemt ticamības intervālu.

```
ggplot(mtcars, aes(qsec, wt)) + stat_smooth(se = FALSE) + geom_point()
```

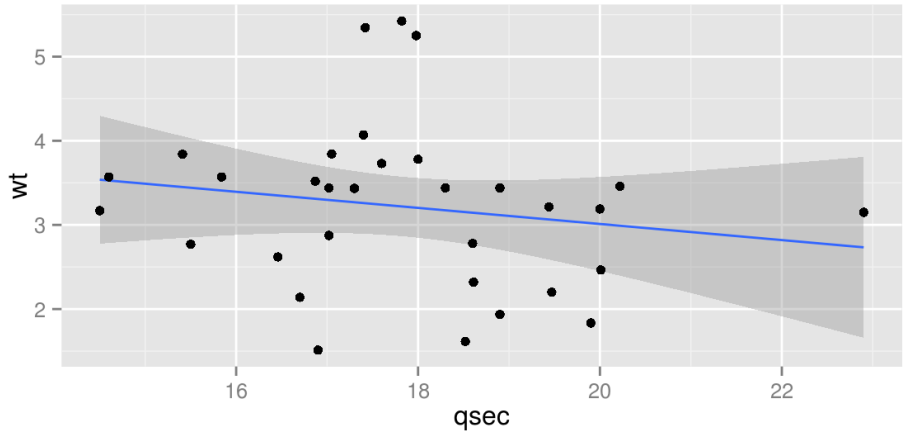
```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method =
```



stat_smooth()

Ja nepieciešams pievienot tieši lineāru trendu, tad jālieto arguments `method="lm"`.

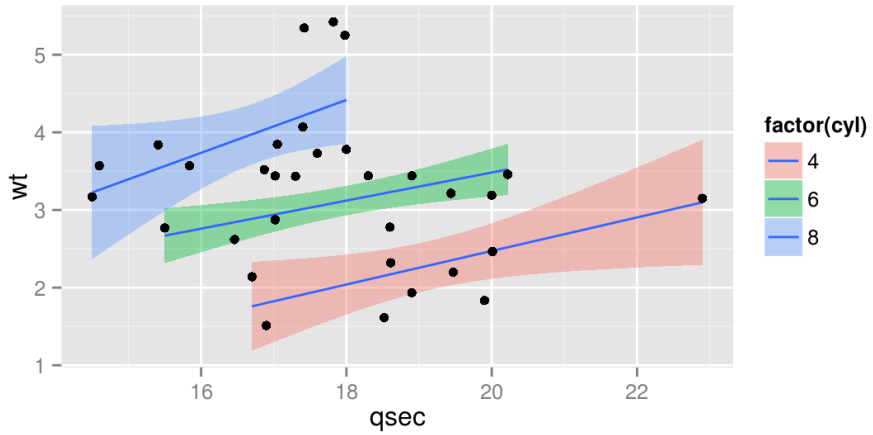
```
ggplot(mtcars, aes(qsec, wt)) + stat_smooth(method = "lm") + geom_point()
```



stat_smooth()

Nosakot krāsu (līnijai) vai aizpildījumu (ticamības intervāliem) atkarībā no mainīgā, trenda līnijas parādīsies katram līmenim atsevišķi.

```
ggplot(mtcars, aes(qsec, wt)) + stat_smooth(method=lm, aes(fill = factor(cyl))) +
      geom_point()
```

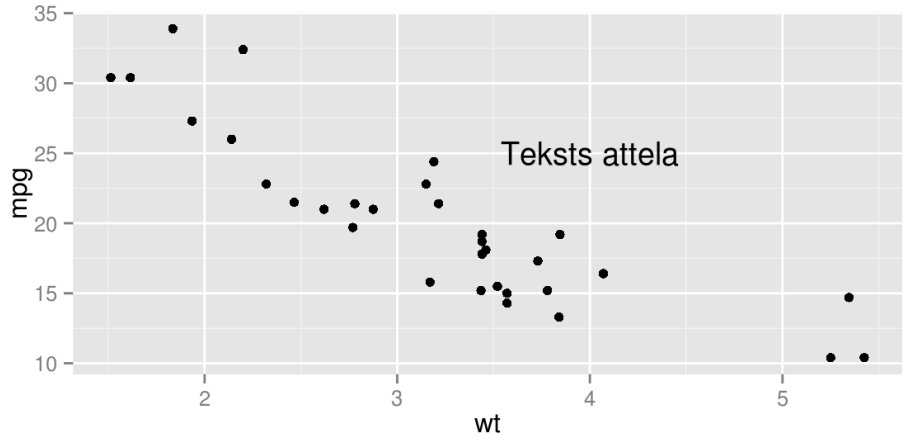


annotate()

Ar funkciju annotate() ir iespējams izvietot attēlā elementus, kas nav nodefinēti atsevišķās datu tabulās.

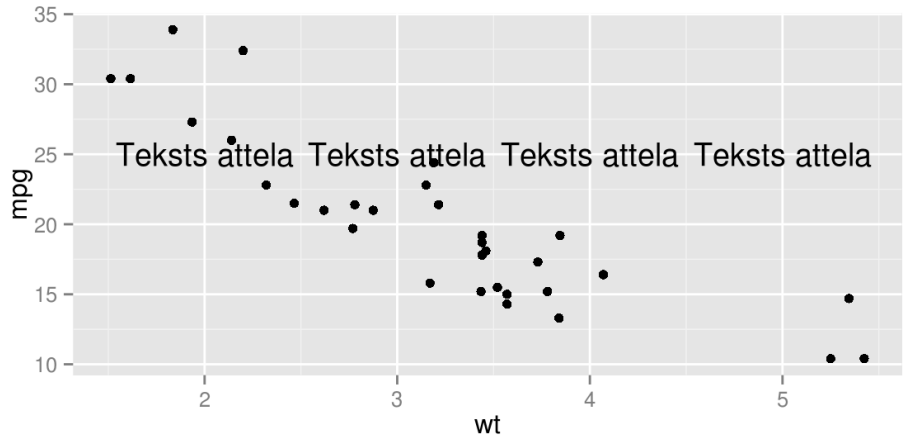
Kā argumenti jānorāda elementa veids, atbilstošā koordinātes.

```
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point() +
      annotate("text", x = 4, y = 25, label = "Teksts attela")
```



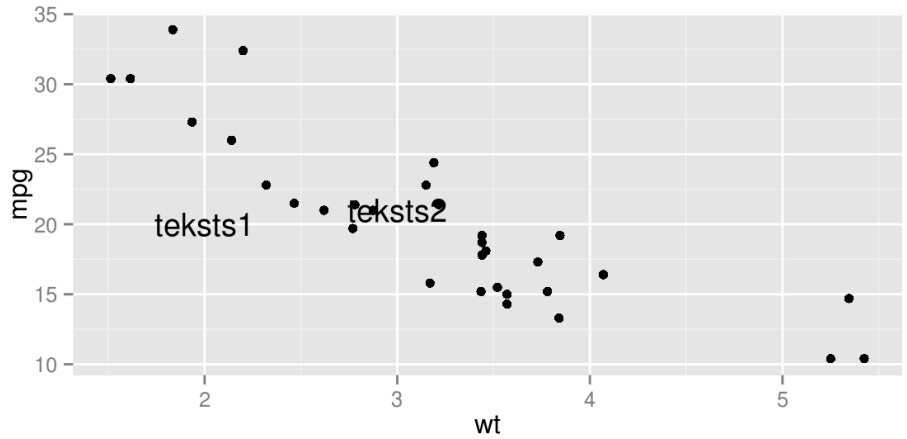
annotate()

```
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point() +
      annotate("text", x = 2:5, y = 25, label = "Teksts attela")
```



annotate()

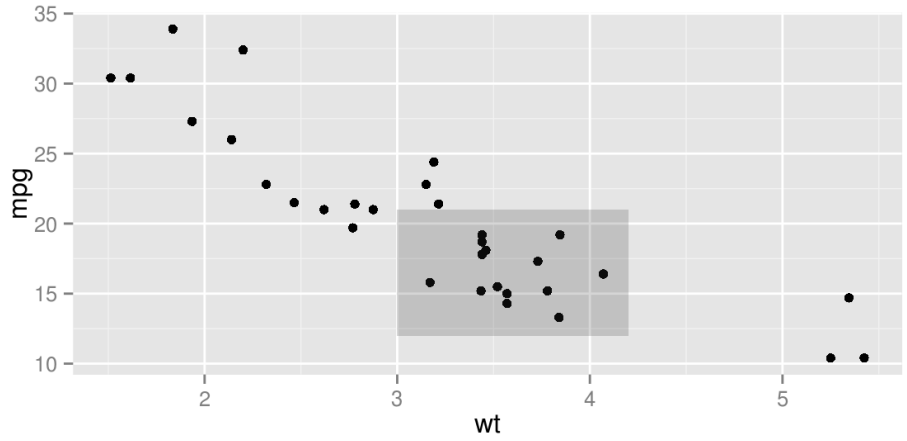
```
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()  +
      annotate("text", x = 2:3, y = 20:21, label = c("teksts1", "teksts2"))
```



annotate()

Ar argumentu "rect" var izveidot taisnstūri.

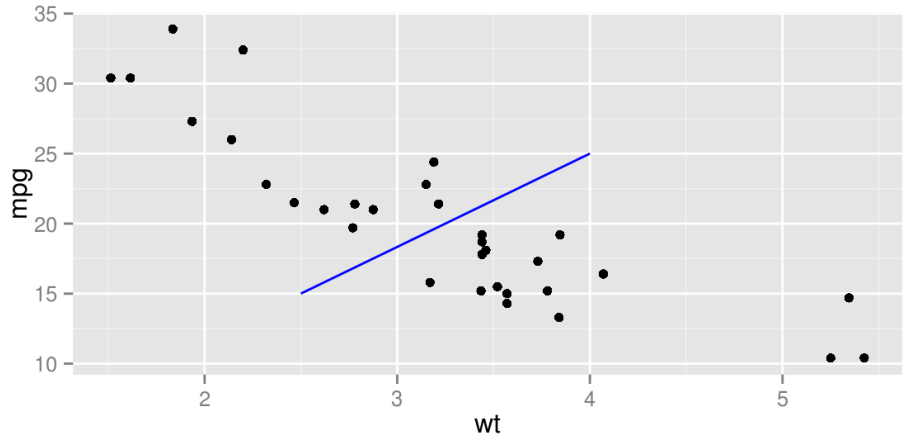
```
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()  +
      annotate("rect", xmin = 3, xmax = 4.2, ymin = 12, ymax = 21, alpha = .2)
```



annotate()

Ar argumentu "segment" var izveidot ierobežota garuma līniju.

```
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()  +
      annotate("segment", x = 2.5, xend = 4, y = 15, yend = 25, colour = "blue")
```



Skalas

Izmantojot funkcijas `scale_..._...()` var mainīt ne tikai x un y asu noformēju un veidu, bet arī ietekmēt visus citus atribūtus, kas tiek noteikti izmantojot `aes()`: krāsas, lielumus, punktu formas, līniju veidus, apzīmējumu caurspīdīgumu.

Skalu piemēri:

- `scale_x_discrete(),scale_y_discrete()`
- `scale_x_continuous(),scale_y_continuous()`
- `scale_colour_discrete(),scale_colour_continuous(),scale_colour_grey()` `scale_colour_gradient()`
- `scale_fill_discrete(),scale_fill_continuous(),scale_fill_grey(),` `scale_fill_gradient()`

Skalas

Skalu piemēri:

- `scale_linetype_discrete()`,`scale_linetype_continuous()`, `scale_linetype_manual()`,`scale_linetype_identity()`
- `scale_x_log10()`,`scale_x_reverse()`,`scale_x_sqrt()`

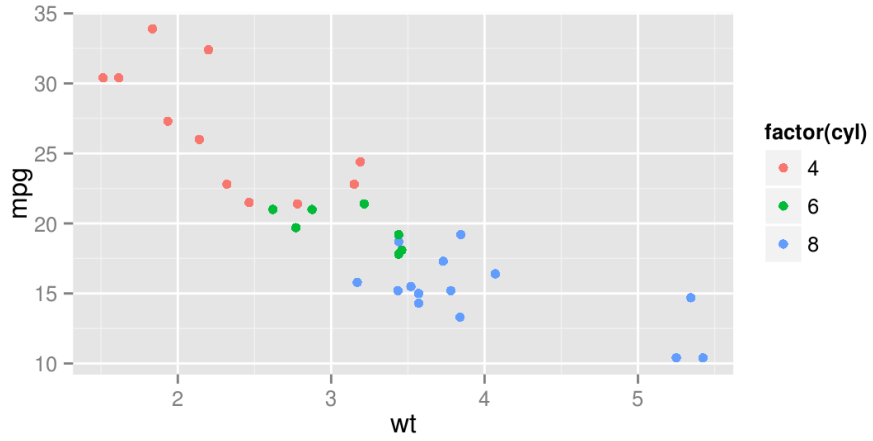
Skalas

Skalām var mainīt sekojošos parametrus:

- name - nosaukums
- breaks - dalījuma vietas
- labels - apzīmējumi dalījuma vietās
- limits - vērtību diapozons
- values - lietāja definētās krāsas/līniju veidi/simbolu veidi - izmanto tikai ar `scale_..._manual()`

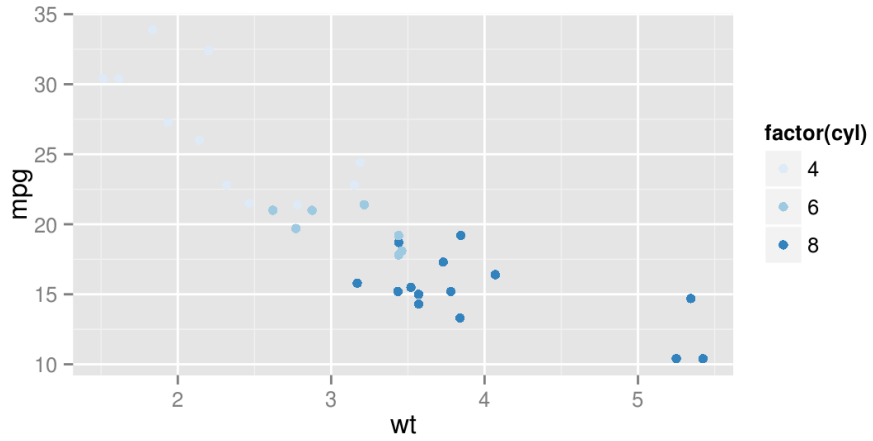
scale_colour_brewer()

```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point()
```



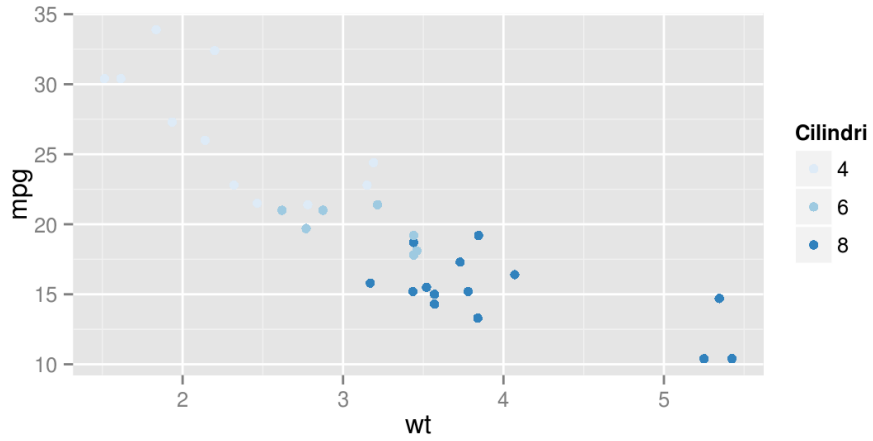
scale_colour_brewer()

```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_colour_brewer()
```



