Advanced Data Visualization in R

Iris Malone

November 6, 2015

Outline

What I'm Covering:

Outline

What I'm Covering:

- pros/cons different graphing packages

Outline

What I'm Covering:

- pros/cons different graphing packages
- ggplot and the grammar of graphics

Outline

What I'm Covering:

- pros/cons different graphing packages
- ggplot and the grammar of graphics
- how to visualize summary stats and regression results

Outline

What I'm Covering:

- pros/cons different graphing packages
- ggplot and the grammar of graphics
- how to visualize summary stats and regression results
- basic spatial visualization

Download code and slides at:

`web.stanford.edu/~imalone/VAM.html`

Choosing a Visualization Package

Toy Example: Plot number of democracies and autocracies over time.

Choosing a Visualization Package

Toy Example: Plot number of democracies and autocracies over time.

Options:

- plot (graphics)

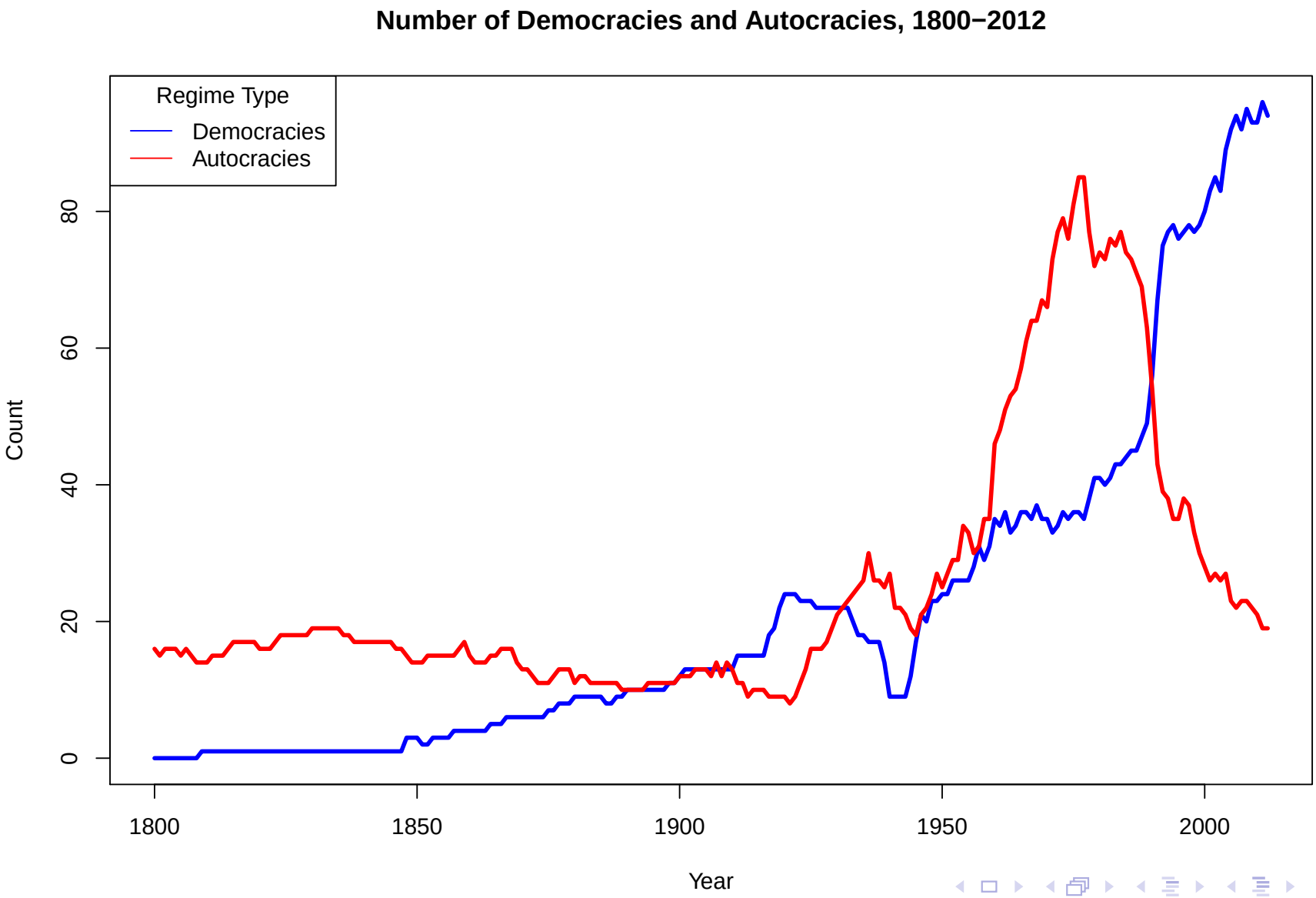
Option 1: Plot

Code:

```
plot(year, numdems, #x, y
  #aesthetics stuff (color, plot type, size)
    col = "blue", type = "l" , lwd=3,
  #main title and axes labels
    main = "Number of Democracies and Autocracies, 1800-2012",
    xlab = "Year",
    ylab = "Count")
lines(year, numauts, type = "l", col = "red", lwd =3)
legend("topleft", #location of legend
  legend=c("Democracies","Autocracies"), #legend labels/var
  col = c(4, 2), # colors for each,
  #legend colors default 1= black, 2 = red, 4=blue
  lty = c(1, 1),
  title="Regime Type") # Name of Legend
```

Option 1: Plot

Visual:



Choosing a Visualization Package

Toy Example.

Options:

- plot (graphics)

Choosing a Visualization Package

Toy Example.

Options:

- plot (graphics) Overly simplistic.

Choosing a Visualization Package

Toy Example.

Options:

- plot (graphics) Overly simplistic. Error interp.

Choosing a Visualization Package

Toy Example.

Options:

- plot (graphics) Overly simplistic. Error interp. Limited online support.

Choosing a Visualization Package

Toy Example.

Options:

- plot (graphics) Overly simplistic. Error interp. Limited online support.
- xyplot (lattice)

Option 2: xyplot

Step 1: Build First Layer for Number of Democracies.

```
library(lattice)
layer1 = xyplot(numdems ~ year, #format: y ~ x
    type = "l",
    # add title
    main = "Number of Democracies and Autocracies, 1800-2012",
    col = "blue", lwd=1,
    #legend
    key=list(space="right", # location
    # aesthetics (aes)
  lines=list(col=c("red","blue"), lty=c(1,1), lwd=2),
    #labels for each line
  text=list(c("Autocracies","Democracies"))))
```

