NYU Politics Data Lab Workshop: Data Visualization with R and ggplot2

Pablo Barberá

Department of Politics
New York University

email: *pablo.barbera@nyu.edu*
twitter: @p_barbera

October 15, 2013

Data Visualization with R and ggplot2

Purpose of workshop: to introduce tools to generate elegant and effective plots for our academic research.

Why `R`?

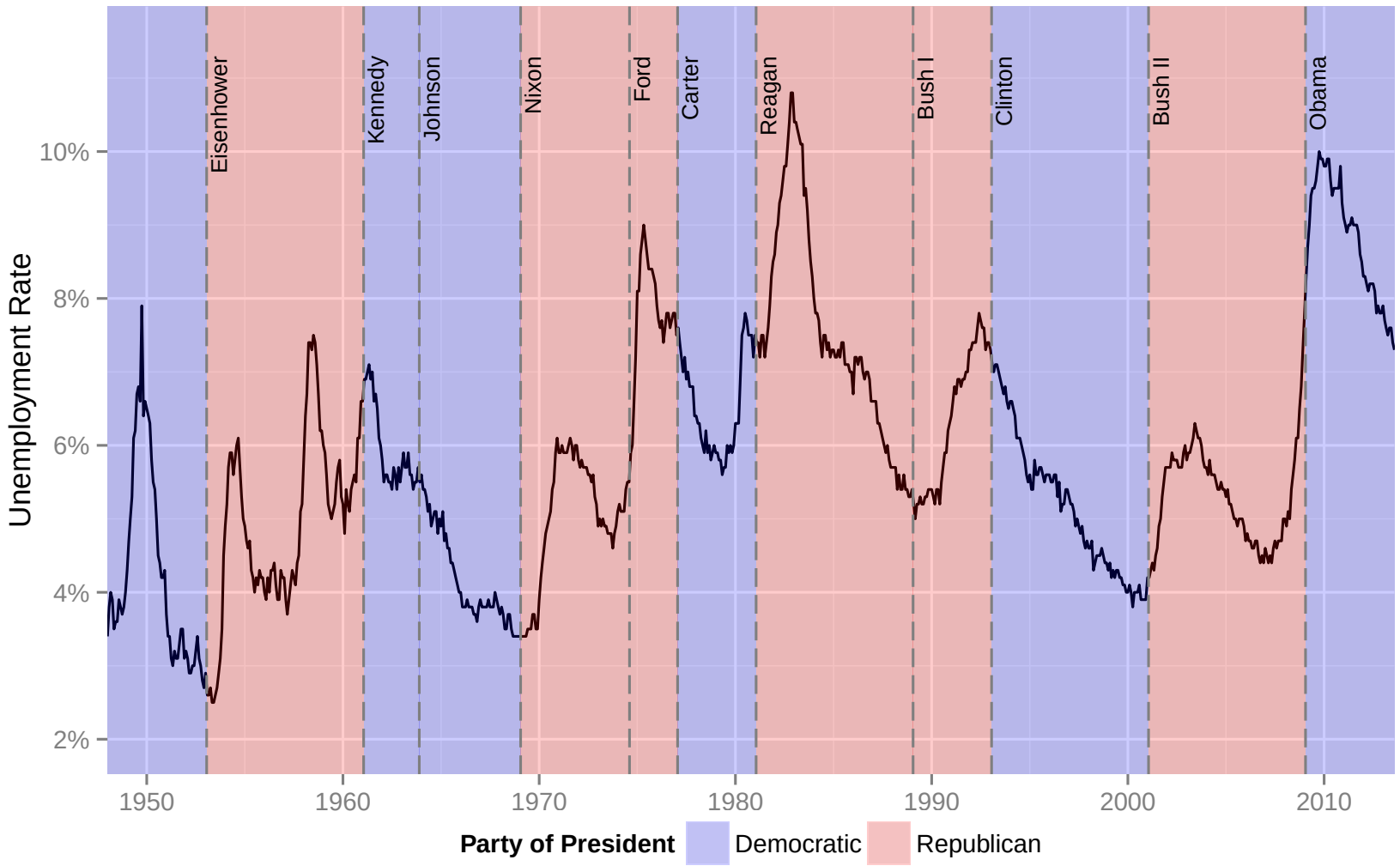
- Mature, widely used, open-source, easily extensible (5K packages on CRAN repository)
- Object-oriented programming language.
- Many built-in basic and advanced statistical tools.

Why `ggplot2`?

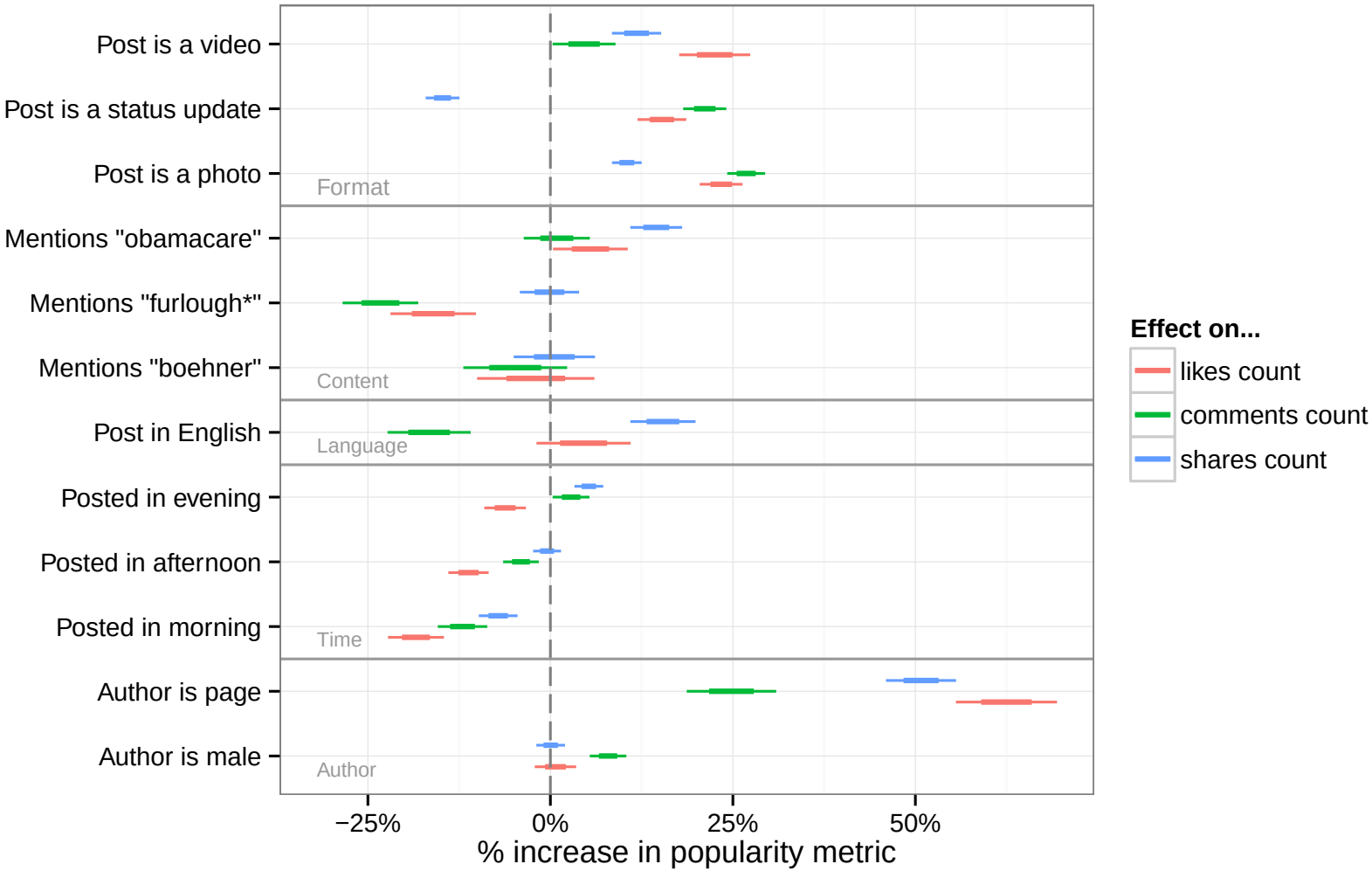
- Based on "Grammar of Graphics" (Wilkinson, 2005)
  $\rightarrow$ powerful, consistent, modular.
- Sensible defaults, but also easy to customize
- Excellent online resources (and easy to google)

Outline

1. Mastering the grammar of graphics
   - Building up a plot layer by layer
   - Scales, axes, legends
   - Themes and other options
2. Applications
   - Annotated line plots
   - Regression coefficient plots
   - Network visualization
   - Maps and spatial analysis
   - Animated plots
3. Beyond `ggplot2`

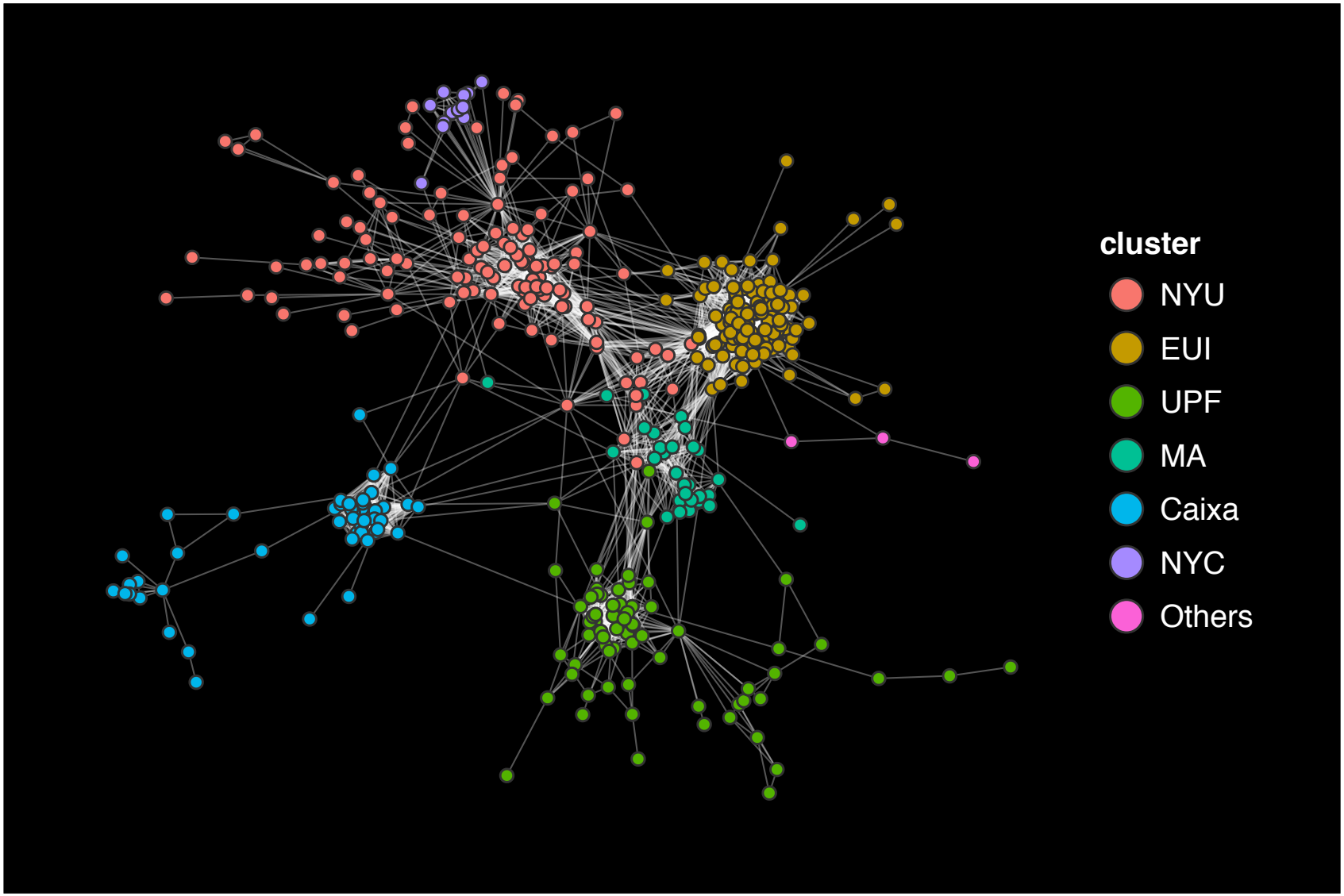

Unemployment Rate in the United States, 1948–2013
Unemployment Rate
10%
8%
6%
4%
2%
1950
1960
1970
1980
1990
2000
2010
Eisenhower
Kennedy
Johnson
Nixon
Ford
Carter
Reagan
Bush I
Clinton
Bush II
Obama
Party of President
Democratic
Republican

What makes Facebook posts about politics popular?

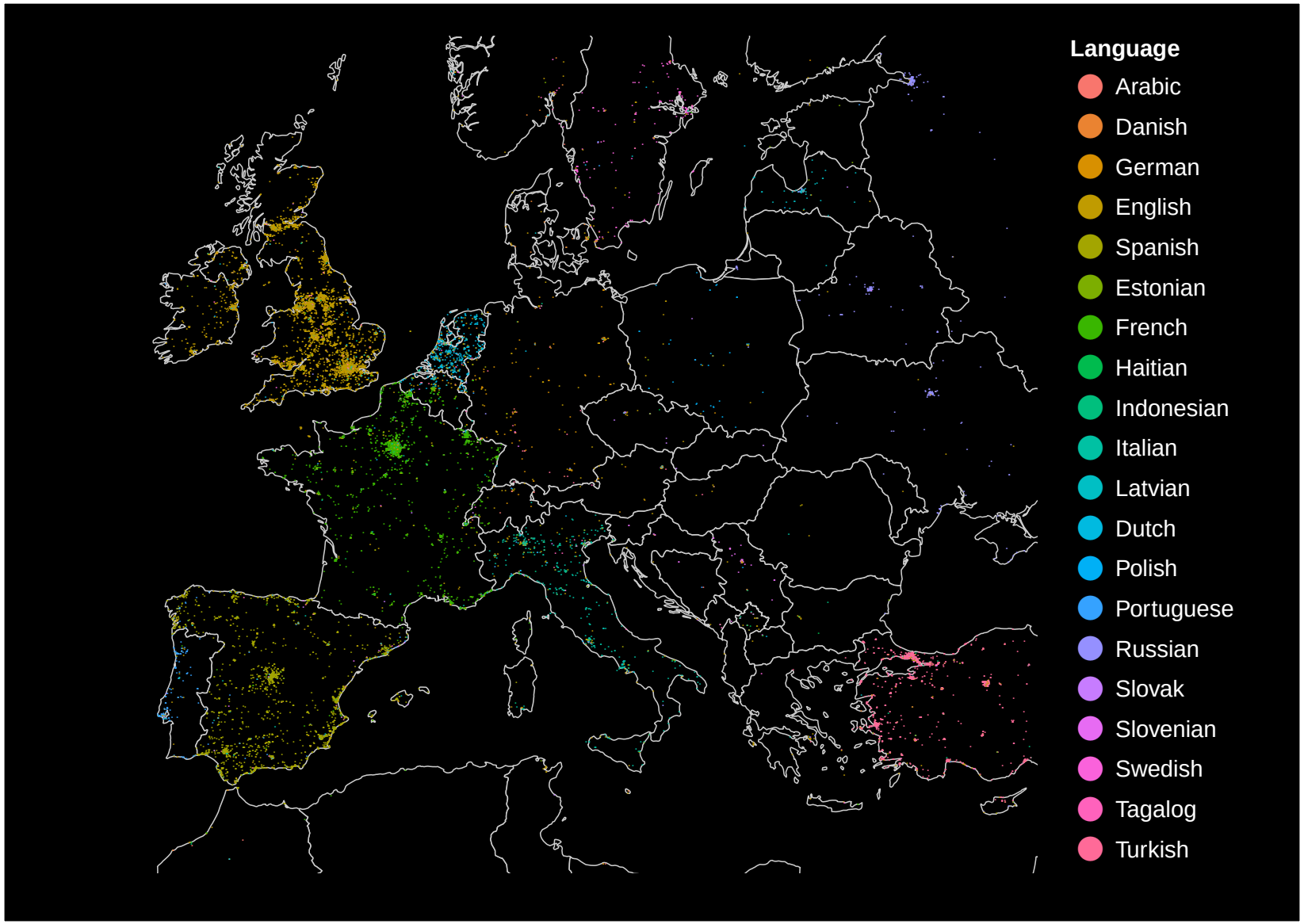


Data: 65K public Facebook posts about govt. shutdown

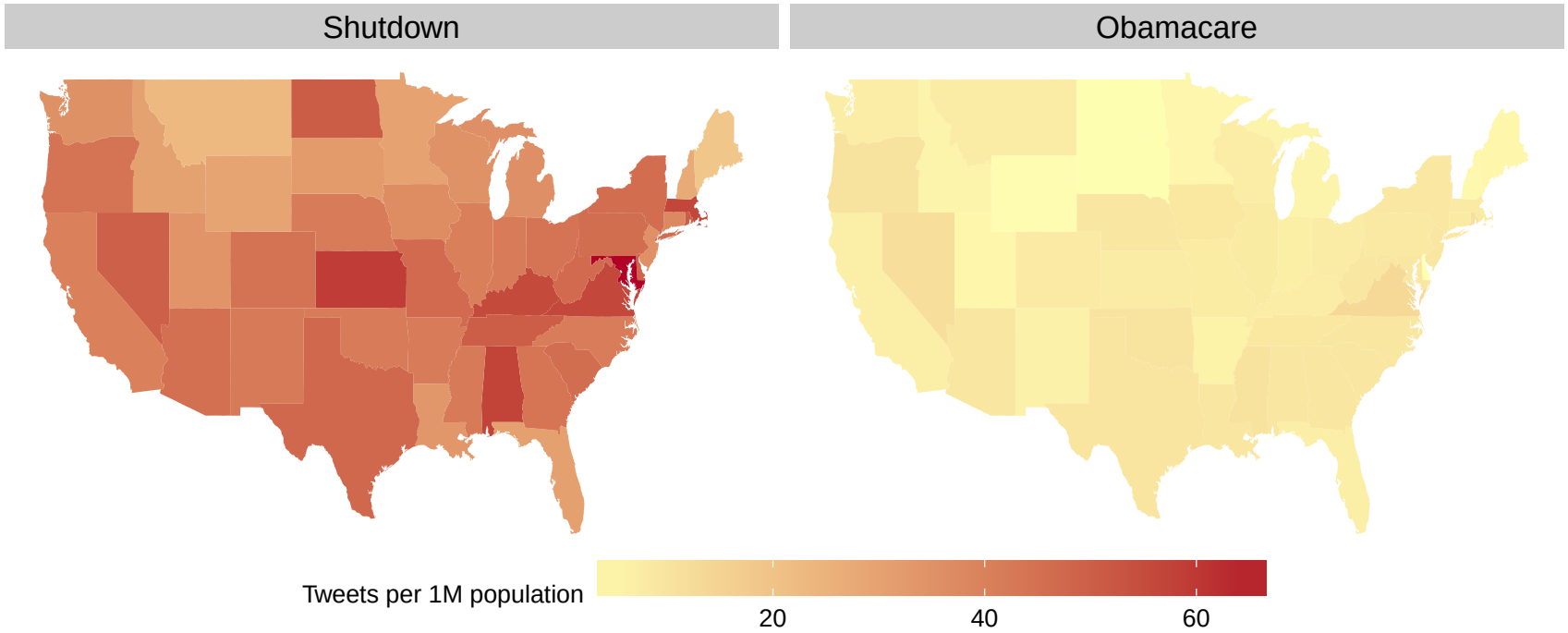
Visualizing your Facebook network

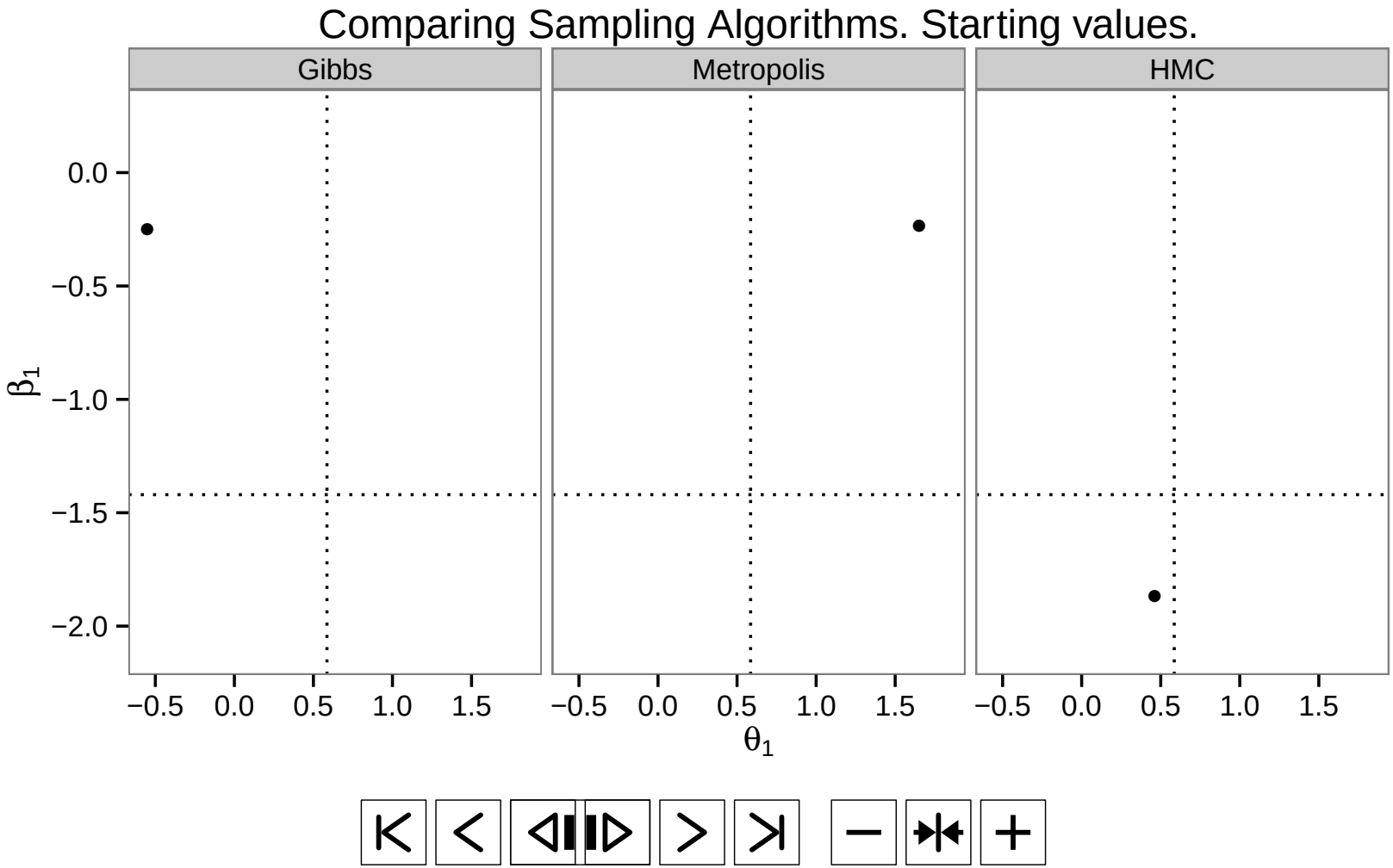


Geolocated tweets, colored by language



Geolocated tweets mentioning ‘shutdown’ and ‘obamacare’, 10/01





Animated plots showing convergence of MCMC algorithms

Materials

Code and data:
http://www.pablobarbera.com/workshop_ggplot2.zip

Short URL:
bit.ly/ggplot2_nyu

Fork my repo!
github.com/pablobarbera/Rdataviz

First R script (please run it now):
`code/00_installing_packages.R`

What is the grammar of graphics?

The grammar of graphics.

A statistical graph is a mapping from data to aesthetic attributes (color, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system. Faceting can be used to generate the same plot for different subsets of the data. It is the combination of these independent components that make up a graphic.

Hadley Wickham, *ggplot2, page 3*

What is the grammar of graphics?

Components of a graph:

data What you want to visualize, including variables (columns) to be mapped to aesthetic attributes.

geom Geometric objects that are drawn to represent the data: bars, lines, points, etc.

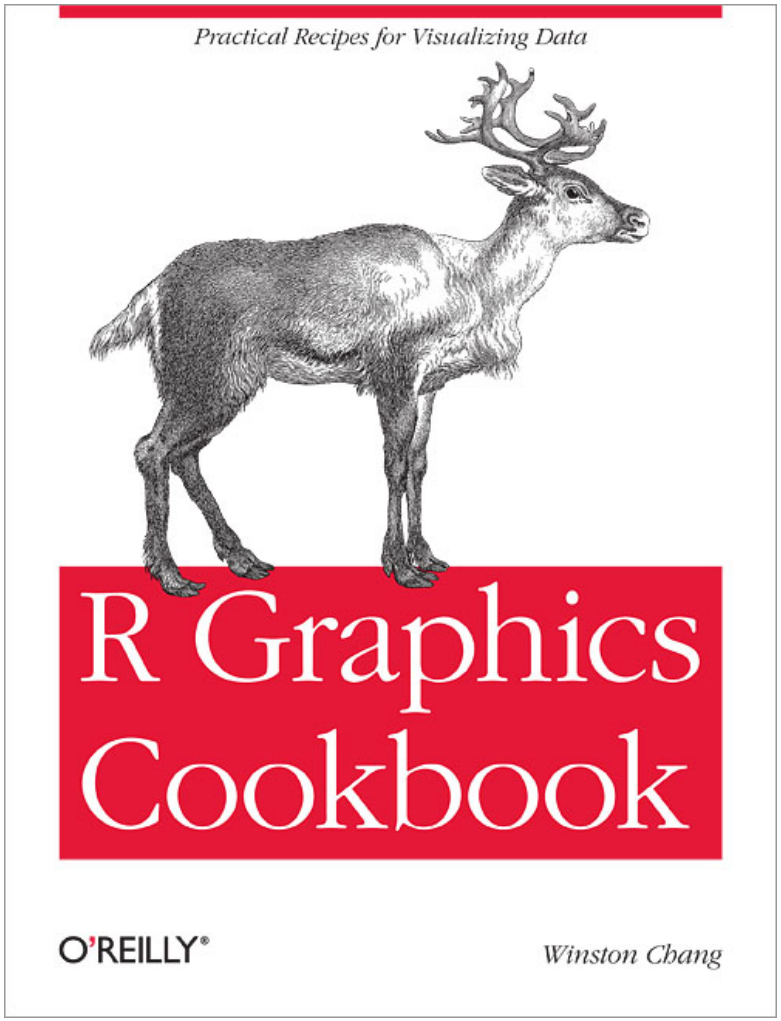
stats Statistical transformations of the data, such as binning or averaging.

scales Map values in the data space to values in an aesthetic space (color, shape, size...)

coord Coordinate system; provides axes and gridlines to make it possible to read the graph.

facets Breaking up the data into subsets, to be displayed independently on a grid

Resources and inspiration



Also: Stack Overflow, ggplot2 listserv & docs

Running example: analysis of Facebook data

Code: `code/01_ggplot2_basics.R`

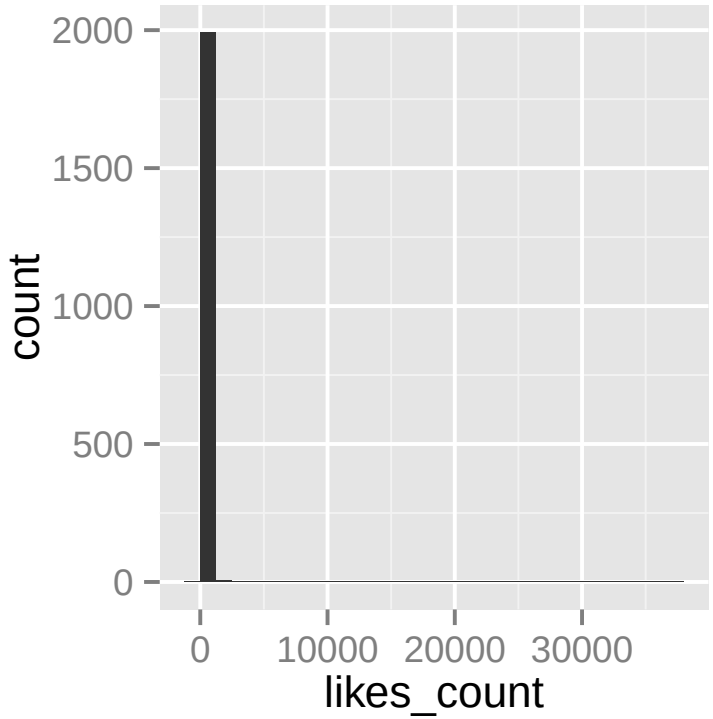
Downloading public Facebook posts about the gov. shutdown.

```
library(devtools)
install_github("Rfacebook", "pablobarbera")
library(Rfacebook)
token <- "XXXXXXXXXX" # (your token here)
posts <- searchFacebook("shutdown", token, n=500)
users <- getUsers(posts$from_id, token)
names(users)[1] <- "from_id"
users <- users[!duplicated(users$from_id),]
fb.data <- merge(posts, users, by="from_id")
```

Univariate analysis: continuous variables

Distribution of number of likes for each post.

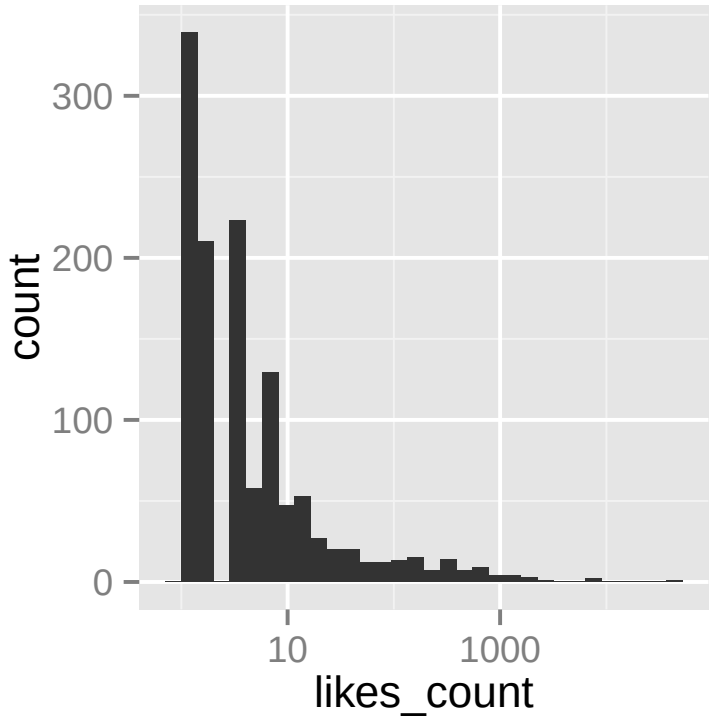
```
# loading library
library(ggplot2)
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count))
# adding histogram
p + geom_histogram()
```



Univariate analysis: continuous variables

Distribution of (logged) number of likes for each post.