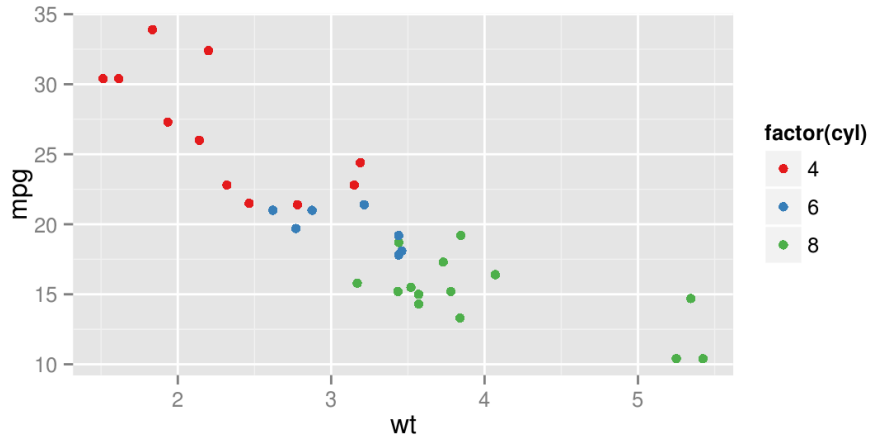
scale_colour_brewer()

```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_colour_brewer("Cilindri")
```



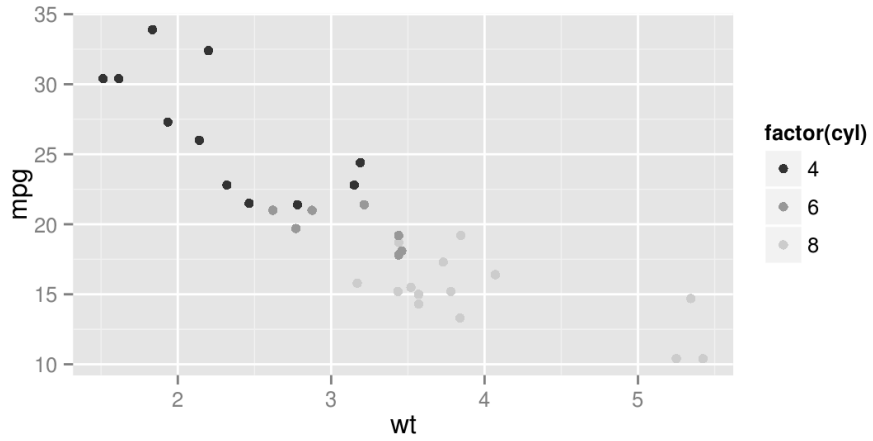
scale_colour_brewer()

```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_colour_brewer(palette="Set1")
```



scale_colour_grey()

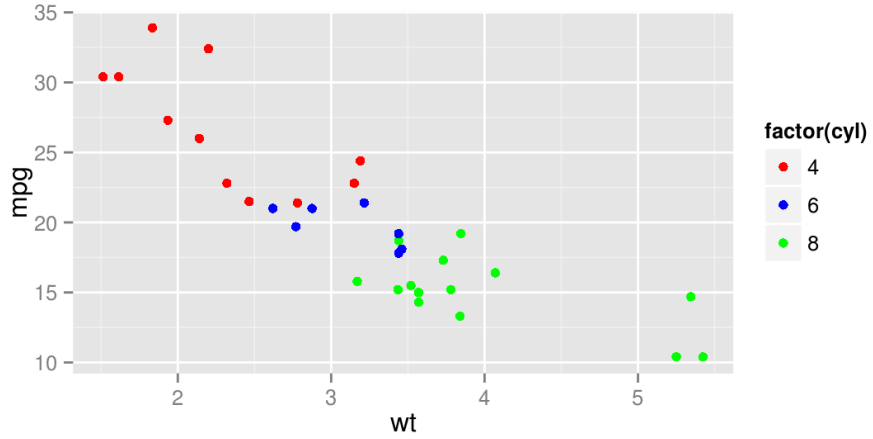
```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_colour_grey()
```



scale_colour_manual()

Izmantojot `scale_..._manual()` ir iespējams izraudzīties paša definētās vērtības (to skaitama jāatbilst līmeņu skaitam).

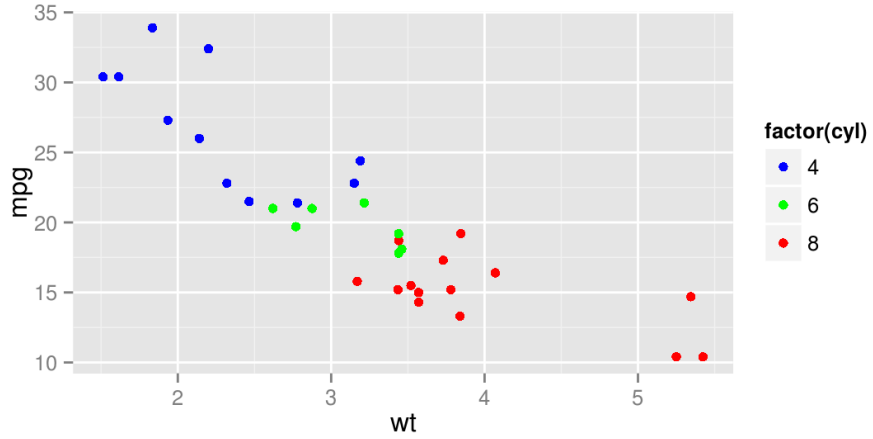
```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_colour_manual(values = c("red","blue", "green"))
```



scale_colour_manual()

Ja pirms vēlamās vērtības (krāsas) norāda arī līmeņa nosaukumu, tad secībai nav nozīmes.

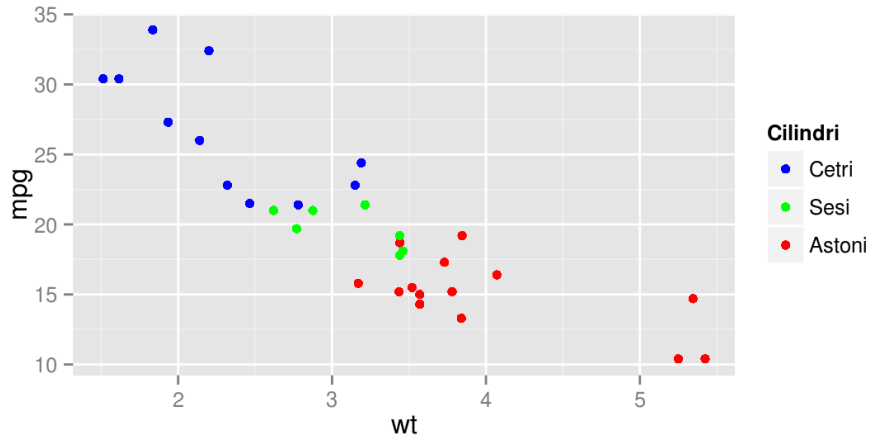
```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_colour_manual(values = c("8" = "red","4" = "blue","6" = "green"))
```



scale_colour_manual()

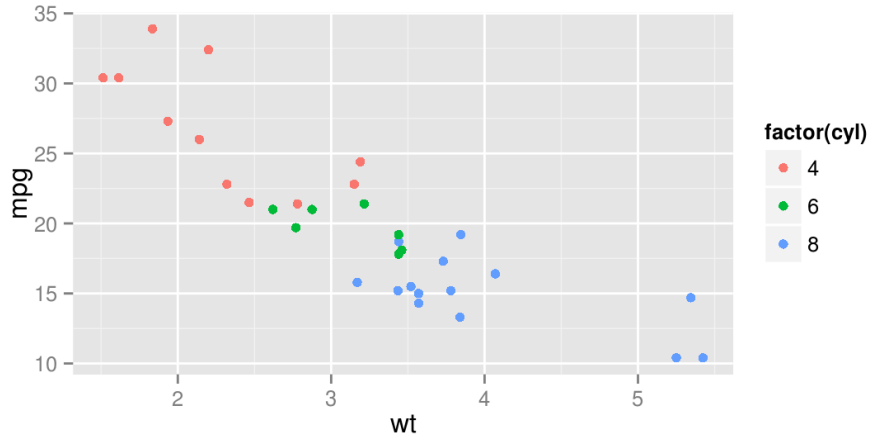
Lai mainītu līmeņu nosaukumus leģendā, jāizmanto argumentu `labels=`.

```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_colour_manual("Cilindri",values = c("4" = "blue","6" = "green","8" = "red"),
                                 labels=c("Cetri","Sesi","Astoni"))
```



scale_continuous()

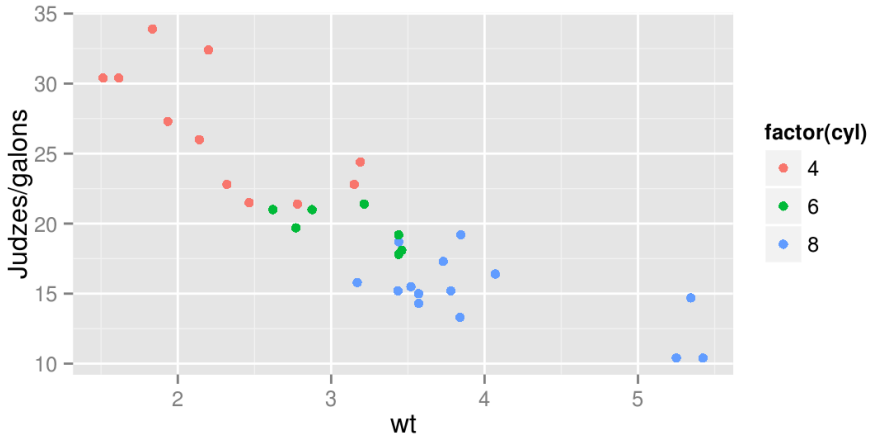
```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point()
```



scale_continuous()

Skaitliskām vērtībām izmanto `scale_.._continuous()`.

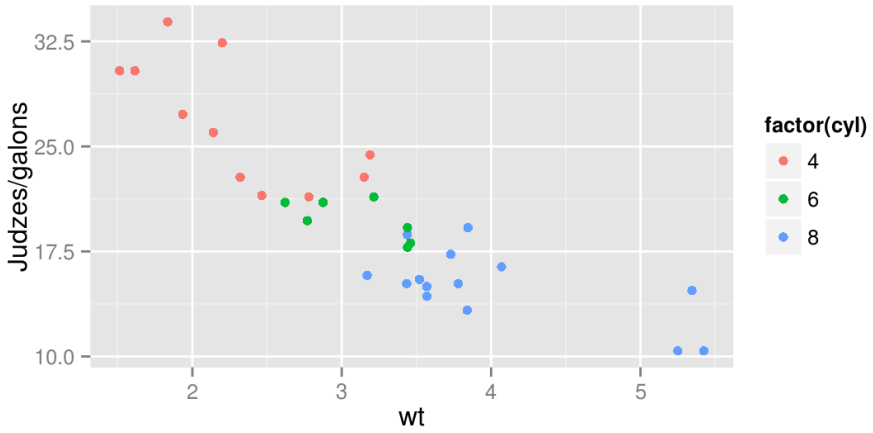
```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_y_continuous("Judzes/galons")
```



scale_continuous()

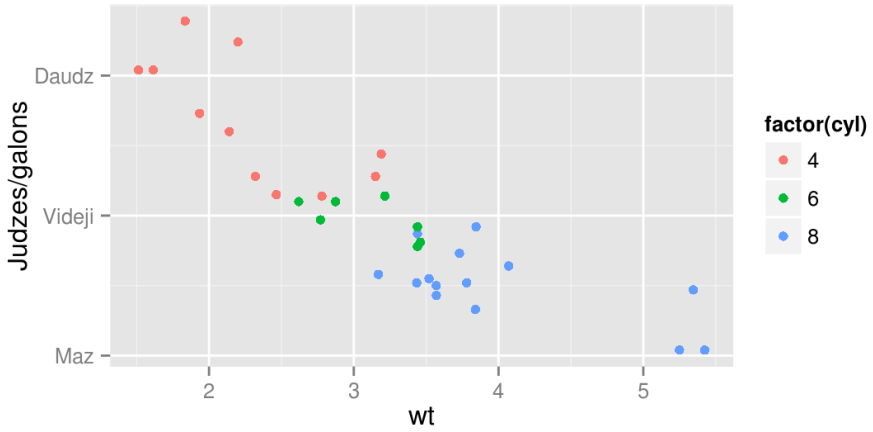
Ar argumentu `breaks=` var noteikt vērtības, pie kādām veikt skalas dalījumu.

```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_y_continuous("Judzes/galons",breaks=seq(10,40,7.5))
```



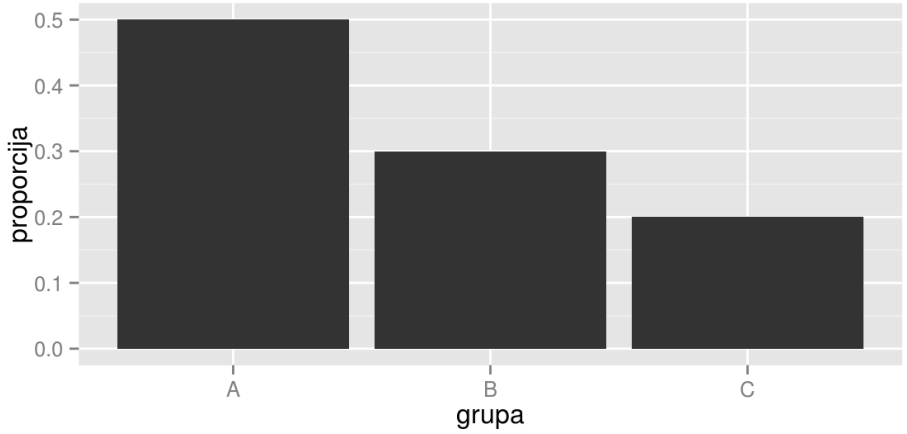
scale_continuous()

```
ggplot(mtcars, aes(x = wt, y = mpg,color=factor(cyl))) + geom_point() +
      scale_y_continuous("Judzes/galons",breaks=c(10,20,30),
                         labels=c("Maz","Videji","Daudz"))
```



scale_continuous()

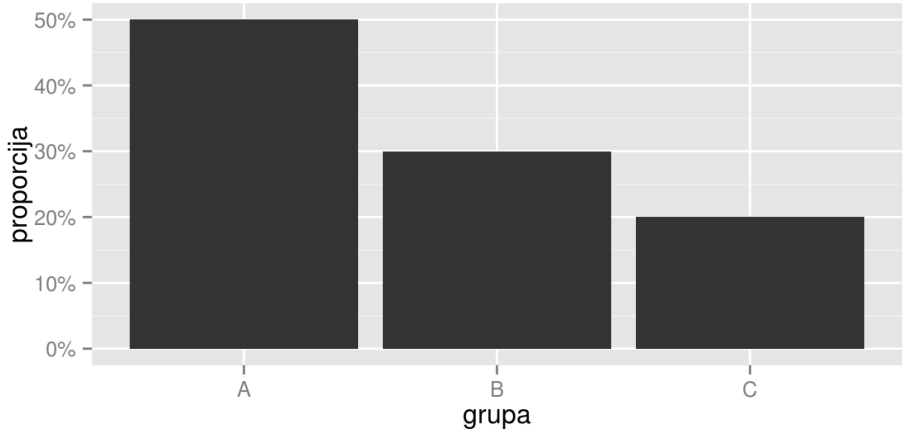
```
df<-data.frame(grupa=c("A","B","C"),proporcija=c(0.5,0.3,0.2))
ggplot(df,aes(grupa,proporcija))+geom_bar(stat="identity")
```



scale_continuous()

Paketē `scales` ir funkcijas, kas ļauj noformēt skalu vērtības, piemēram, funkcija `percent()` skaitļus pārvēšs procentos.