Option 2: xyplot

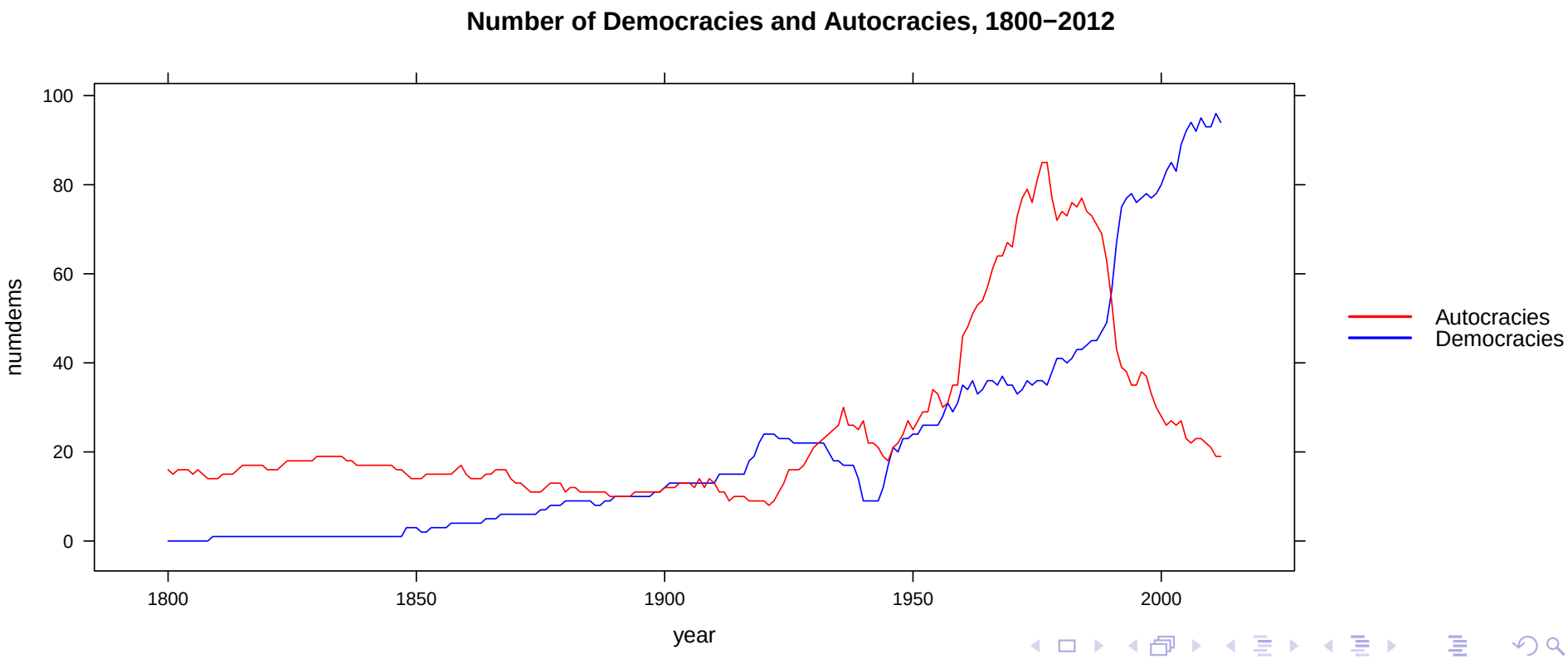
Step 2: Build Second Layer for Number of Autocracies.

```
layer2 = xyplot(numauts ~ year,
         type = "l",
         col = "red")
```

Option 2: xyplot

Step 3: Add the layers.

```
#need extra package to put layers on top
suppressMessages(library(latticeExtra))
layer1 + layer2
```



Graphics Packages

Options:

- plot (graphics) Overly simplistic. Error Interp. Limited online support.
- xyplot (lattice)

Graphics Packages

Options:

- plot (graphics) Overly simplistic. Error Interp. Limited online support.
- xyplot (lattice) Supplemental packages.

Graphics Packages

Options:

- plot (graphics) Overly simplistic. Error Interp. Limited online support.
- xyplot (lattice) Supplemental packages. Default look could be nicer.

Graphics Packages

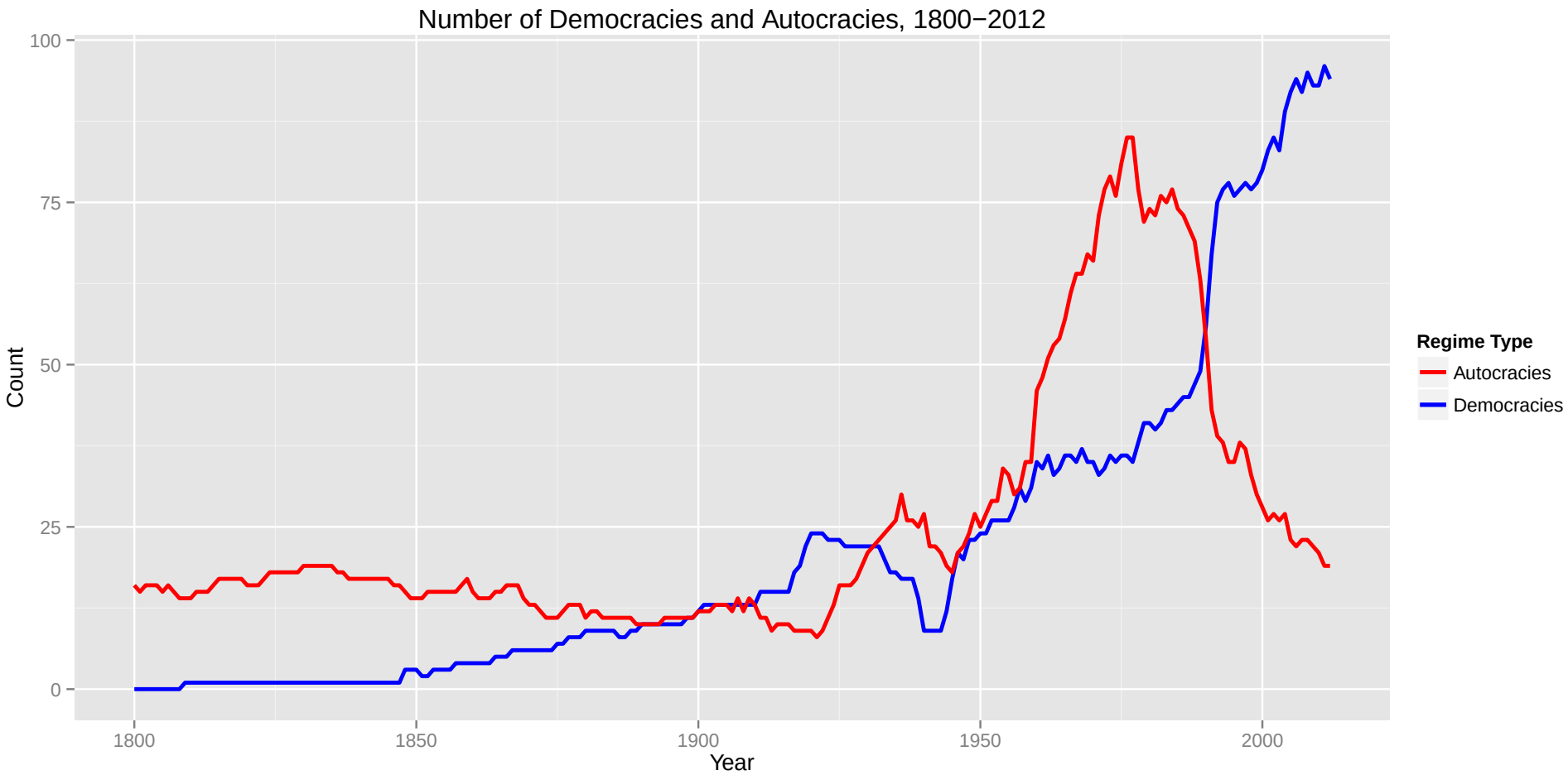
Options:

- plot (graphics) Overly simplistic. Error Interp. Limited online support.
- xyplot (lattice) Supplemental packages. Default look could be nicer.
- ggplot (ggplot2)

Option 3: ggplot

```
suppressMessages(library(ggplot2))
ggplot(data = NULL) +
  #line for dems and auts
 geom_line(aes(x=year, y = numdems, colour = "numdems")) +
 geom_line(aes(x=year, y = numauts, colour = "numauts")) +
 xlab("Year") + ylab("Count") +
 ggtitle("Number of Democracies and Autocracies, 1800-2012") +
 # legend aesthetics
 scale_color_manual(name = "Regime Type", # Name
 labels = c(numdems="Democracies", numauts = "Autocracies"), #
    values=c(numdems=4,numauts=2))
```

Option 3: ggplot



Summary of Options:

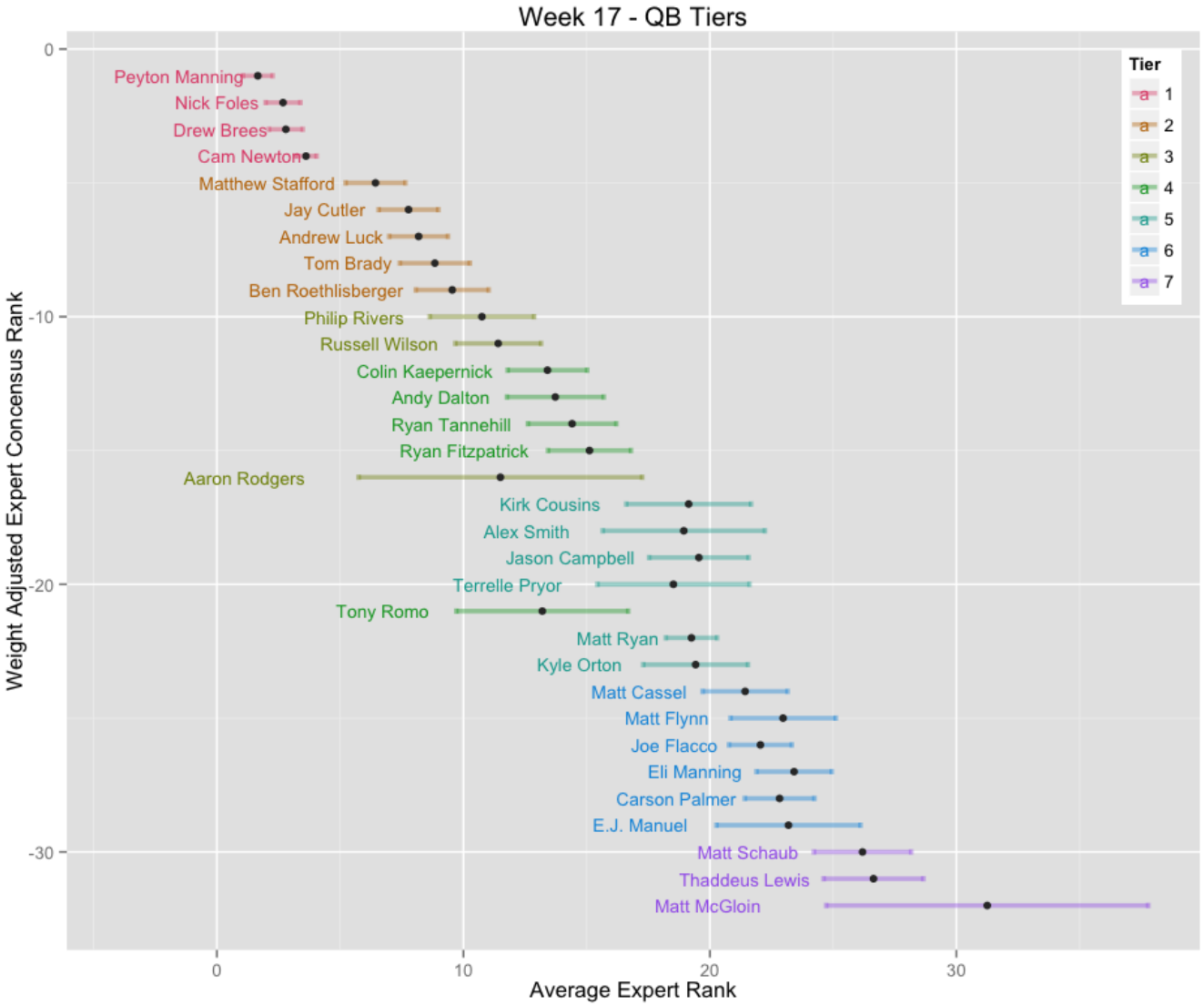
- plot (graphics)
- xyplot (lattice)
- ggplot (ggplot2)

Why ggplot?

- Used professionally

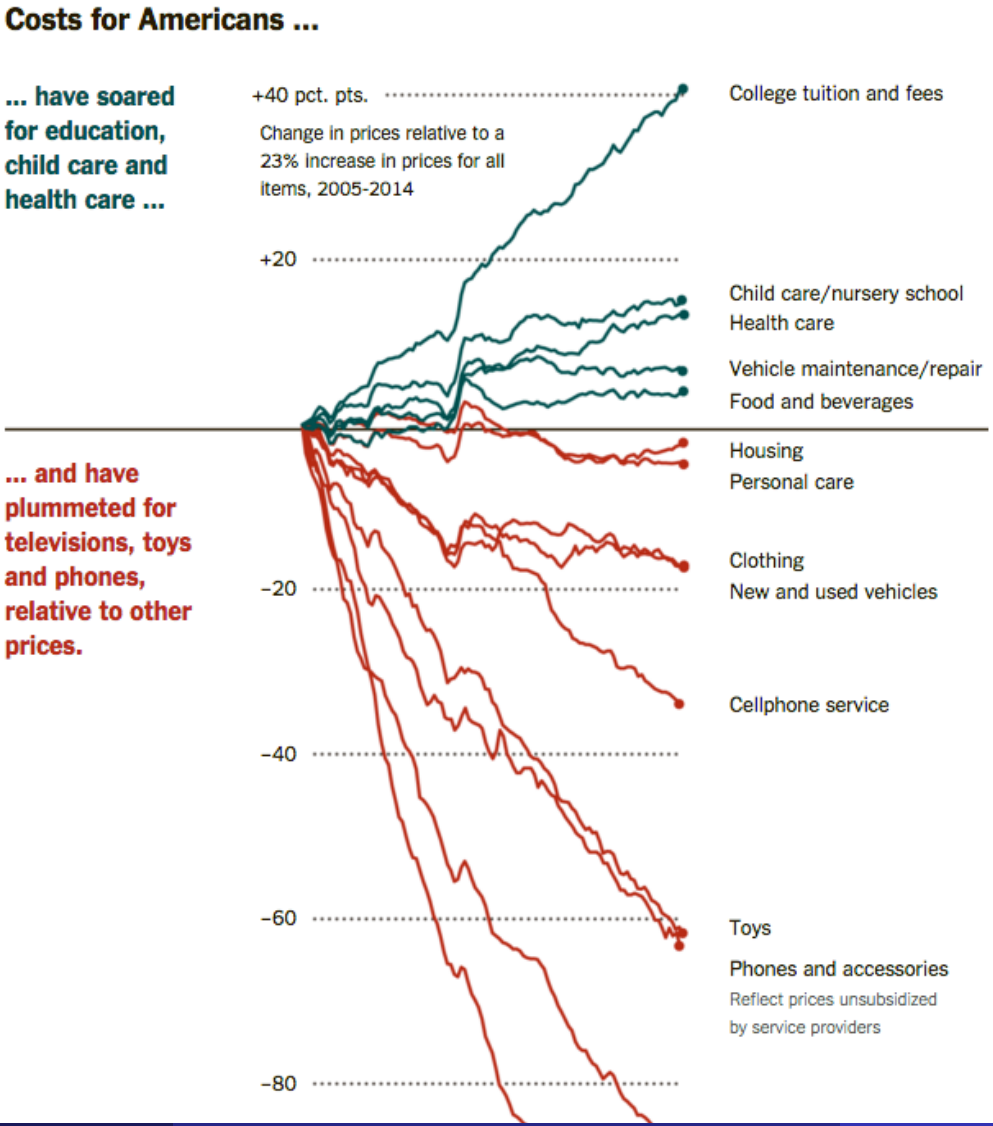
Why ggplot?

Example 1



Why ggplot?

Example 2



Why ggplot?

- Used professionally

Why ggplot?