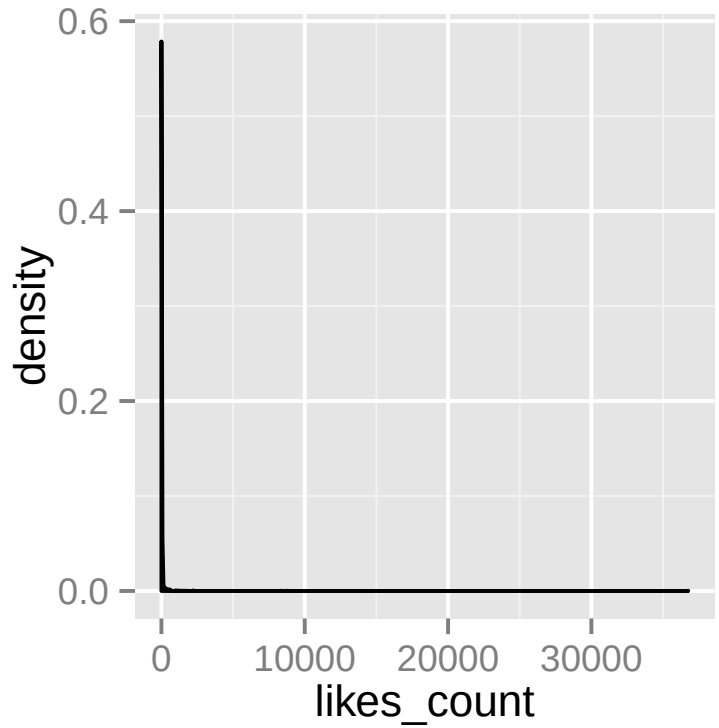
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count))
# adding histogram
p + geom_histogram() +
    scale_x_log10()
```



Univariate analysis: continuous variables

Distribution of number of likes for each post.

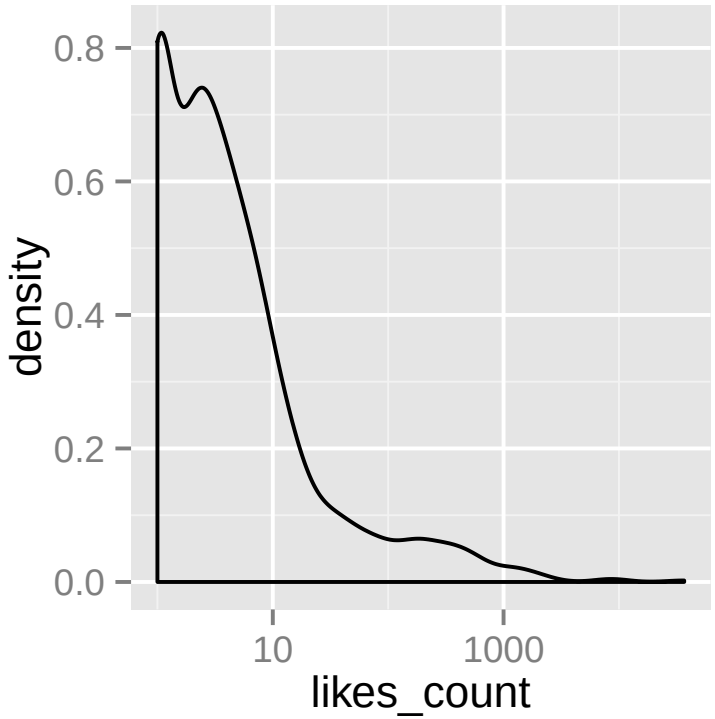
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count))
# adding density plot
p + geom_density()
```



Univariate analysis: continuous variables

Distribution of (logged) number of likes for each post.

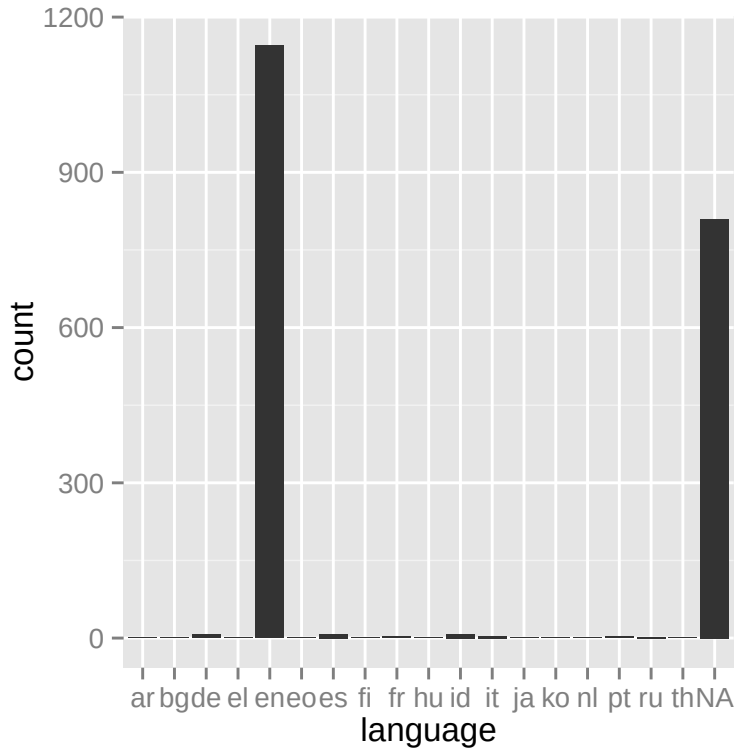
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count))
# adding density plot
p + geom_density() +
    scale_x_log10()
```



Univariate analysis: categorical variables

Distribution of posts by language

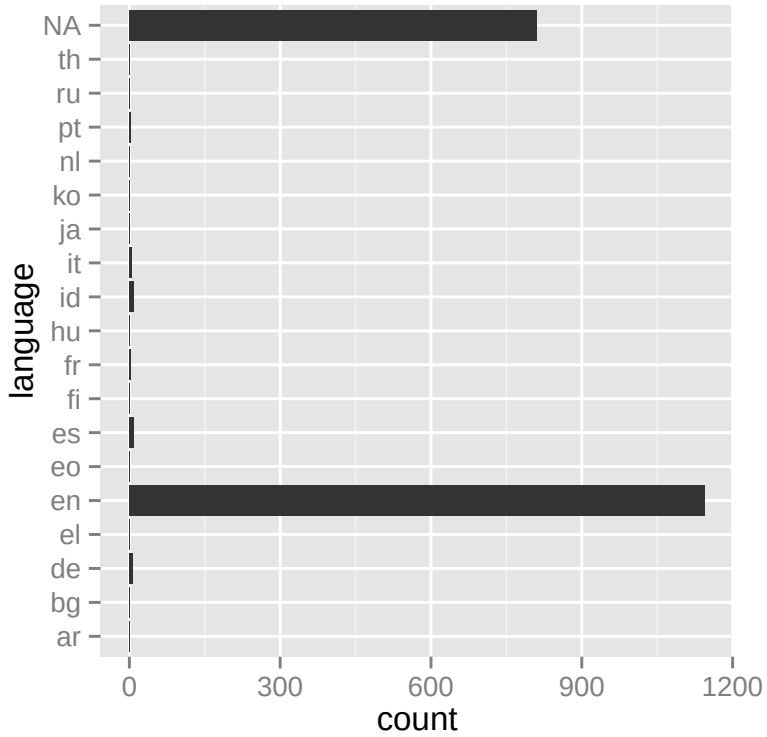
```
# first two characters of
# 'locale' are lang. of user
fb.data$language <- substr(
   fb.data$locale, 1, 2)
# base layer
p <- ggplot(fb.data,
   aes(x=language))
# bar chart
p + geom_bar()
```



Univariate analysis: categorical variables

Distribution of posts by language

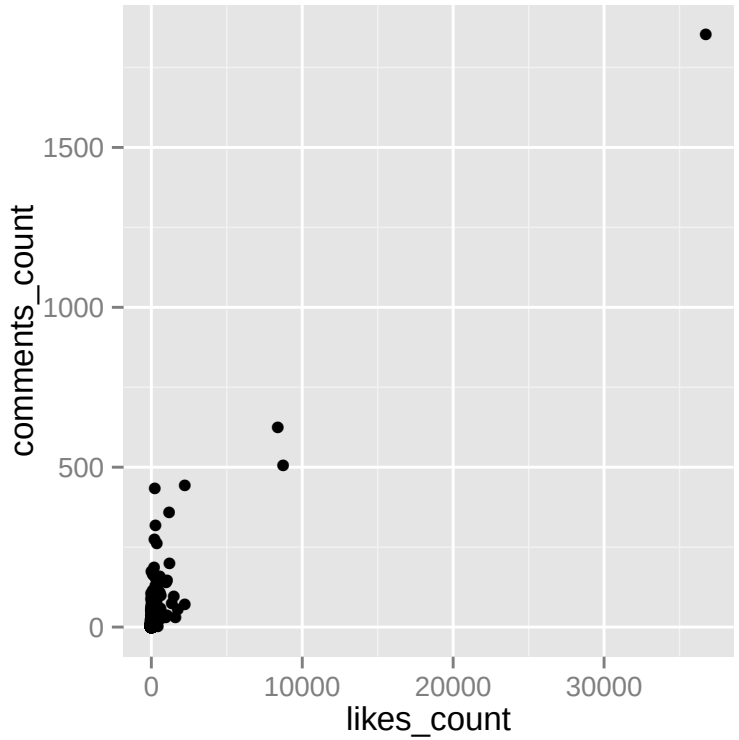
```
# first two characters of
# 'locale' are lang. of user
fb.data$language <- substr(
   fb.data$locale, 1, 2)
# base layer
p <- ggplot(fb.data,
   aes(x=language))
# bar chart (horizontal)
p + geom_bar() +
   coord_flip()
```



Bivariate analysis: two continuous variables

Counts of likes and counts of comments for posts

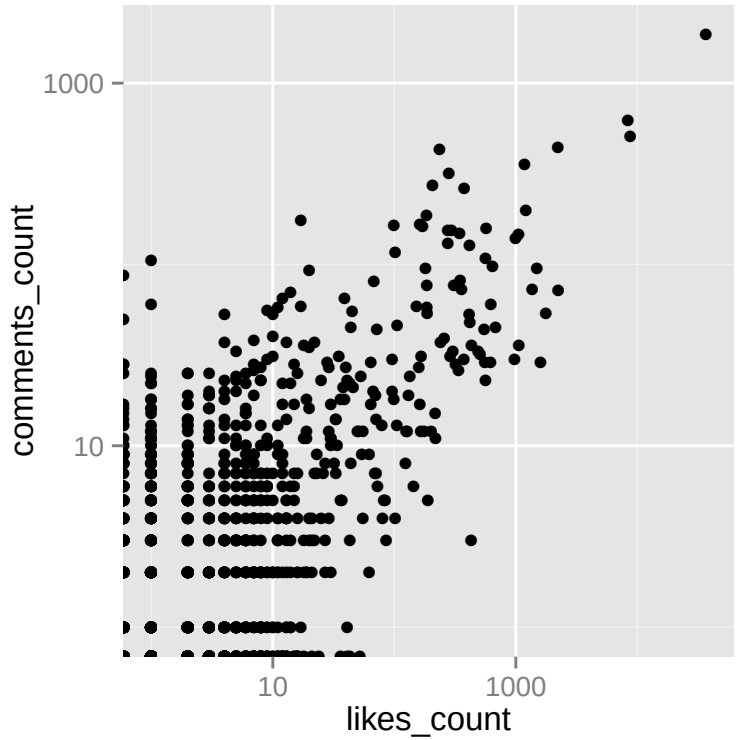
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count,
    y=comments_count))
# scatter plot
p + geom_point()
```



Bivariate analysis: two continuous variables

(Logged) counts of likes and (logged) counts of comments for posts

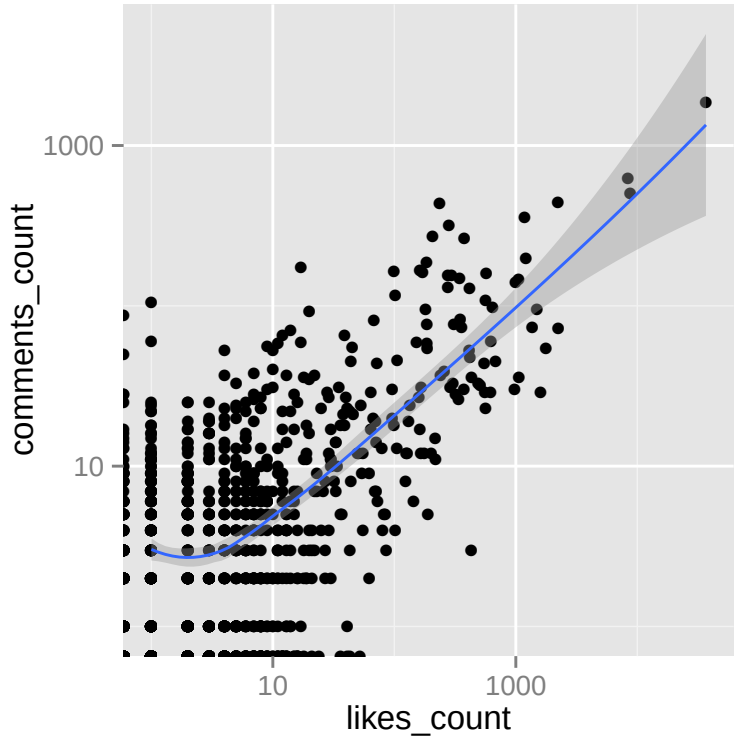
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count,
    y=comments_count))
# scatter plot
p + geom_point() +
    scale_x_log10() +
    scale_y_log10()
```



Bivariate analysis: two continuous variables

(Logged) counts of likes and (logged) counts of comments for posts

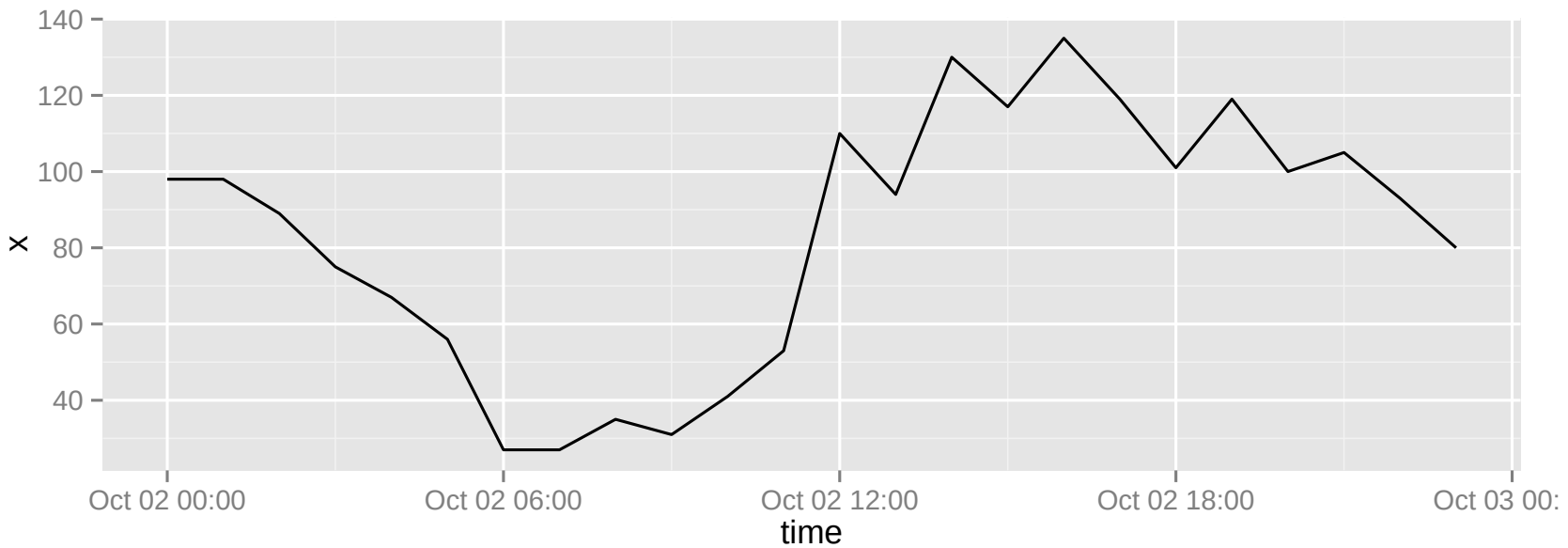
```
p <- ggplot(fb.data,
    aes(x=likes_count,
    y=comments_count))
# scatter plot
p + geom_point() +
    scale_x_log10() +
    scale_y_log10() +
    stat_smooth(
      na.rm=T,
      data=fb.data[
      fb.data$likes_count>0 &
      fb.data$comments_count>0,]
      )
```



Bivariate analysis: two continuous variables

Number of posts about shutdown, per hour

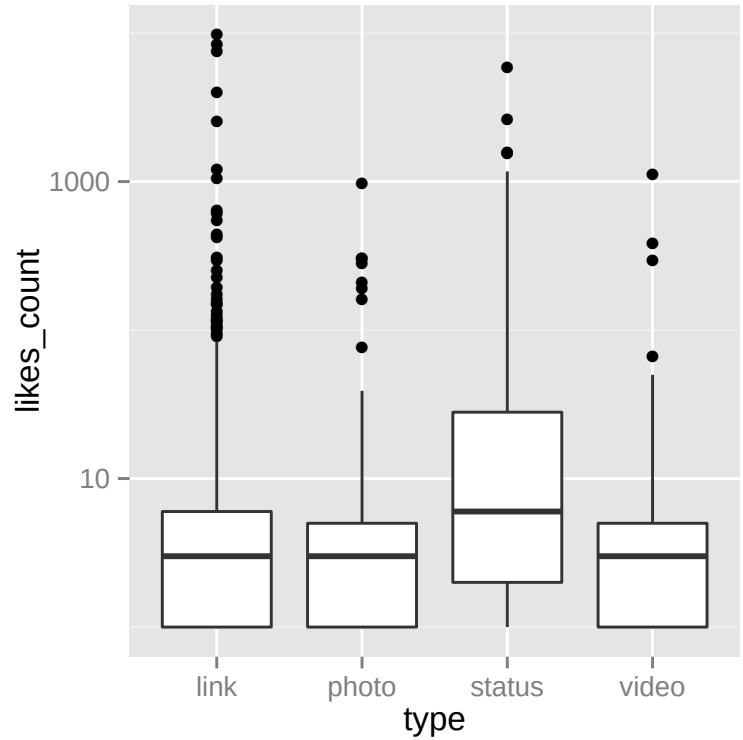
```
fb.data$time <- substr(fb.data$created_time, 1, 13) ## date + hour
fb.data$time <- as.POSIXct(fb.data$time, format="%Y-%m-%dT%H")
fb.data$count <- 1 ## counting number of posts per hour
counts <- aggregate(fb.data$count, list(time=fb.data$time), sum)
p <- ggplot(counts, aes(x=time, y=x))
p + geom_line()
```



Bivariate analysis: continuous + categorical variables

Number of likes by type of post

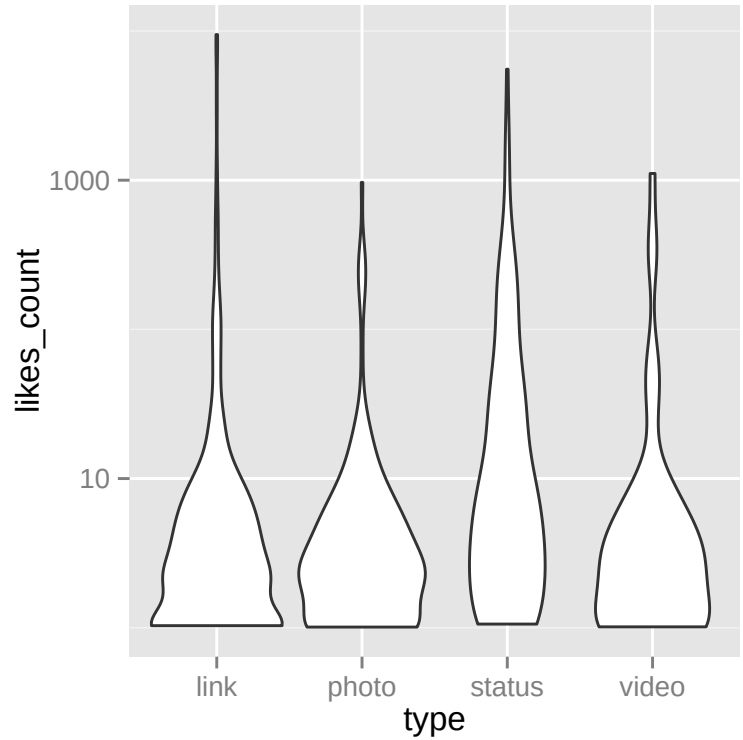
```
p <- ggplot(fb.data,
    aes(x=type,
    y=likes_count))
# box plot
p + geom_boxplot() +
    scale_y_log10()
```



Bivariate analysis: continuous + categorical variables

Number of likes by type of post

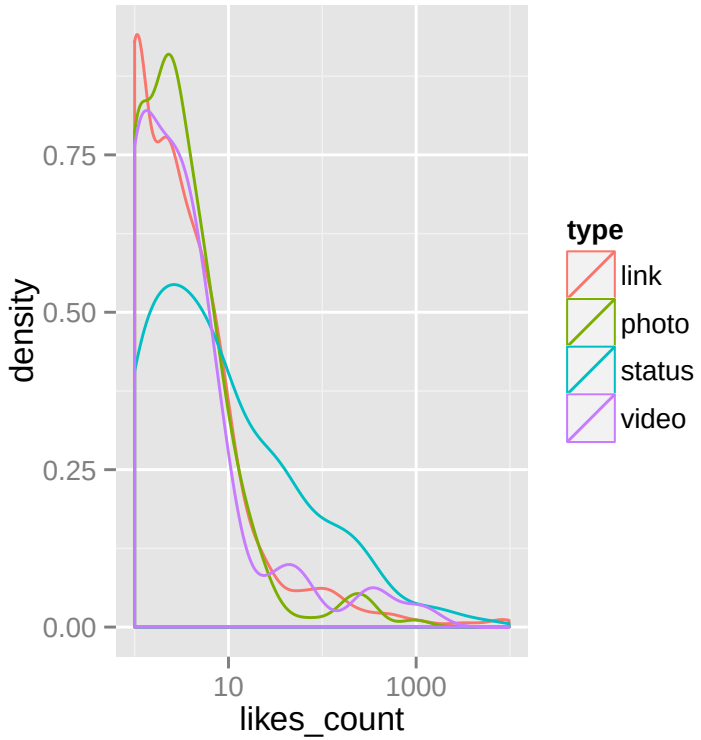
```
p <- ggplot(fb.data,
    aes(x=type,
    y=likes_count))
# violin plot
p + geom_violin() +
    scale_y_log10()
```



Bivariate analysis: continuous + categorical variables

Number of likes by type of post

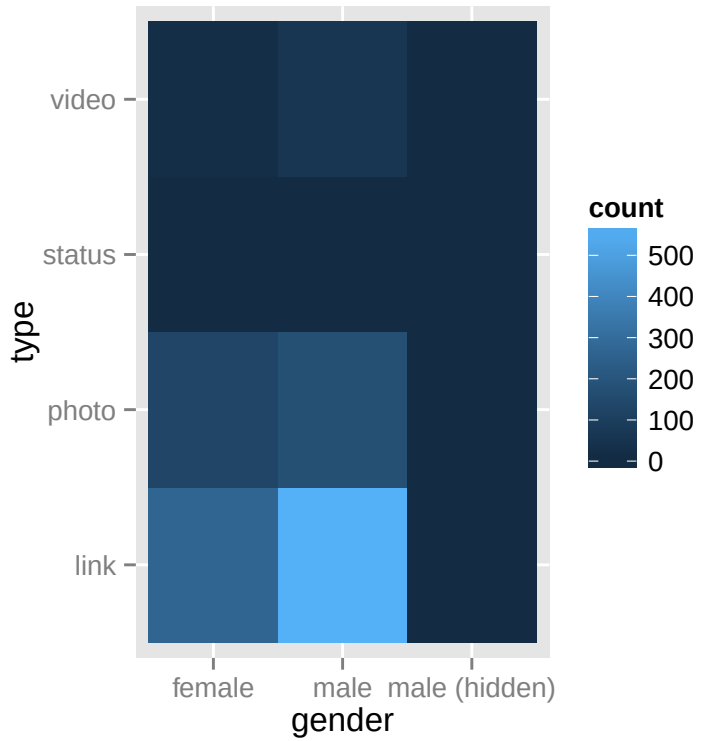
```
p <- ggplot(fb.data,
    aes(x=likes_count))
# density plot
p + geom_density(
    aes(color=type)) +
    scale_x_log10()
```



Bivariate analysis: two categorical variables

Number of posts, by type and gender

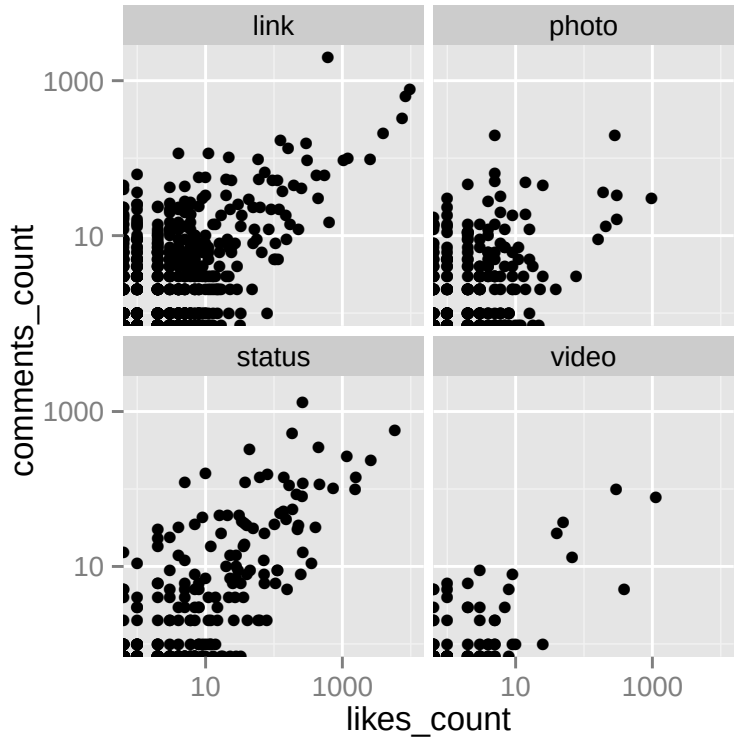
```
# counts data.frame
tab <- data.frame(
   table(fb.data$gender,
     fb.data$type))
names(tab) <- c(
   "gender", "type", "count")
# base layer
p <- ggplot(tab,
   aes(x=gender, y=type))
# tile plot
p + geom_tile(
   aes(fill=count))
```



Multivariate analysis: three continuous variables

Number of likes and comments, by type of post

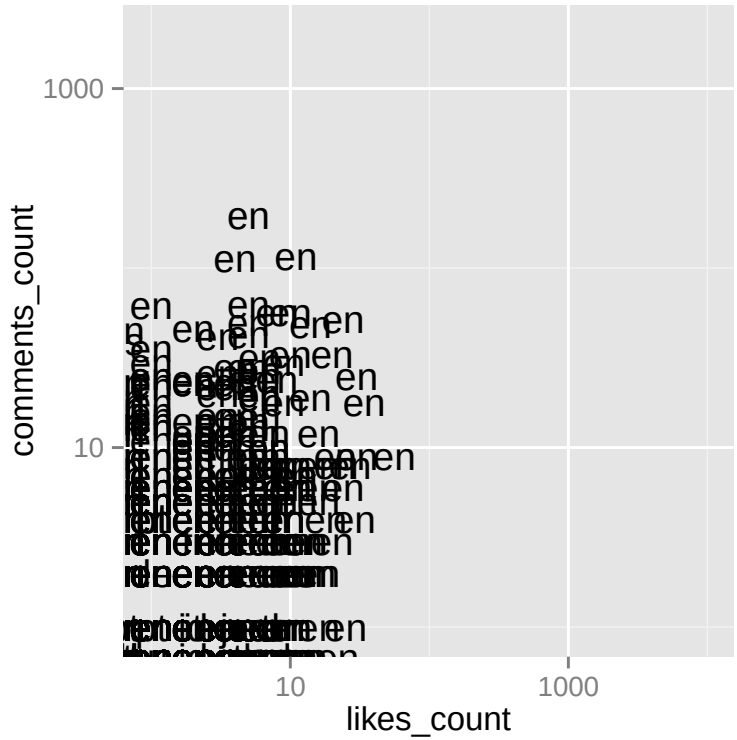
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count,
    y=comments_count))
p + geom_point() +
    scale_x_log10() +
    scale_y_log10() +
    facet_wrap(~type, nrow=2)
```



Multivariate analysis: continuous + categorical variables

Number of likes and comments, by country

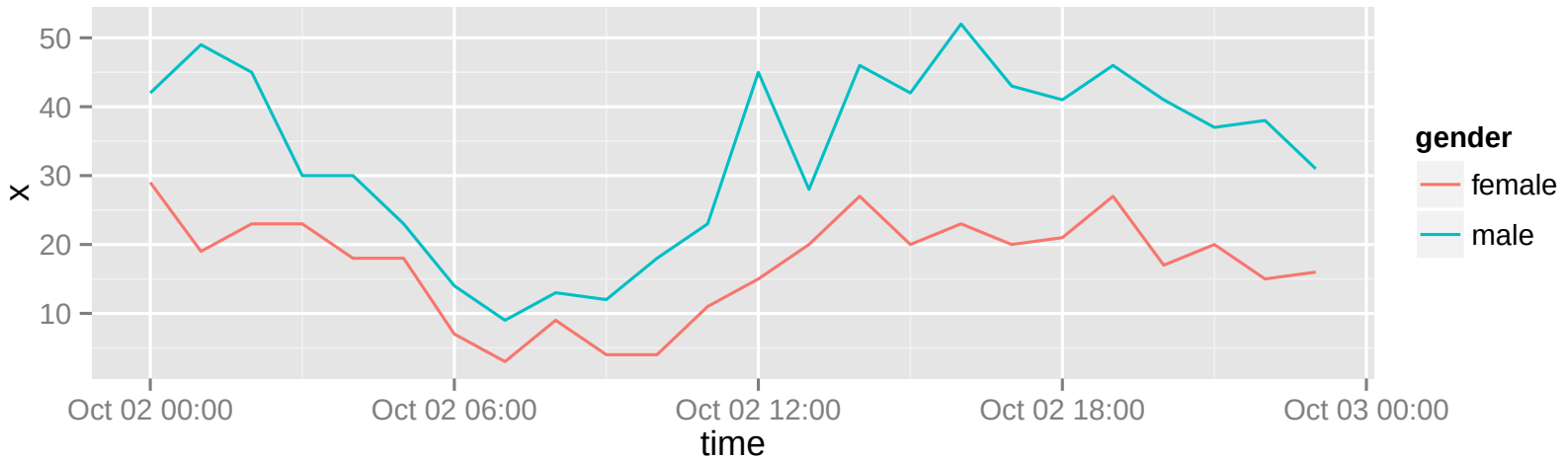
```
# base layer
p <- ggplot(fb.data,
   aes(x=likes_count,
   y=comments_count,
   label=language))
p + geom_text() +
   scale_x_log10() +
   scale_y_log10()
```