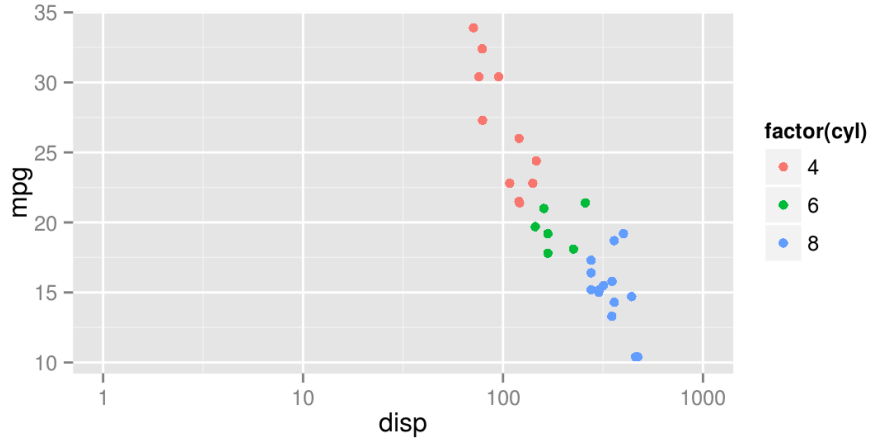
```
library(scales)
ggplot(df,aes(grupa,proporcija))+geom_bar(stat="identity") +
      scale_y_continuous(labels=percent)
```



scale_x_log10()

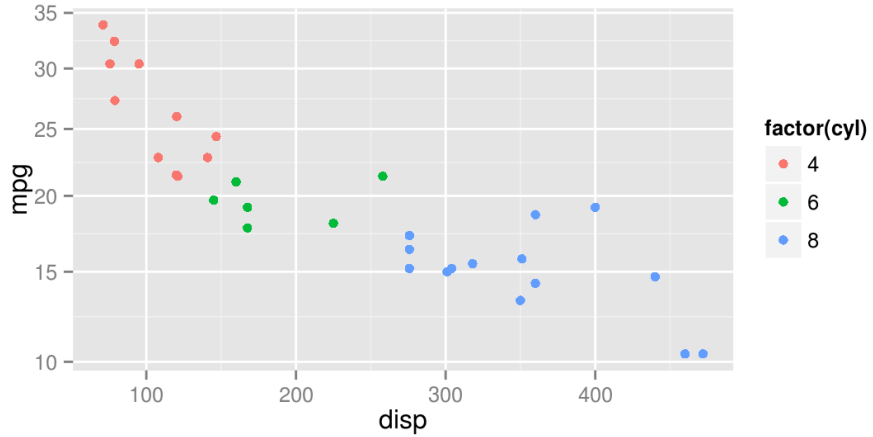
Ar funkcijām `scale_.._log10()`, `scale_.._sqrt()` var veikt automātisku skalu (asu) logaritmisko vai kvadrātsaknes transformāciju.

```
ggplot(mtcars, aes(x = disp, y = mpg,color=factor(cyl))) + geom_point()  +
      scale_x_log10(limits=c(1,1000),breaks=c(1,10,100,1000))
```



scale_y_sqrt()

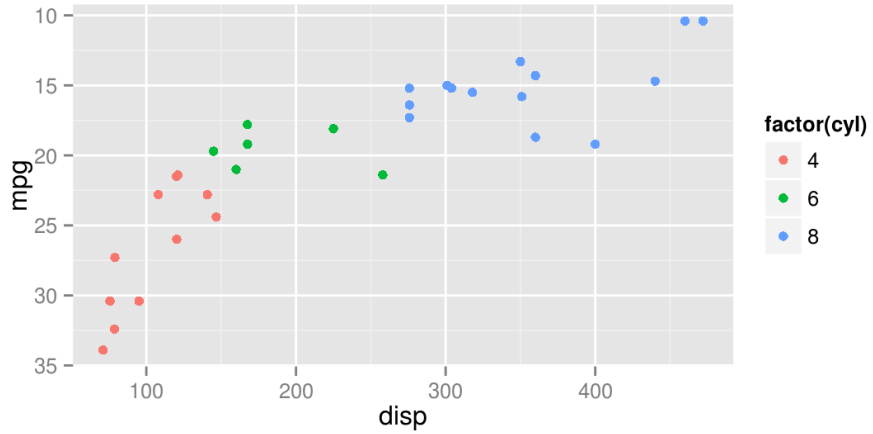
```
ggplot(mtcars, aes(x = disp, y = mpg,color=factor(cyl)))+ geom_point() + scale_y_sqrt()
```



scale_y_reverse()

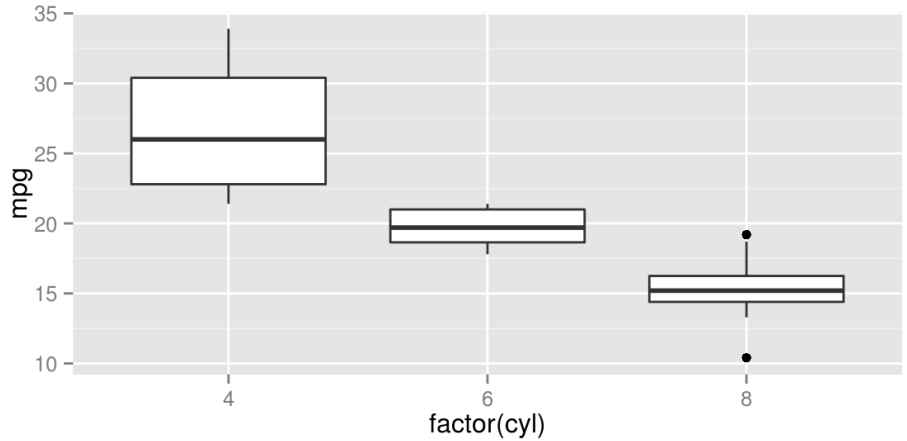
Ar `scale_.._reverse()` var apgriezt vērības pretējā secībā.

```
ggplot(mtcars, aes(x = disp, y = mpg,color=factor(cyl)))+ geom_point() +
      scale_y_reverse()
```



scale_x_discrete()

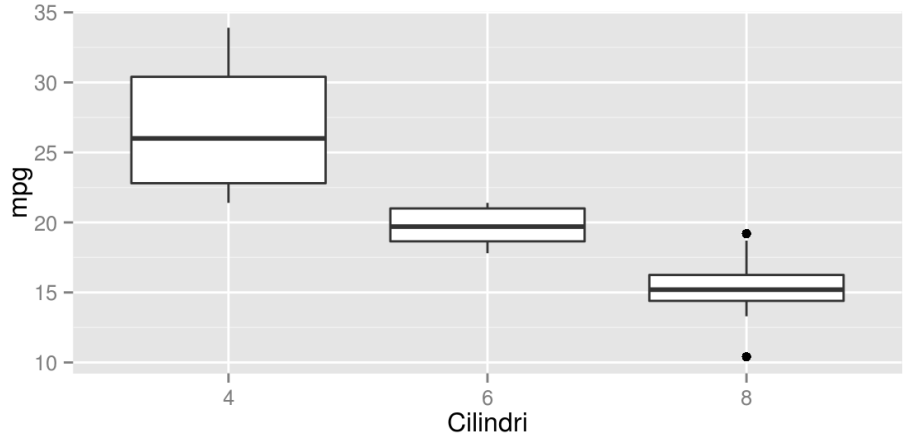
```
ggplot(mtcars, aes(x = factor(cyl), y = mpg))+ geom_boxplot()
```



scale_x_discrete()

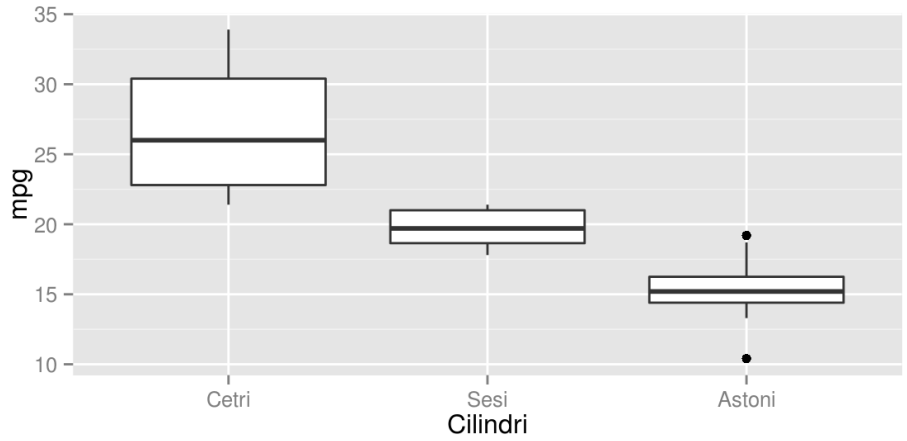
Ja ass vērtības ir faktori (der arī veseliem skaitļiem), tad izmanto `scale_..._discrete()`

```
ggplot(mtcars, aes(x = factor(cyl), y = mpg))+ geom_boxplot() +
      scale_x_discrete("Cilindri")
```



scale_x_discrete()

```
ggplot(mtcars, aes(x = factor(cyl), y = mpg))+ geom_boxplot() +
      scale_x_discrete("Cilindri",labels=c("Cetri","Sesi","Astoni"))
```



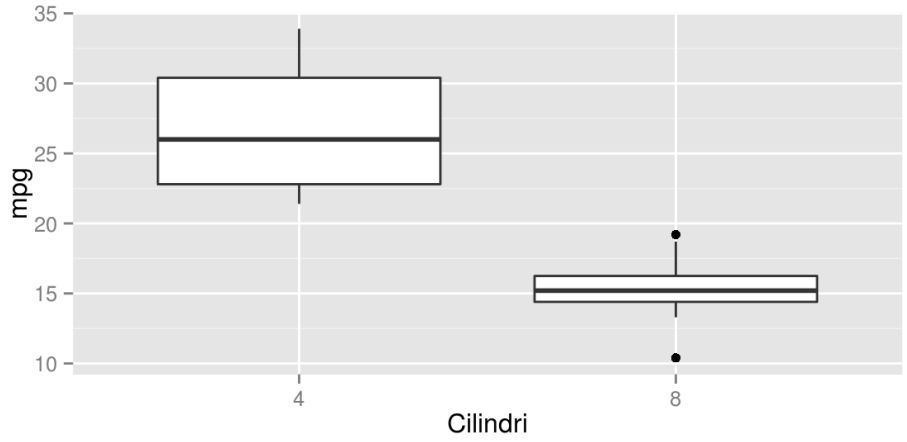
scale_x_discrete()

Ja diskrētai asij norāda argumentu `limits=` un iekļauj tikai dažus no līmeņiem, tad pārējie līmeņi attēlā neparādās.

```
ggplot(mtcars, aes(x = factor(cyl), y = mpg))+ geom_boxplot() +
      scale_x_discrete("Cilindri",limits=c("4","8"))
```

```
## Warning: Removed 2 rows containing missing values (geom_segment).
```

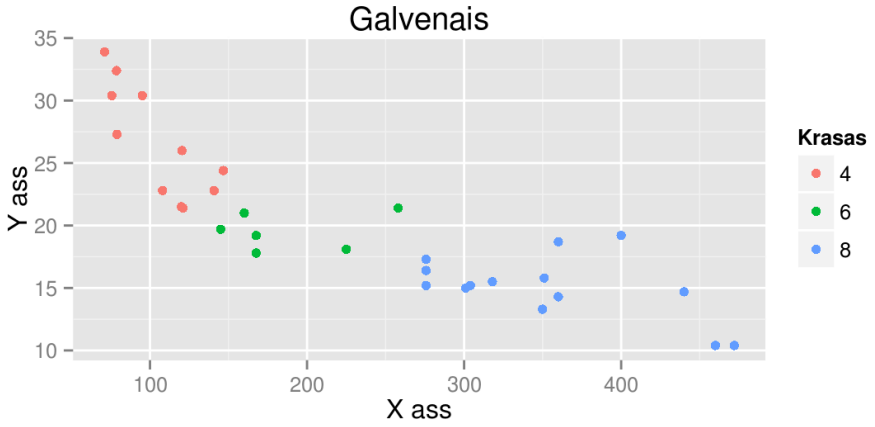
```
## Warning: Removed 1 rows containing missing values (geom_segment).
```



Nosaukumi

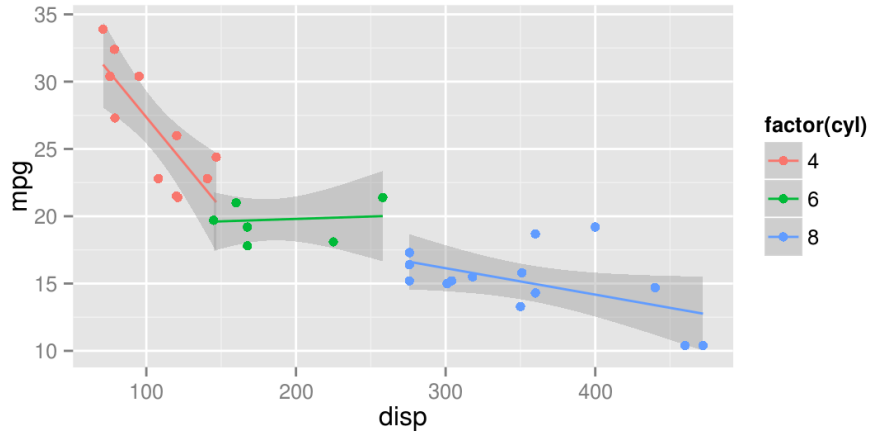
Nosaukums asīm, attēlam leģendai var mainīt ne tikai ar skalu funkcijām, bet arī ar funkciju `labs()` un argumentiem, x=, y=, `title=`, `color=`, `fill=`, `size=`, ...

```
ggplot(mtcars, aes(x = disp, y = mpg,color=factor(cyl)))+ geom_point() +
      labs(x = "X ass",y="Y ass",title="Galvenais",color="Krasas")
```