- Used professionally
- Very pretty

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate
- Great support online

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate
- Great support online
- Knowledge transfers to other packages/languages (ggvis, Shiny, Python)

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate
- Great support online
- Knowledge transfers to other packages/languages (ggvis, Shiny, Python)
- Steep Learning Curve

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate
- Great support online
- Knowledge transfers to other packages/languages (ggvis, Shiny, Python)
- Steep Learning Curve
- Lots of syntax

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate
- Great support online
- Knowledge transfers to other packages/languages (ggvis, Shiny, Python)
- Steep Learning Curve
- Lots of syntax
- Can be slow

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate
- Great support online
- Knowledge transfers to other packages/languages (ggvis, Shiny, Python)
- Steep Learning Curve
- Lots of syntax
- Can be slow
- Defaults to weird colors

Why ggplot?

- Used professionally
- Very pretty
- Easy to manipulate
- Great support online
- Knowledge transfers to other packages/languages (ggvis, Shiny, Python)
- Steep Learning Curve
- Lots of syntax
- Can be slow
- Defaults to weird colors
- **Summary: Worth it.**

ggplot Syntax

Following next few slides adapated from Samantha Tyner

- Based on Grammar of Graphics book by Leland Wilkinson hence 'gg'

ggplot Syntax

Following next few slides adapated from Samantha Tyner

- Based on Grammar of Graphics book by Leland Wilkinson hence 'gg'
- New Zealander Hadley Wickham $\rightarrow$ R

ggplot Syntax

Following next few slides adapated from Samantha Tyner

- Based on Grammar of Graphics book by Leland Wilkinson hence 'gg'
- New Zealander Hadley Wickham $\rightarrow$ R
- Analogy: Think of parts of a plot like parts of a sentence

ggplot Syntax

Following next few slides adapated from Samantha Tyner

- Based on Grammar of Graphics book by Leland Wilkinson hence 'gg'
- New Zealander Hadley Wickham $\rightarrow$ R
- Analogy: Think of parts of a plot like parts of a sentence
- Warning: qplot

ggplot Syntax

- Noun $\rightarrow$ Data

```
ggplot(data = df)
```

ggplot Syntax

- Noun → Data

```
ggplot(data = df)
```

- Verb → "geom_" + Plot Type

```
ggplot(data = df) + geom_bar()
```

Table 1: Geom Types

abline	area	boxplot
errorbar	histogram	line
point	ribbon	smooth
blank	density	jitter
polygon	quantile	vline

ggplot Syntax

- Noun → Data

```
ggplot(data = df)
```

- Verb → "geom_" + Plot Type

```
ggplot(data = df) + geom_bar()
```

- Adjectives → Aesthetics ("aes") (x, y, fill, colour, linetype)

```
ggplot(data = df, aes(x=categorical.var, fill=group.var)) +
  geom_bar()
```

ggplot Syntax

Adjectives → Aesthetics ("aes")

	colour	size	linetype	alpha	fill	weight	shape	width	height	angle	hjust	vjust	fa
abline	black	0.5	1	yes	--	--	--	--	--	--	--	--	
area	yes	0.5	1	yes	grey20	--	--	--	--	--	--	--	
bar	yes	0.5	1	yes	grey20	1	--	--	--	--	--	--	
bin2d	yes	0.5	1	yes	grey60	1	--	--	--	--	--	--	
boxplot	grey20	0.5	solid	yes	white	1	16	--	--	--	--	--	
contour	#3366FF	0.5	1	yes	--	1	--	--	--	--	--	--	
crossbar	black	0.5	1	yes	yes	--	--	--	--	--	--	--	
density	black	0.5	1	yes	yes	1	--	--	--	--	--	--	
density2d	#3366FF	0.5	1	yes	--	1	--	--	--	--	--	--	
errorbar	black	0.5	1	yes	--	--	--	0.5	--	--	--	--	
errorbarh	black	0.5	1	yes	--	--	--	--	0.5	--	--	--	
freqpoly	black	0.5	1	yes	--	--	--	--	--	--	--	--	
hex	yes	0.5	--	yes	grey50	--	--	--	--	--	--	--	
hline	black	0.5	1	yes	--	--	--	--	--	--	--	--	
linerange	black	0.5	1	yes	--	--	--	--	--	--	--	--	
path	black	0.5	1	yes	--	--	--	--	--	--	--	--	
point	black	2	--	yes	yes	--	16	--	--	--	--	--	
pointrange	black	0.5	1	yes	yes	--	16	--	--	--	--	--	
polygon	NA	0.5	1	yes	grey20	--	--	--	--	--	--	--	
quantile	#3366FF	0.5	1	yes	--	1	--	--	--	--	--	--	
raster	--	--	--	yes	grey20	--	--	--	--	--	--	--	
rect	yes	0.5	1	yes	grey20	--	--	--	--	--	--	--	
ribbon	yes	0.5	1	yes	grey20	--	--	--	--	--	--	--	
rug	black	0.5	1	yes	--	--	--	--	--	--	--	--	
segment	black	0.5	1	yes	--	--	--	--	--	--	--	--	
smooth	#3366FF	0.5	1	0.4	grey60	1	--	--	--	--	--	--	
step	black	0.5	1	yes	--	--	--	--	--	--	--	--	
text	black	5	--	yes	--	--	--	--	--	0	0.5	0.5	
tile	yes	0.1	1	yes	grey20	--	--	--	--	--	--	--	
violin	grey20	0.5	solid	yes	white	1	--	--	--	--	--	--	
vline	black	0.5	1	yes	--	--	--	--	--	--	--	--	

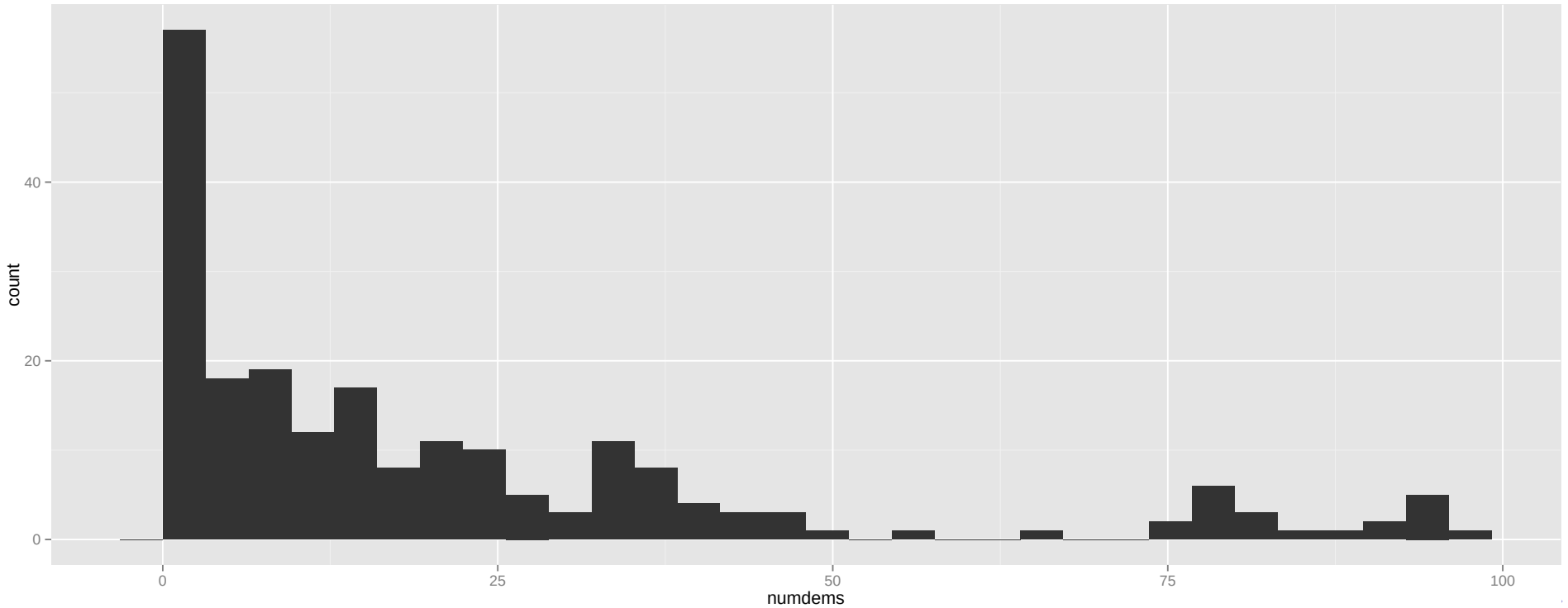
ggplot Syntax. Aesthetics Sidebar.