Multivariate analysis: continuous + categorical variables

Number of posts per hour, by gender

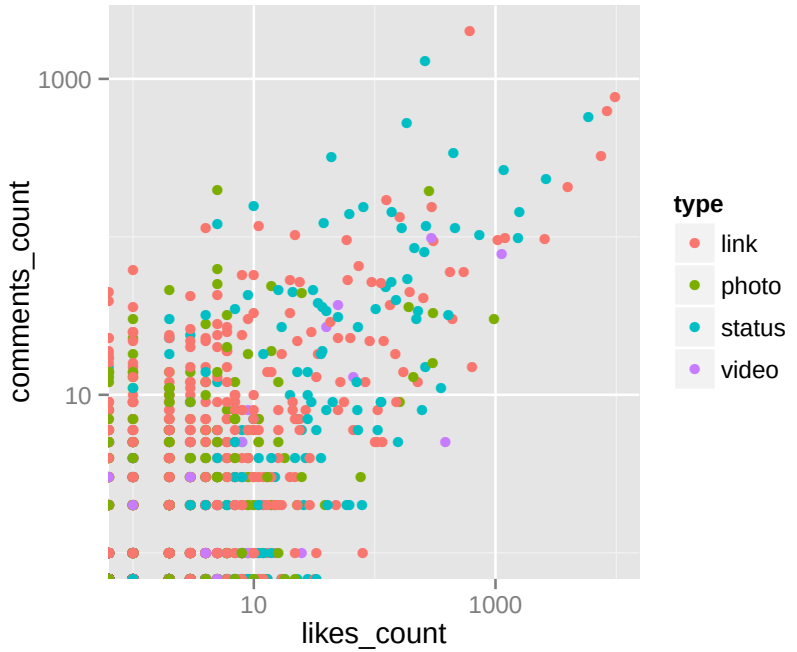
```
counts.m <- aggregate(fb.data$count[fb.data$gender=="male"],
   by=list(time=fb.data$time[fb.data$gender=="male"]), FUN=sum)
counts.f <- aggregate(fb.data$count[fb.data$gender=="female"],
   by=list(time=fb.data$time[fb.data$gender=="female"]), FUN=sum)
counts.m$gender <- "male"; counts.f$gender <- "female"
counts <- rbind(counts.m, counts.f)
p <- ggplot(counts, aes(x=time, y=x, group=gender))
p + geom_line(aes(color=gender))
```



Multivariate analysis: continuous + categorical variables

Number of likes and comments, by type of post

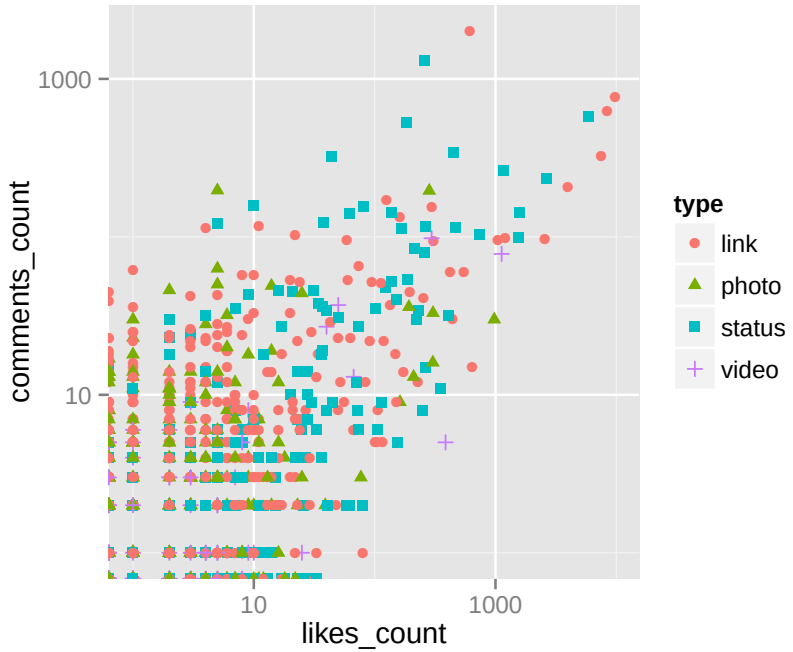
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count,
    y=comments_count))
p + geom_point(
    aes(color=type)) +
    scale_x_log10() +
    scale_y_log10()
```



Multivariate analysis: continuous + categorical variables

Number of likes and comments, by type of post

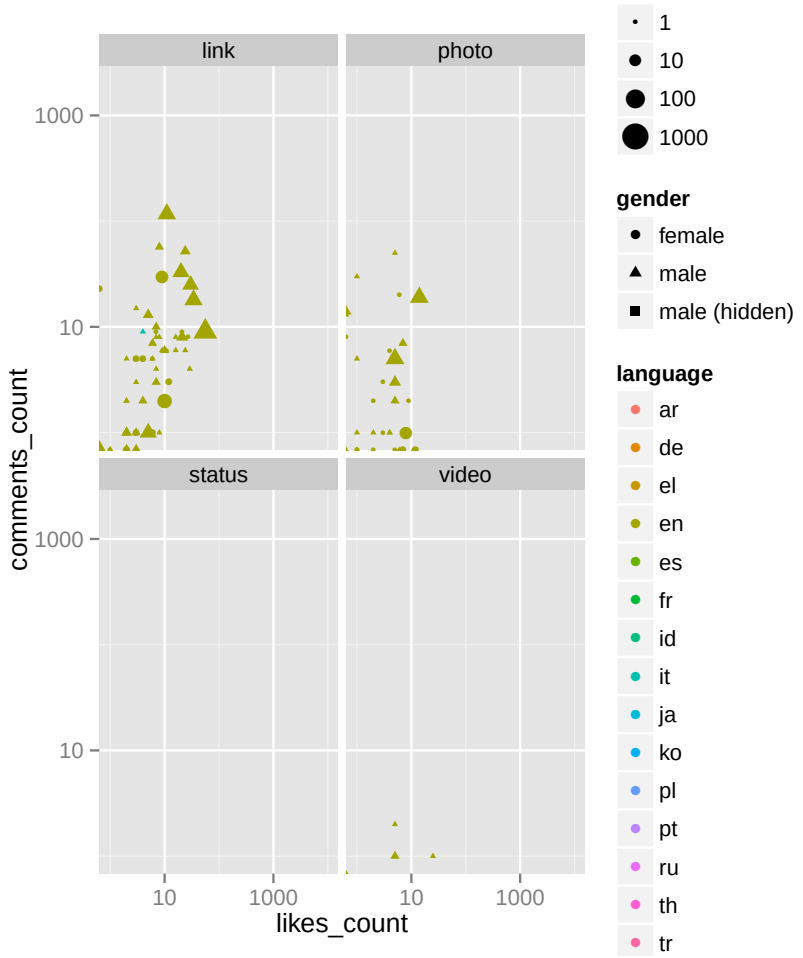
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count,
    y=comments_count))
p + geom_point(
    aes(color=type,
    shape=type)) +
    scale_x_log10() +
    scale_y_log10()
```



Multivariate analysis: continuous + categorical variables

Number of likes and comments, with multiple scales

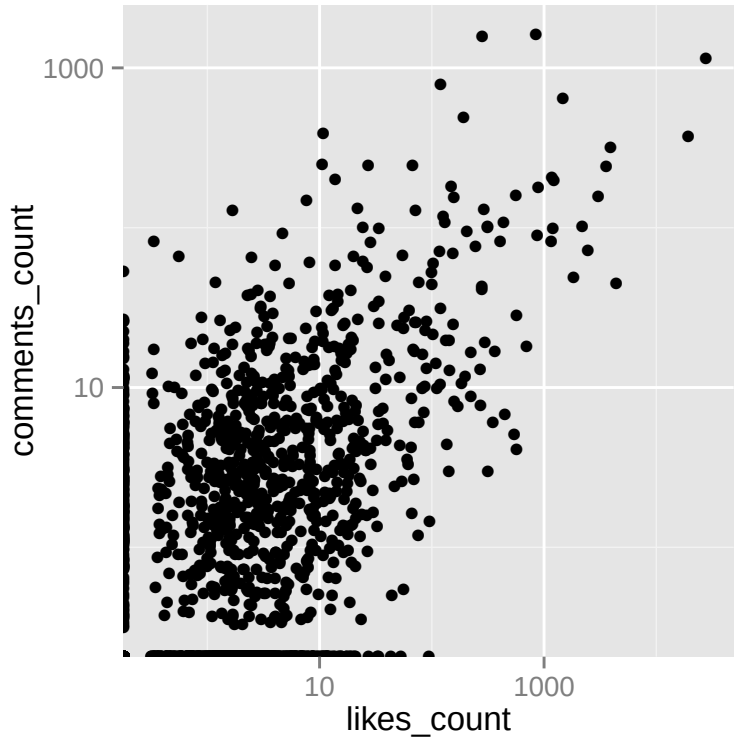
```
# base layer
p <- ggplot(fb.data,
    aes(x=likes_count,
    y=comments_count))
p + geom_point(
    aes(shape=gender,
    color=language,
    size=shares_count)) +
    scale_x_log10() +
    scale_y_log10() +
    scale_size(
    trans="log10") +
    facet_wrap(~type,nrow=2)
```



Dealing with overplotting

Counts of likes and counts of comments for posts

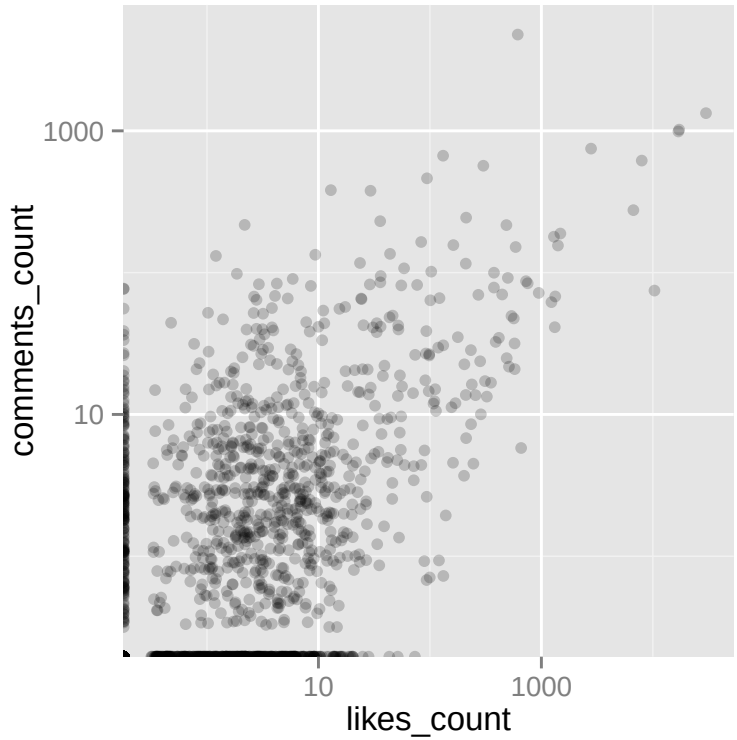
```
# base layer
p <- ggplot(fb.data,
   aes(x=likes_count,
   y=comments_count))
# jittered dots
p + geom_jitter(
   position = position_jitter(
     width = .5, height=.5)) +
    scale_x_log10() +
    scale_y_log10()
```



Dealing with overplotting

Counts of likes and counts of comments for posts

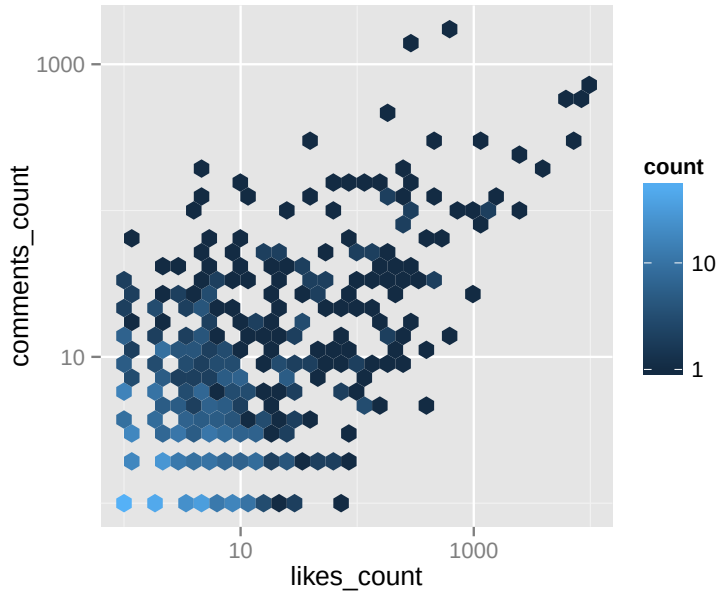
```
# base layer
p <- ggplot(fb.data,
   aes(x=likes_count,
   y=comments_count))
# adding transparency
p + geom_jitter(
   position = position_jitter(
     width = .5, height=.5),
     alpha=1/5) +
    scale_x_log10() +
    scale_y_log10()
```



Dealing with overplotting

Counts of likes and counts of comments for posts

```
# base layer
p <- ggplot(fb.data[
    fb.data$likes_count>0 &
    fb.data$comments_count>0,],
    aes(x=likes_count,
    y=comments_count))
# hexagon binning
p + geom_hex() +
     scale_x_log10() +
     scale_y_log10() +
     scale_fill_continuous(
     trans="log10")
```

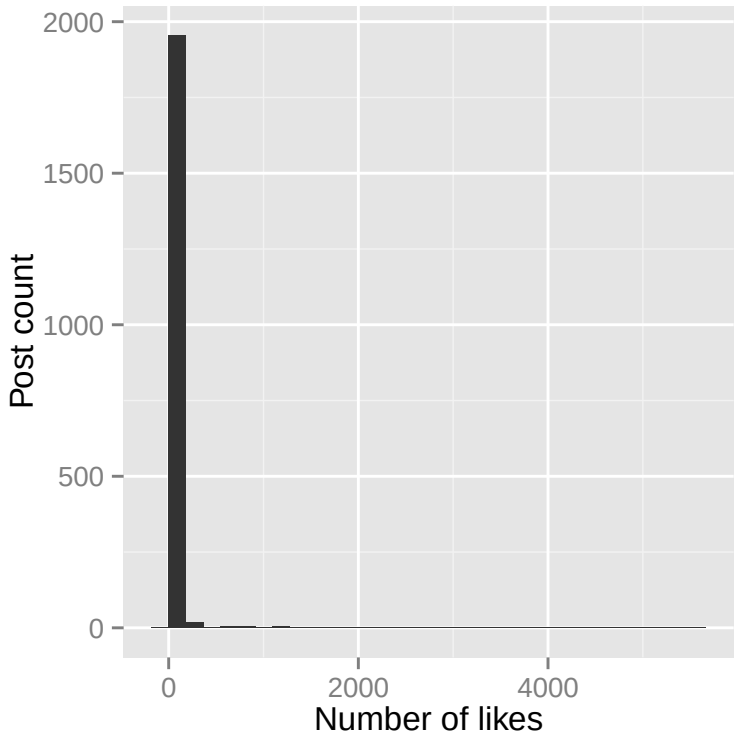


Customizing axes

Code: `code/02_scales_axes_legends.R`

Changing axis titles

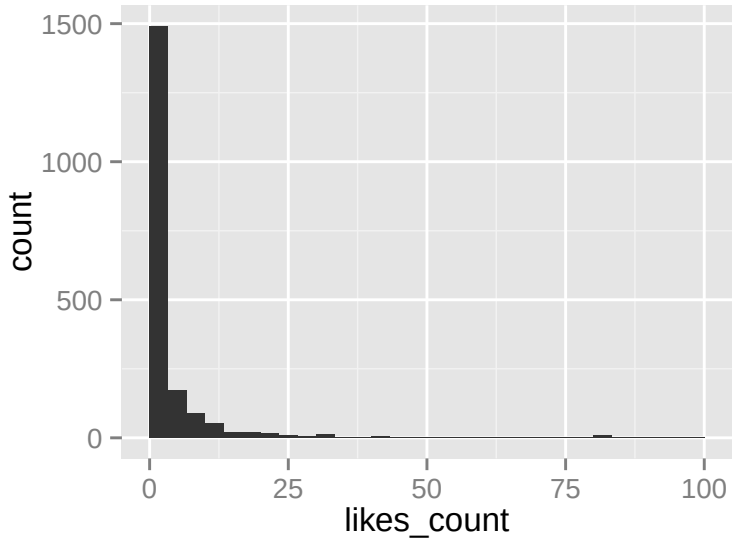
```
p <- ggplot(fb.data,
aes(x=likes_count))
p + geom_histogram() +
    scale_x_continuous(
      "Number of likes") +
    scale_y_continuous(
      "Post count")
```



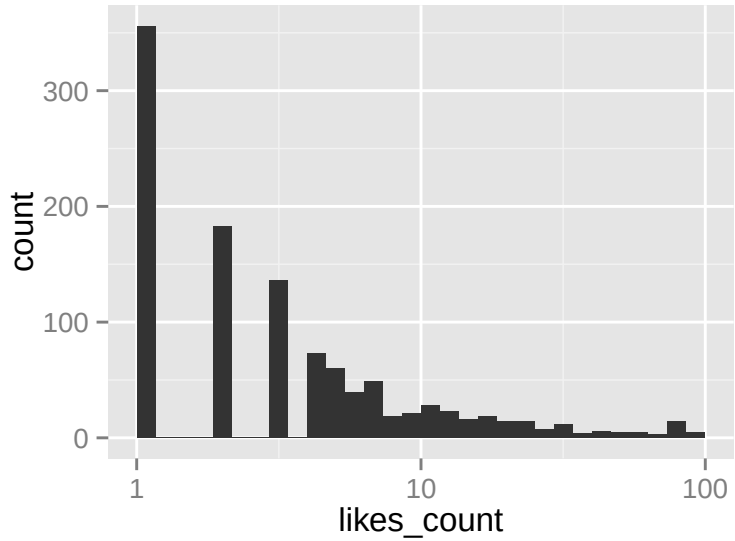
Customizing axes

Changing axis limits (continuous variable)

```
p + geom_histogram() +
   scale_x_continuous(
     limits=c(0, 100))
```



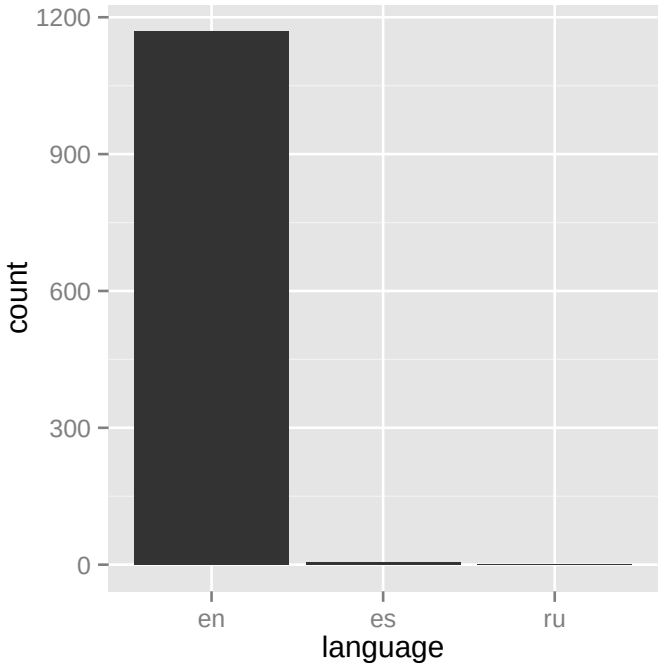

```
p + geom_histogram() +
   scale_x_log10(
     limits=c(1, 100))
```



Customizing axes

Changing axis limits (categorical variable)

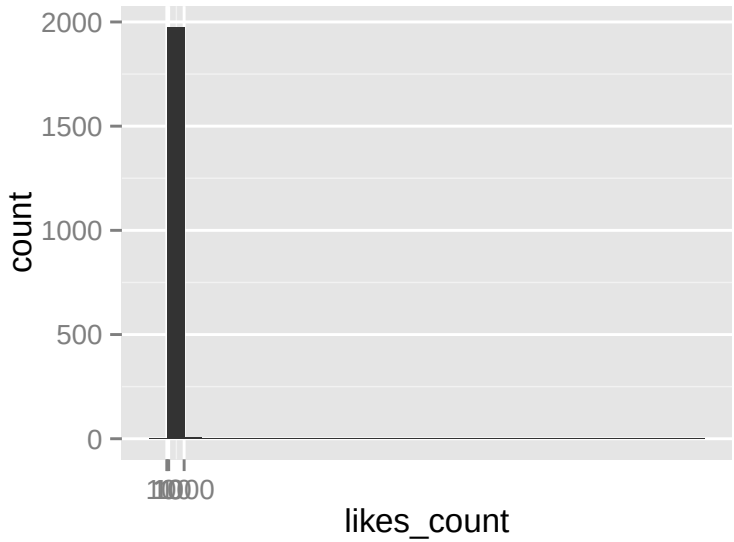
```
fb.data$language <- substr(
   fb.data$locale, 1, 2)
p <- ggplot(fb.data,
   aes(x=language))
p + geom_bar() +
   scale_x_discrete(
      limits=c("en", "es", "ru"))
```



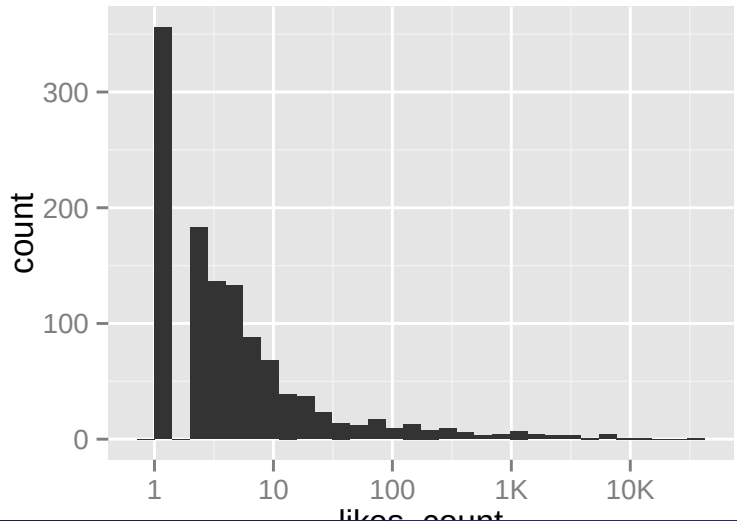
Customizing axes

Changing axis breaks and labels (continuous variable)

```
p <- ggplot(fb.data,
aes(x=likes_count))
p + geom_histogram() +
 scale_x_continuous(
  breaks=c(1, 10, 100, 1000))
```



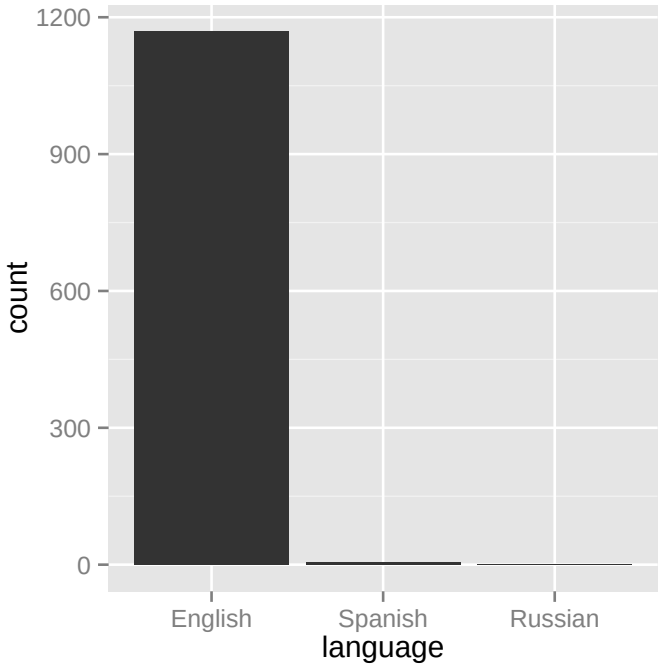

```
p <- ggplot(fb.data,
  aes(x=likes_count))
p + geom_histogram() +
 scale_x_log10(
  breaks=c(1, 10, 100, 1000, 10000),
  labels=c(1, 10, 100, "1K", "10K"))
```



Customizing axes

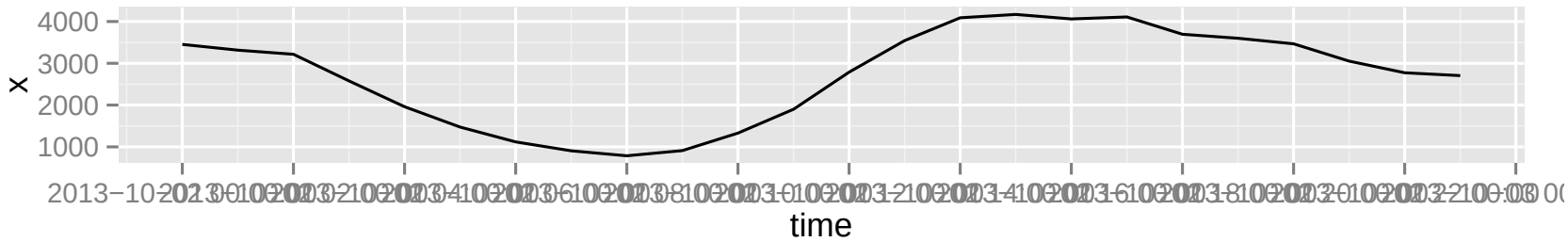
Changing axis limits (categorical variable)

```
p <- ggplot(fb.data,
   aes(x=language))
p + geom_bar() +
   scale_x_discrete(
      limits=c("en", "es", "ru"),
      labels=c("en" = "English",
      "es" = "Spanish",
      "ru" = "Russian"))
```

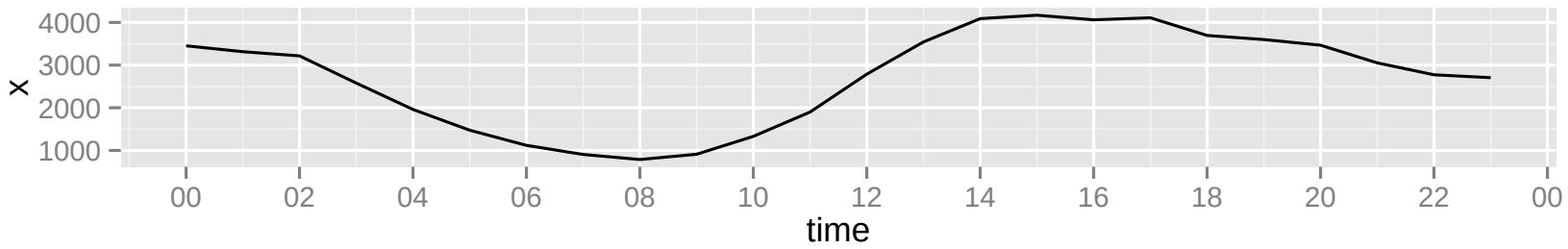


Customizing axes

```
library(scales)
p <- ggplot(counts, aes(x=time, y=x))
p + geom_line() + scale_x_datetime(breaks="2 hours")
```



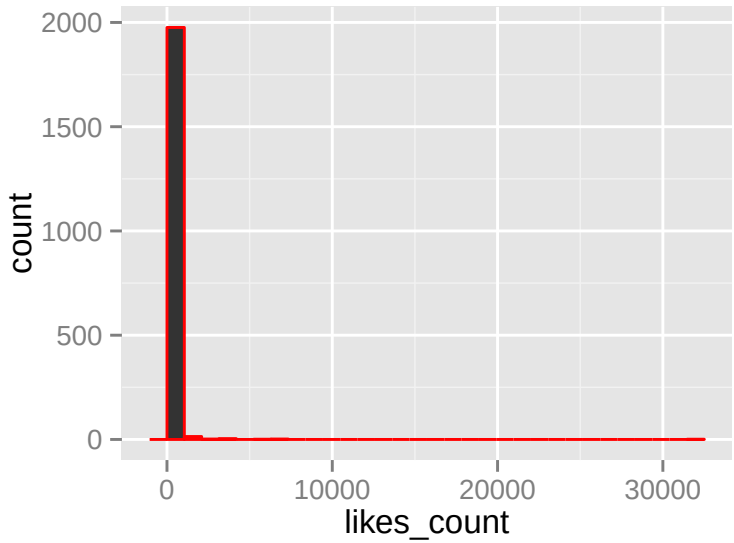

```
p + geom_line() + scale_x_datetime(breaks="2 hours",
    labels = date_format("%H"))
```



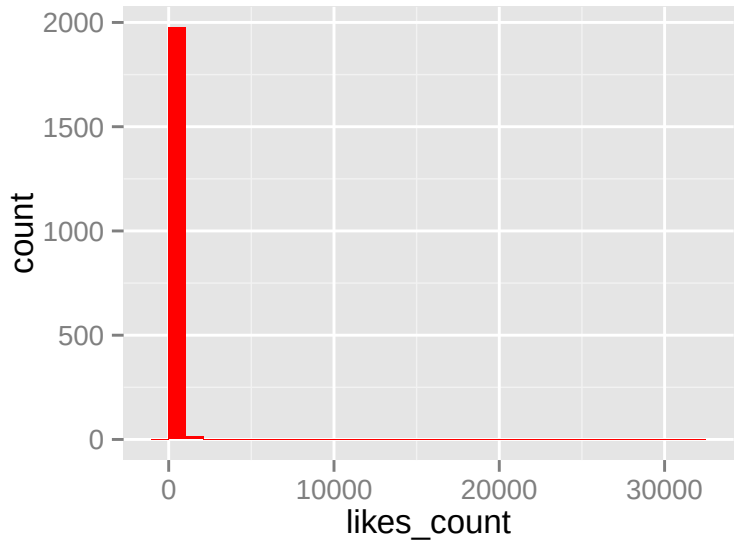
Customizing scales

Customizing properties of geoms

```
p <- ggplot(fb.data,
aes(x=likes_count))
p + geom_histogram(
    color="red")
```

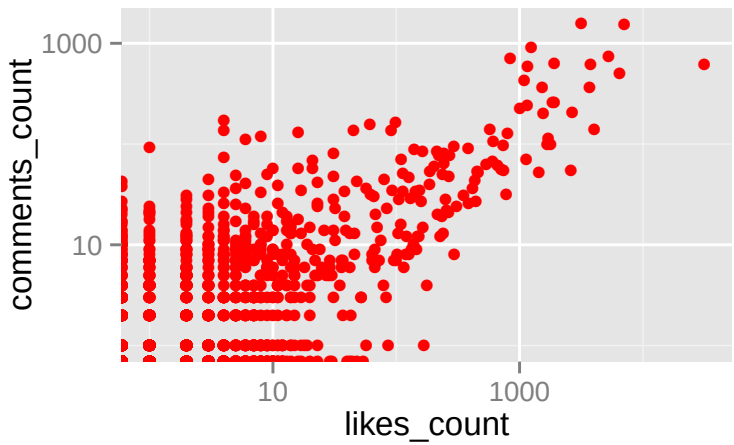



```
p <- ggplot(fb.data,
aes(x=likes_count))
p + geom_histogram(
    fill="red")
```