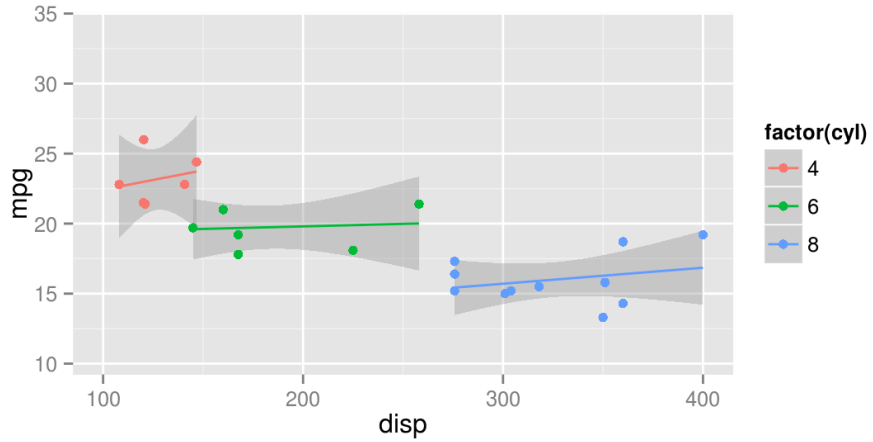
Vērtību diapozons

```
ggplot(mtcars, aes(x = disp, y = mpg,color=factor(cyl)))+
       geom_smooth(method="lm")+geom_point()
```



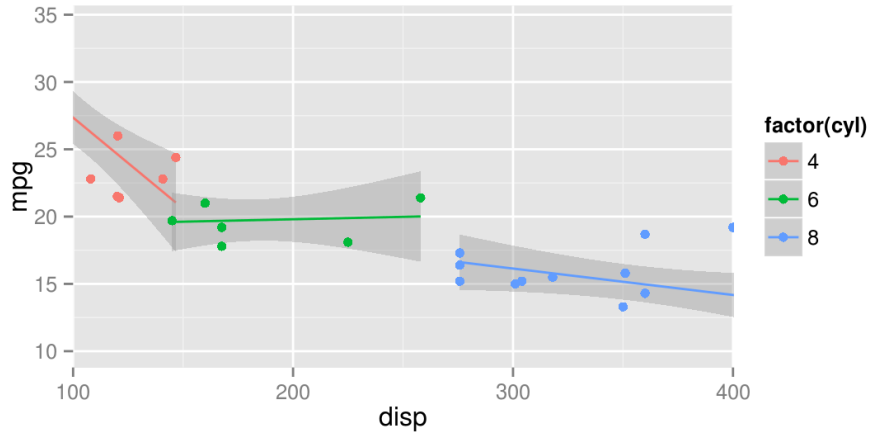
Vērtību diapozons

```
ggplot(mtcars, aes(x = disp, y = mpg,color=factor(cyl)))+
       geom_smooth(method="lm")+geom_point()+
       scale_x_continuous(limits=c(100,400))
```



Vērtību diapozons

```
ggplot(mtcars, aes(x = disp, y = mpg,color=factor(cyl)))+
       geom_smooth(method="lm")+geom_point()+
       coord_cartesian(xlim = c(100, 400))
```



Vērtību diapozons

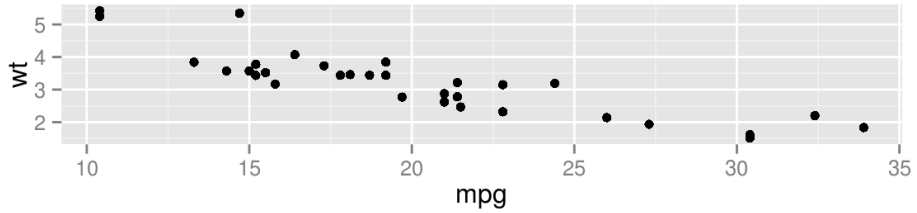
Ja vērtību diapozons tiek noteikts ar argumentu `limits=` kādā no `scale_..._...()` funkcijām, tad visas pārējās vērtības tiek IZSLĒGTAS no datiem.

Ja vērtību diapozons tiek noteikts ar funkciju `coord_cartesian()`, tad esošais grafiks tiek palielināts/samazināts (zoom) līdz noteiktam diapozonam, bet attēla veidošanai izmanto VISAS vērtības.

coord_fixed()

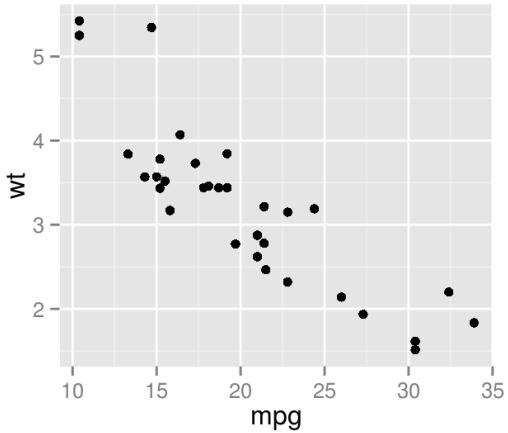
Ar funkciju `coord_fixed()` un argumentu `ratio=` ir iespējams norādīt kādai ir jābūt vērtību attiecībai starp x un y asi. Piemēram, `ratio=1` nozīmē, ka viena vienībā uz x ass ir tikpat gara cik uz y ass.

```
ggplot(mtcars, aes(x = mpg, y = wt)) +geom_point()+coord_fixed(ratio = 1)
```



coord_fixed()

```
ggplot(mtcars, aes(x = mpg, y = wt)) +geom_point()+coord_fixed(ratio = 5)
```



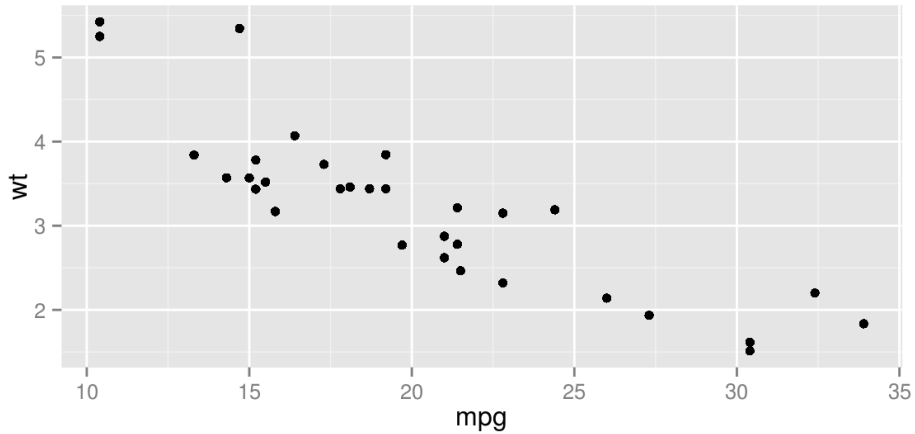
Attēlu sadalīšana daļās

`ggplot2` sistēmā ir iespējams automātiski sadalīt attēlu vairākās daļās balstoties uz vienu vai vairākiem mainīgajiem. To panāk ar funkcijām `facet_grid()` un `facet_wrap()`.

Pirmajā gadījumā tiek izveidots rāmis, kur jānorāda mainīgais, kas dala x ass virzienā un y ass virzienā, bet `facet_wrap()` gadījumā dalījums notiek pēc viena mainīgā, norādot nepieciešamo rindu vai kolonnu skaitu.

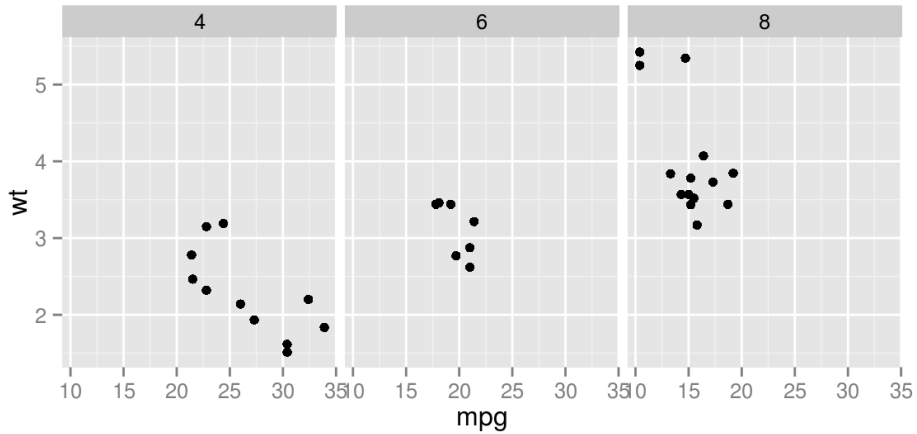
facet_grid()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point()
```



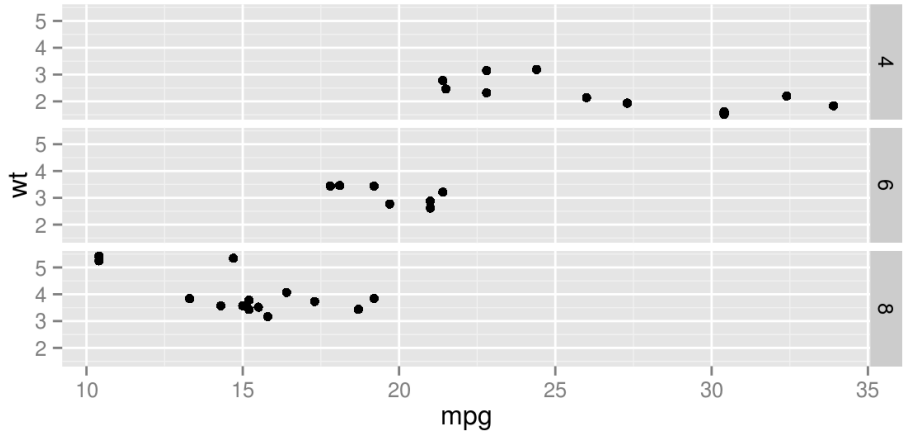
facet_grid()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_grid(. ~ cyl)
```



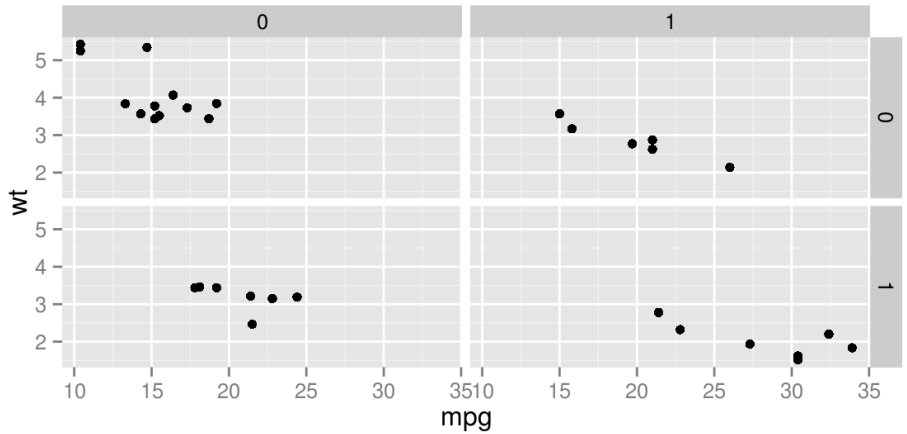
facet_grid()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_grid(cyl ~ .)
```



facet_grid()

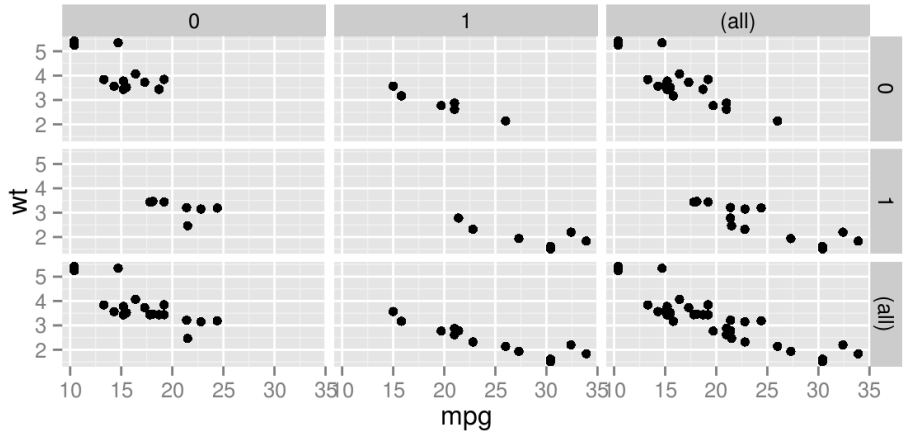
```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_grid(vs ~ am)
```



facet_grid()

Ja funkcijai `facet_grid()` pieliek argumentu margins=TRUE, tad tiek izveidoti arī faktoru kombināciju attēli.

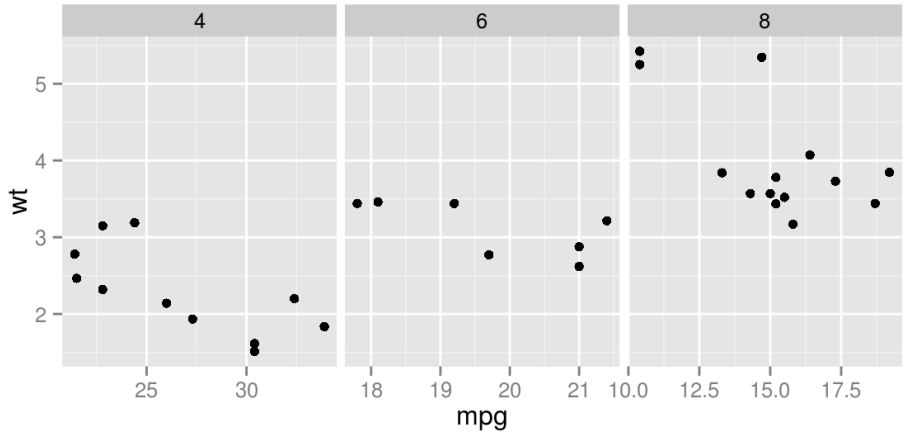
```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_grid(vs ~ am, margins=TRUE)
```



facet_grid()

Ar argumentu `scales="free"` panāk, ka mazajiem attēliem ir atšķirīgas vērtību skalas.

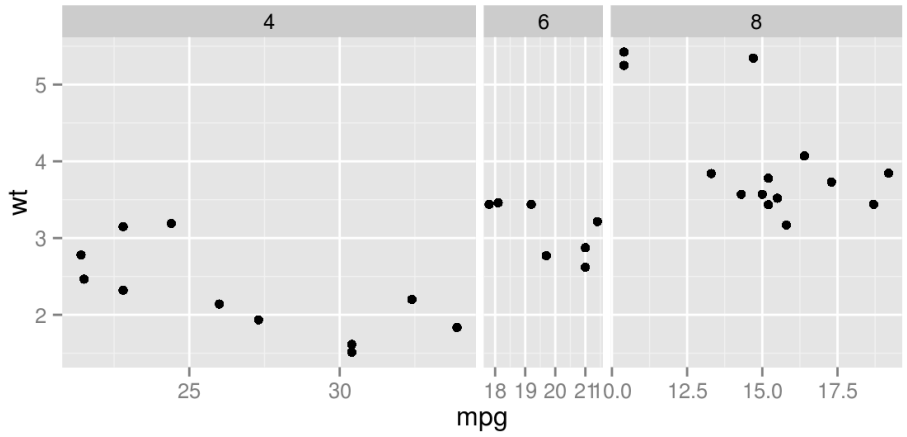
```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_grid(. ~ cyl, scales = "free")
```



facet_grid()

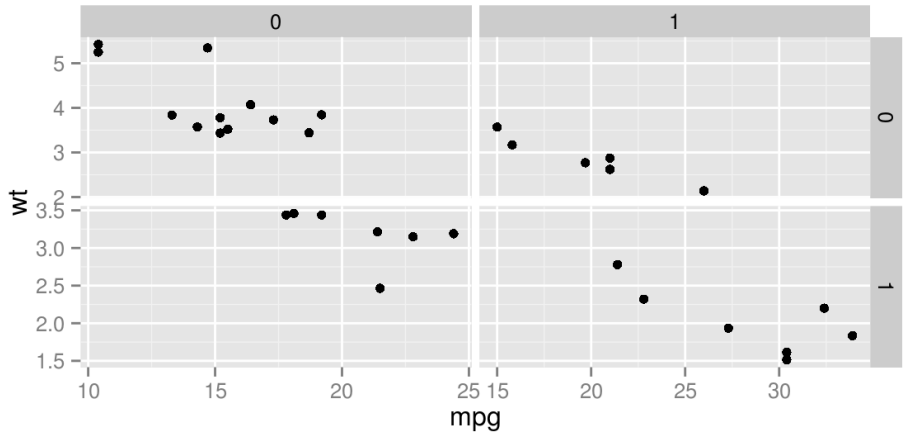
Arguments `space="free"` nodrošina, ka mazajiem attēliem atvēlētā vieta mainās atkarībā no attēlojamo datu diapazona.