Note. Difference between fill, colour, and placement.

Default.

```
ggplot(data = NULL, aes(x=numdems)) + geom_bar()
```

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

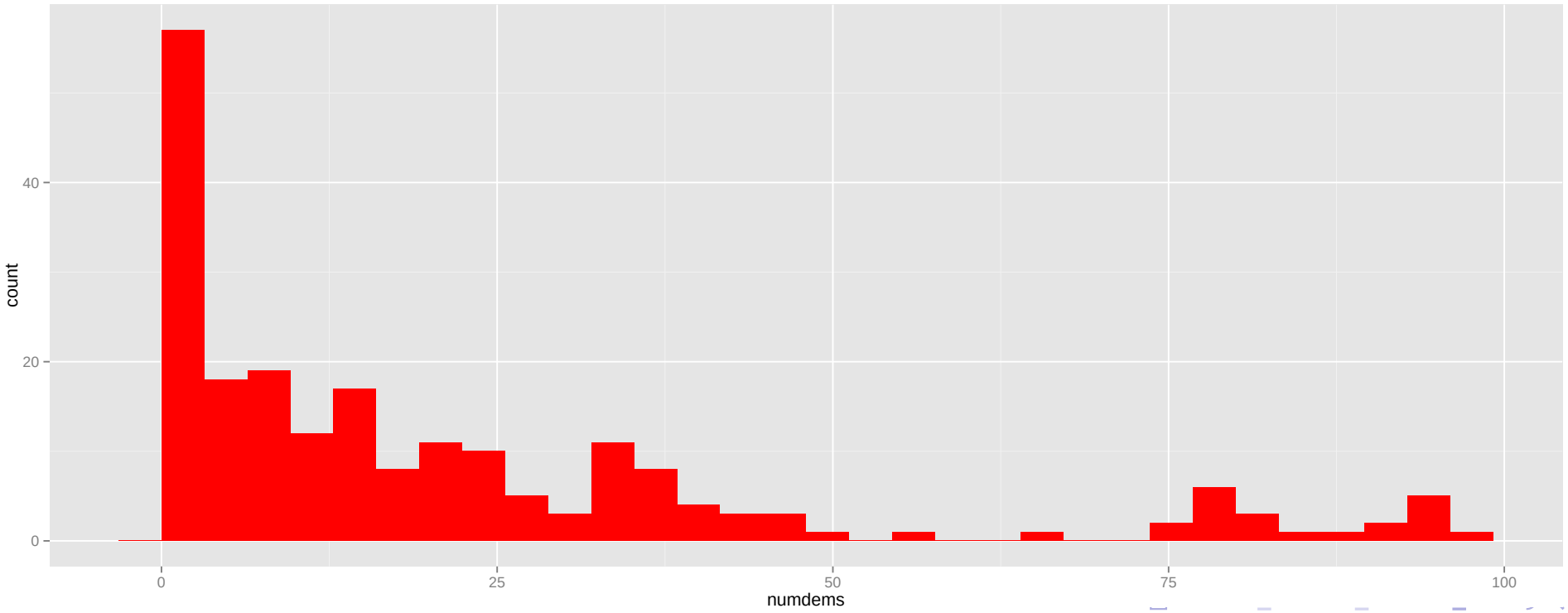


ggplot Syntax. Aesthetics Sidebar.

Fill.

```
ggplot(data = NULL, aes(x=numdems)) +
  geom_bar(fill="red")
```

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

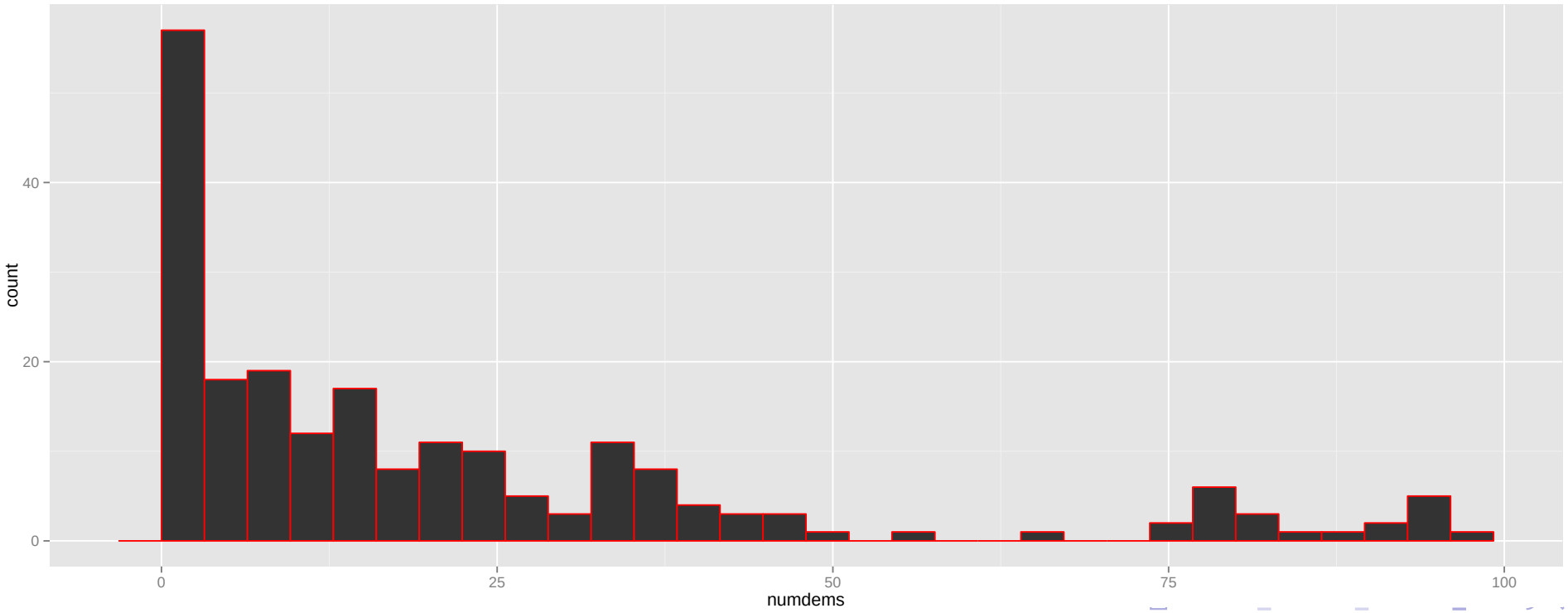


ggplot Syntax. Aesthetics Sidebar.