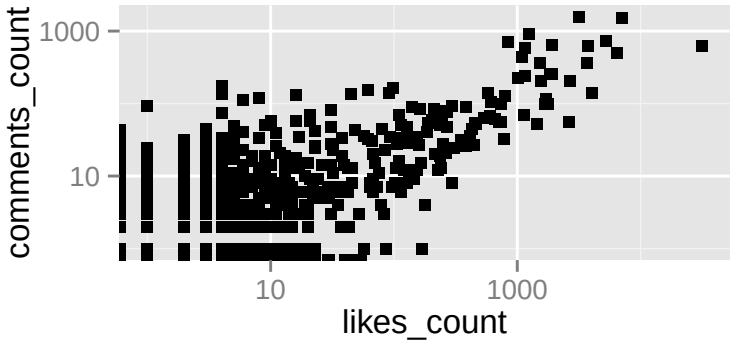


Customizing scales

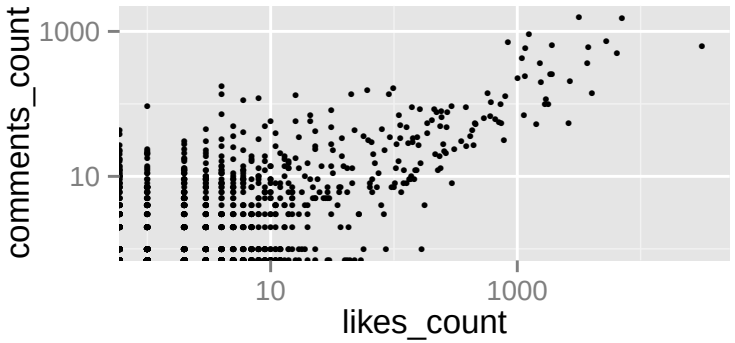
```
p <- ggplot(fb.data,
aes(x=likes_count,
  y=comments_count)) +
  scale_x_log10() +
  scale_y_log10()
p + geom_point( color="red")
```




```
p + geom_point(shape=15)
```



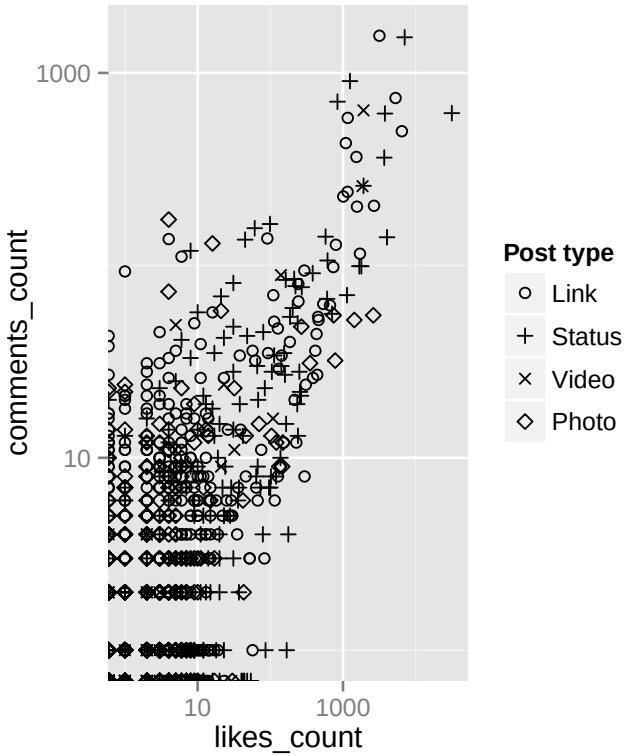

```
p + geom_point( size=1)
```



Customizing scales

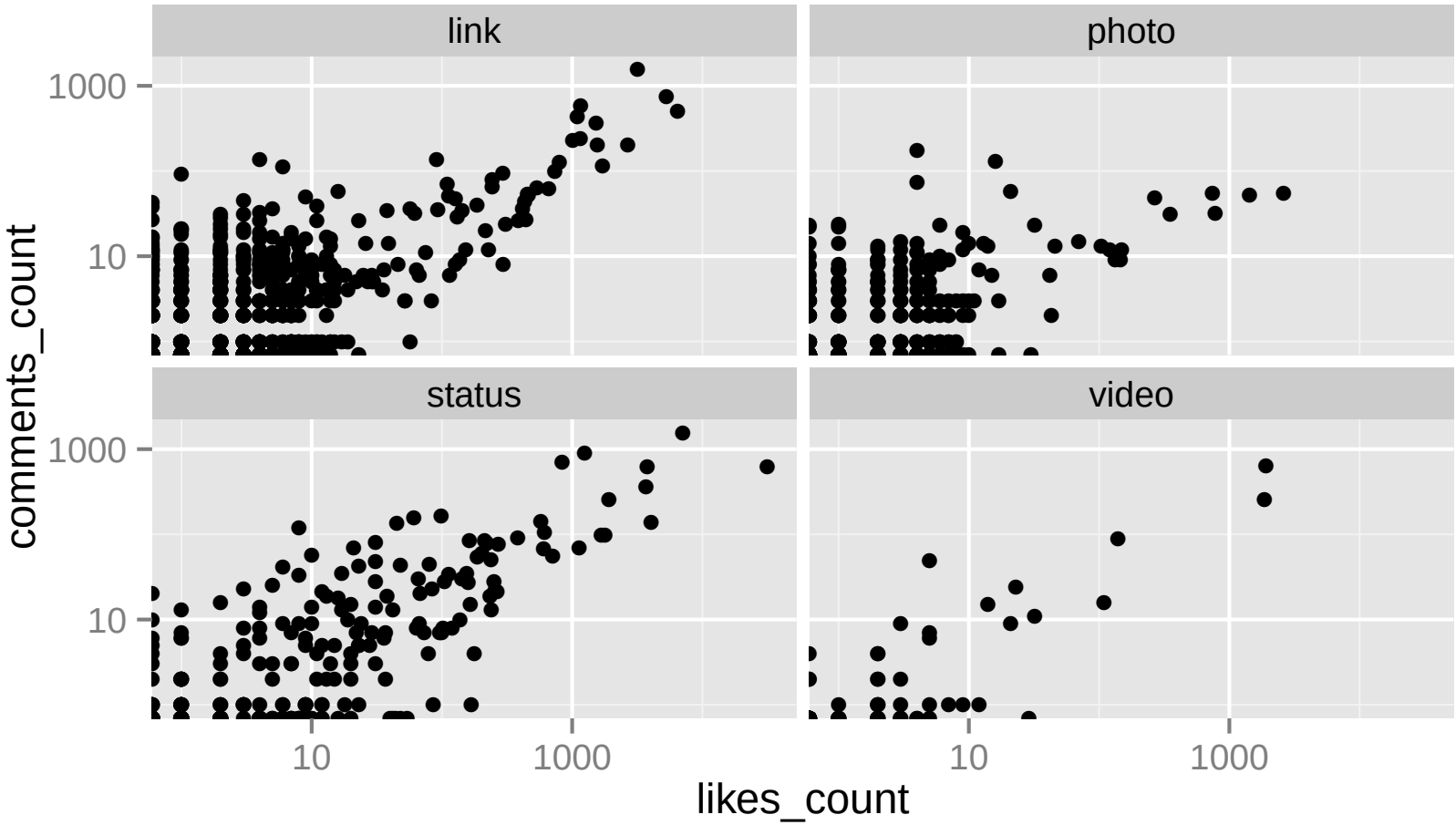
Changing labels in legend

```
p + geom_point(
    aes(shape=type)) +
    scale_shape_manual(
    "Post type",
      limits = c("link", "status",
        "video", "photo"),
      labels = c("Link", "Status",
        "Video", "Photo"),
      values=c(1, 3, 4, 5))
```



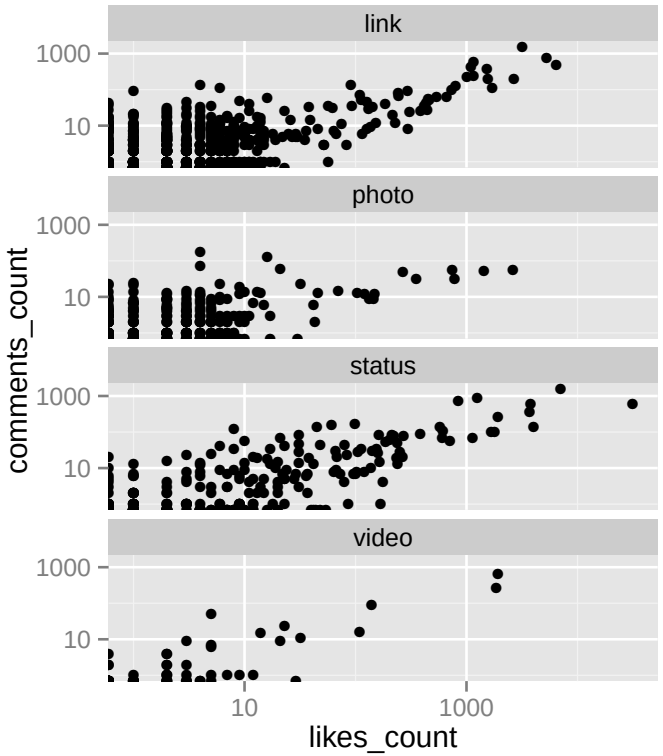
Facets

```
p + geom_point() + facet_wrap(~type)
```



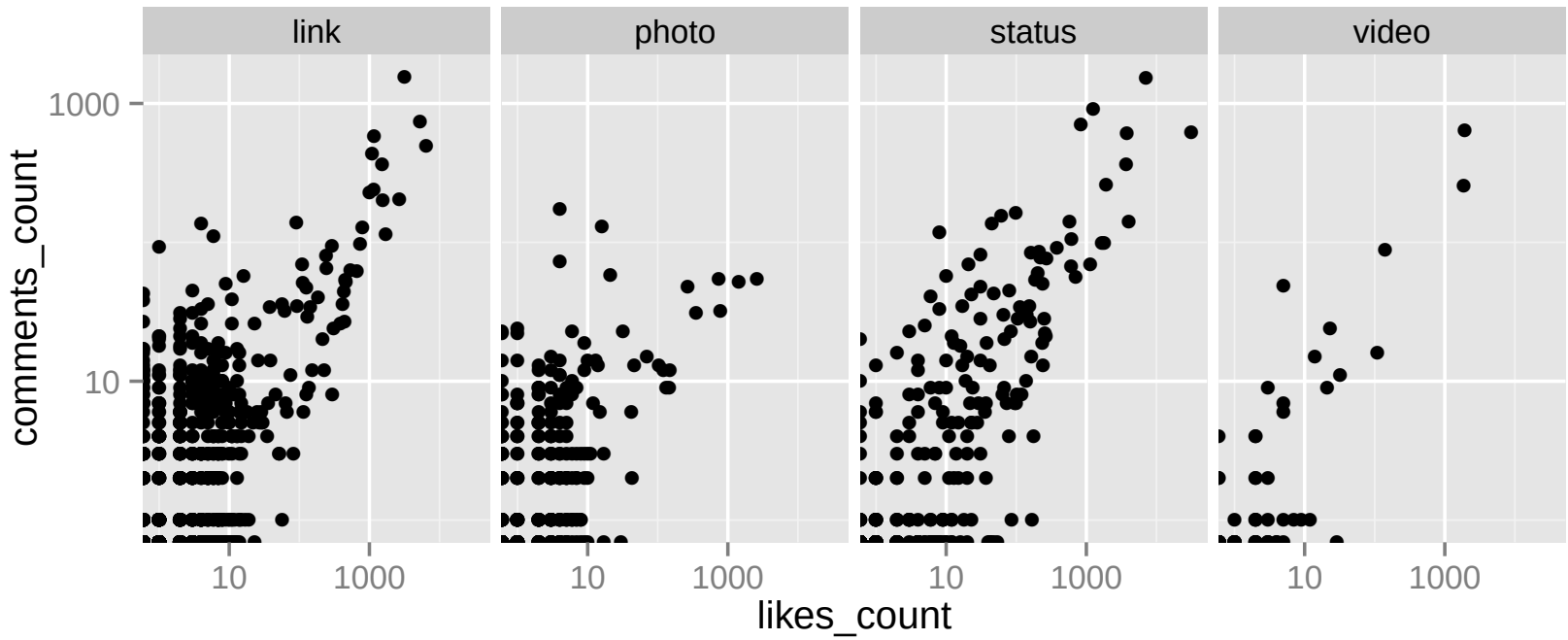
Facets

```
p + geom_point() +
  facet_wrap(~type, nrow=4)
```



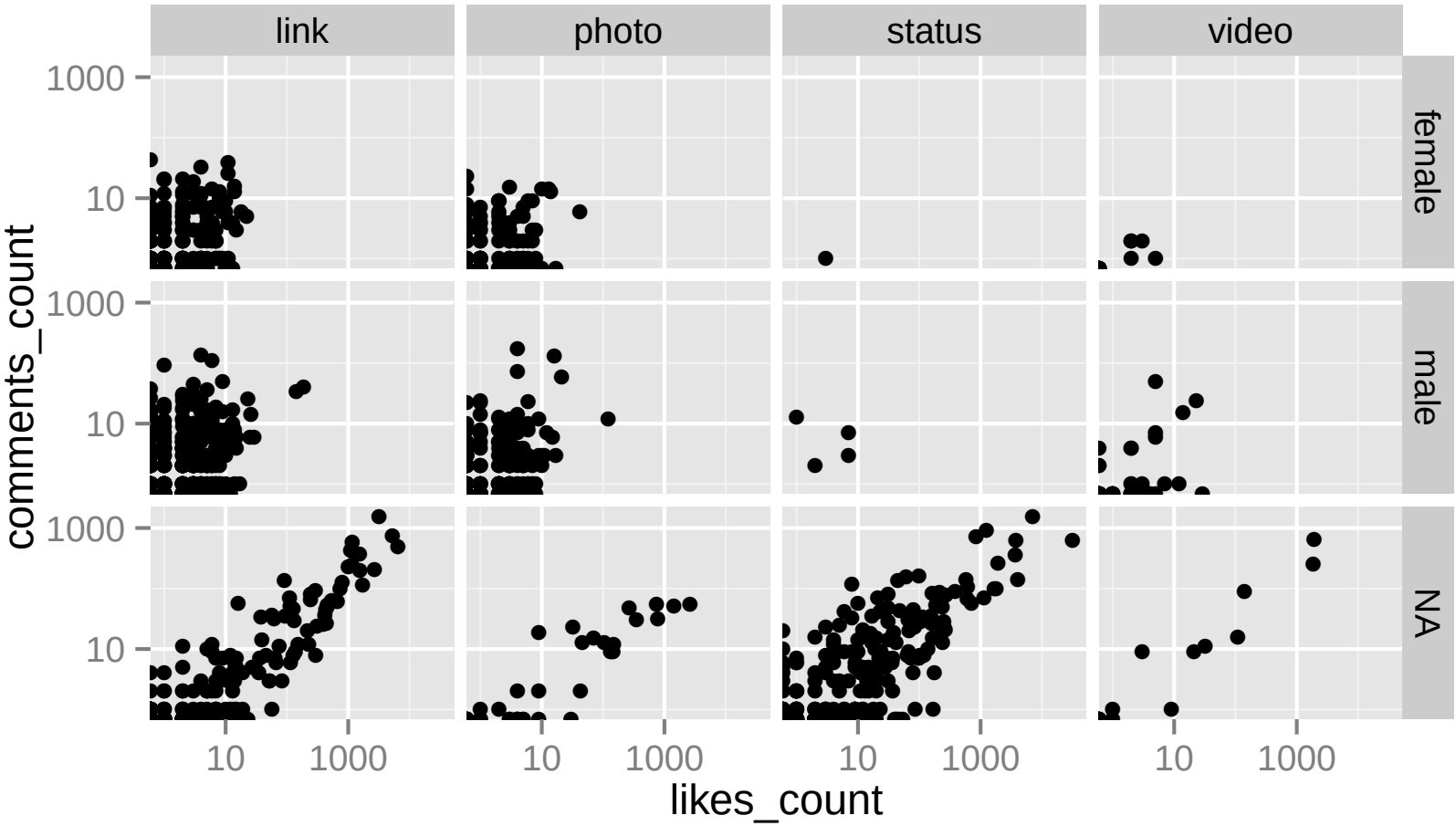
Facets

```
p + geom_point() + facet_wrap(~type, ncol=4)
```



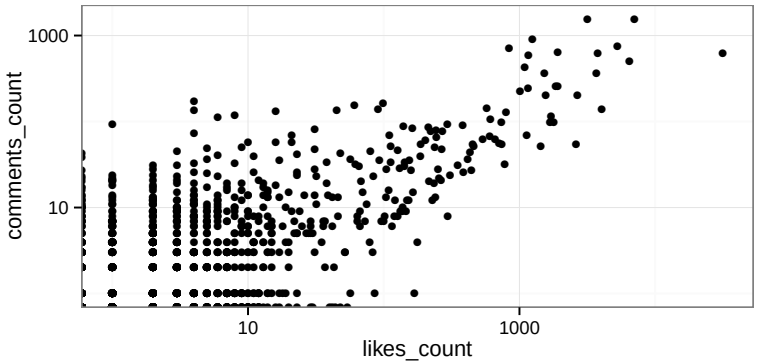
Facets

```
p + geom_point() + facet_grid(gender~type)
```

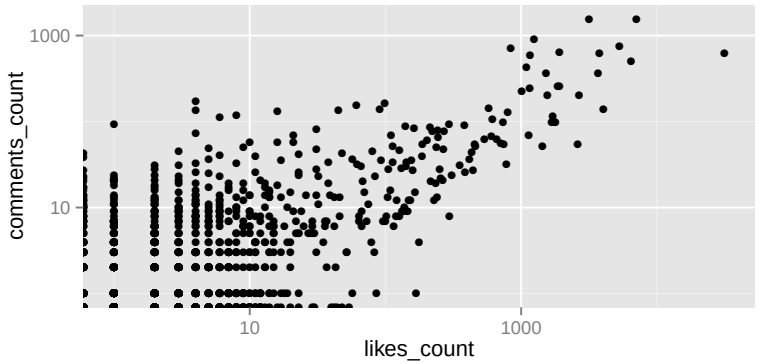


Themes

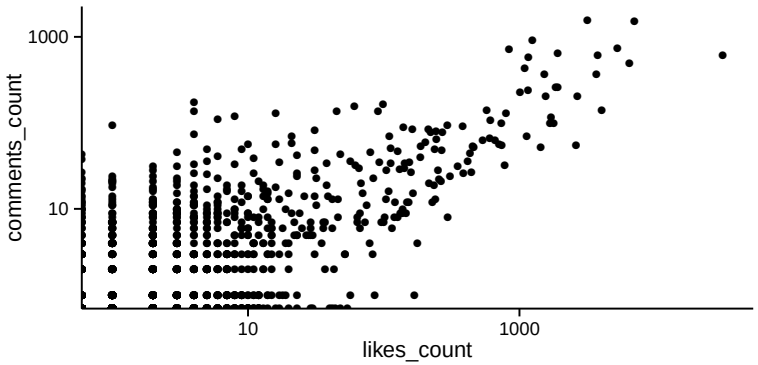
```
p + geom_point() + theme_bw()
```



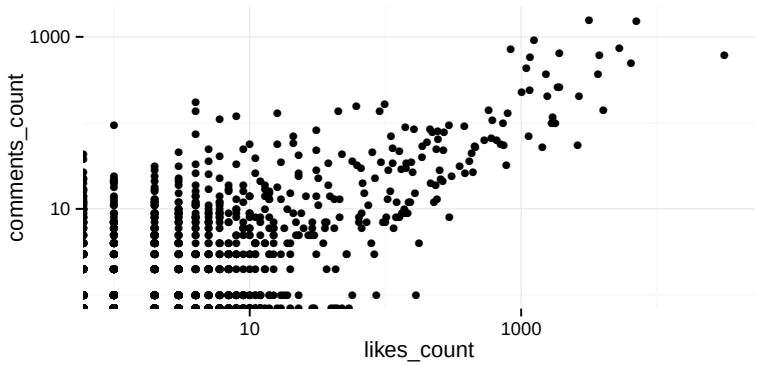

```
p + geom_point() + theme_grey()
```




```
p + geom_point() + theme_classic()
```

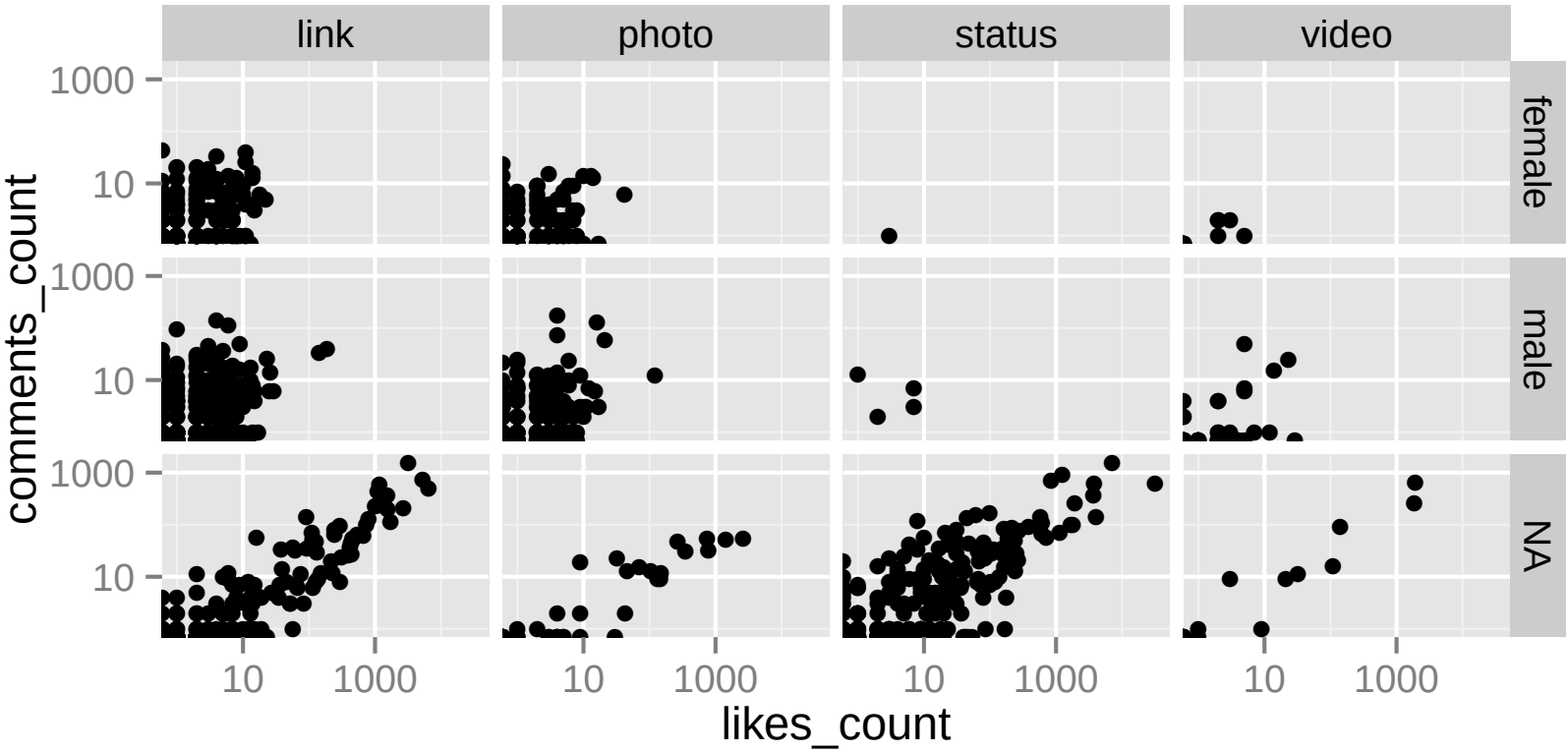



```
p + geom_point() + theme_minimal()
```



Saving plots

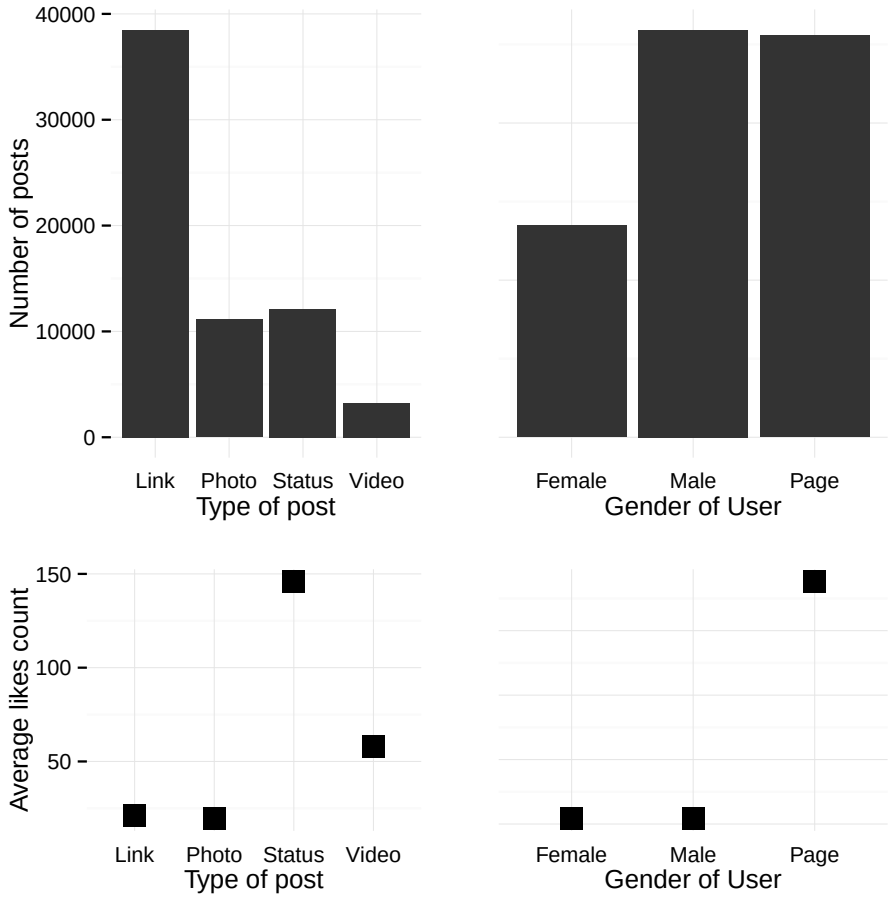
```
p + geom_point() + facet_grid(gender~type)
```




```
ggsave(pq, file="plots/plot.pdf", height=6, width=6)
```

Creating grid of plots

1. Store each plot as an object in memory
2. Use `pdf`, `grid arrange` and `arrangeGrob` (from `gridExtra` package) to create plot.



Example: unemployment rate in the U.S.

Code: `code/03_line_plots.R`

```
library(gdata)
d <- read.xls("../data/unemployment_data.xls", stringsAsFactors=F)
# subsetting only what we need
months <- as.character(d[9, 2:13])
years <- as.character(d[10:75, 1])
d <- as.numeric(unlist(t(d[10:75, 2:13])))
# putting it together into a data frame
df <- data.frame(expand.grid(months, years))
names(df) <- c("month", "year")
df$unemp <- d/100
# removing missing values
df <- df[!is.na(df$unemp),]
```

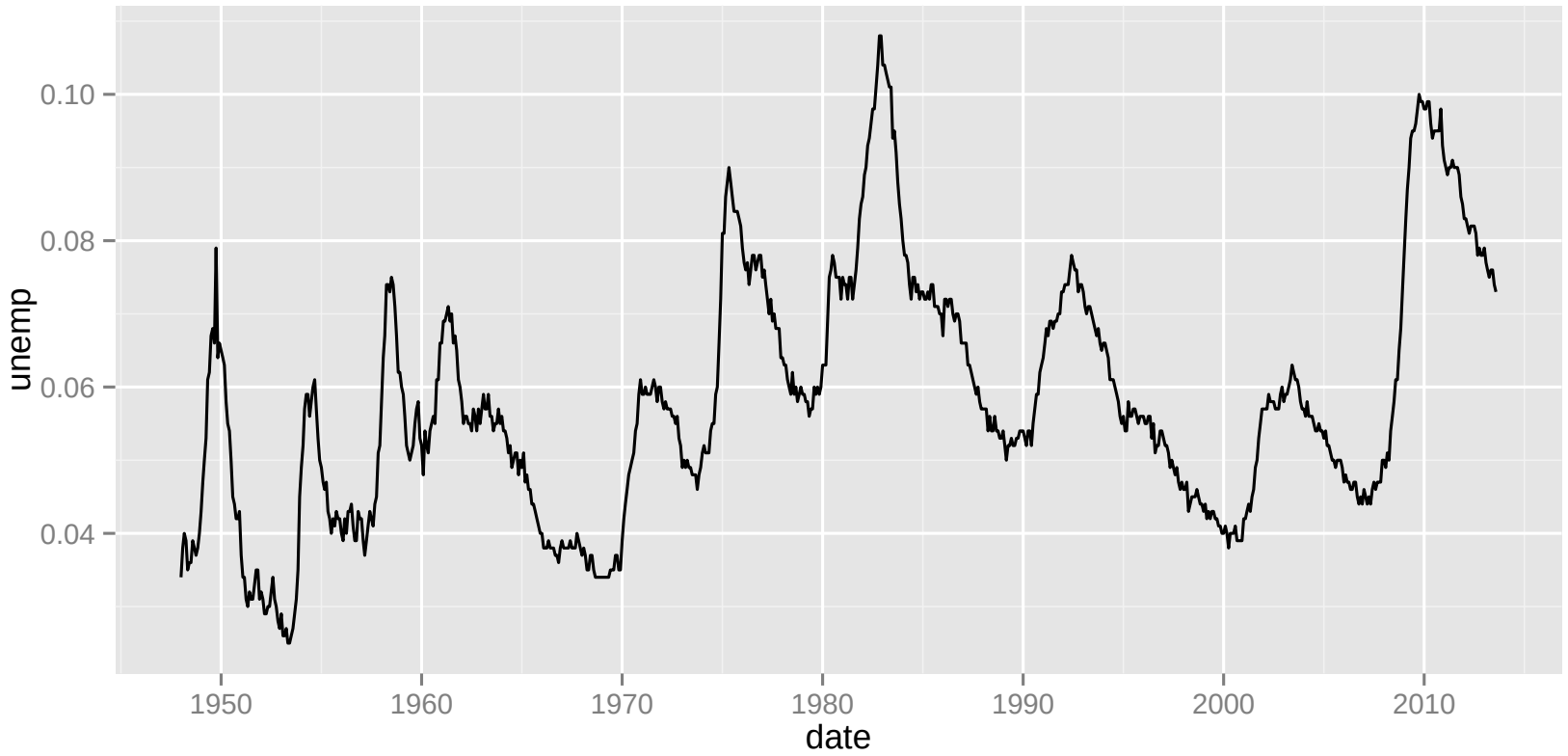
Example: unemployment rate in the U.S.

```
# creating a date variable
df$date <- paste("01", df$month, df$year)
df$date <- as.Date(df$date, format="%d %b %Y")
# this is what the data looks like...
str(df)

## 'data.frame': 788 obs. of  4 variables:
##  $ month: Factor w/ 12 levels "Jan","Feb","Mar",..: 1 2 3 4 5 6 7 8
##  $ year : Factor w/ 66 levels "1948","1949",..: 1 1 1 1 1 1 1 1 1 1
##  $ unemp: num  0.034 0.038 0.04 0.039 0.035 0.036 0.036 0.039 0.038
##  $ date : Date, format: "1948-01-01" "1948-02-01" ...
```

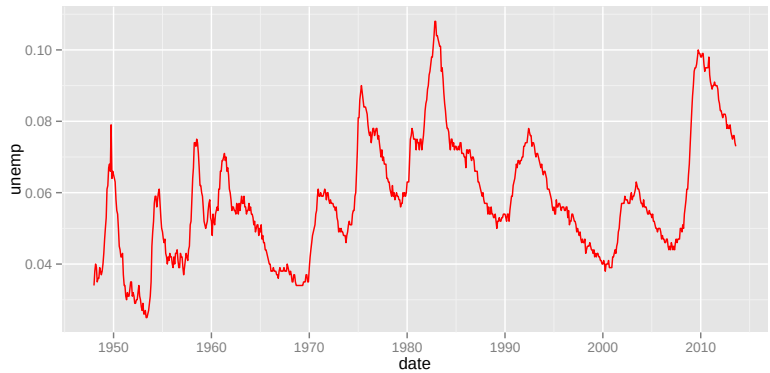
Example: unemployment rate in the U.S.

```
# plot with default options
library(ggplot2)
p <- ggplot(df, aes(x=date, y=unemp))
p + geom_line()
```

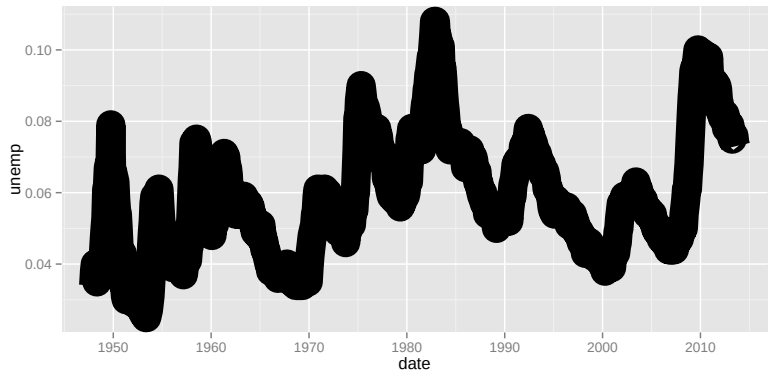


Example: unemployment rate in the U.S.