```
ggplot(mtcars, aes(mpg, wt)) + geom_point() +
      facet_grid(. ~ cyl, scales = "free", space = "free")
```



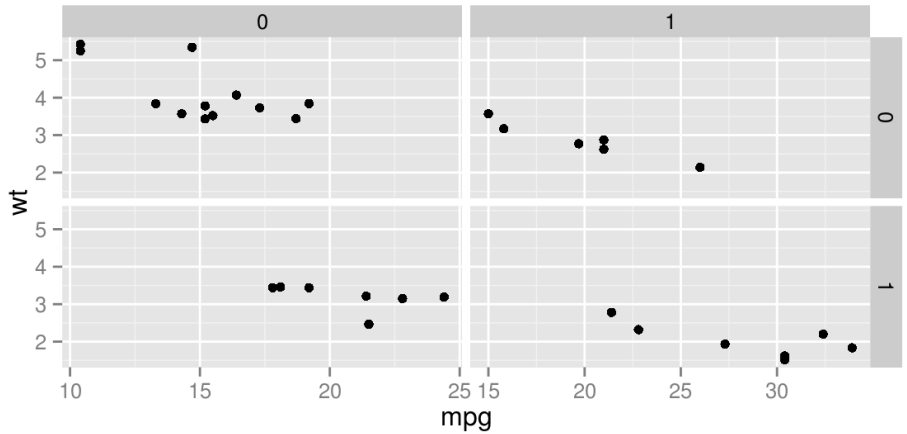
facet_grid()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_grid(vs ~ am, scales = "free")
```



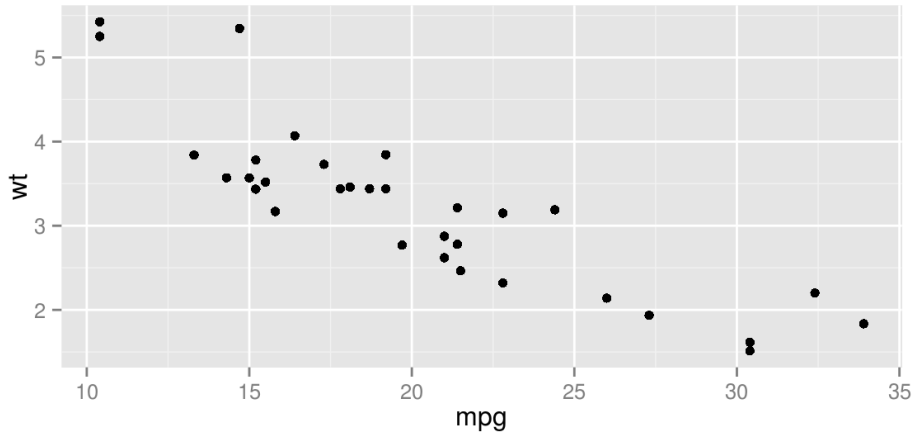
facet_grid()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_grid(vs ~ am, scales = "free_x")
```



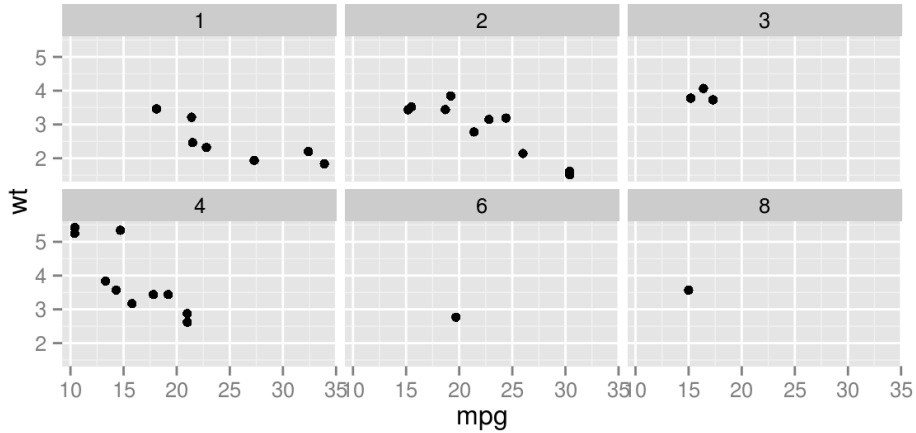
facet_wrap()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point()
```



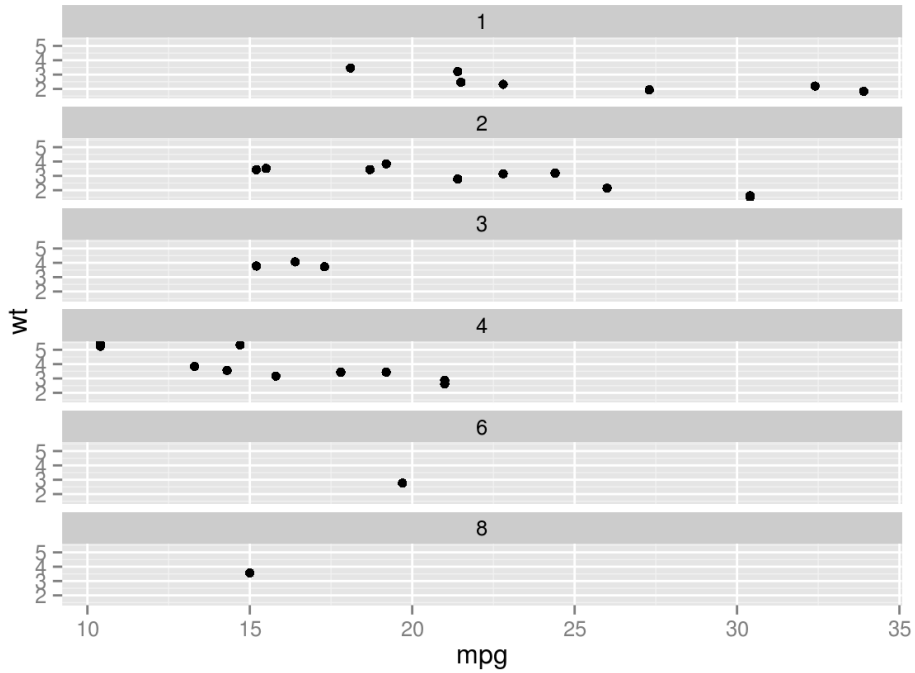
facet_wrap()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_wrap(~ carb)
```



facet_wrap()

```
ggplot(mtcars, aes(mpg, wt)) + geom_point() + facet_wrap(~ carb,ncol=1)
```



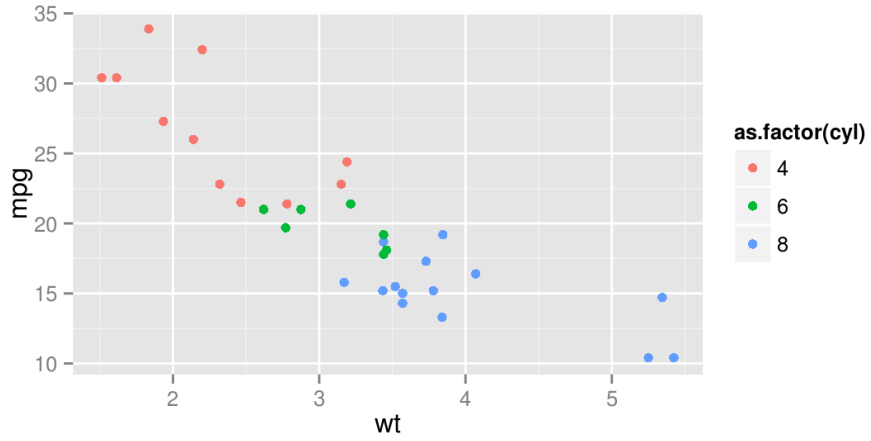
Attēla izskata maiņa

Lai mainītu attēla izskatu, var izmantot gatavas tēmas vai arī mainīt katru elementu atsevišķi izmantojot funkciju `theme()`.

Gatavās tēmas, piemēram, ir `theme_bw()` vai `theme_minimal()`.

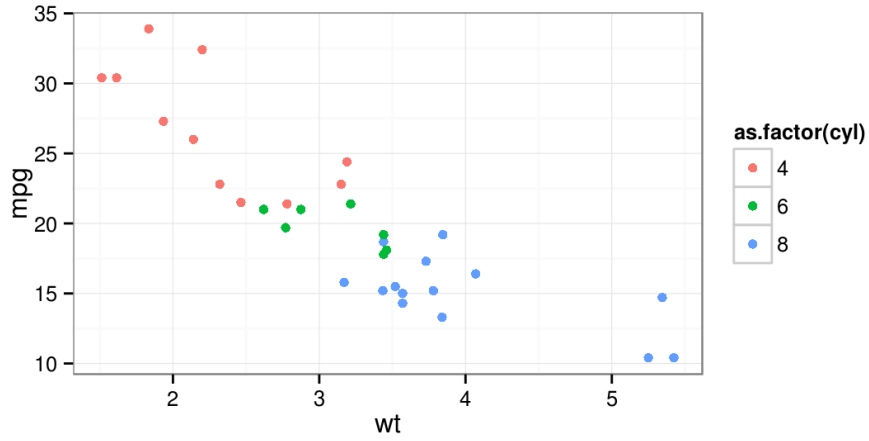
theme_bw()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point()
```



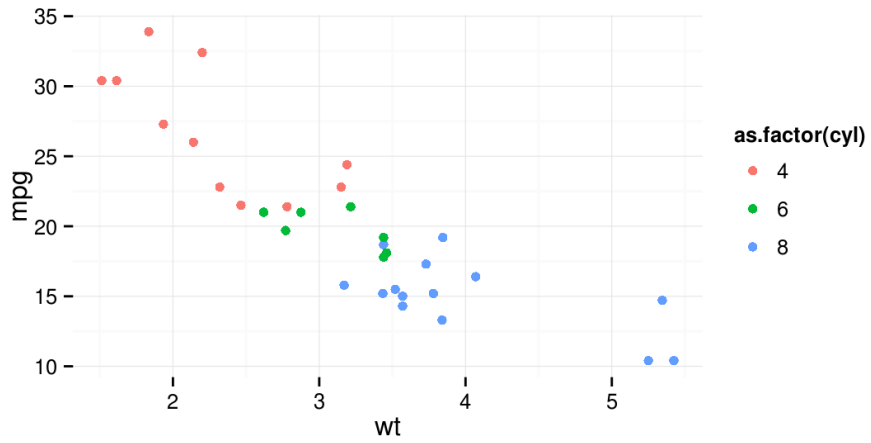
theme_bw()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() + theme_bw()
```



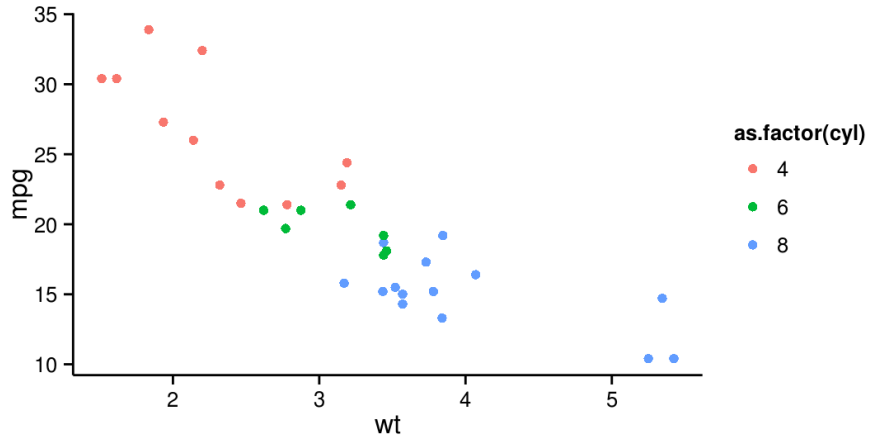
theme_minimal()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme_minimal()
```



theme_classic()

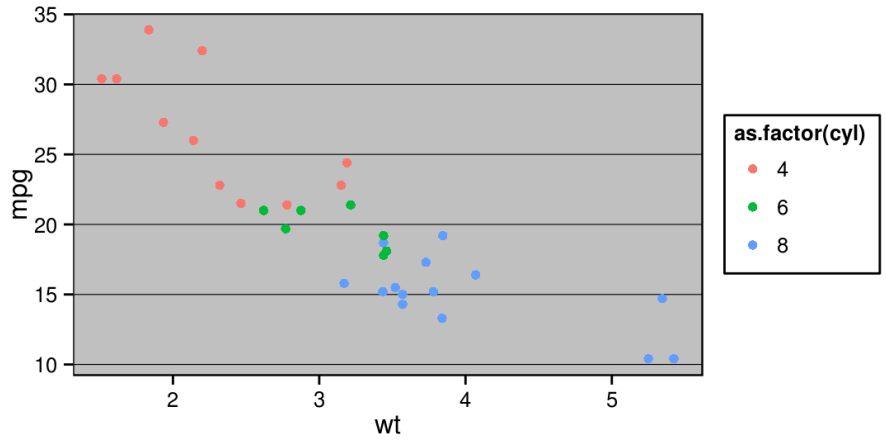
```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme_classic()
```



theme_excel()

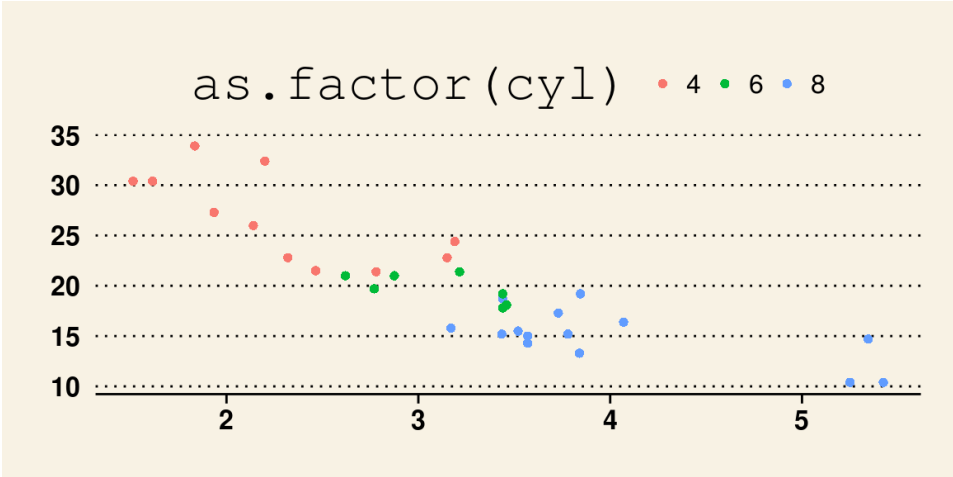
Paketē ggthemes ir vairākas papildus noformējuma tēmas.

```
library(ggthemes)
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme_excel()
```



theme_wsj()

```
library(ggthemes)
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() + theme_wsj()
```



theme()