Colour.

```
ggplot(data = NULL, aes(x=numdems)) +
  geom_bar(colour="red")
```

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

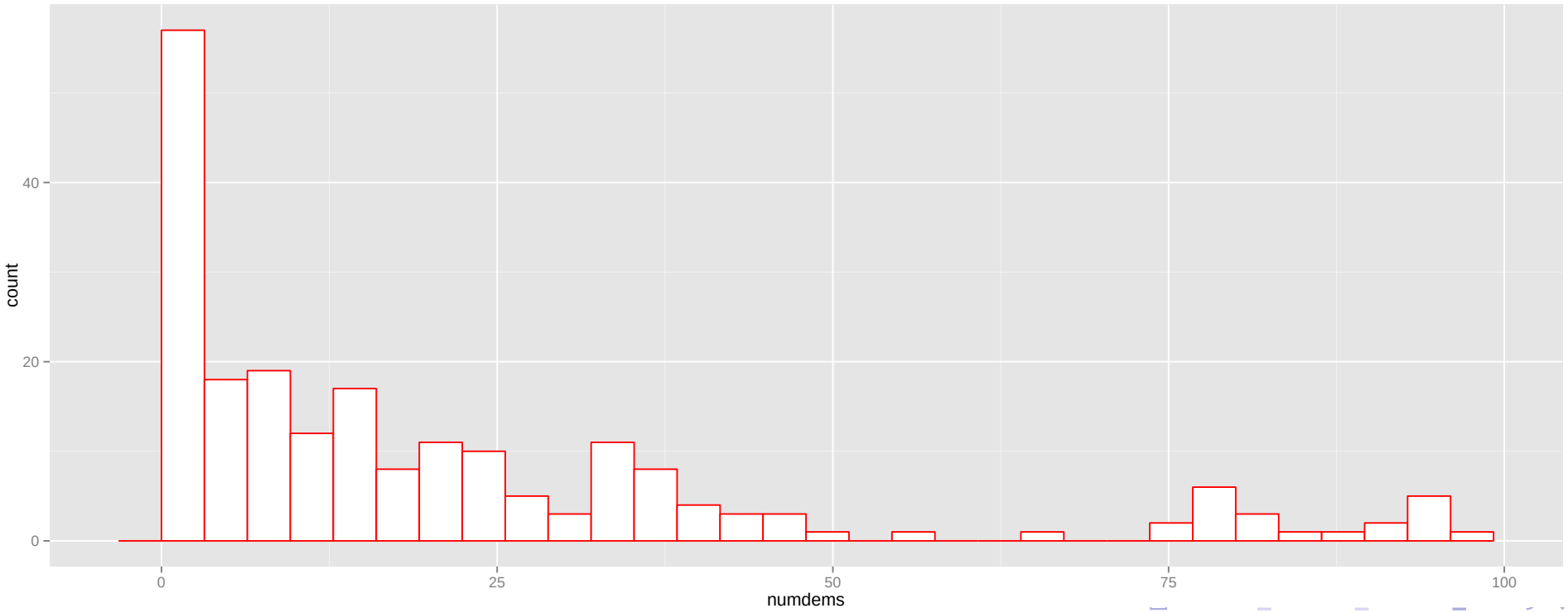


ggplot Syntax. Aesthetics Sidebar.

Fill and Colour.

```
ggplot(data = NULL, aes(x=numdems)) +
  geom_bar(fill="white", colour="red")
```

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

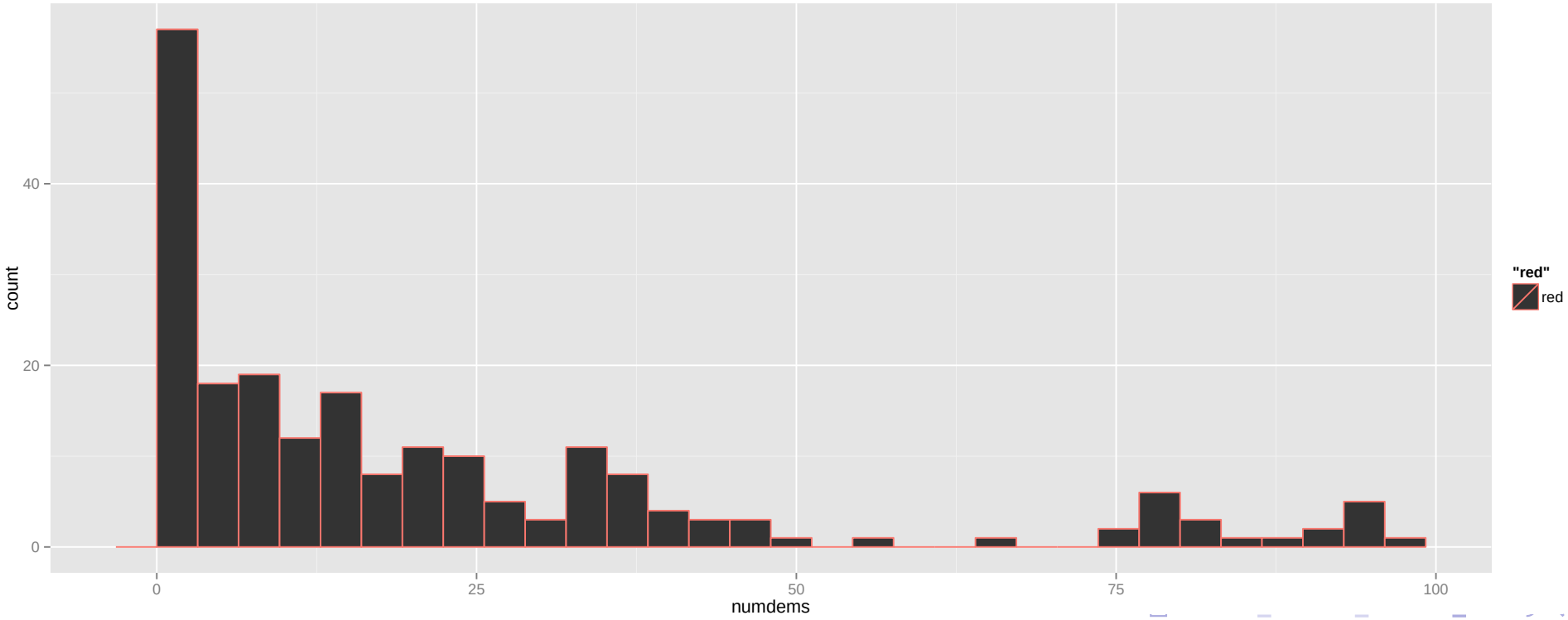


ggplot Syntax. Aesthetics Sidebar

Defined inside the aesthetics argument. Ack!

```
ggplot(data = NULL) +
  geom_bar(aes(x=numdems, colour ="red"))
```