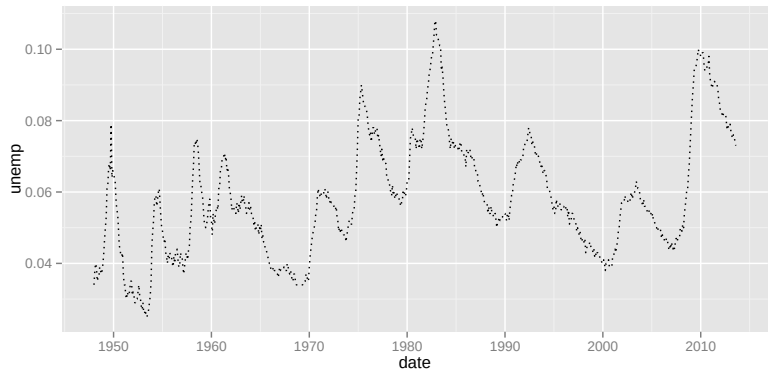
```
p + geom_line(color='red')
```



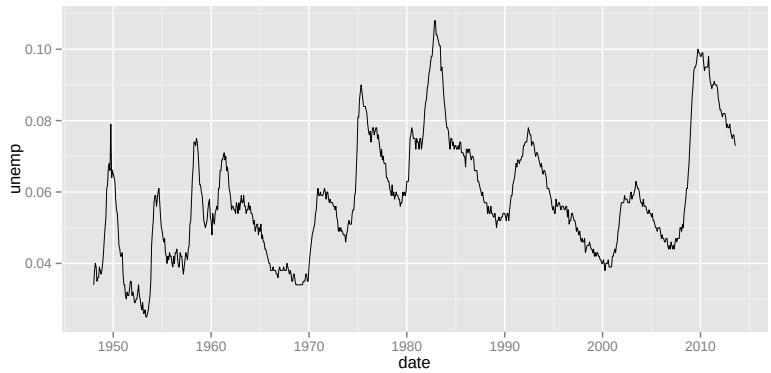

```
p + geom_line(size=10)
```




```
p + geom_line(linetype=3)
```

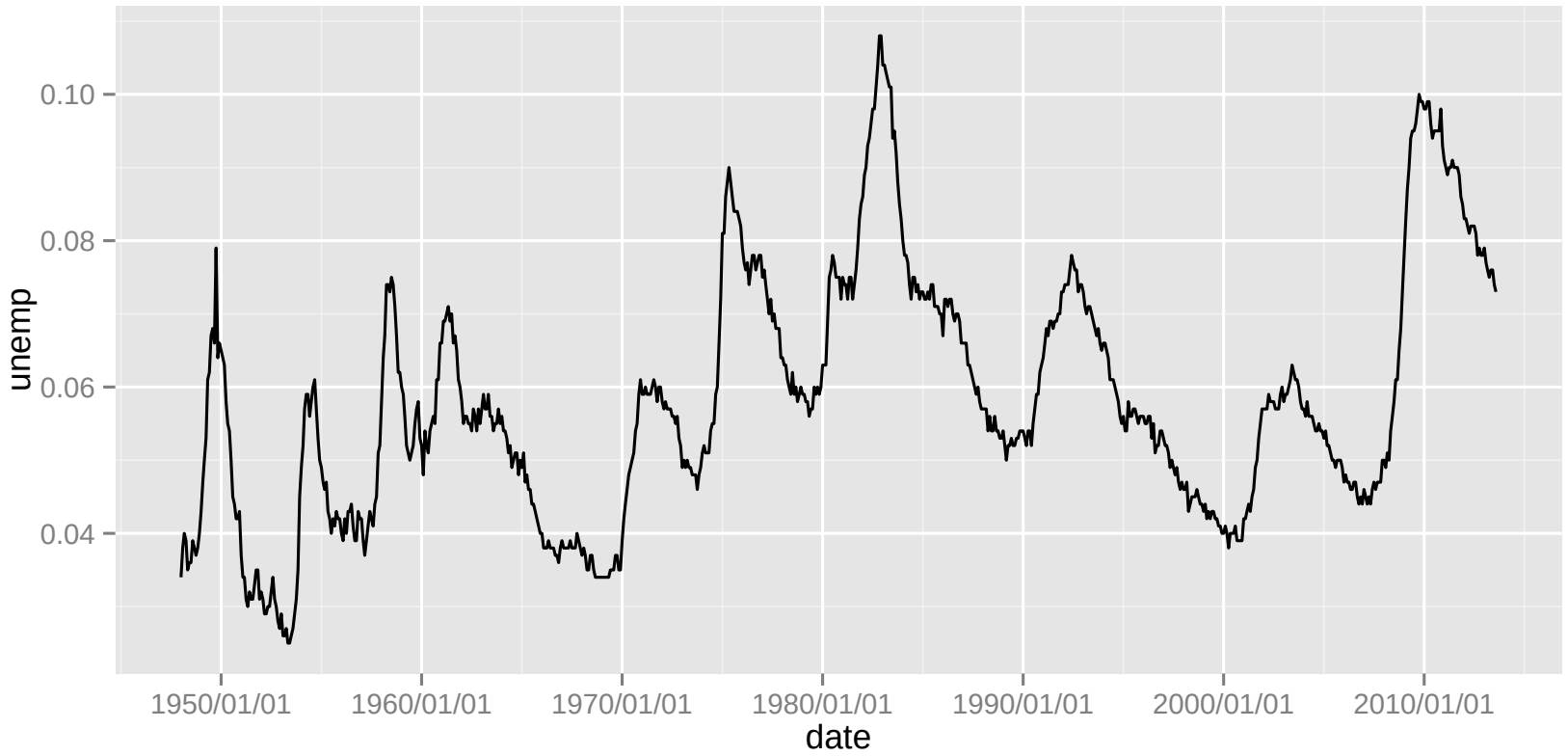



```
p + geom_line(size=0.1)
```



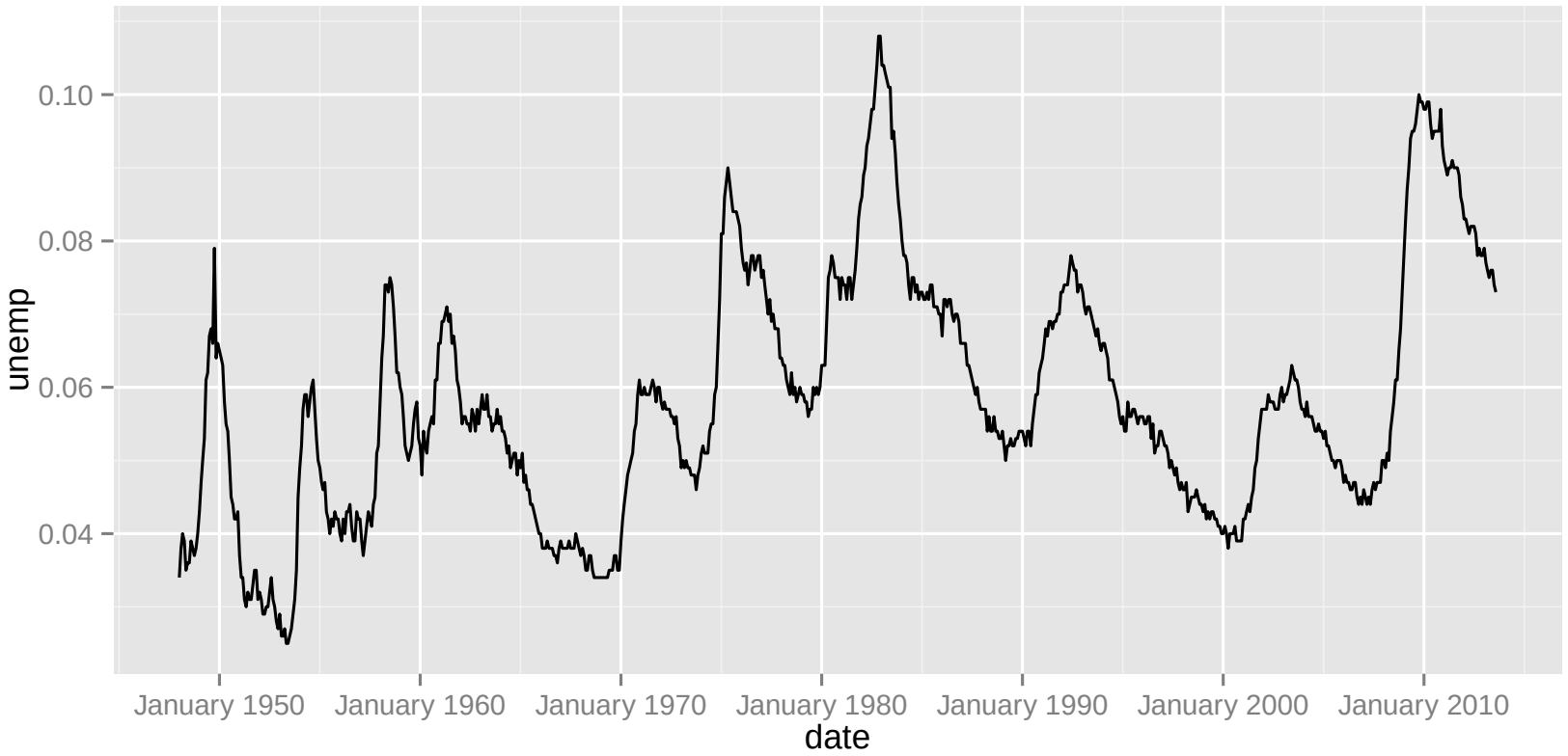
Example: unemployment rate in the U.S.

```
library(scales)
# customizing scale
p + geom_line() +  ## full date
   scale_x_date(labels = date_format("%Y/%m/%d"))
```



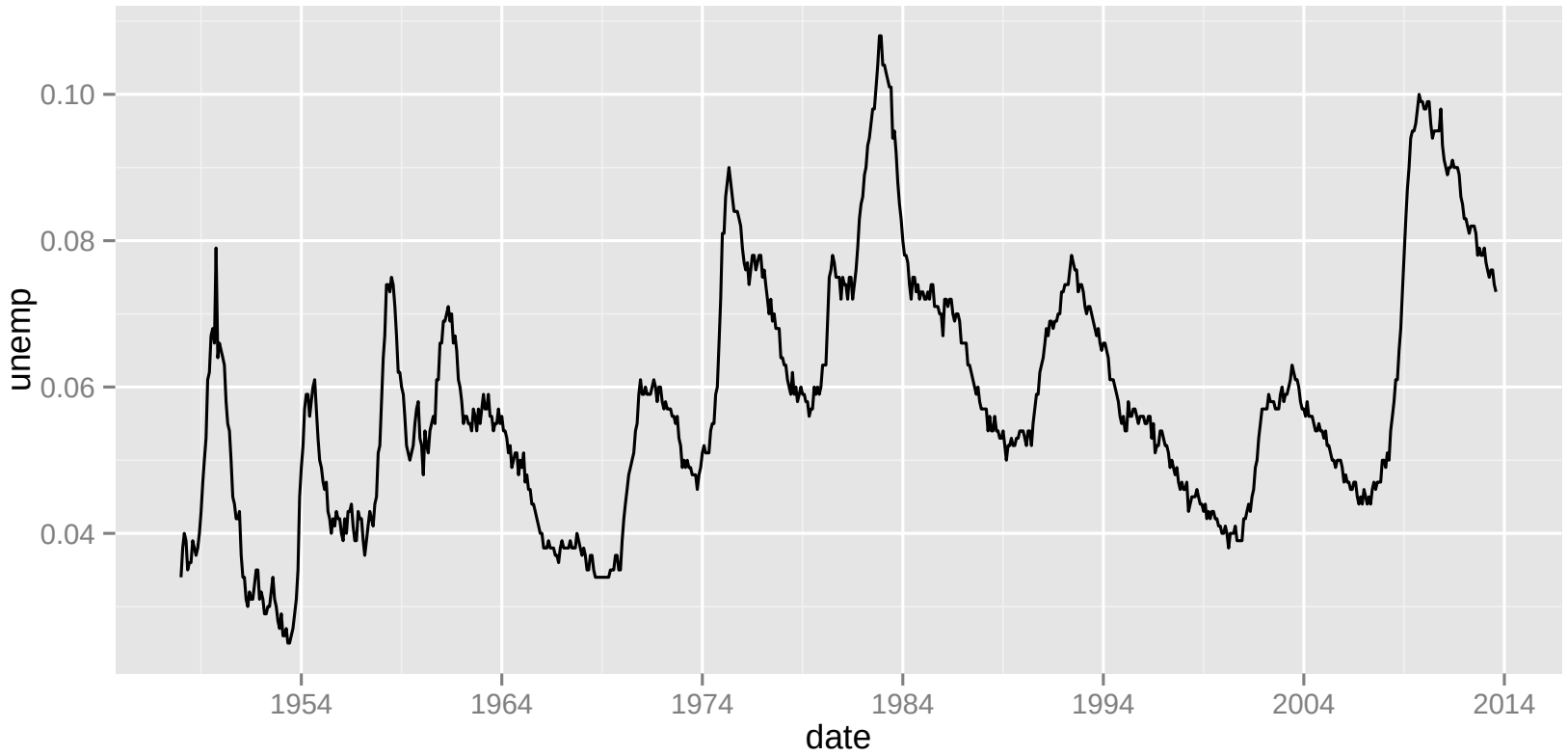
Example: unemployment rate in the U.S.

```
# customizing scale
p + geom_line() +  ## full month and year
   scale_x_date(labels = date_format("%B %Y"))
```



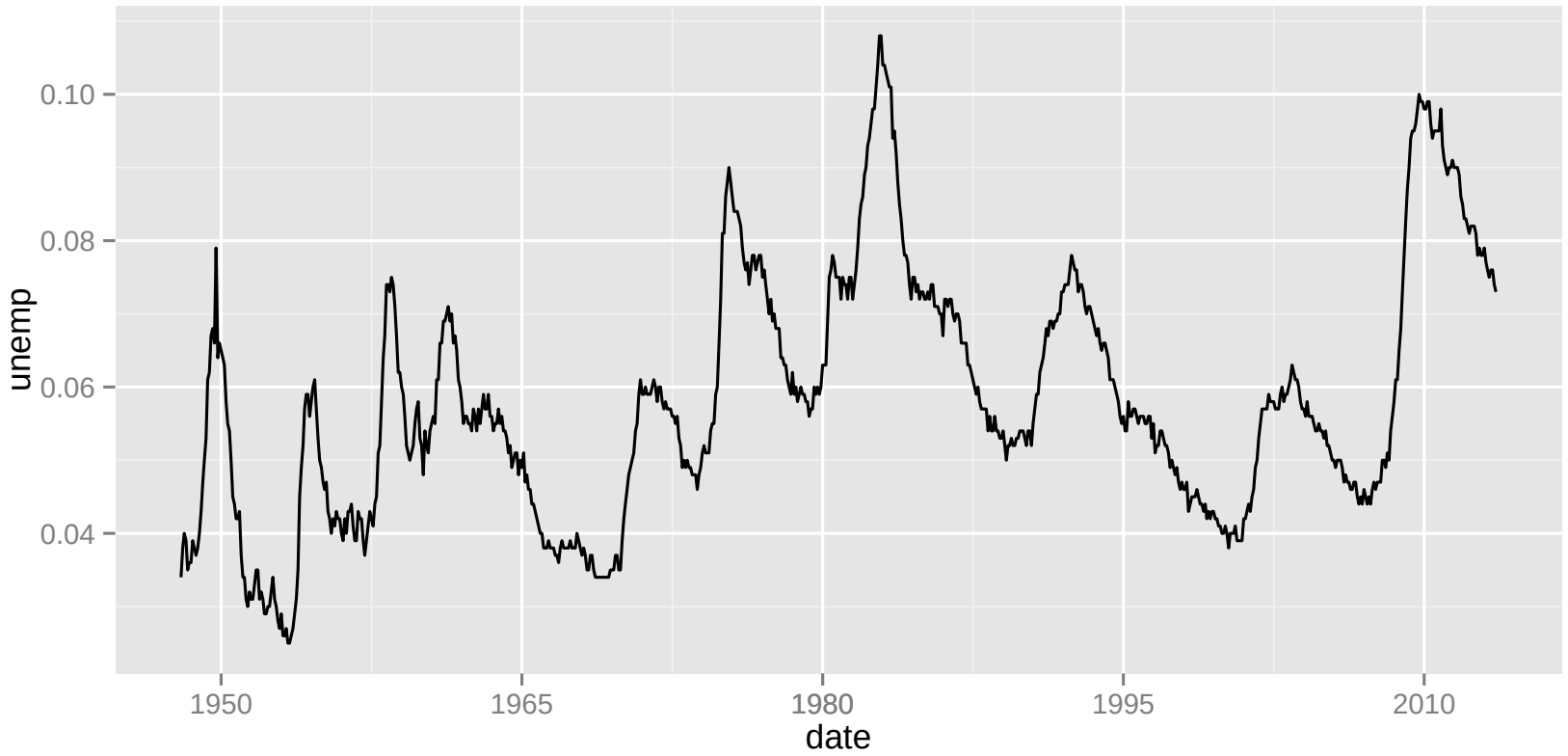
Example: unemployment rate in the U.S.

```
# customizing scale
p + geom_line() +  ## every 10 years
   scale_x_date(labels = date_format("%Y"),
    breaks = date_breaks("10 years"))
```



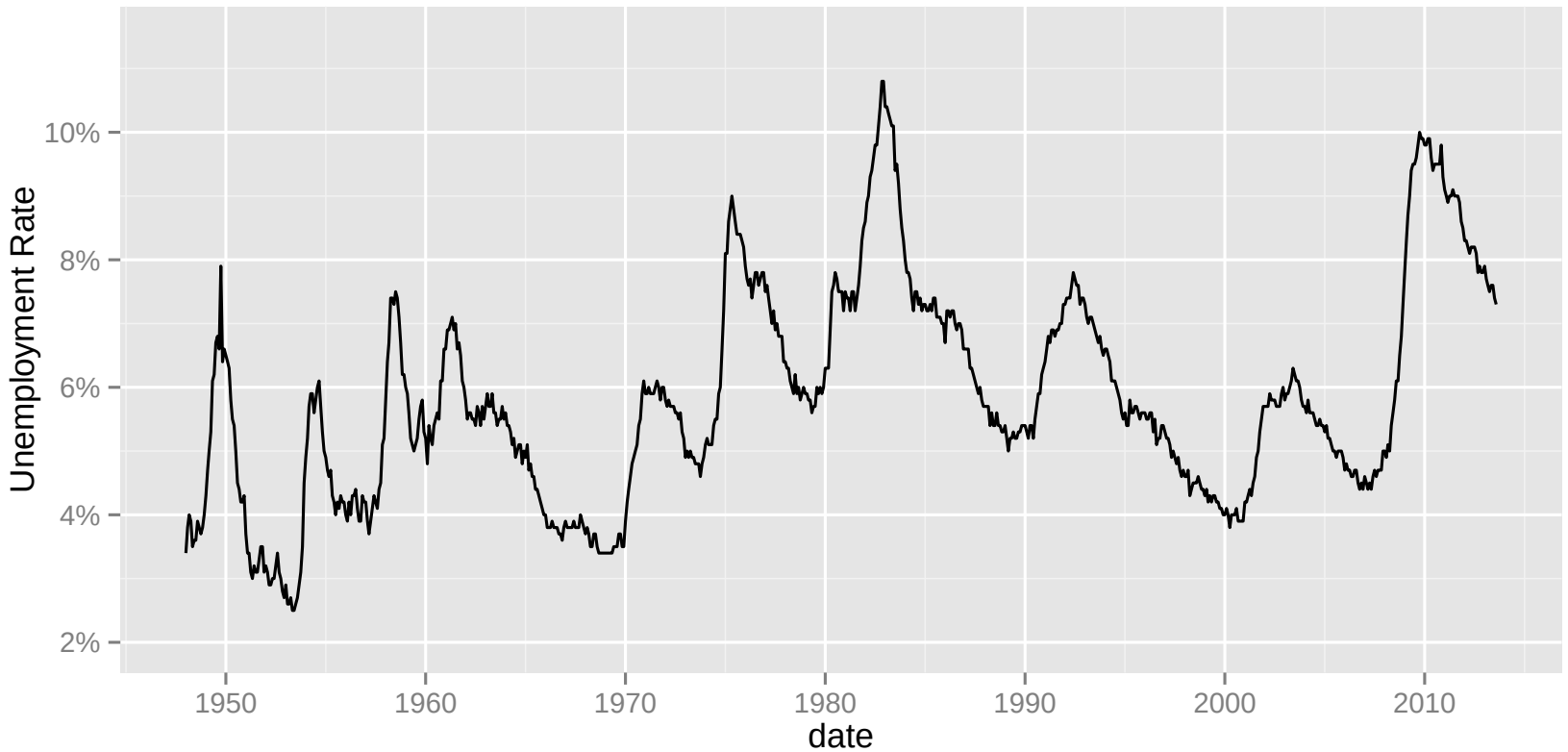
Example: unemployment rate in the U.S.

```
p + geom_line() +  ## manual breaks
scale_x_date(labels = date_format("%Y"),
    breaks = as.Date(c("1950-01-01", "1965-01-01", "1980-01-01",
        "1980-01-01", "1995-01-01", "2010-01-01")))
```



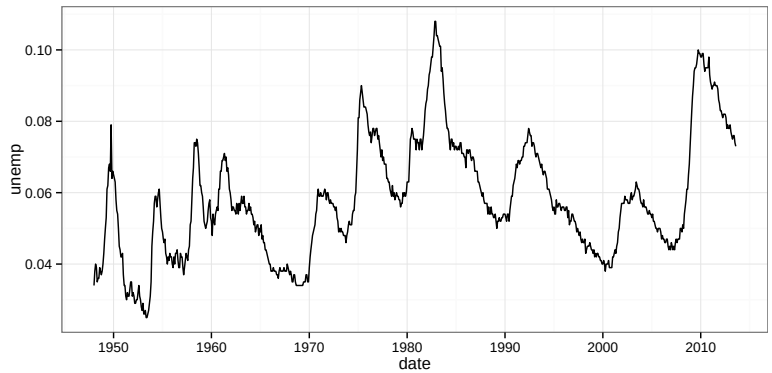
Example: unemployment rate in the U.S.

```
p + geom_line() +  ## customizing y scale
scale_y_continuous("Unemployment Rate", labels = percent,
 limits=c(0.02, 0.115), breaks=c(0.02, 0.04, 0.06, 0.08, 0.10))
```

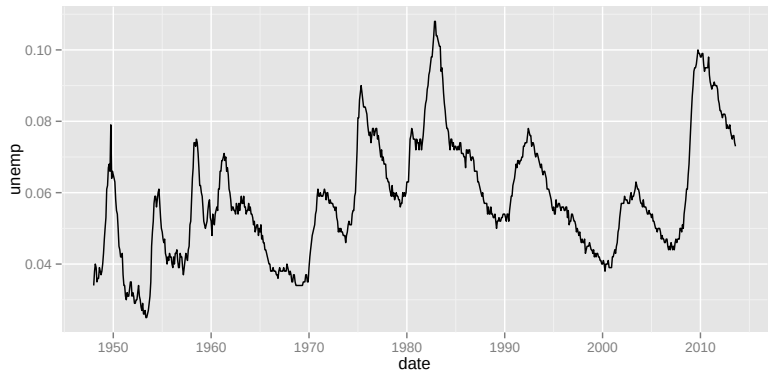


Example: unemployment rate in the U.S.

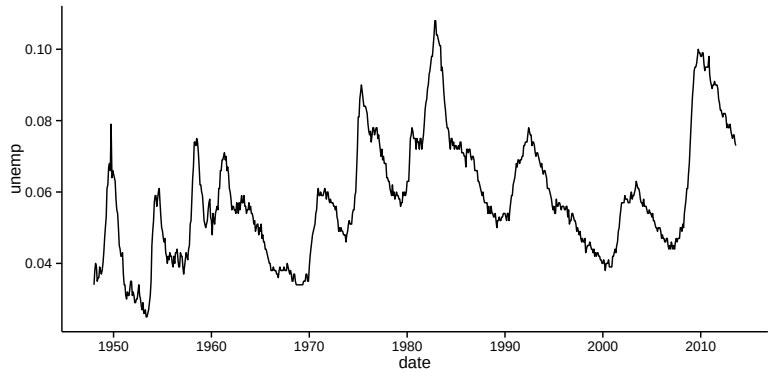
```
p + geom_line() + theme_bw()
```



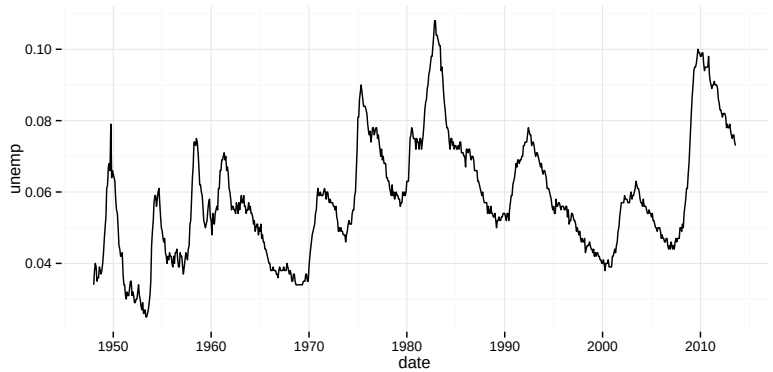

```
p + geom_line() + theme_grey()
```




```
p + geom_line() + theme_classic()
```




```
p + geom_line() + theme_minimal()
```

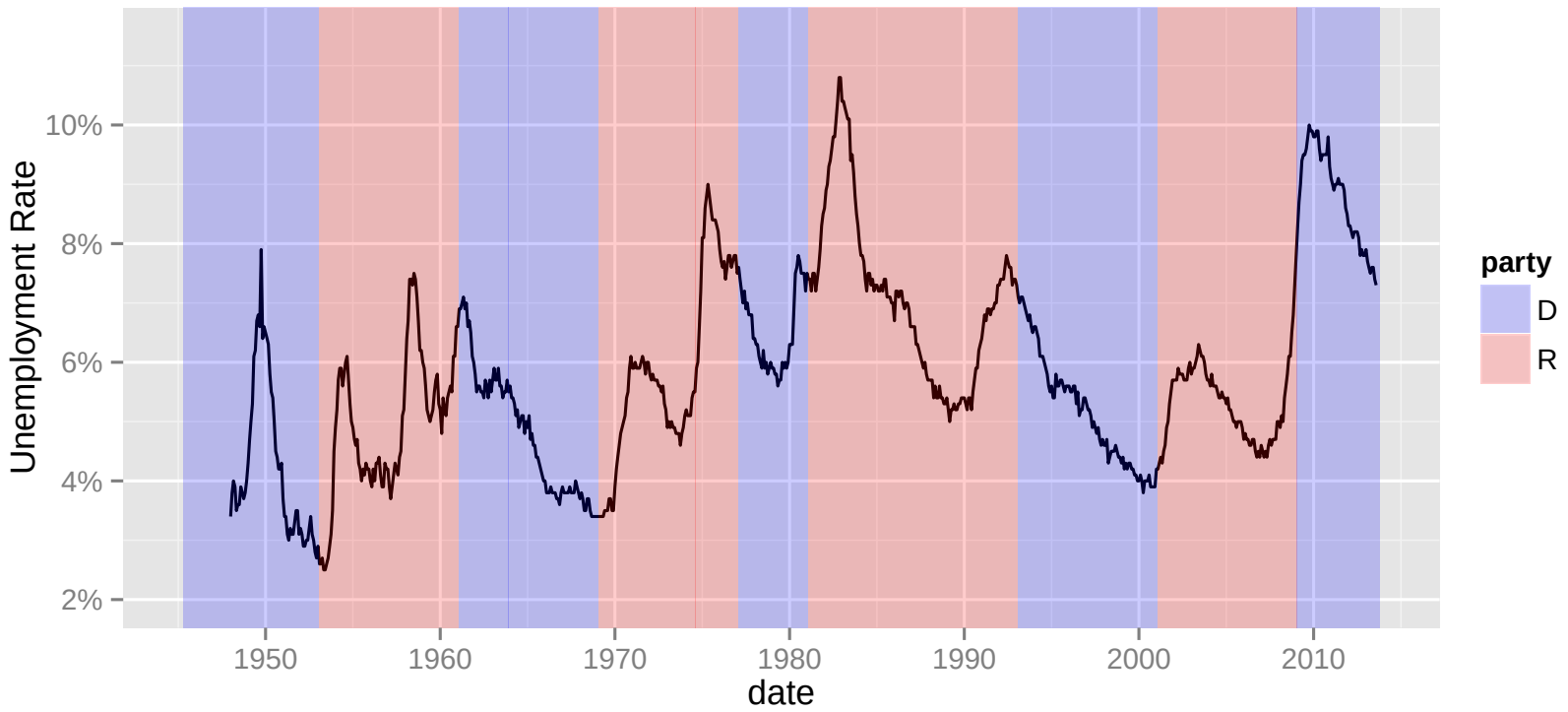


Example: unemployment rate in the U.S.

```
# background colors for different presidents
name <- c("Truman", "Eisenhower", "Kennedy", "Johnson", "Nixon",
        "Ford", "Carter", "Reagan", "Bush I", "Clinton",
        "Bush II", "Obama")
start <- as.Date(c("1945-04-12", "1953-01-20", "1961-01-20",
        "1963-11-22", "1969-01-20", "1974-08-09", "1977-01-20",
        "1981-01-20", "1989-01-20", "1993-01-20", "2001-01-20",
        "2009-01-20"))
end <- c(start[-1], as.Date("2013-10-15"))
party <- c("D", "R", "D", "D", "R", "R", "D", "R", "R", "D",
        "R", "D")
pres <- data.frame(name, start, end, party, stringsAsFactors=F)
pq <- p + geom_line() + scale_y_continuous("Unemployment Rate",
   labels = percent, limits = c(0.02, 0.115),
   breaks = c(0.02, 0.04, 0.06, 0.08, 0.10))
```