Argumenti attēla elementu mainīšanai

- axis.title - asu paraksti (element_text)
- axis.title.x - x ass paraksts (element_text)
- axis.title.y - y ass paraksts (element_text)
- axis.text - apzīmējumi pie asīm (element_text)
- axis.text.x - apzīmējumi pie x ass (element_text)
- axis.text.y - apzīmējumi pie y ass (element_text)
- axis.ticks - nogriežņi pie asīm (element_line)
- axis.ticks.x - nogriežņi pie x ass (element_line)
- axis.ticks.y - nogriežņi pie y ass (element_line)
- axis.ticks.length - nogriežņu garums pie asīm (unit)
- axis.ticks.margin - atstarpe starp ass apzīmējumu un nogriežņiem (unit)
- axis.line - līnijas gar asīm (element_line)
- axis.line.x - līnijas pie x ass (element_line)
- axis.line.y - līnijas pie y ass (element_line)

theme()

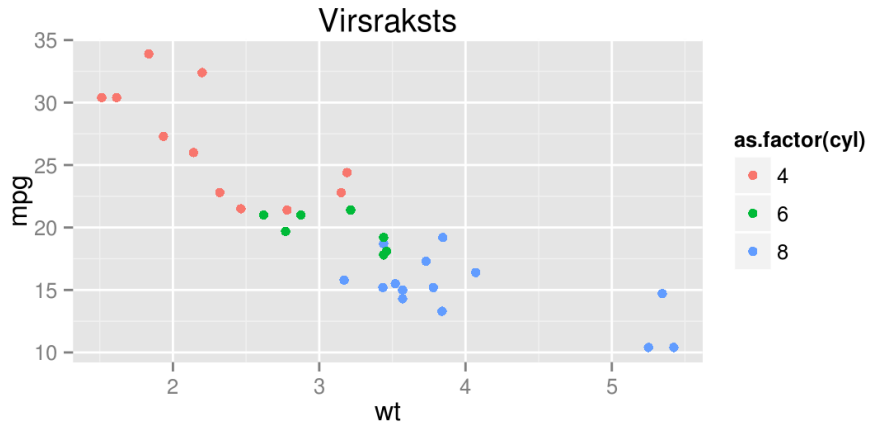
- legend.background - leģendas pamatne (element_rect)
- legend.margin - papildus atstarpe apkārt leģendai (unit)
- legend.key - pamatne zem leģendas ierakstiem (element_rect)
- legend.key.size - leģendas ierakstu izmērs (unit)
- legend.key.height - leģendas ieraksta pamatnes augstums (unit)
- legend.key.width - leģendas ieraksta pamatnes platums (unit)
- legend.text - leģendas ieraksti (element_text)
- legend.text.align - leģendas teksta novietojums (skaitlis no 0 līdz 1)
- legend.title - leģendas virsraksts (element_text)
- legend.title.align - leģendas virsraksta novietojums (skaitlis no 0 līdz 1)
- legend.position - leģendas novietojums ("left", "right", "bottom", "top", vai divu skaitļu vektors)
- legend.direction - leģendas ierakstu izvietojums ("horizontal" or "vertical")
- legend.justification - leģendas novietojums grafika iekšienē ("center" vai divu skaitļu vektors)
- legend.box - vairāku leģendu novietojums ("horizontal" or "vertical")
- panel.background - grafika iekšienes pametne (element_rect)
- panel.border - robeža apkārt grafika iekšienei (element_rect)

theme()

- panel.margin - mala ap atsevišķiem grafikiem (facet) (unit)
- panel.grid - grid lines (element_line)
- panel.grid.major - galvenās grid lines (element_line)
- panel.grid.minor - mazās grid lines (element_line)
- panel.grid.major.x - vertikālās galvenās grid lines (element_line)
- panel.grid.major.y - horizontālās galvenās grid lines (element_line)
- panel.grid.minor.x - vertikālās mazās grid lines (element_line)
- panel.grid.minor.y - horizontālās mazās grid lines (element_line)
- plot.background - visa grafika pamatne (element_rect)
- plot.title - grafika virsraksts (element_text)
- plot.margin - mala apkārt visam grafikam (unit)
- strip.background - atsevišķu grafiku (facet) uzrakstu pamatne (element_rect)
- strip.text - atsevišķu grafiku (facet) uzraksts (element_text)
- strip.text.x - atsevišķu grafiku (facet) uzraksts horizontālā virzienā (element_text)
- strip.text.y - atsevišķu grafiku (facet) uzraksts vertikālā virzienā (element_text)

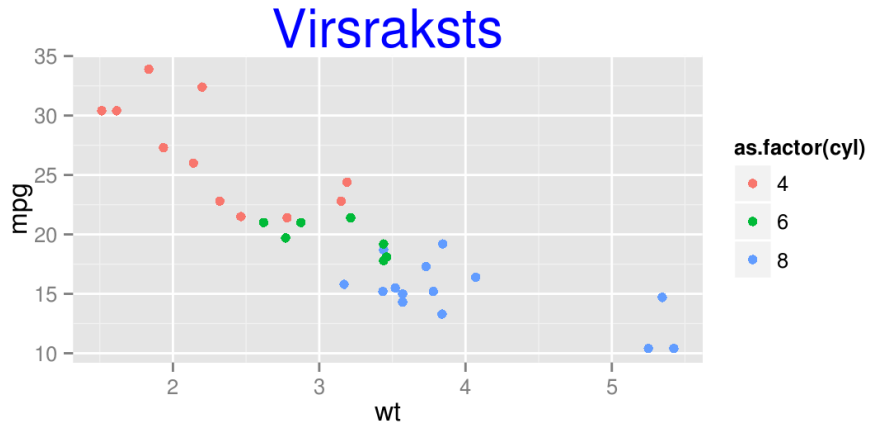
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point()+
      labs(title="Virsraksts")
```



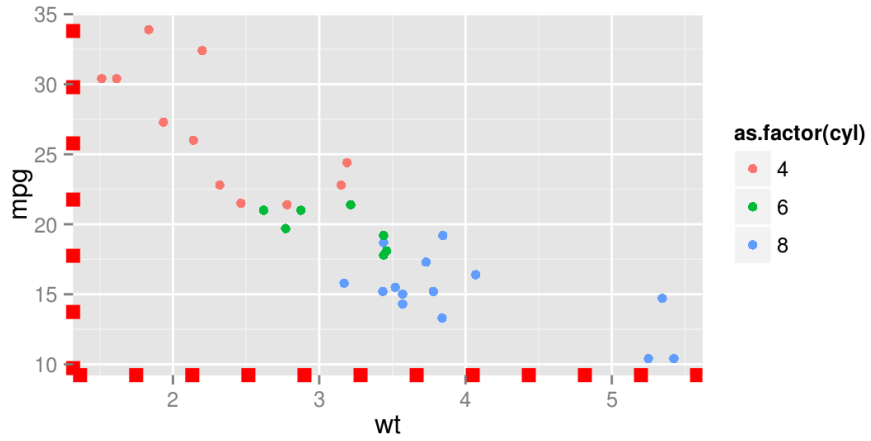
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point()+
      labs(title="Virsraksts") +
      theme(plot.title = element_text(size = rel(2),colour="blue"))
```



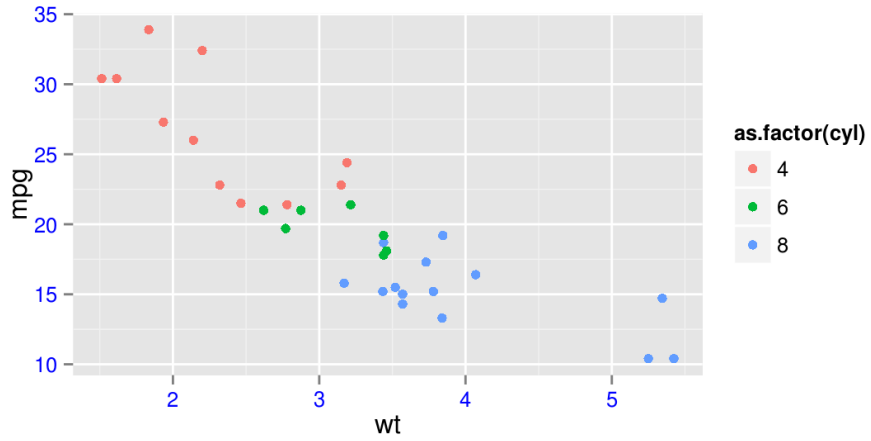
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(axis.line = element_line(size = 3, colour = "red", linetype = "dotted"))
```



theme()

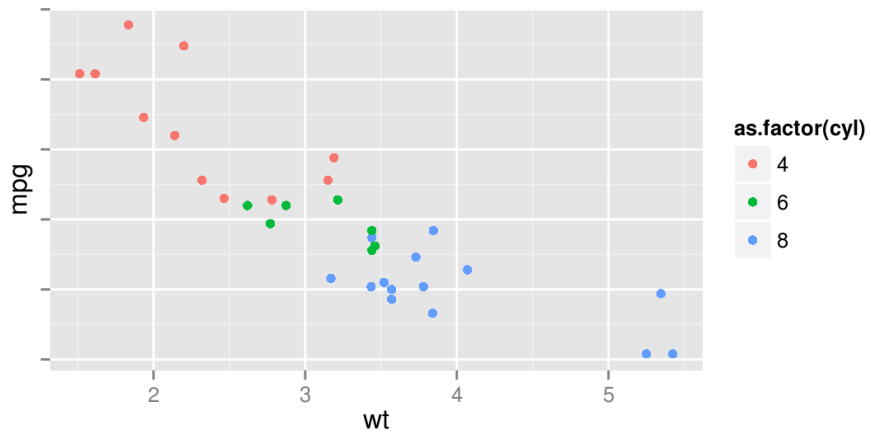
```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(axis.text = element_text(colour = "blue"))
```



theme()

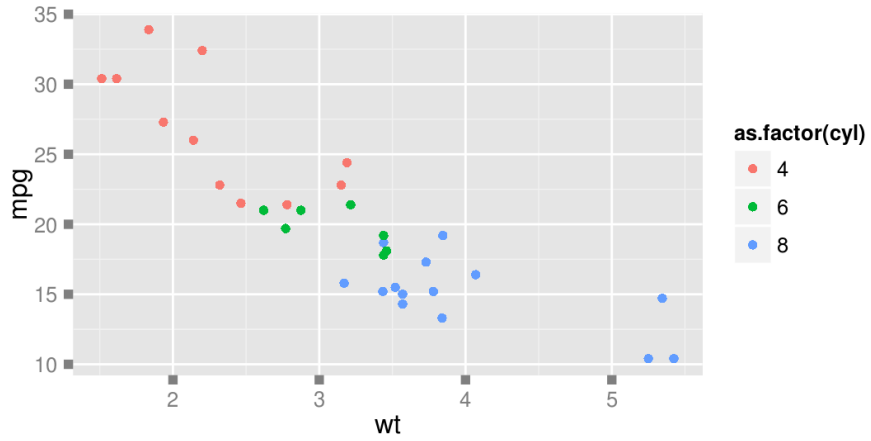
Jā kādu no elementiem nepieciešams pilnībā izslēgt/paslēpt, izmanto argumentu `element_blank()`.

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(axis.text.y = element_blank())
```



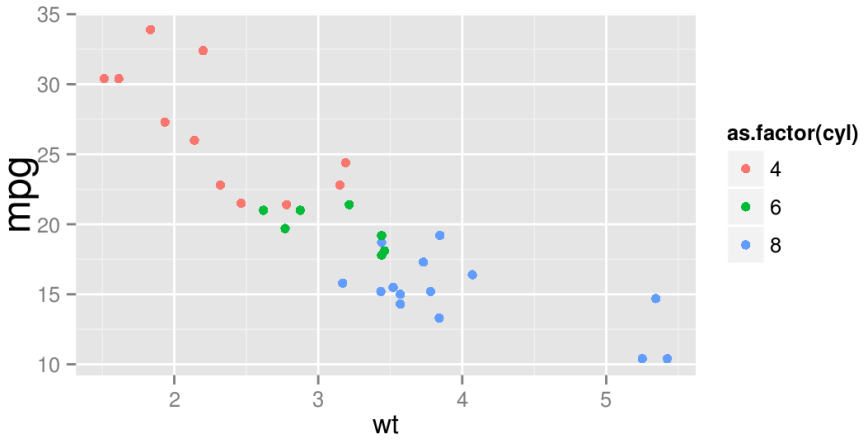
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(axis.ticks = element_line(size = 2))
```



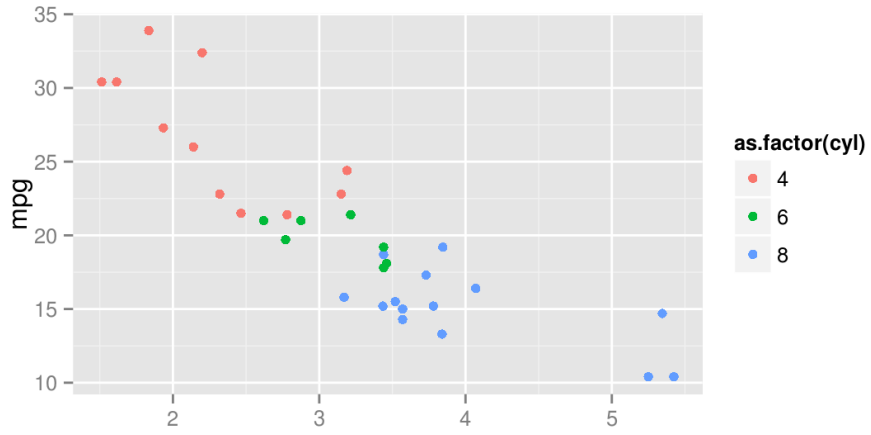
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(axis.title.y = element_text(size = rel(1.5), angle = 90))
```