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

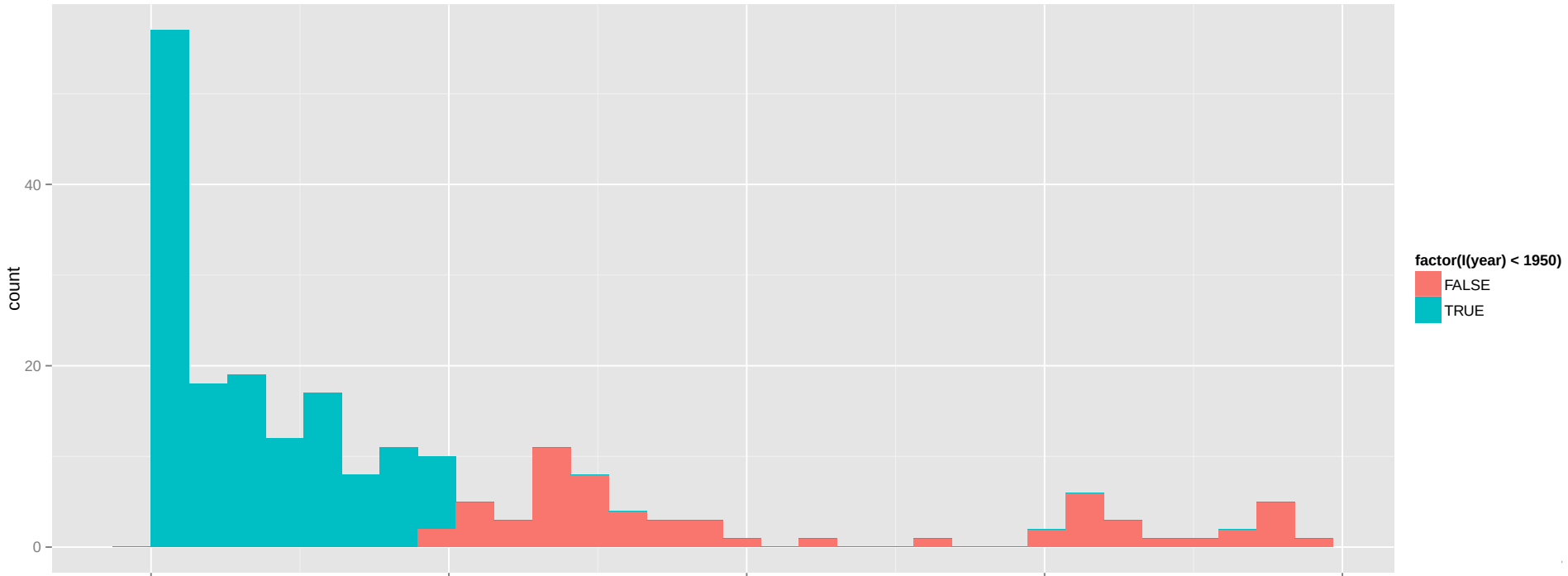


ggplot Syntax. Aesthetics Sidebar

Defined inside the aesthetics argument.

```
ggplot(data = NULL) +
  geom_bar(aes(x=numdems,
  fill=factor(I(year)<1950)))
```

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



ggplot Syntax

- Adverb $\rightarrow$ stat (e.g. identity, bin)

```
ggplot(data = df, aes(x=categorical.var, fill=group.var))
  + geom_bar(stat = "bin")
```

ggplot Syntax

- Adverb → stat (e.g. identity, bin)

```
ggplot(data = df, aes(x=categorical.var, fill=group.var))
  + geom_bar(stat = "bin")
```

- Preposition → position (e.g. fill, dodge, identity)

```
ggplot(data = df, aes(x=categorical.var, fill=group.var)) +
  geom_bar(stat="bin", position = "identity", binwidth=5)
```

Toy Example for Learning Syntax.

Fearon and Laitin (2003). Let's suppose we would like to get a feel for their data by first just looking at the number of civil wars in their dataset as a function of two variables: (1) region and (2) time.

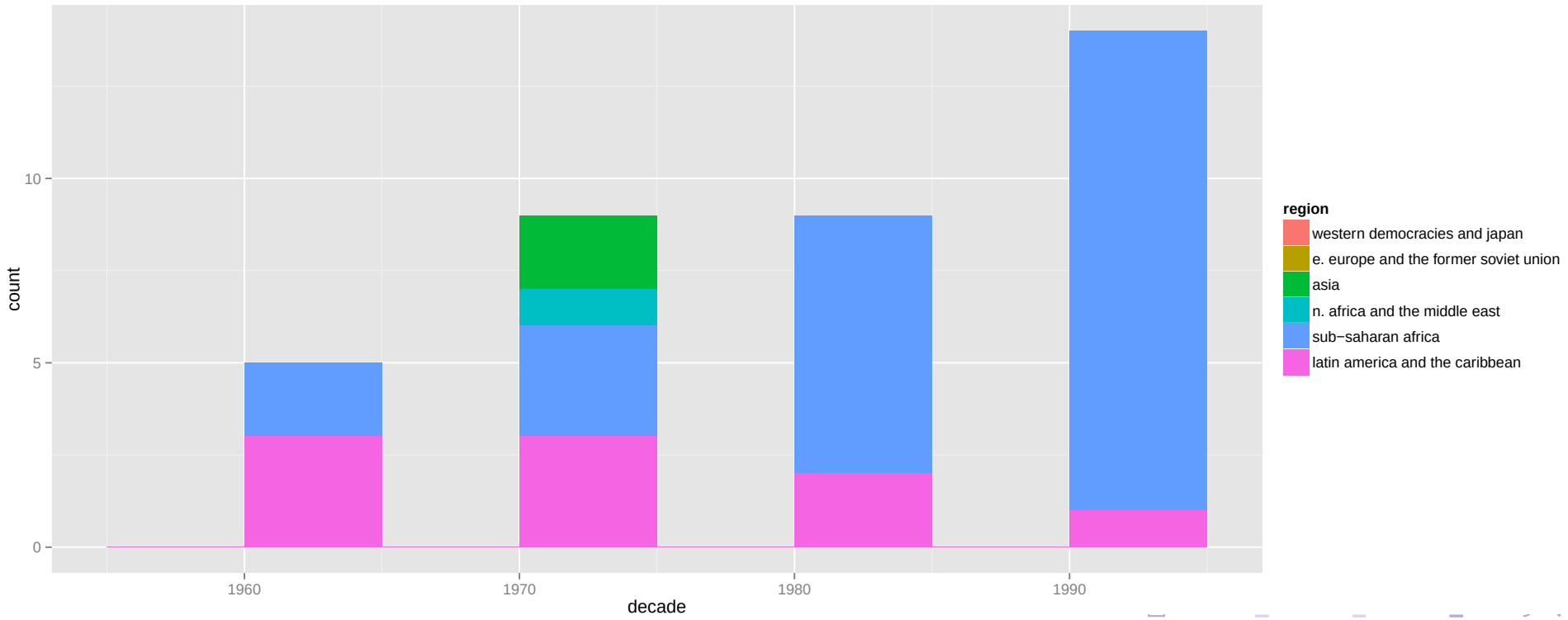
```
library(foreign)
df = read.dta("repdata.dta")
#subset data so 1 obs/civil war
dfonset = subset(df, df$onset == 1)
```

D.V. Civil War Onset

I.V. Ethnic Fractionalization, Mountainous Terrain, Oil, New State, Others

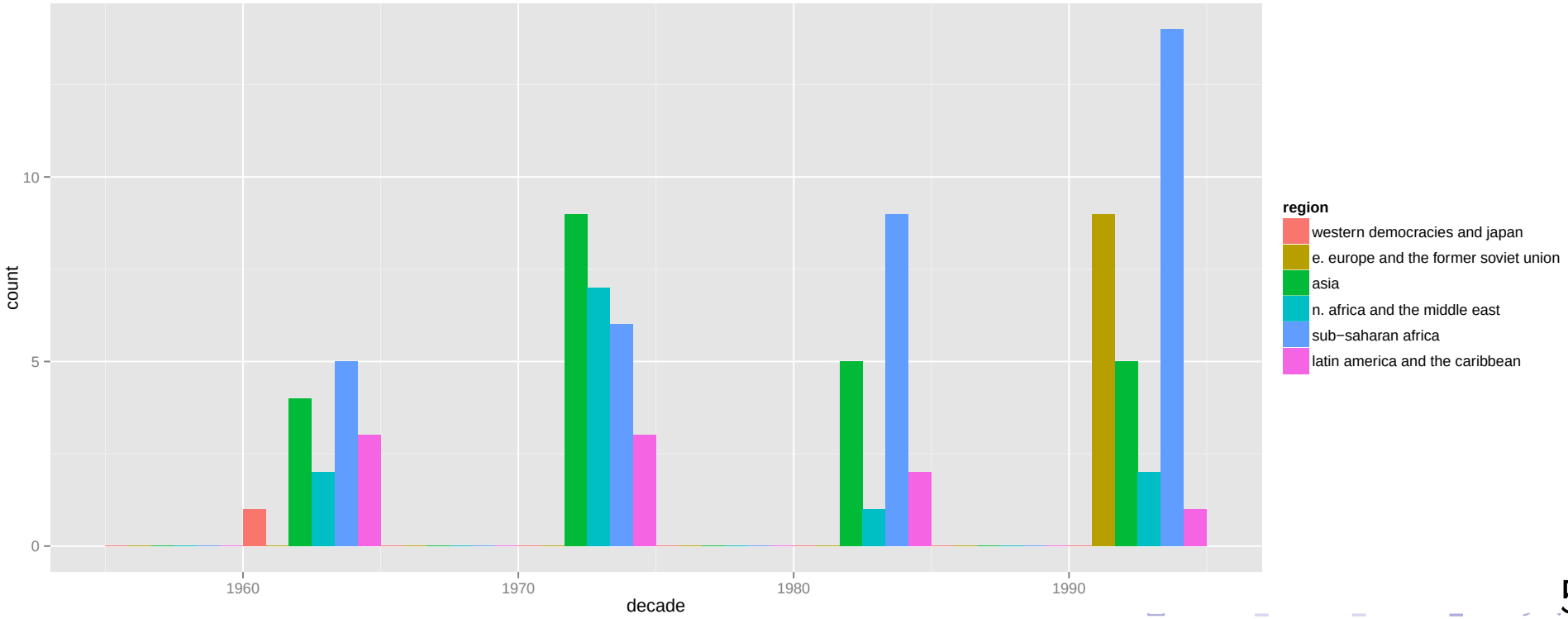
ggplot Syntax. Example. Civil War by Region and Decade

```
p = ggplot(data = dfonset, aes(x=decade,
                               fill = region)) +
  geom_bar(position = "identity",
           binwidth=5)
p
```