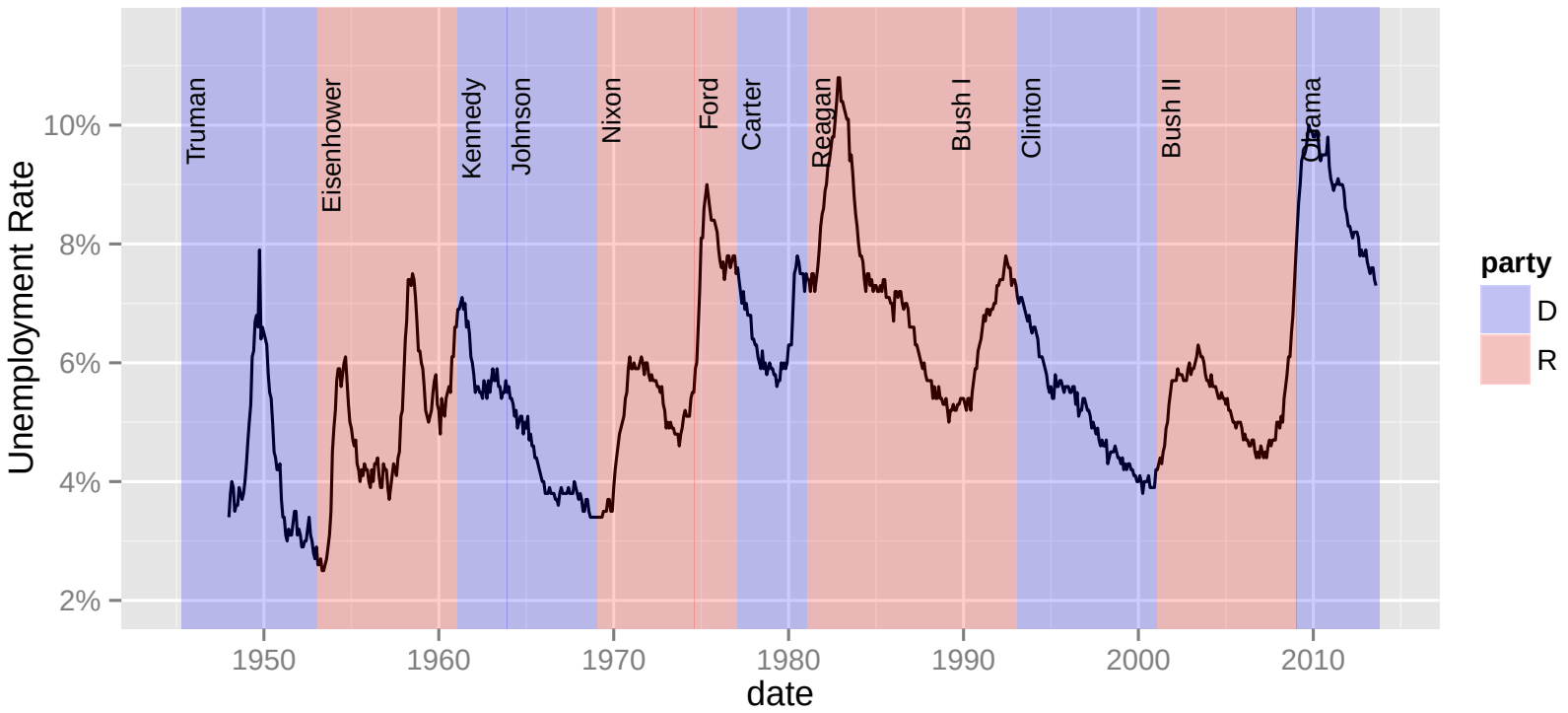
Example: unemployment rate in the U.S.

```
yrng <- range(df$unemp, na.rm=TRUE)
xrng <- range(df$date, na.rm=TRUE)
pq + geom_rect(data=pres, aes(NULL, NULL, xmin=start, xmax=end,
    fill=party), ymin=yrng[1]-0.05, ymax=yrng[2]+0.05) +
    scale_fill_manual(values = alpha(c("blue", "red"), 0.2))
```



Example: unemployment rate in the U.S.

```
pq + geom_rect(data=pres, aes(NULL, NULL, xmin=start, xmax=end,
    fill=party), ymin=yrng[1]-0.05, ymax=yrng[2]+0.05) +
    scale_fill_manual(values = alpha(c("blue", "red"), 0.2)) +
    geom_text(data=pres, aes(x=start, y=yrng[2], label=name),
    size=3, hjust=1, vjust=1.25, angle=90)
```

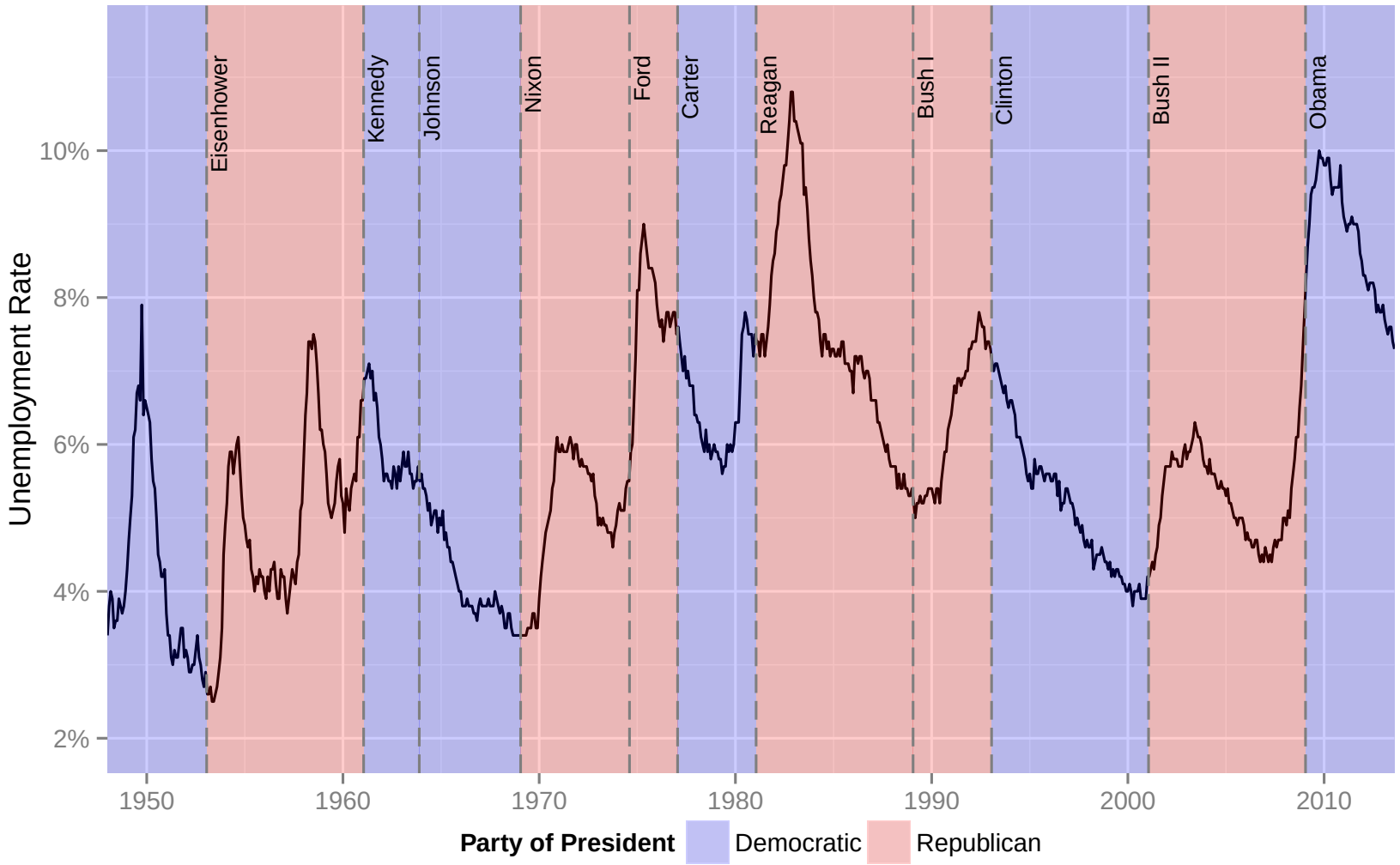


Example: unemployment rate in the U.S.

```
# final touches
library(grid)
yrng[2] <- yrng[2] + 0.05 # moving names up
pq <- pq + geom_rect(data=pres, aes(NULL, NULL, xmin=start,
    xmax=end, fill=party), ymin=yrng[1]-0.05, ymax=yrng[2]) +
scale_fill_manual(values = alpha(c("blue", "red"), 0.2))  +
scale_y_continuous("Unemployment Rate", labels = percent,
   limits = c(.02, .115), breaks=c(.02, .04, .06, .08, .10)) +
geom_text(data=pres, aes(x=start, y=yrng[2], label=name),
    size=3, hjust=1, vjust=1.25, angle=90) +
geom_vline(data=pres, aes(xintercept=as.numeric(start)),
    color="grey50", linetype=5, size=0.5) + # separation lines
theme(axis.title.x = element_blank(), # removing x title
    legend.position = "bottom", # legend to bottom
    legend.margin = unit(-1, "cm") ) + # less margin
 scale_fill_manual("Party of President",  # fixing labels
        values=alpha(c("blue", "red"), 0.2),
        labels= c("Democratic", "Republican"))
```

Example: unemployment rate in the U.S.

```
pq
```



Example: determinants of popularity on Facebook

Code: `code/04_coefficients_plots.R`

```
library(Rfacebook) # downloading public FB. posts
token <- "XXXXXXXXXX" # (your token here)
fb.data <- searchFacebook("shutdown", token, n=2000)
# recoding data
fb.data$gender[is.na(fb.data$gender)] <- "Page"
fb.data$gender[fb.data$gender == "male (hidden)"] <- "male"
fb.data$language <- substr(fb.data$locale, 1, 2)
fb.data$english <- ifelse(
    fb.data$language == "en" & !is.na(fb.data$language),
    "English", "Others")
```

Example: determinants of popularity on Facebook

```
fb.data$obamacare <- grepl("obamacare", fb.data$message,
   ignore.case=TRUE)
fb.data$boehner <- grepl("boehner", fb.data$message,
   ignore.case=TRUE)
fb.data$furlough <- grepl("furlough", fb.data$message,
   ignore.case=TRUE)
fb.data$time <- substr(fb.data$created_time, 12, 13)
fb.data$night <- fb.data$time %in%
   c("00", "01", "02", "03", "04", "05")
fb.data$morning <- fb.data$time %in%
   c("06", "07", "08", "09", "10", "11")
fb.data$afternoon <- fb.data$time %in% as.character(12:17)
fb.data$evening <- fb.data$time %in% as.character(18:23)
```

Example: determinants of popularity on Facebook

```
# Running regressions
r1 <- summary(lm(log(likes_count+1) ~ gender + type + english +
   obamacare + boehner + furlough + morning + afternoon +
   evening, data=fb.data))
r2 <- summary(lm(log(comments_count+1) ~ gender + type +
   english + obamacare + boehner + furlough + morning +
   afternoon + evening, data=fb.data))
r3 <- summary(lm(log(shares_count+1) ~ gender + type +
   english + obamacare + boehner + furlough + morning +
   afternoon + evening, data=fb.data))
```

Example: determinants of popularity on Facebook

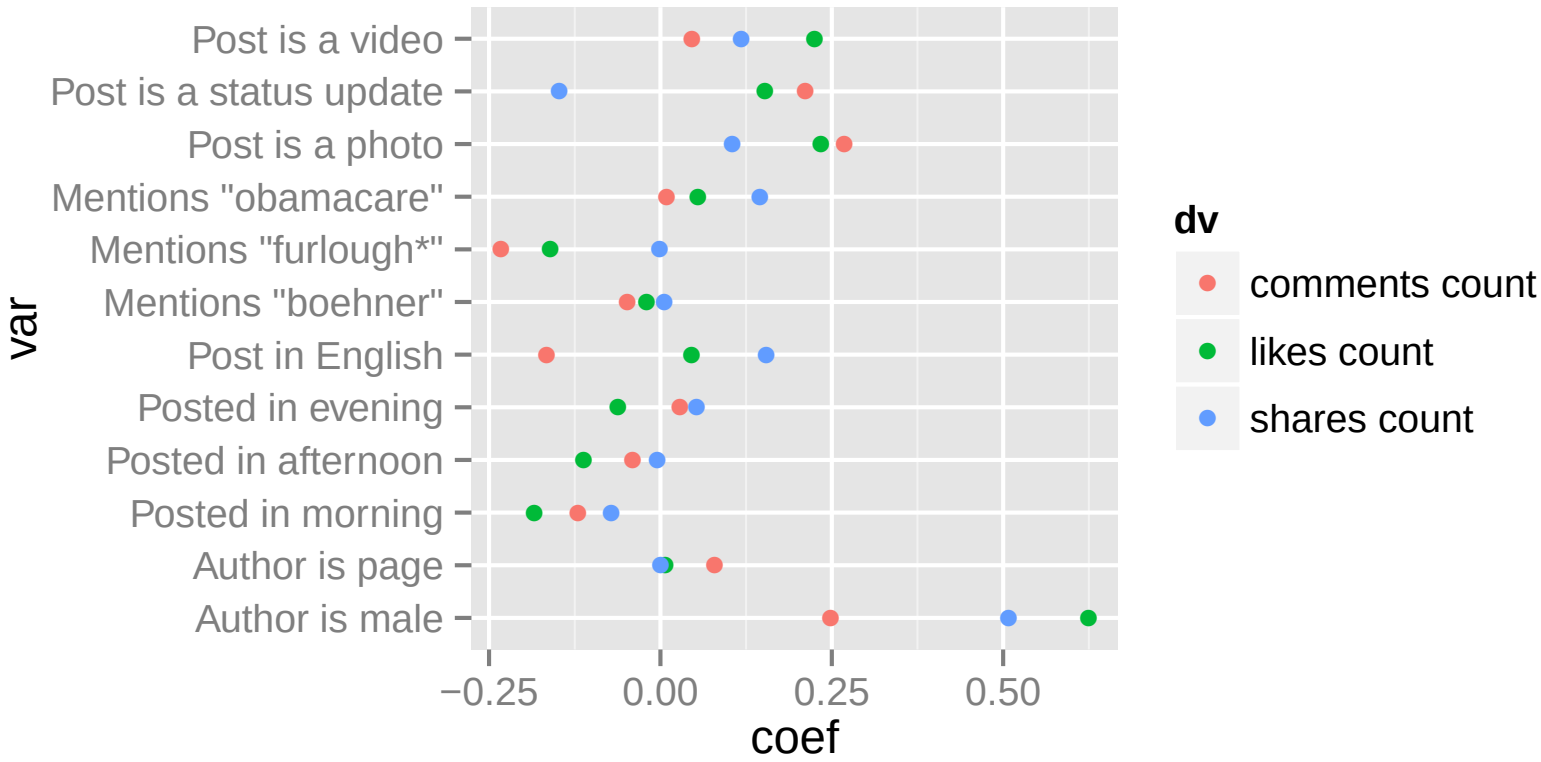
```
# Saving regressions into a data frame
df <- lapply(list(r1, r2, r3), function(x)
     data.frame(
    var = rownames(x$coefficients)[2:13],
    coef = x$coefficients[2:13, "Estimate"],
    sd = x$coefficients[2:13, "Std. Error"])
)
df <- do.call(rbind, df)
df$dv <- rep(c("likes count", "comments count",
    "shares count"), each=12)
```

Example: determinants of popularity on Facebook

```
# changing variable labels
levels(df$var) <- c("Posted in afternoon", 'Mentions "boehner"',
    "Post in English", "Posted in evening", 'Mentions "furlough*"',
    "Author is male", "Author is page", "Posted in morning",
    'Mentions "obamacare"', "Post is a photo",
    "Post is a status update", "Post is a video")
df$var <- factor(df$var, levels=c("Author is male", "Author is page",
    "Posted in morning", "Posted in afternoon", "Posted in evening",
    "Post in English", 'Mentions "boehner"', 'Mentions "furlough*"',
    'Mentions "obamacare"', "Post is a photo",
    "Post is a status update", "Post is a video"))
```

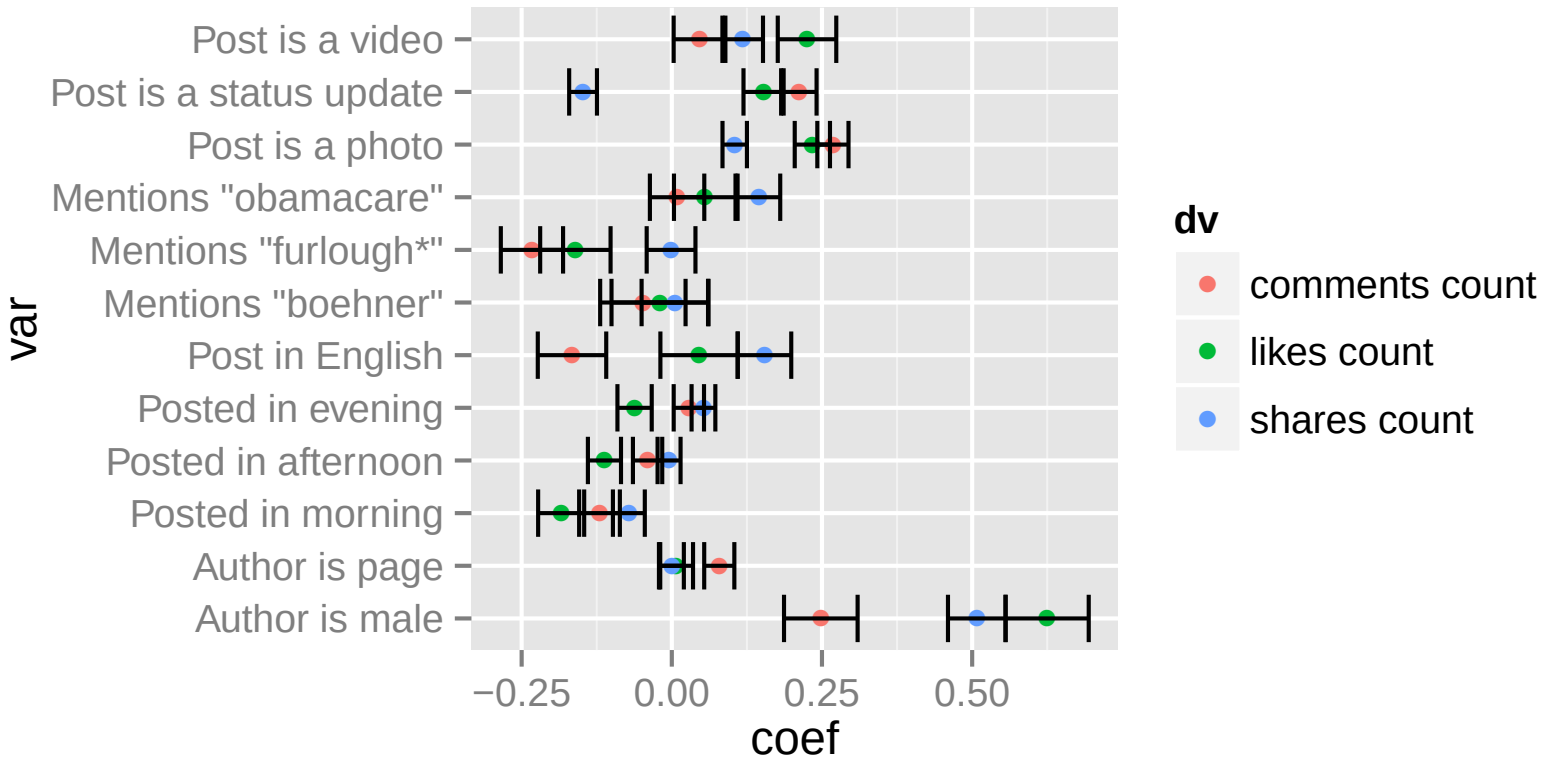
Example: determinants of popularity on Facebook

```
library(ggplot2)
p <- ggplot(df, aes(y=coef, x=var))
p + geom_point(aes(color=dv)) + coord_flip()
```



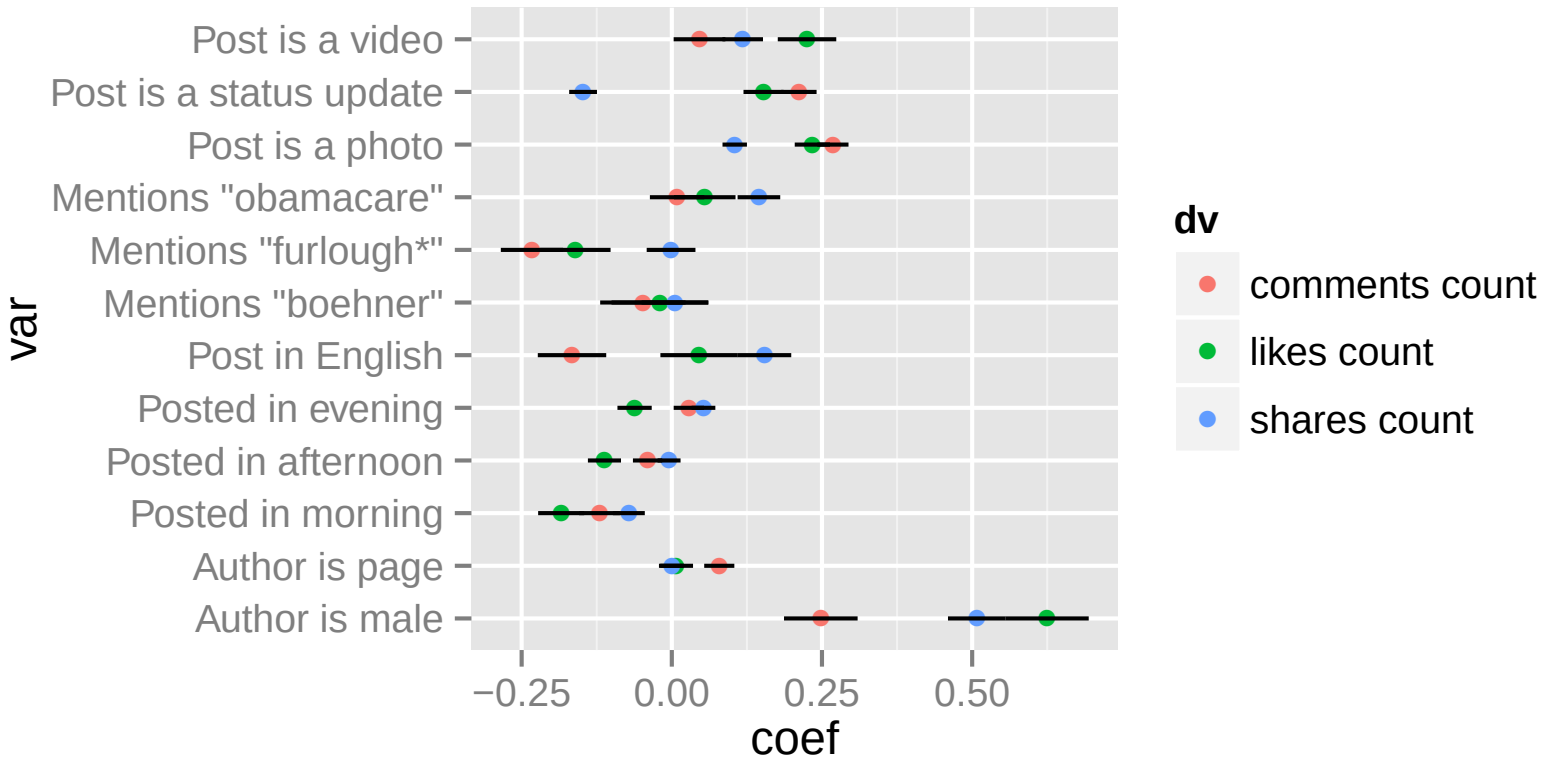
Example: determinants of popularity on Facebook

```
# adding error bars
p  + geom_point(aes(color=dv)) + coord_flip() +
    geom_errorbar(aes(x=var, ymin=coef-2*sd, ymax=coef+2*sd))
```



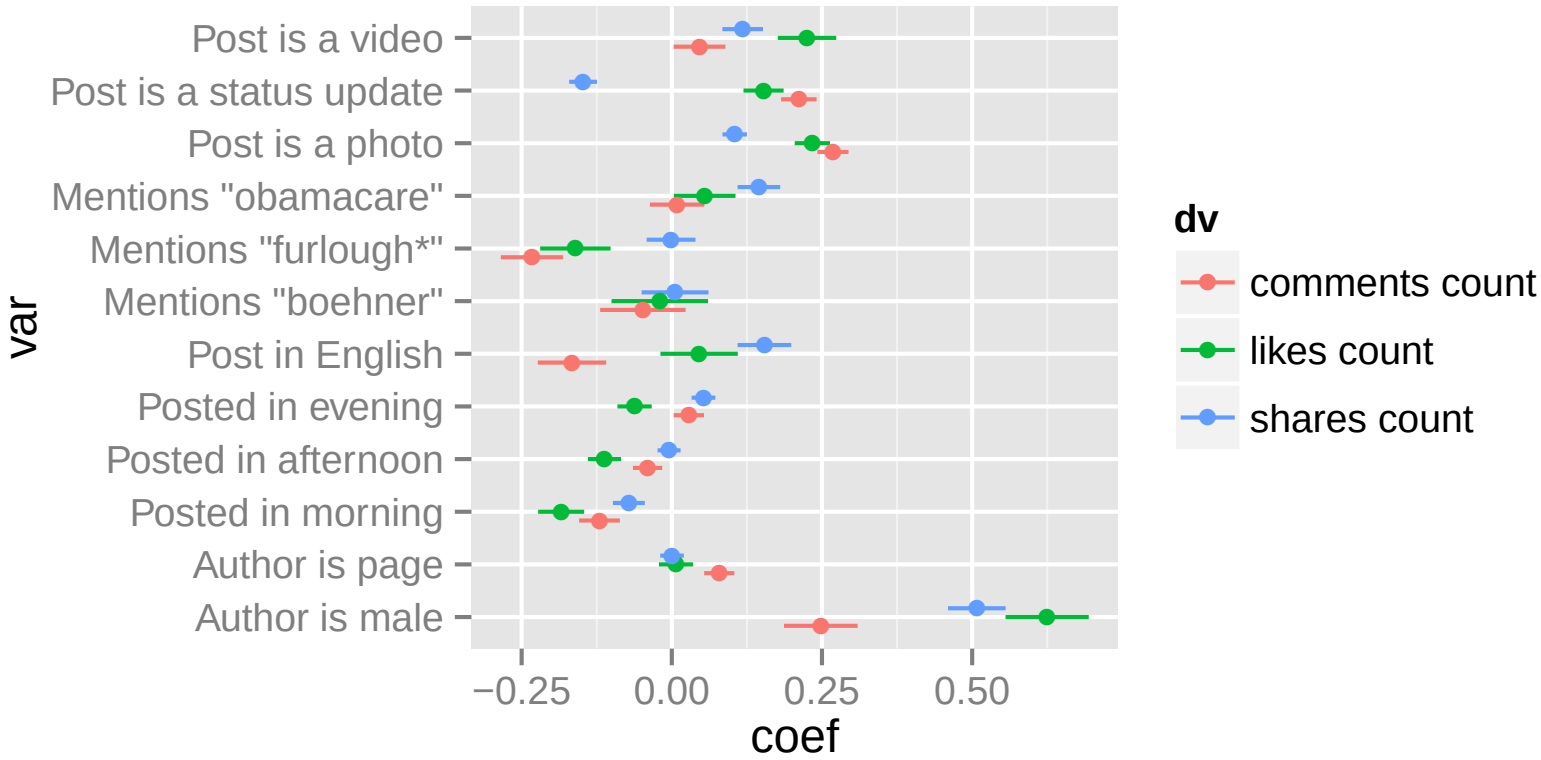
Example: determinants of popularity on Facebook

```
# adding lines to indicate uncertainty
p  + geom_point(aes(color=dv)) + coord_flip() +
    geom_linerange(aes(x=var, ymin=coef-2*sd, ymax=coef+2*sd))
```



Example: determinants of popularity on Facebook

```
# order matters! also use position_dodge to avoid overlap
p <- ggplot(df, aes(y=coef, x=var, group=dv))
p + coord_flip() + geom_linerange(aes(ymin=coef-2*sd,
   ymax=coef+2*sd, color=dv), position=position_dodge(.5)) +
    geom_point(aes(color=dv), position = position_dodge(.5))
```



Example: determinants of popularity on Facebook

```
# fixing axes, legends, line at 0... and putting it all together
library(scales)
p <- ggplot(df, aes(y=coef, x=var))
pq <- p + coord_flip() +
    # adding first (long, thin) line for coef +- 2 sd
    geom_linerange(aes(ymin=coef-2 *sd, ymax=coef+2*sd, color=dv),
    position=position_dodge(.5)) +
    # adding second (short, thick) line for coef +- 1 sd
    geom_linerange(aes(ymin=coef-sd, ymax=coef+sd, color=dv),
    position=position_dodge(.5), size=1) +
    # changing y axis title and scale (y and not bc of coord_flip)
    scale_y_continuous("% increase in popularity metric",
      labels = percent) +
    # adding line at 0
    geom_hline(yintercept=0, linetype=5, color="grey50")
```