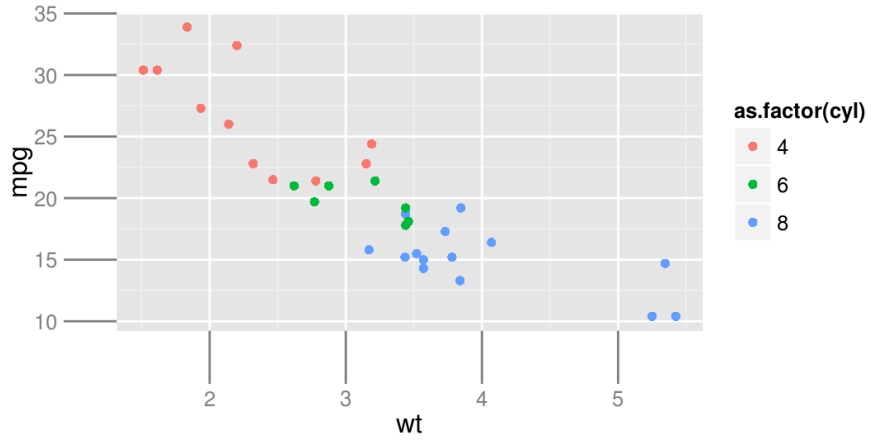
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(axis.title.x = element_blank())
```



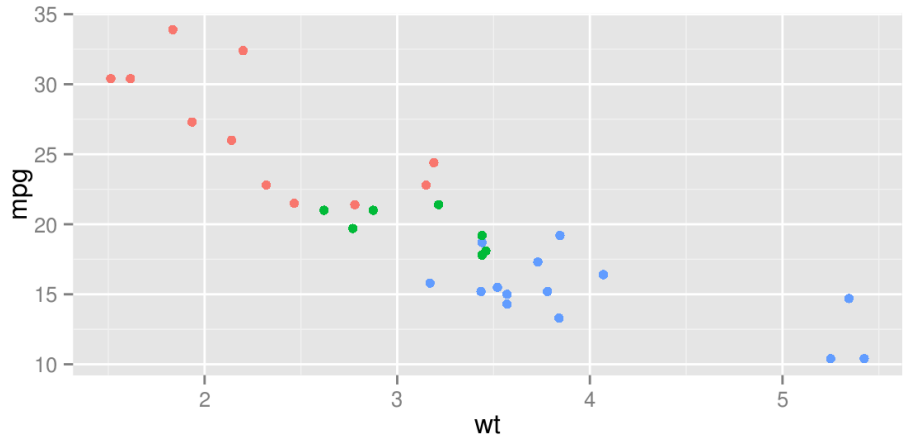
theme()

```
library(grid)
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(axis.ticks.length = unit(.85, "cm"))
```



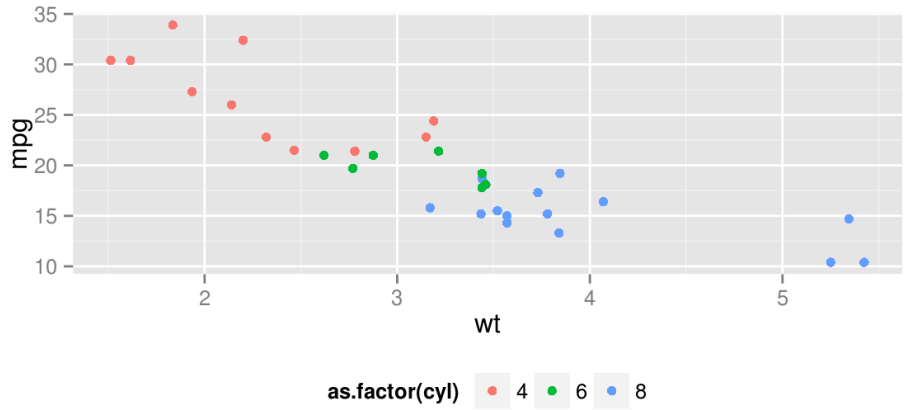
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.position = "none")
```



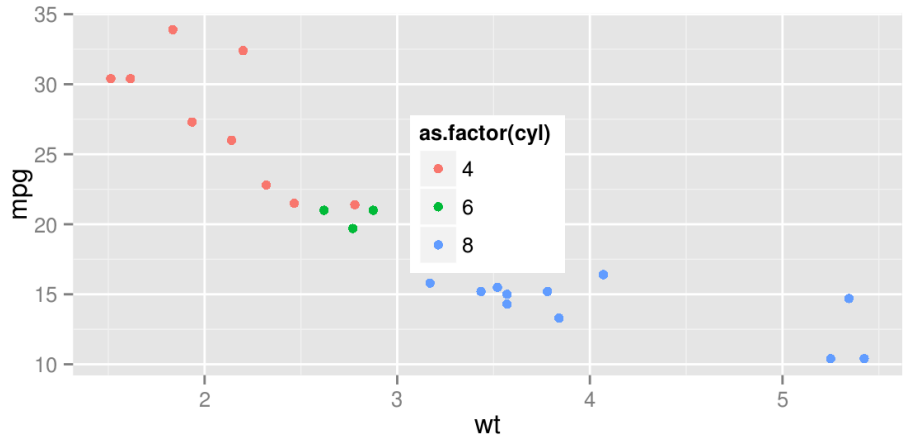
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.position = "bottom")
```



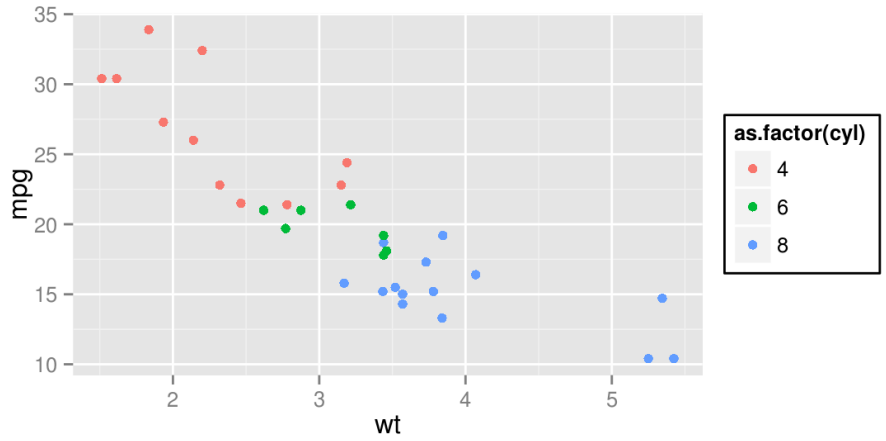
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.position = c(.5, .5))
```



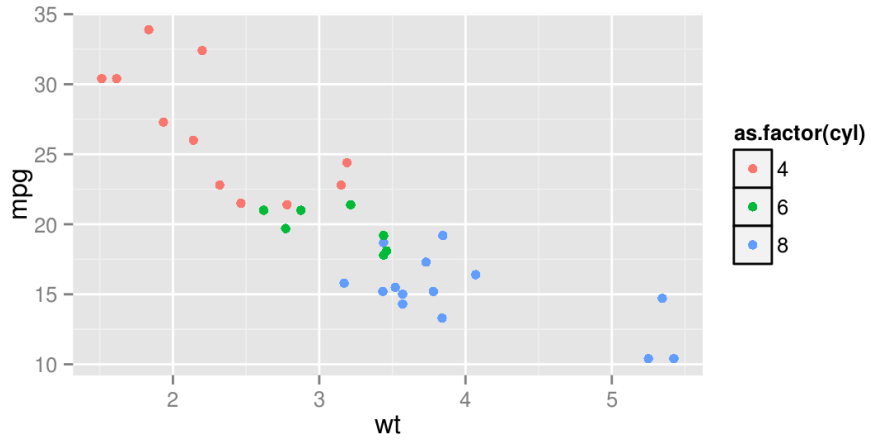
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.background = element_rect(colour = "black"))
```



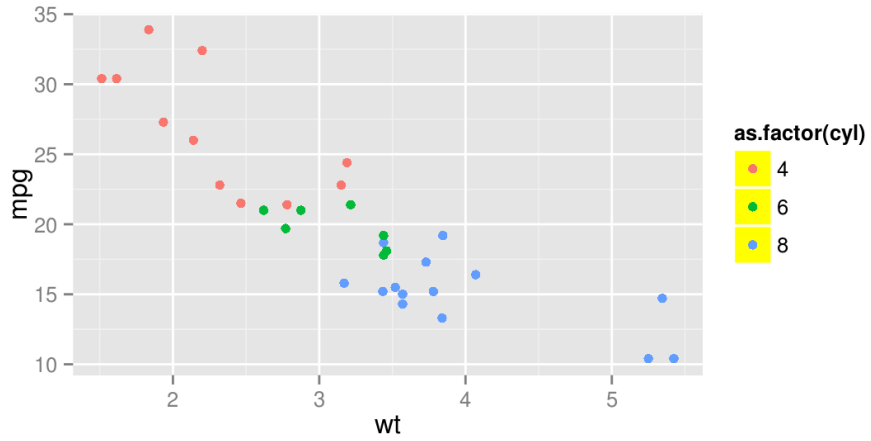
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.key = element_rect(colour = "black"))
```



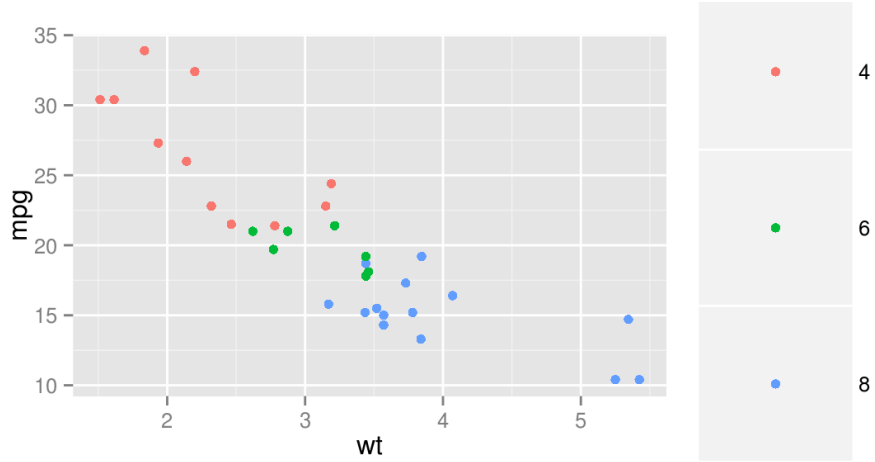
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.key = element_rect(fill = "yellow"))
```



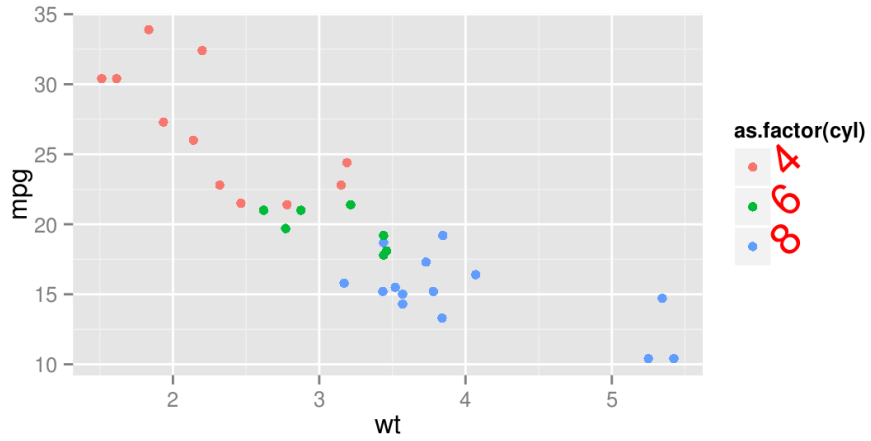
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.key.size = unit(2.5, "cm"))
```



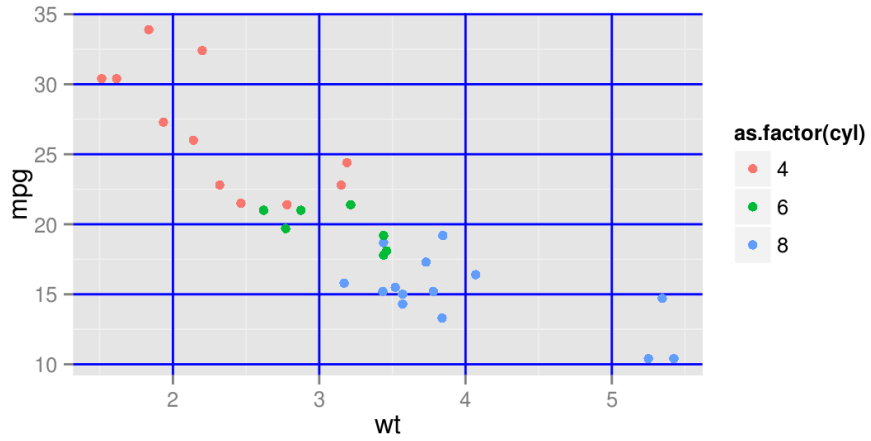
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(legend.text = element_text(size = 20, colour = "red", angle = 45))
```



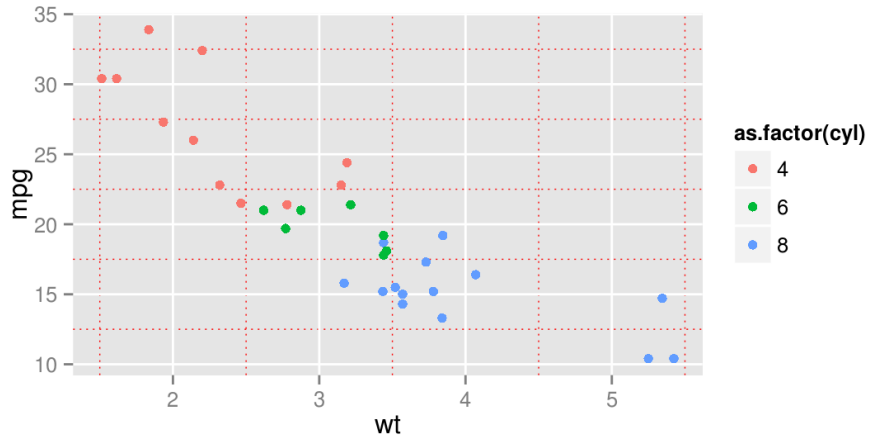
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(panel.grid.major = element_line(colour = "blue"))
```



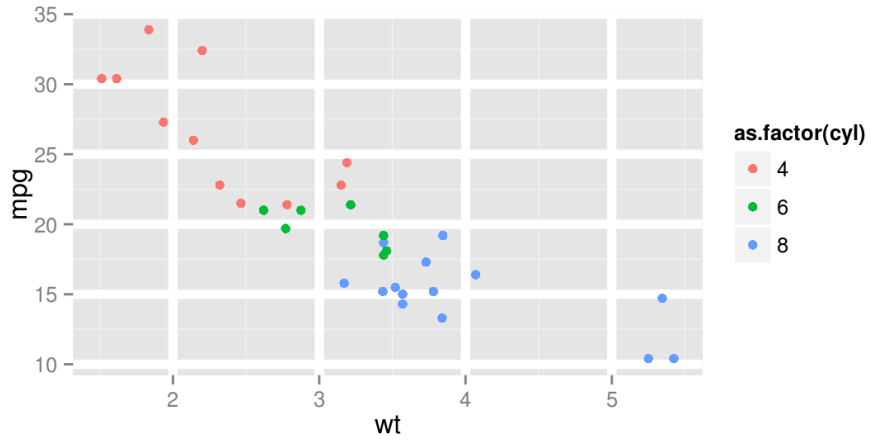
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(panel.grid.minor = element_line(colour = "red", linetype = "dotted"))
```



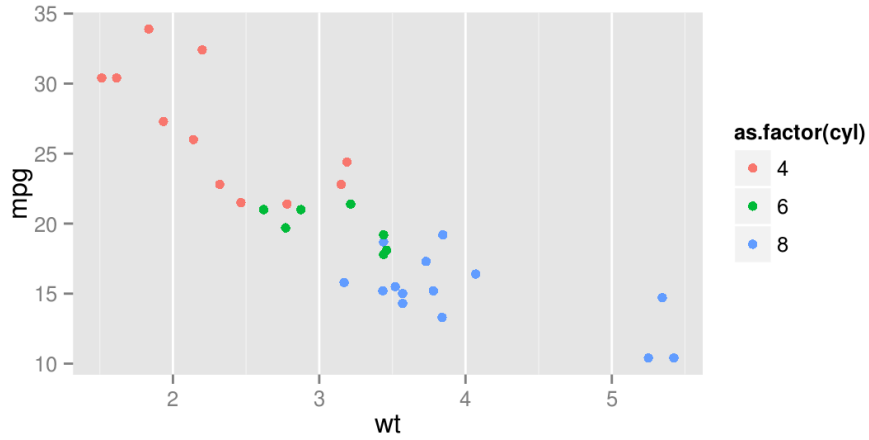
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(panel.grid.major = element_line(size = 2))
```