Example: determinants of popularity on Facebook

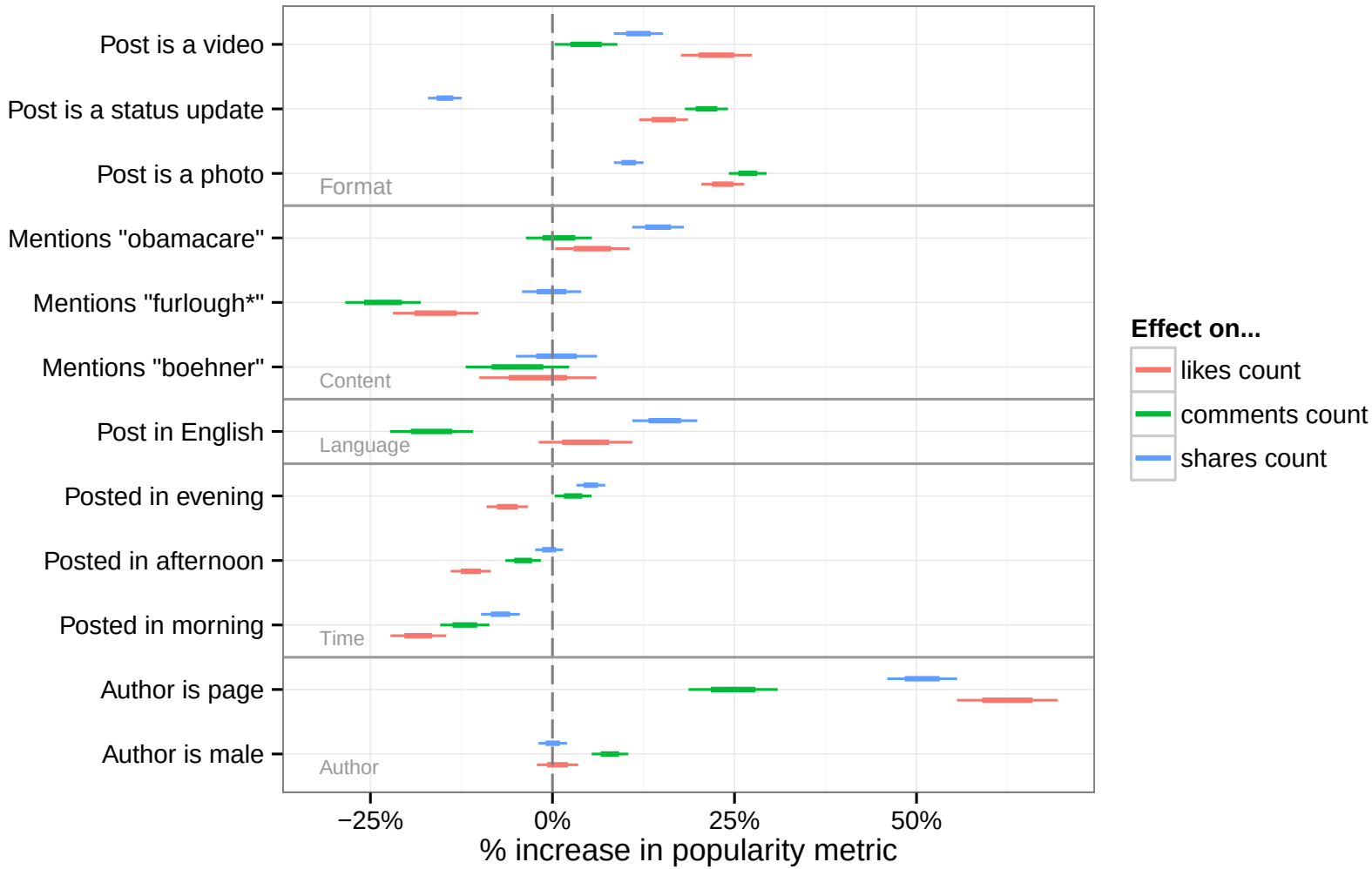
```
# changing theme and removing axis title
pq <- pq + theme_bw() + theme(
    axis.title.y=element_blank() ) +
    # changing title of legend
    scale_color_discrete("Effect on...") +
    # horizontal lines to separate different types of variables
    geom_vline(xintercept=2.5, color="grey60") +
    geom_vline(xintercept=5.5, color="grey60") +
    geom_vline(xintercept=6.5, color="grey60") +
    geom_vline(xintercept=9.5, color="grey60")
```

Example: determinants of popularity on Facebook

```
# adding labels indicating types of variables
pq <- pq + annotate("text", x=9.8, y=-.32, label="Format",
        size=3, color="grey60", hjust=0) +
    annotate("text", x=6.8, y=-.32, label="Content",
        size=2.8, color="grey60", hjust=0) +
    annotate("text", x=5.8, y=-.32, label="Language",
        size=2.8, color="grey60", hjust=0) +
    annotate("text", x=2.8, y=-.32, label="Time",
        size=2.8, color="grey60", hjust=0) +
    annotate("text", x=0.8, y=-.32, label="Author",
        size=2.8, color="grey60", hjust=0)
```

Example: determinants of popularity on Facebook

```
pq
```



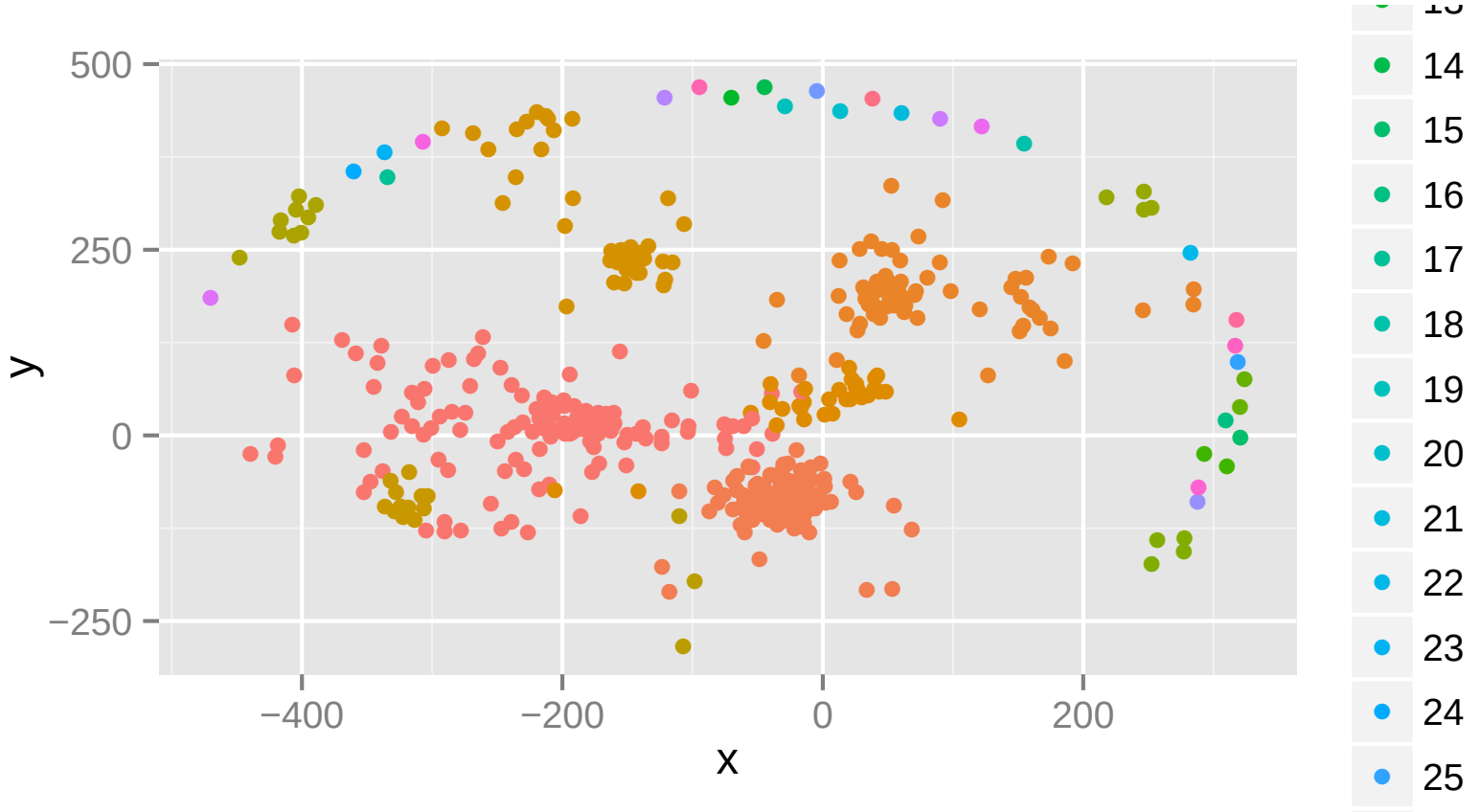
Example: visualizing your Facebook network

Code: `code/05_networks.R`

```
library(Rfacebook) # downloading network of FB friends
token <- "XXXXXXXXXX" # (your token here)
mat <- getNetwork(token, format="adj.matrix")
# preparing node list and layout with igraph
library(graph)
network <- graph.adjacency(mat, mode="undirected")
fc <- fastgreedy.community(network) ## communities
l <- layout.fruchterman.reingold(network, niter=1000, coolexp=0.5)
# preparing data for plot
d <- data.frame(l); names(d) <- c("x", "y")
d$cluster <- factor(fc$membership)
```

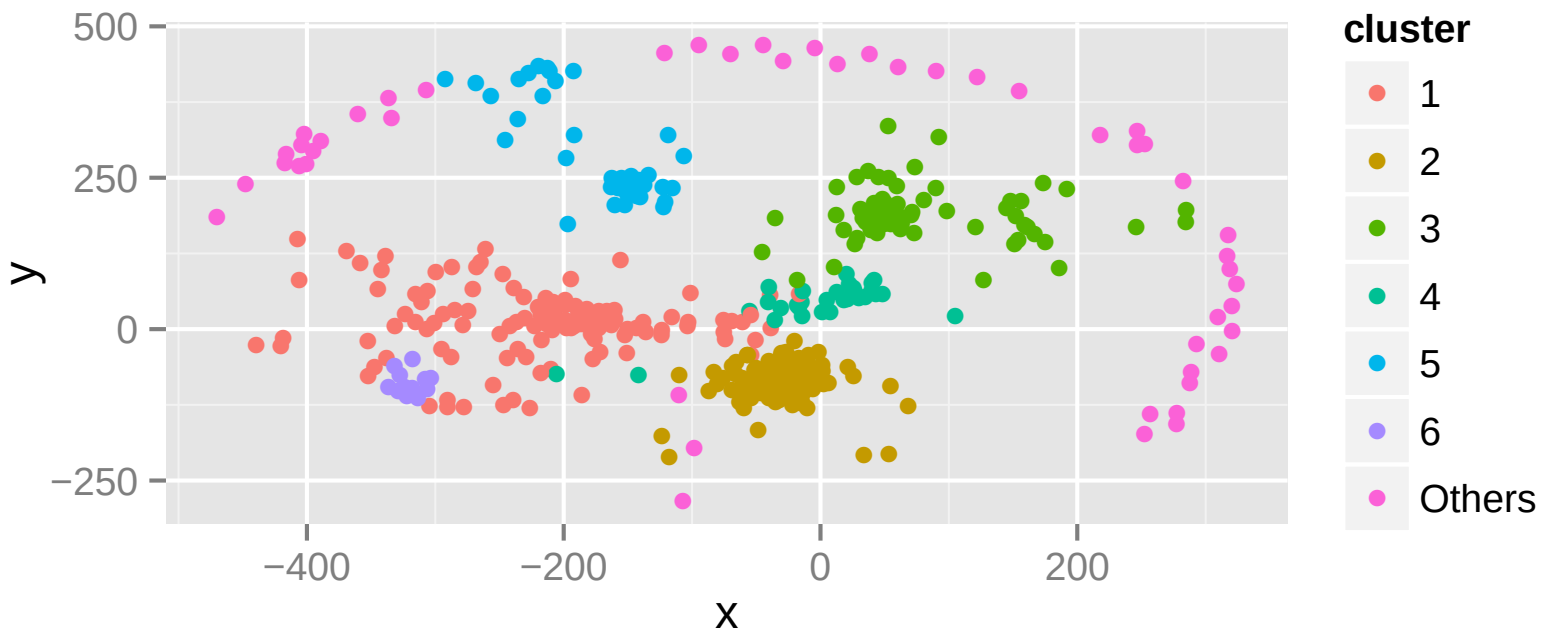
Example: visualizing your Facebook network

```
p <- ggplot(d, aes(x=x, y=y, color=cluster))
(pq <- p + geom_point())
```



Example: visualizing your Facebook network

```
## too many clusters! let's pick just those with 10 friends or more
large.clusters <- which(table(fc$membership)>=10)
fc$membership[fc$membership %in% large.clusters == FALSE] <- "Others"
d$cluster <- factor(fc$membership)
p <- ggplot(d, aes(x=x, y=y, color=cluster))
(pq <- p + geom_point())
```

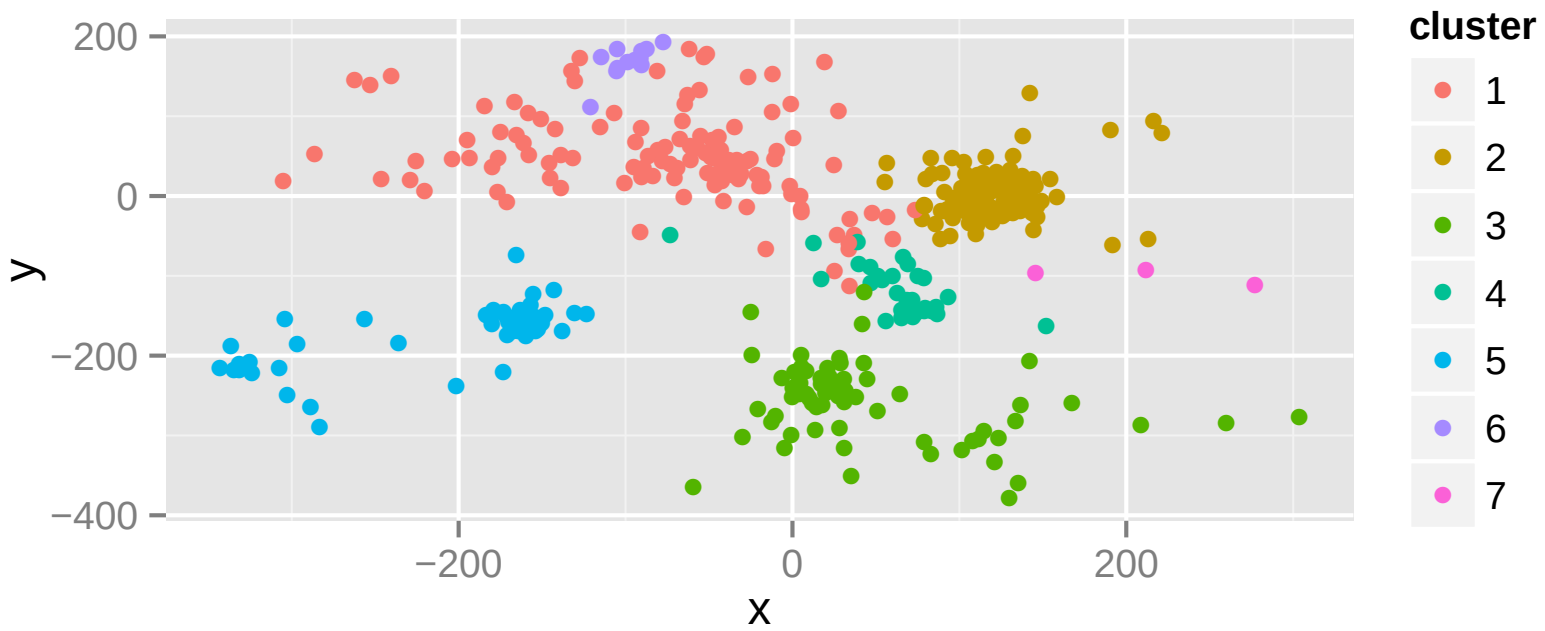


Example: visualizing your Facebook network

```
## simplify even further by keeping only nodes in giant component
cl <- clusters(network)
gc <- which(cl$membership == 1)
mat <- mat[gc, gc]
network <- graph.adjacency(mat, mode="undirected")
fc <- fastgreedy.community(network)
set.seed(123)
l <- layout.fruchterman.reingold(network, niter=1000, coolexp=0.5)
d <- data.frame(l); names(d) <- c("x", "y")
d$cluster <- factor(fc$membership)
```

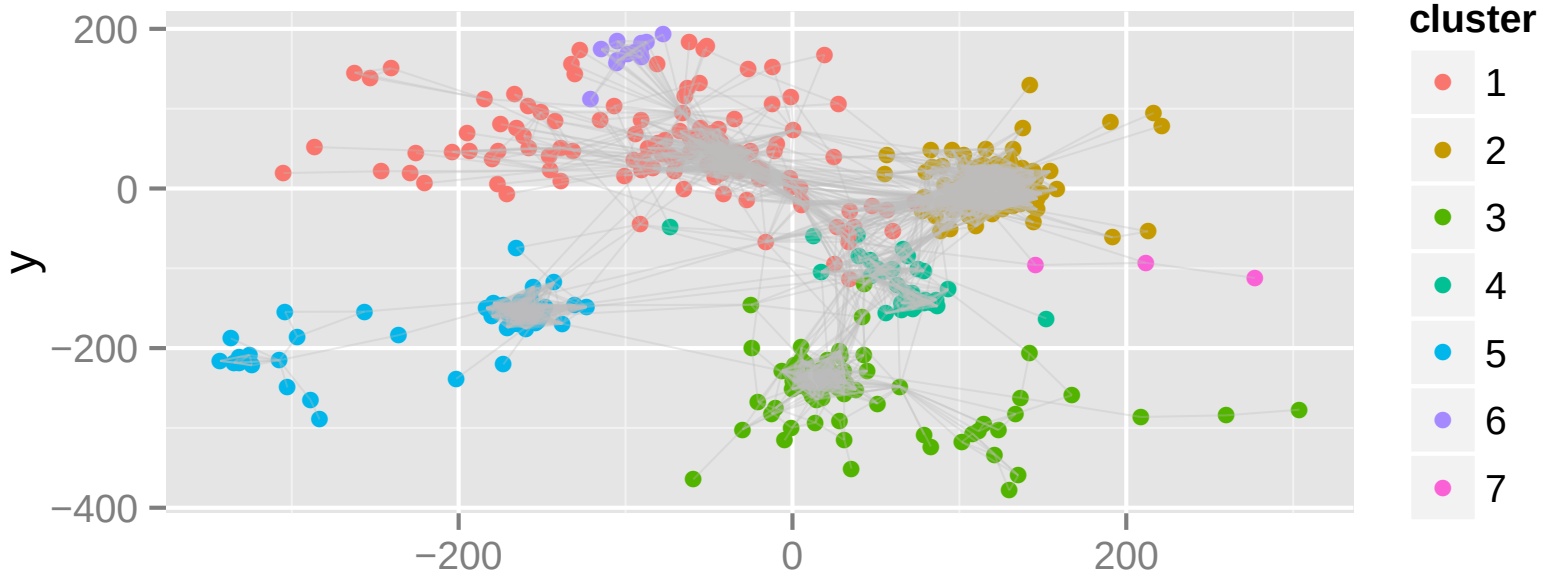
Example: visualizing your Facebook network

```
p <- ggplot(d, aes(x=x, y=y, color=cluster))
(pq <- p + geom_point())
```



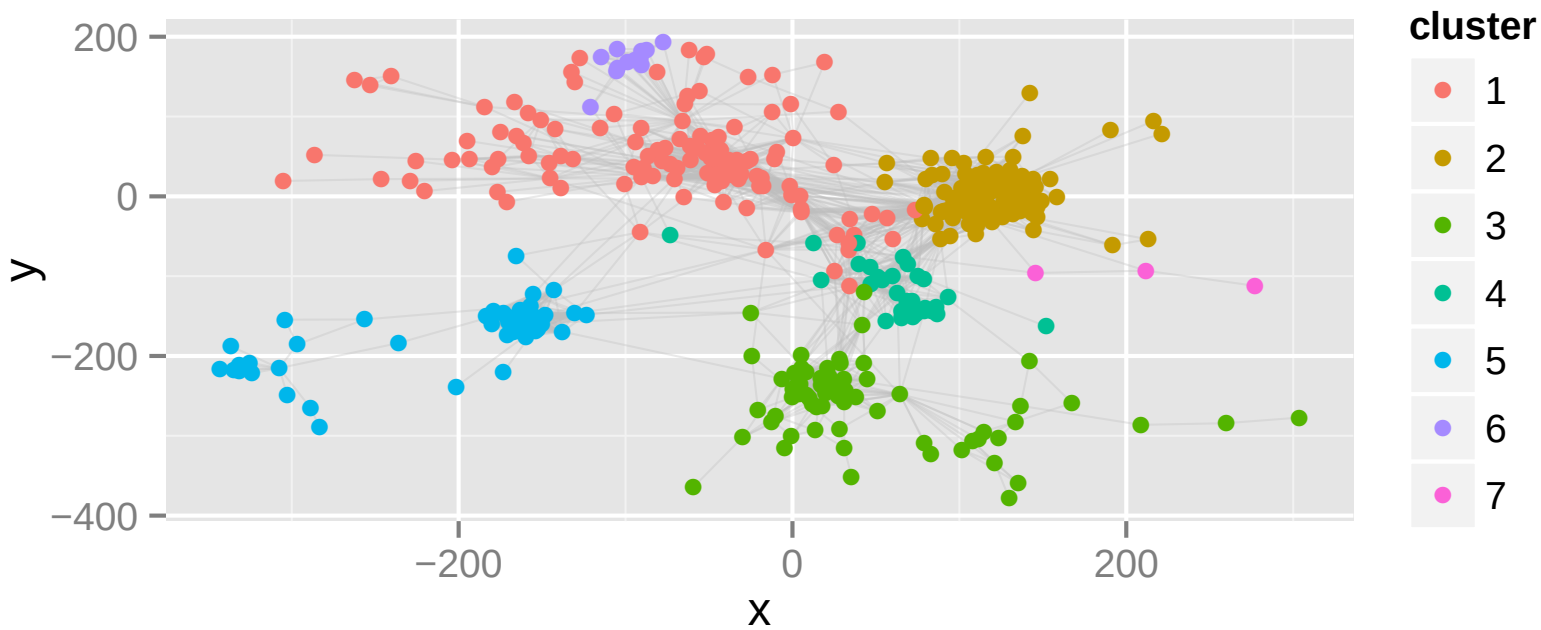
Example: visualizing your Facebook network

```
## now let's add the edges
edgelist <- get.edgelist(network, names=FALSE)
edges <- data.frame(d[edgelist[,1],c("x", "y")],
        d[edgelist[,2],c("x", "y")])
names(edges) <- c("x1", "y1", "x2", "y2")
(pq <- pq + geom_segment(
        aes(x=x1, y=y1, xend=x2, yend=y2),
        data=edges, size=0.25, color="grey", alpha=1/3))
```



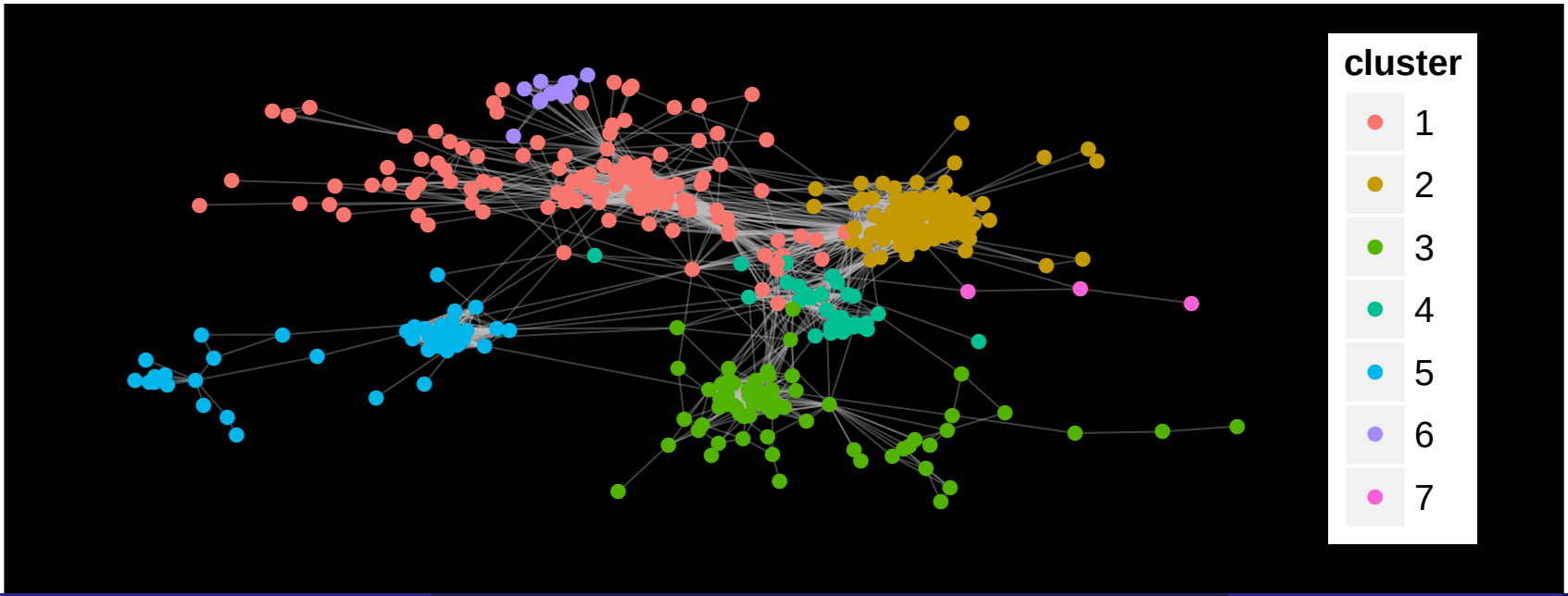
Example: visualizing your Facebook network

```
## note that the order matters!
p <- ggplot(d, aes(x=x, y=y, color=cluster))
(pq <- p + geom_segment(aes(x=x1, y=y1, xend=x2,
   yend=y2), data=edges, size=0.25, color="grey",
   alpha=1/3) + geom_point())
```



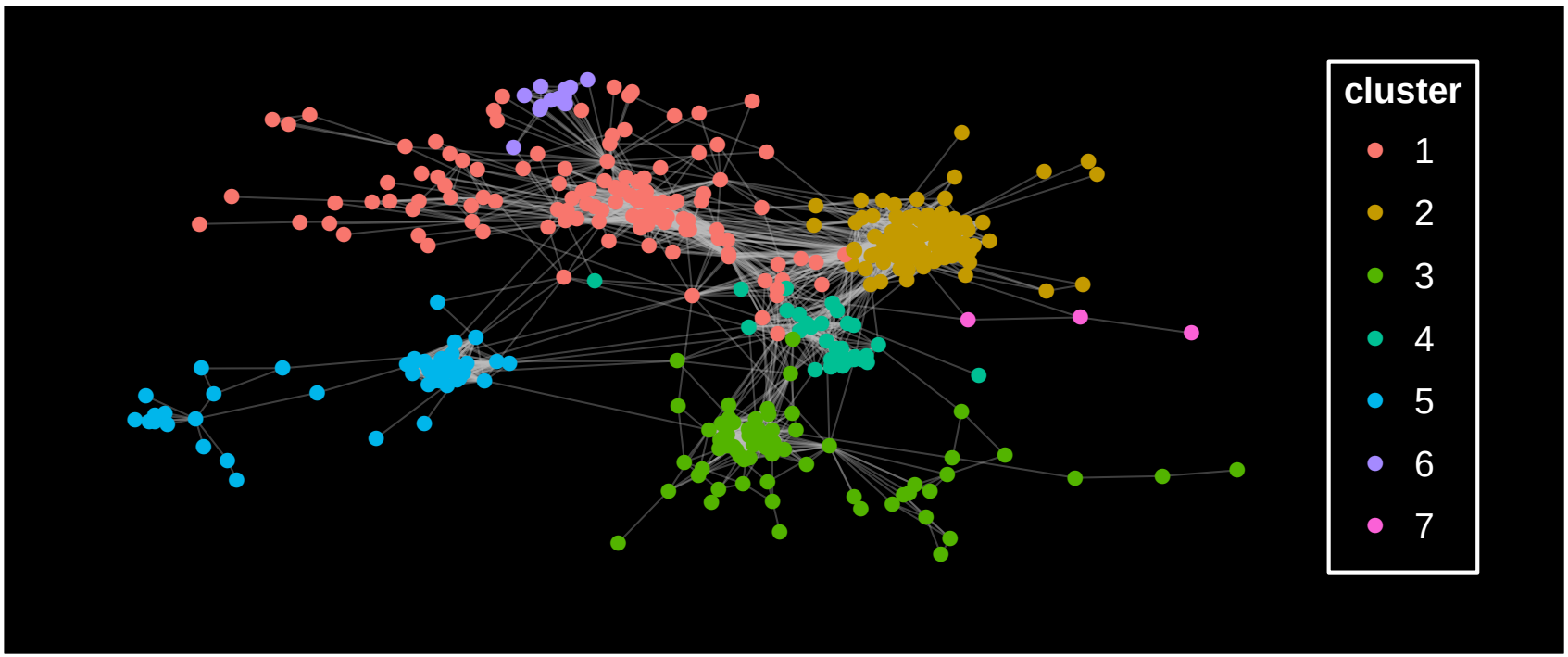
Example: visualizing your Facebook network

```
## change a few of the theme options to make it look better
(pq <- pq + theme(
    panel.background = element_rect(fill = "black"),
    plot.background = element_rect(fill="black"),
    axis.line = element_blank(), axis.text = element_blank(),
    axis.ticks = element_blank(), axis.title = element_blank(),
    panel.border = element_blank(), panel.grid.major =
      element_blank(), panel.grid.minor = element_blank()))
```



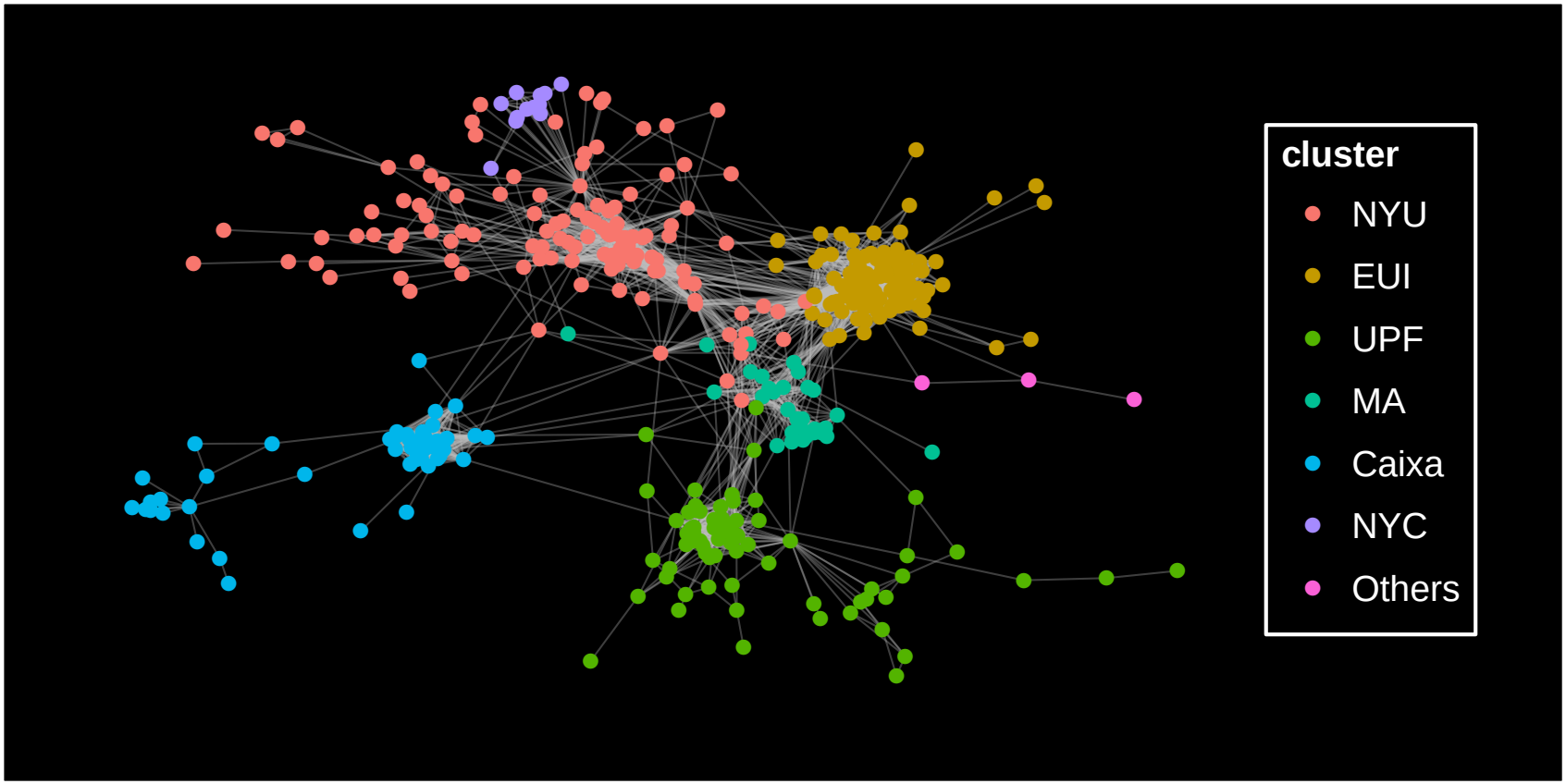
Example: visualizing your Facebook network

```
## now let's customize the legend
(pq <- pq + theme(legend.background =
      element_rect(colour = "white", fill = "black"),
    legend.key = element_rect(fill = "black", colour = F),
    legend.title = element_text(color="white"),
    legend.text = element_text(color="white")))
```



Example: visualizing your Facebook network

```
## after exploring who is in each cluster,
## labeling each of them by its category
labels <- c("NYU", "EUI", "UPF", "MA", "Caixa", "NYC", "Others")
(pq <- pq + scale_color_discrete(labels=labels))
```

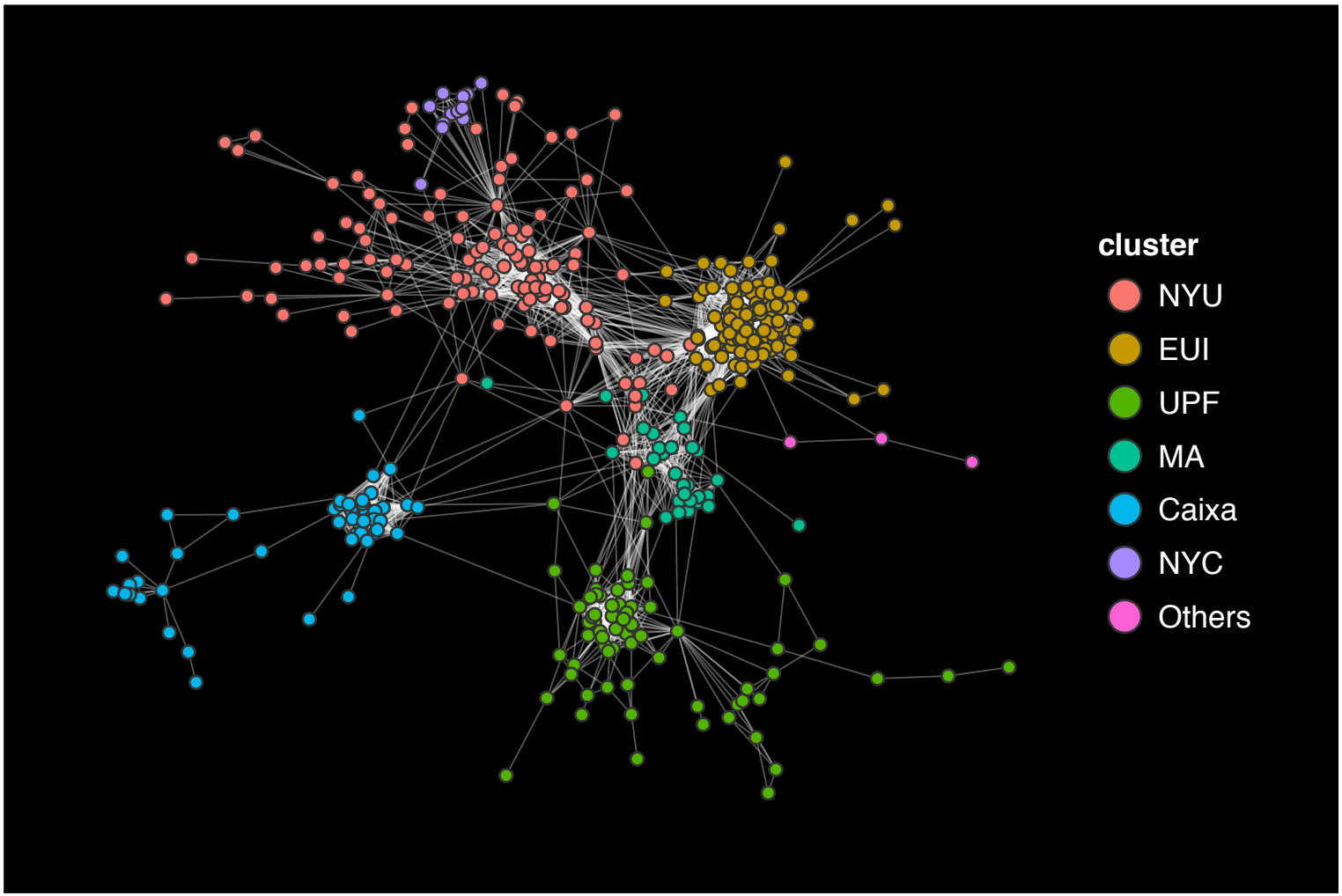


Example: visualizing your Facebook network

```
## putting it all together...
p <- ggplot(d, aes(x=x, y=y, color=cluster))
pq <- p + geom_segment(
        aes(x=x1, y=y1, xend=x2, yend=y2),
        data=edges, size=0.25, color="white", alpha=1/3) +
            ## note that here I add a border to the points
        geom_point(color="grey20", aes(fill=cluster), shape=21, size=2) +
        scale_fill_discrete(labels=labels) +
        theme(
            panel.background = element_rect(fill = "black"),
            plot.background = element_rect(fill="black"),
            axis.line = element_blank(), axis.text = element_blank(),
            axis.ticks = element_blank(),
            axis.title = element_blank(), panel.border = element_blank(),
            panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            legend.background = element_rect(colour = F, fill = "black"),
            legend.key = element_rect(fill = "black", colour = F),
            legend.title = element_text(color="white"),
            legend.text = element_text(color="white")
            ) +
    ## changing size of points in legend
    guides(fill = guide_legend(override.aes = list(size=5)))
```

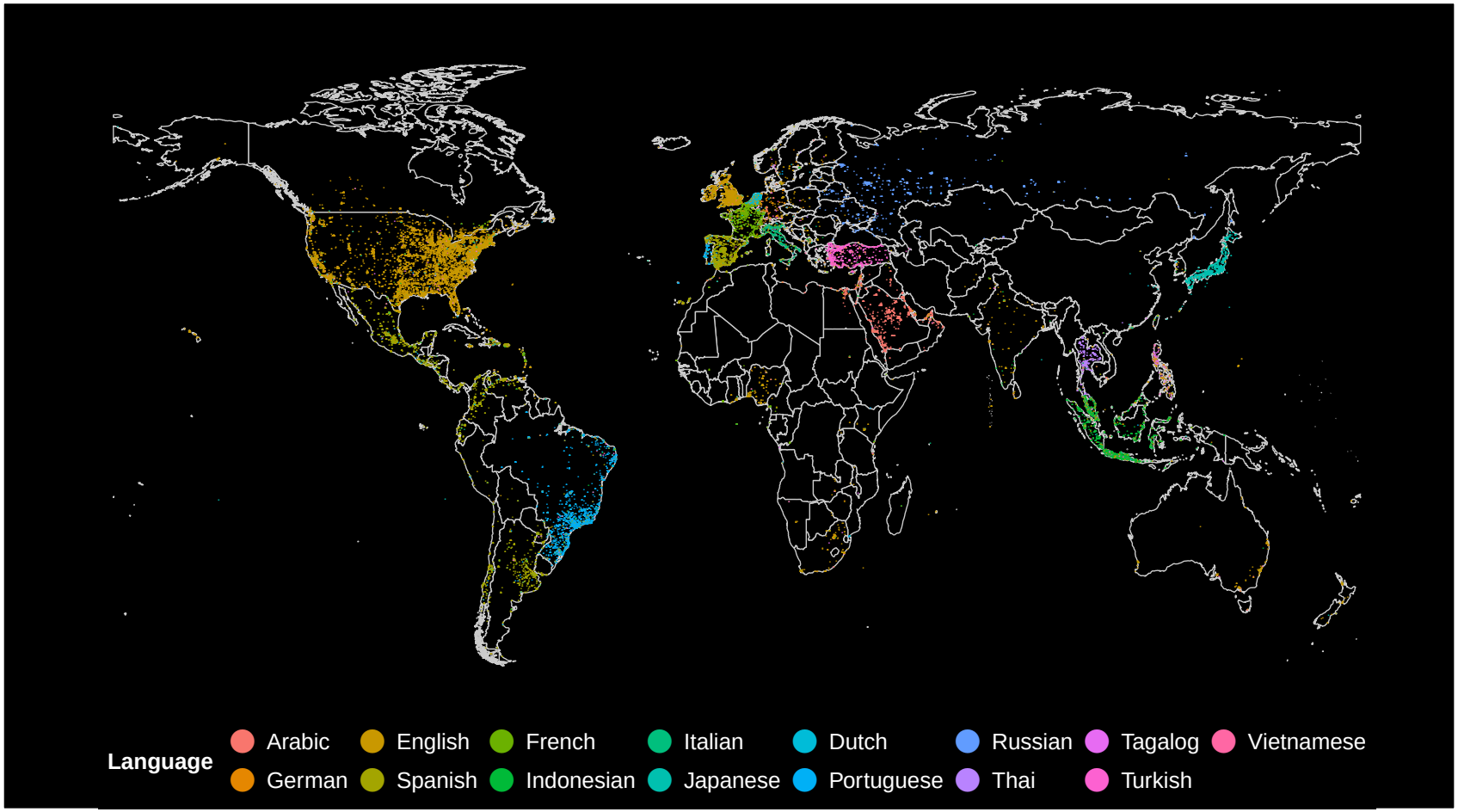
Example: visualizing your Facebook network

```
pq
```



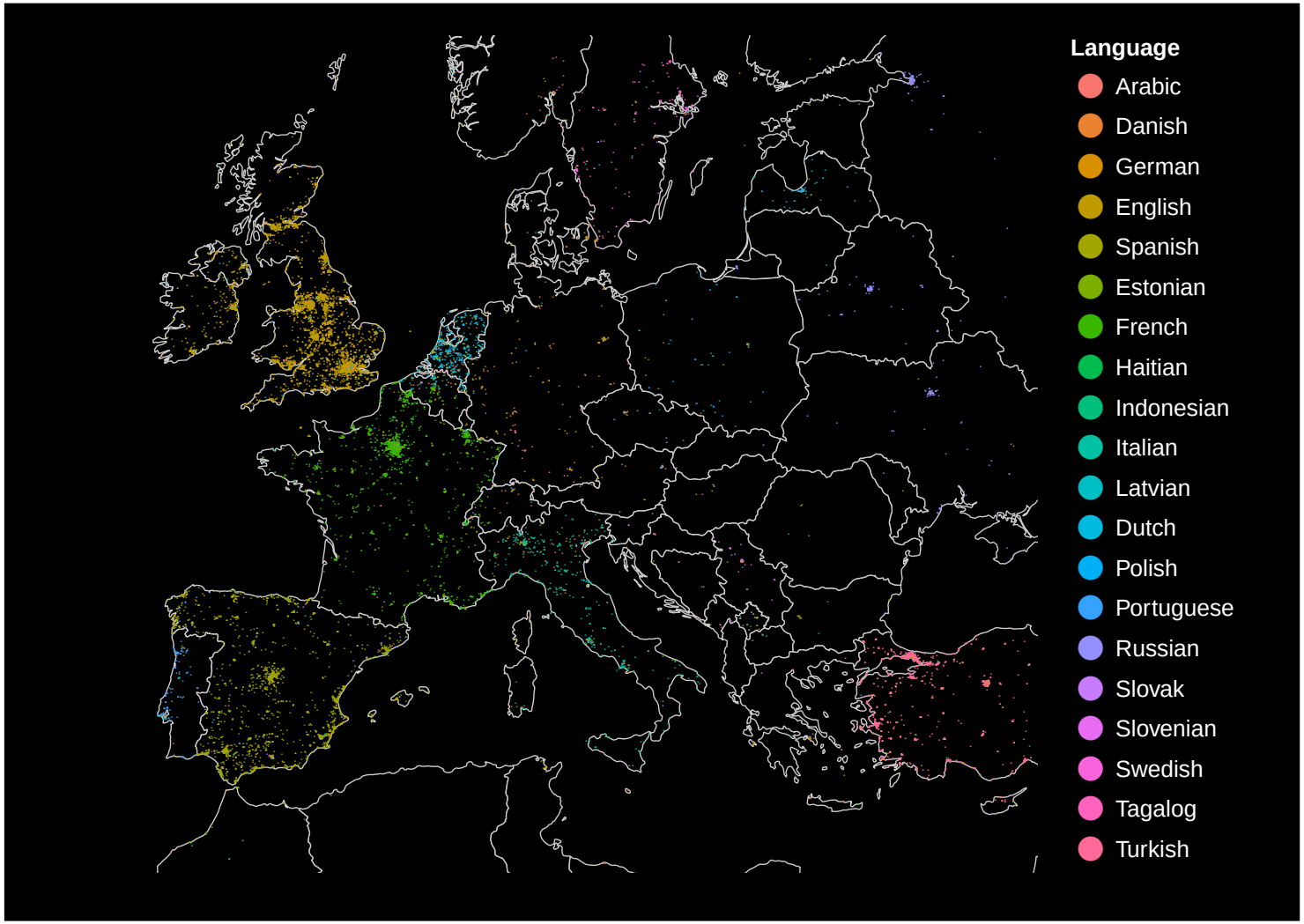
Geolocated tweets in top 15 languages

Code: `code/06_maps.R`. Data: all geolocated tweets, Sept. 15



Geolocated tweets in top 20 languages

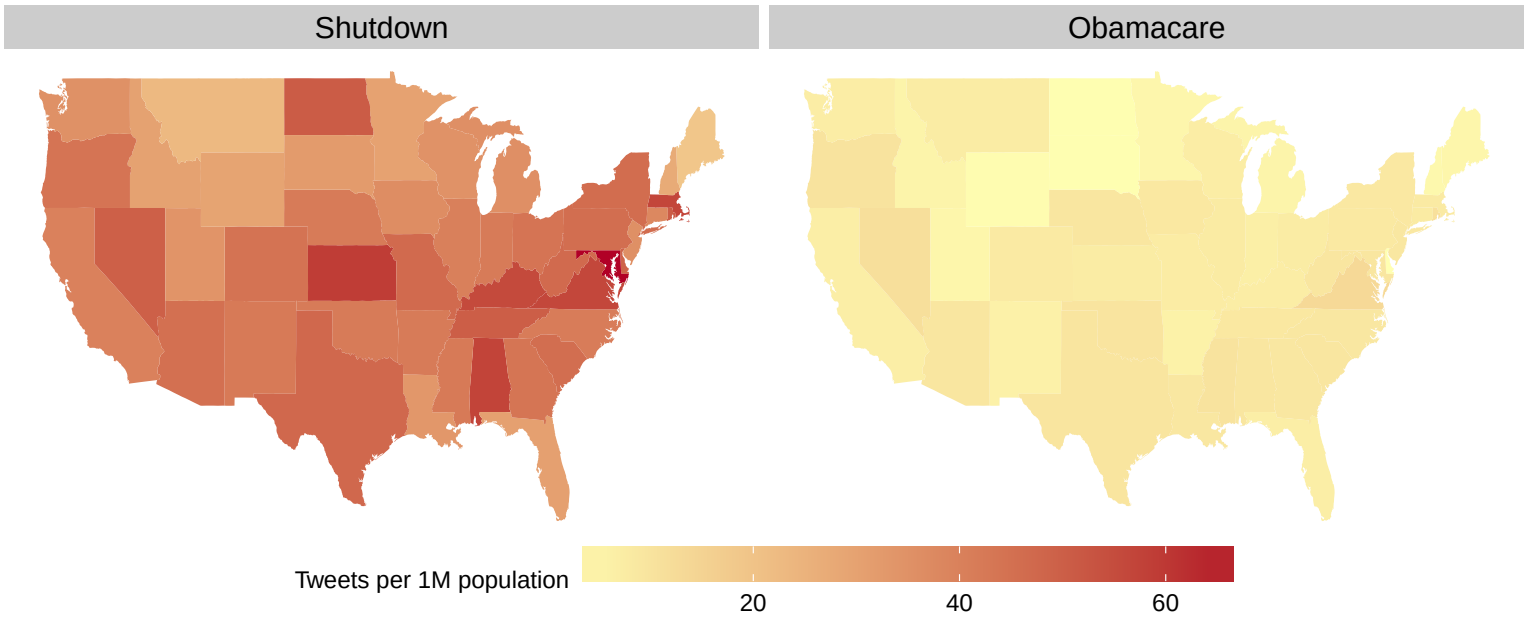
Code: `code/06_maps.R`. Data: all geolocated tweets, Sept. 15



Geolocated tweets about shutdown

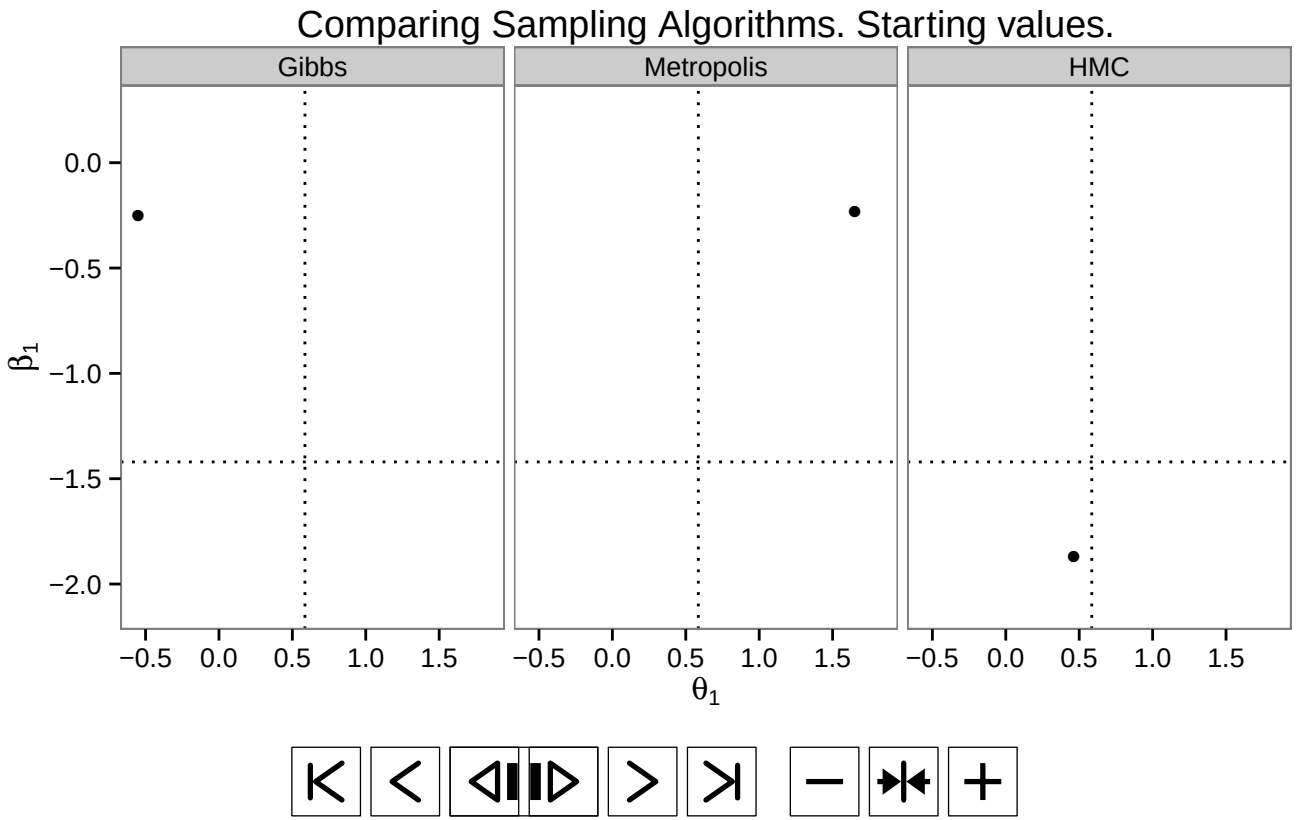
Code: `code/06_maps.R`.

Data: tweets mentioning 'shutdown' and 'obamacare', 10/01



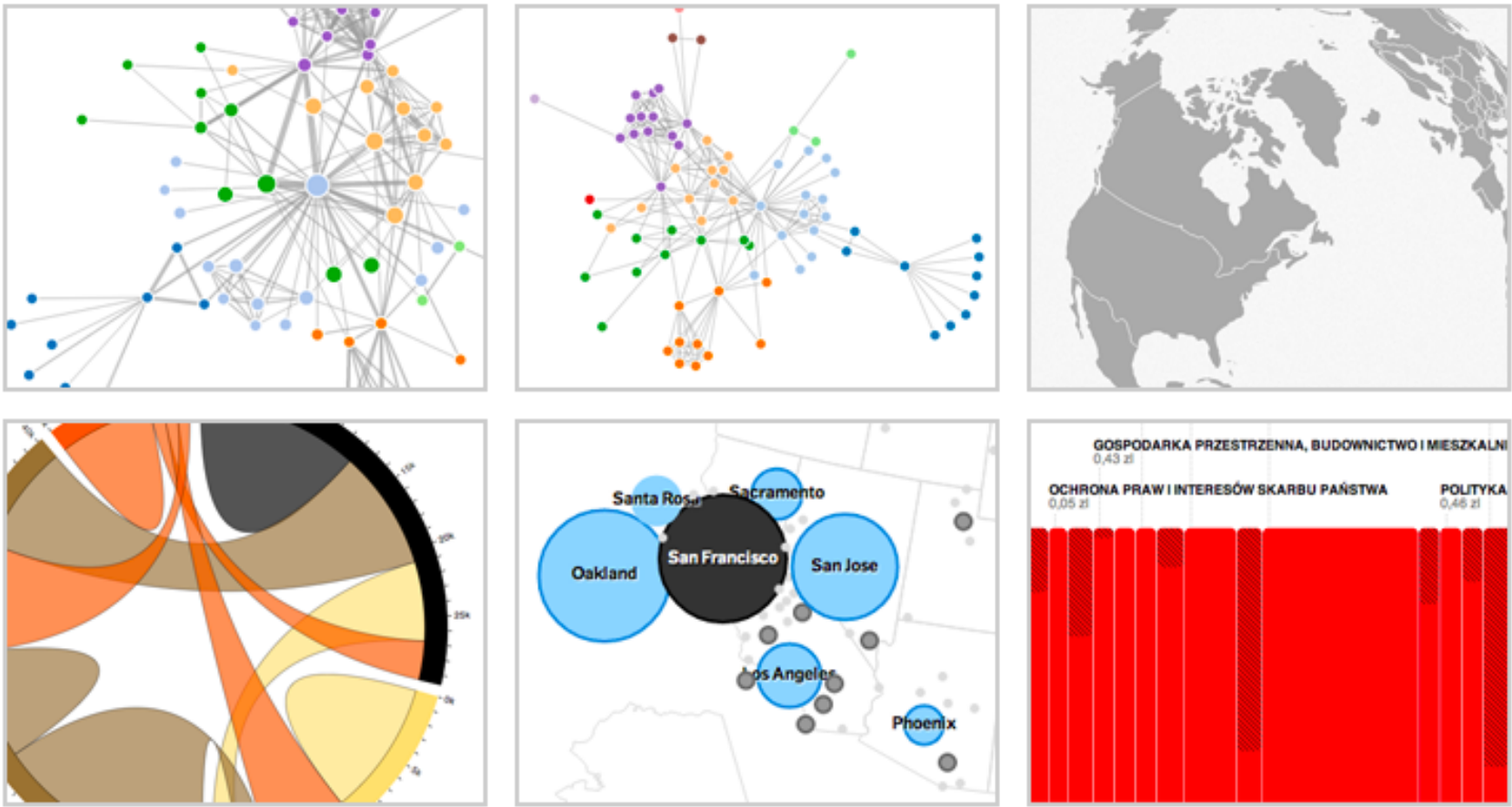
Comparison of sampling algorithms

Code: `07_animated_plots.R` and `sampling_algorithms.R`
Data: simulated item-response data (2-parameter model)



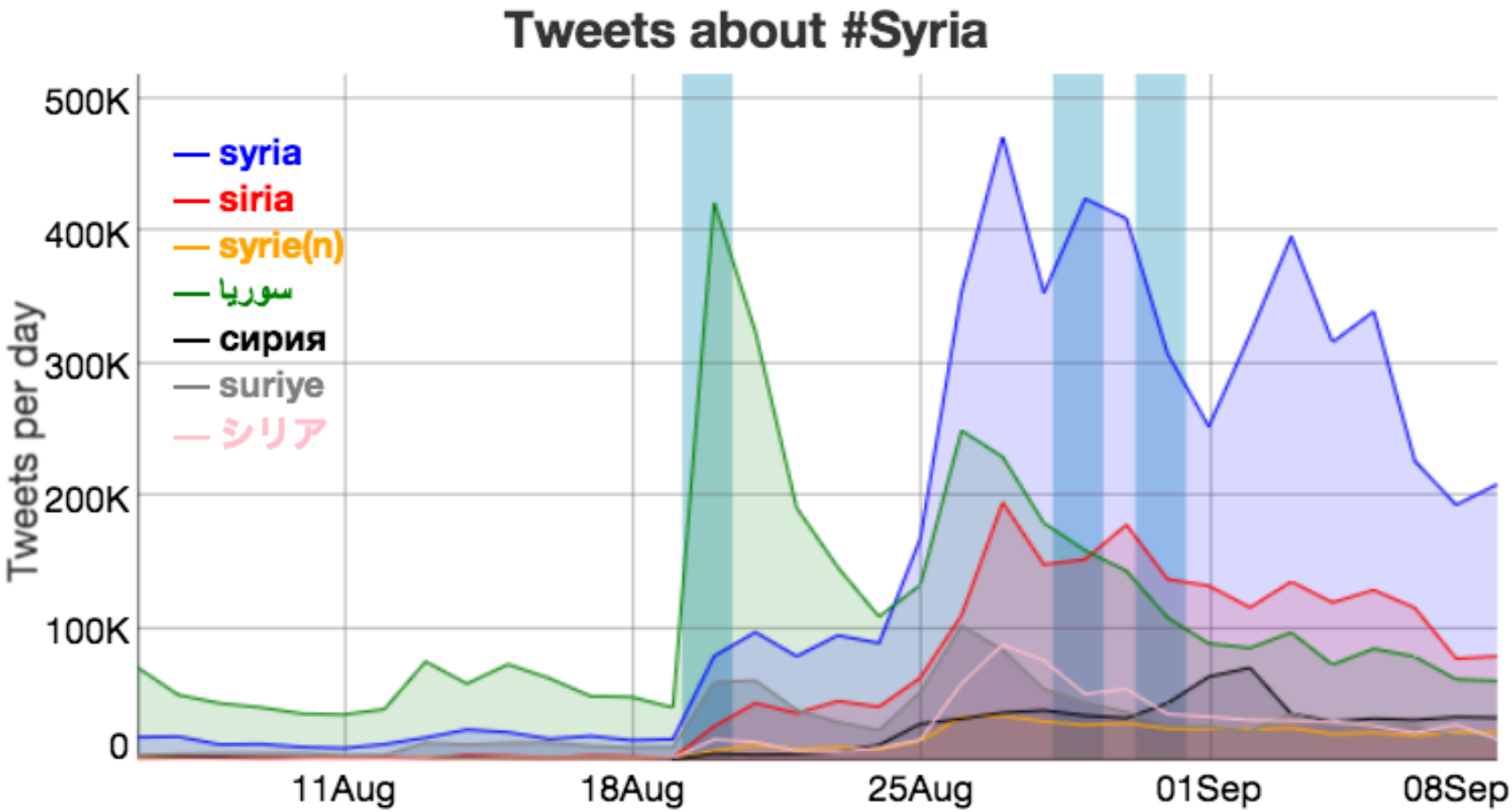
Samplers: Gibbs with data augmentation, Metropolis with uniform jumping distribution, and Hamiltonian Monte-Carlo

Beyond ggplot2



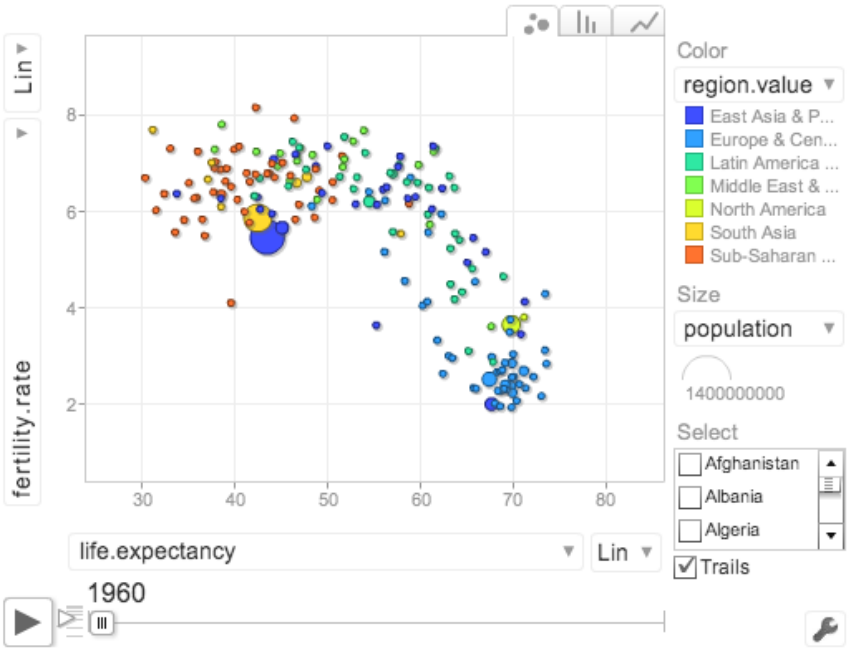
Data-Driven Documents (D3), in javascript

Beyond ggplot2

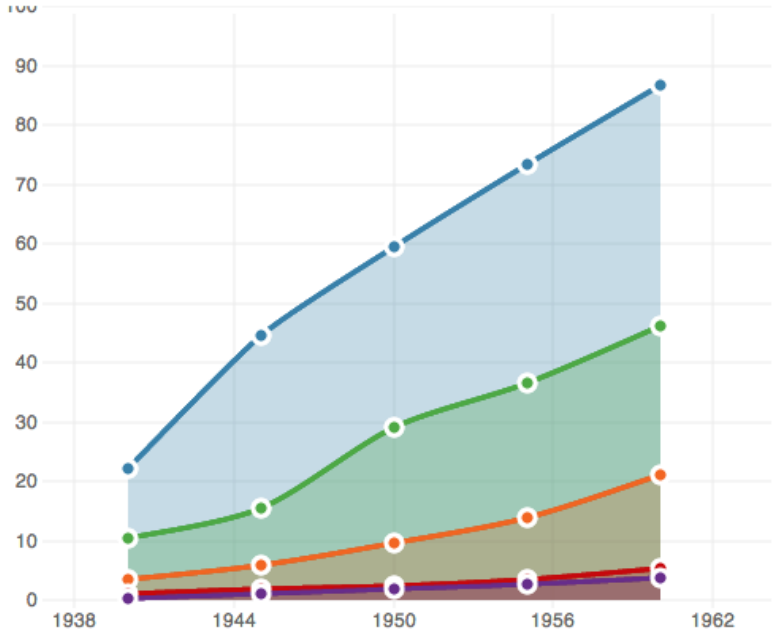


Dygraphs (javascript)

Beyond ggplot2



googleVis for R



rCharts

NYU Politics Data Lab Workshop: Data Visualization with R and ggplot2

Pablo Barberá

Department of Politics
New York University

email: *pablo.barbera@nyu.edu*
twitter: @p_barbera

October 15, 2013