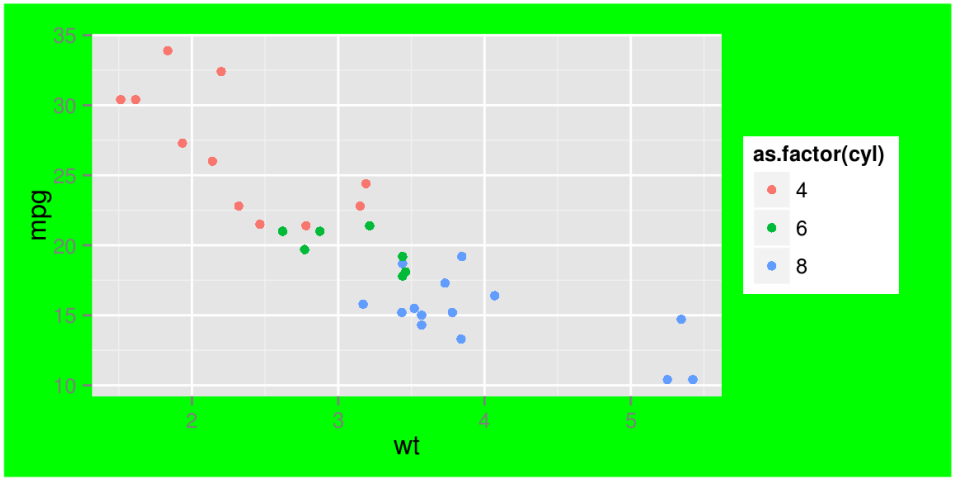
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(panel.grid.major.y = element_blank(),
            panel.grid.minor.y = element_blank())
```



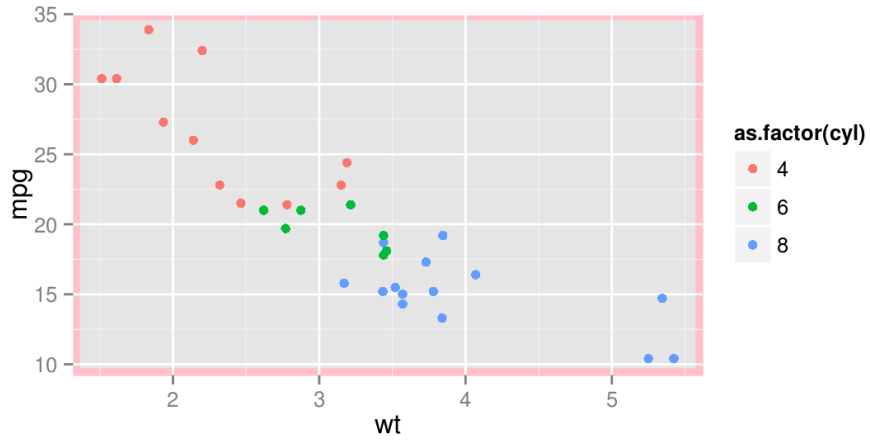
theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(plot.background = element_rect(fill = "green"))
```



theme()

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point() +
      theme(panel.background = element_rect(colour = "pink",size=3))
```

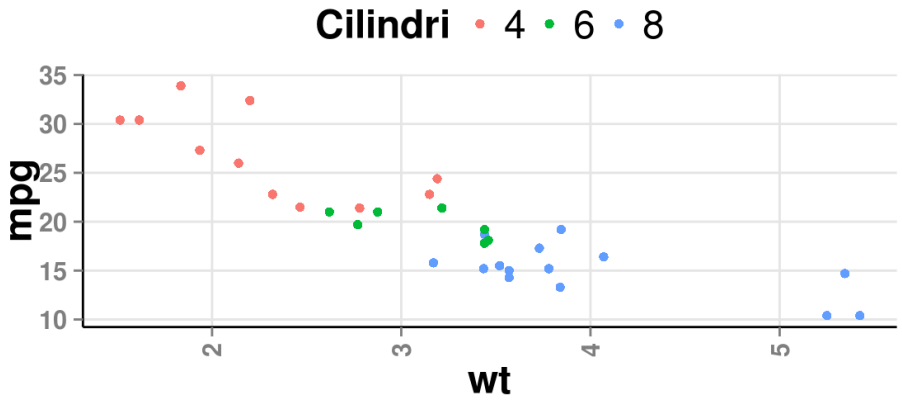


theme()

Katrs elements nav jānorāda savā `theme()` funkcijā, bet tos visus var iekļaut vienā funkcijā.

```
ggplot(mtcars, aes(x = wt, y = mpg,colour = as.factor(cyl))) + geom_point()+
      labs(color="Cilindri")+
  theme(axis.text.y=element_text(size=rel(1.2),face="bold"),
        axis.text.x=element_text(size=rel(1.2),face="bold",angle=90,vjust=0.5),
        axis.title=element_text(size=rel(1.5),face="bold"),
        axis.line=element_line(color="black"),
        panel.background=element_blank(),
        panel.grid.minor=element_blank(),
        panel.grid.major=element_line(color="grey90"),
        legend.position="top",
        legend.key=element_rect(fill="white"),
        legend.title=element_text(size=rel(1.5)),
        legend.text=element_text(size=rel(1.5)))
```

theme()



Attēlu saglabāšana

ggplot2 attēlus var saglabāt arī pēc to izveidošanas.

Lai saglabātu attēlu, izmanto funkciju `ggsave()`, kurā jānorāda vēlamais faila nosaukums ar paplašinājumu, izmērs (pēc noklusējuma collās). Šo komandu izpilda kā pēdējo.

```
ggsave("1_attels.png",width=8,height=5)
```

Attēla veidošana no objektiem

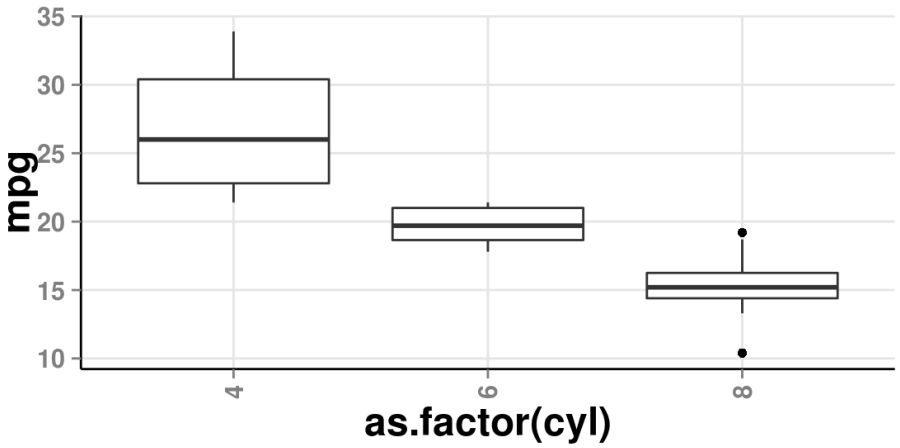
Gan pašu attēlu, gan arī noformējuma nosacījumus var saglabāt kā atsevišķus argumentus un izmantot atkārtoti.

```
p<-ggplot(mtcars,aes(as.factor(cyl),mpg))+geom_boxplot()

noformejums<-  theme(axis.text.y=element_text(size=rel(1.2),face="bold"),
        axis.text.x=element_text(size=rel(1.2),face="bold",angle=90,vjust=0.5),
        axis.title=element_text(size=rel(1.5),face="bold"),
        axis.line=element_line(color="black"),
        panel.background=element_blank(),
        panel.grid.minor=element_blank(),
        panel.grid.major=element_line(color="grey90"),
        legend.position="top",
        legend.key=element_rect(fill="white"),
        legend.title=element_text(size=rel(1.5)),
        legend.text=element_text(size=rel(1.5)))
```

Attēla veidošana no objektiem

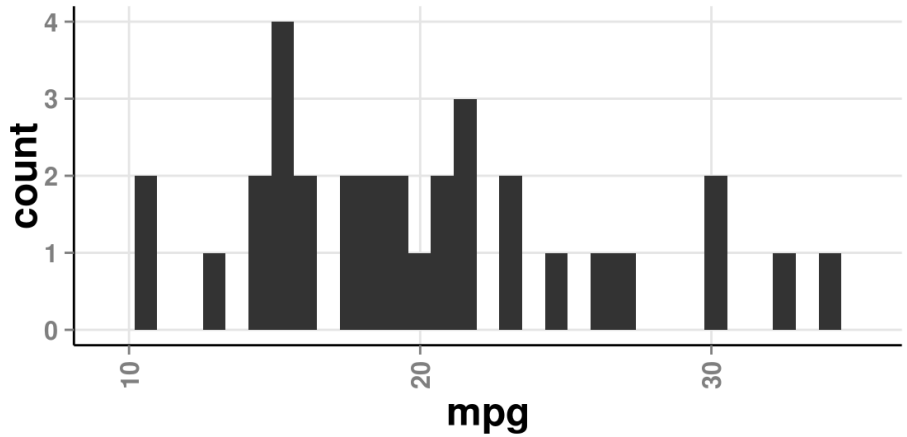
```
p+noformejums
```



Attēla veidošana no objektiem

```
ggplot(mtcars,aes(mpg))+geom_histogram()+noformejums
```

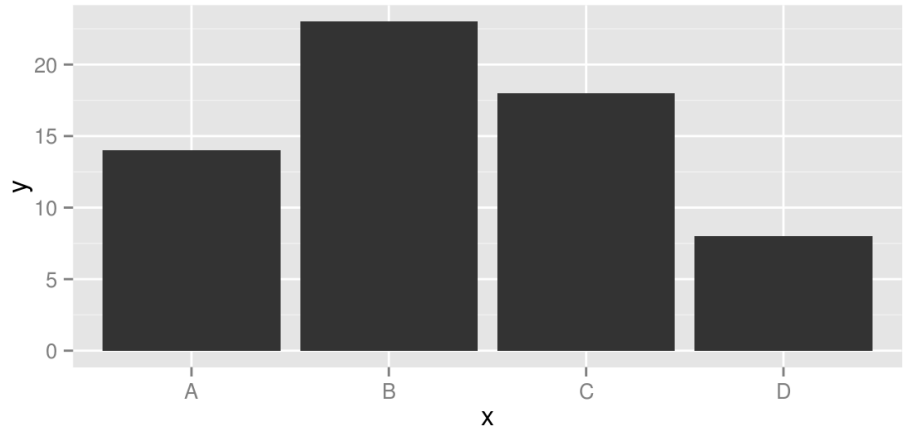
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



Tipiskās problēmas veidojot ggplot2 attēlus

Stabiņu secība

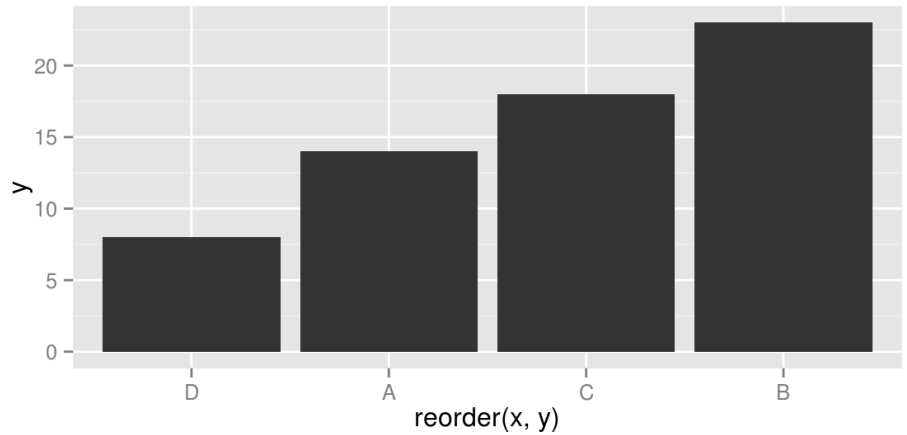
```
df<-data.frame(x=c("A","B","C","D"),y=c(14,23,18,8))
ggplot(df,aes(x,y))+geom_bar(stat="identity")
```



Stabiņu secība

Jāizmanto funkcija `reorder()`, kurā norāda mainīgo pēc kura kārtot

```
ggplot(df,aes(reorder(x,y),y))+geom_bar(stat="identity")
```



Vairākas līnijas attēlā

```
df<-data.frame(
  gads=1900:2000,
  viens=rep(1,101),
  divi=rep(2,101),
  tris=rep(3,101))
head(df)
```

```
##   gads viens divi tris
## 1 1900     1    2    3
## 2 1901     1    2    3
## 3 1902     1    2    3
## 4 1903     1    2    3
## 5 1904     1    2    3
## 6 1905     1    2    3
```

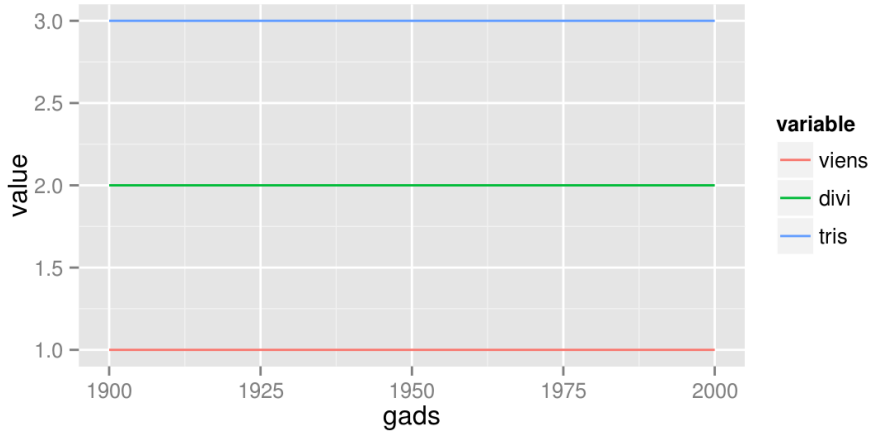
Vairākas līnijas attēlā

```
library(reshape2)
df.long<-melt(df,id.vars="gads")
head(df.long)
```

```
##   gads variable value
## 1 1900    viens     1
## 2 1901    viens     1
## 3 1902    viens     1
## 4 1903    viens     1
## 5 1904    viens     1
## 6 1905    viens     1
```

Vairākas līnijas attēlā

```
ggplot(df.long,aes(gads,value,color=variable))+geom_line()
```



Vairākas līnijas attēlā

```
ggplot(df,aes(x=gads))+
  geom_line(aes(y=viens,color="viens"))+
  geom_line(aes(y=divi,color="divi"))+
  geom_line(aes(y=tris,color="tris"))
```