ggplot Syntax. Example. Civil War by Region and Decade

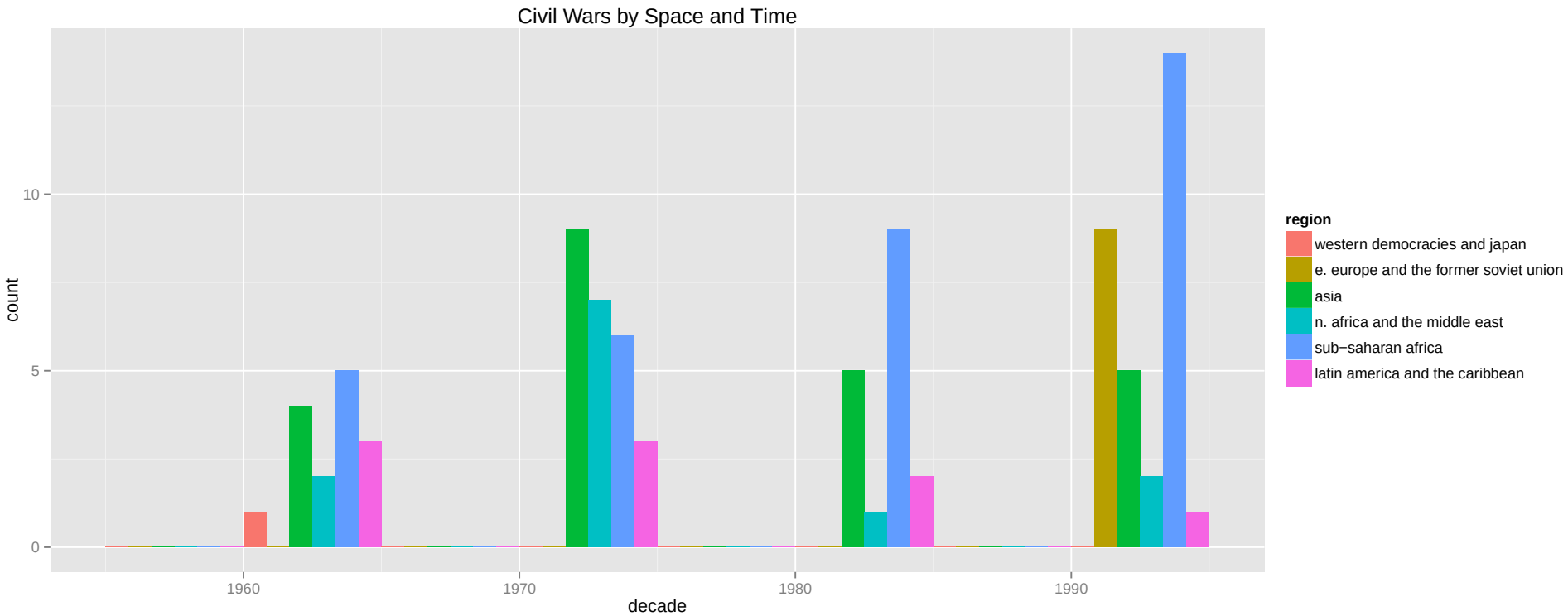
```
p = ggplot(data = dfonset, aes(x=decade,
                               fill = region)) +
  geom_bar(position = "dodge",
           binwidth=5)
p
```



ggplot Syntax: Title

Add Title.

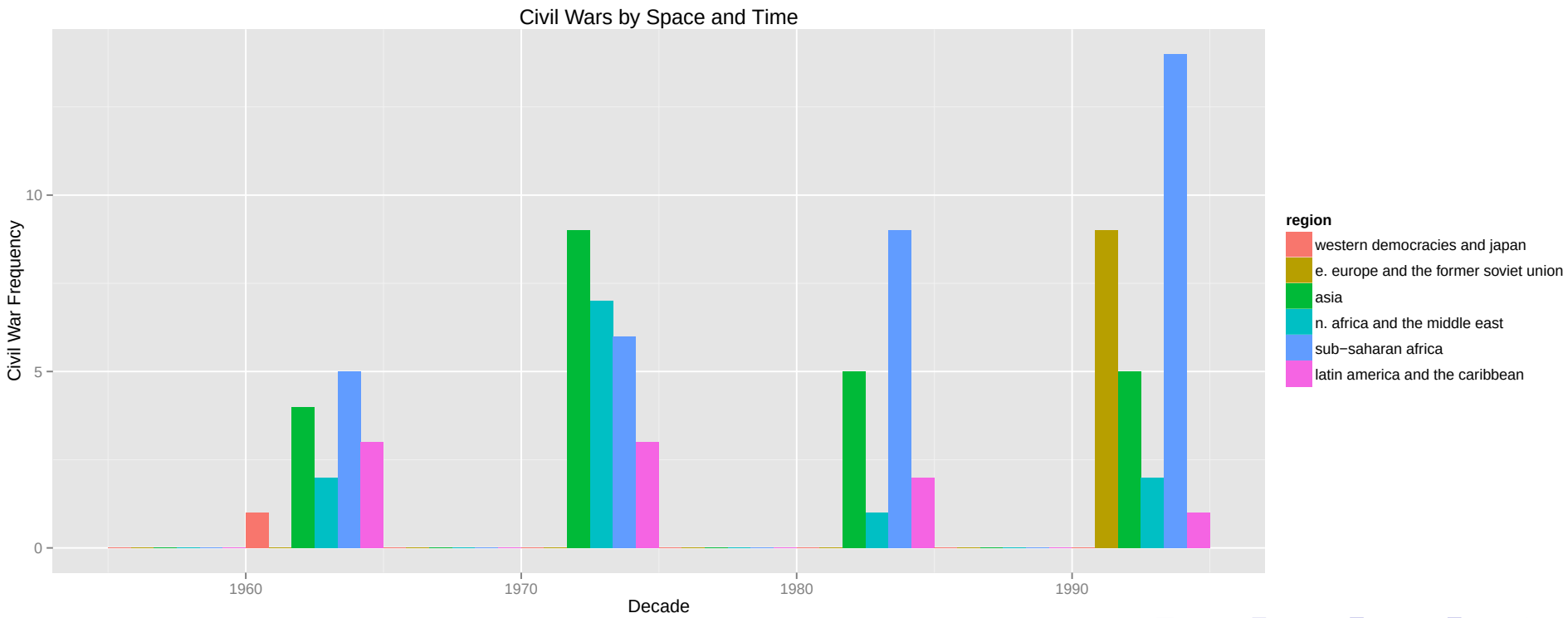
```
p = p +
  ggtitle("Civil Wars by Space and Time")
p
```



ggplot Syntax: Axes

Add Axes.

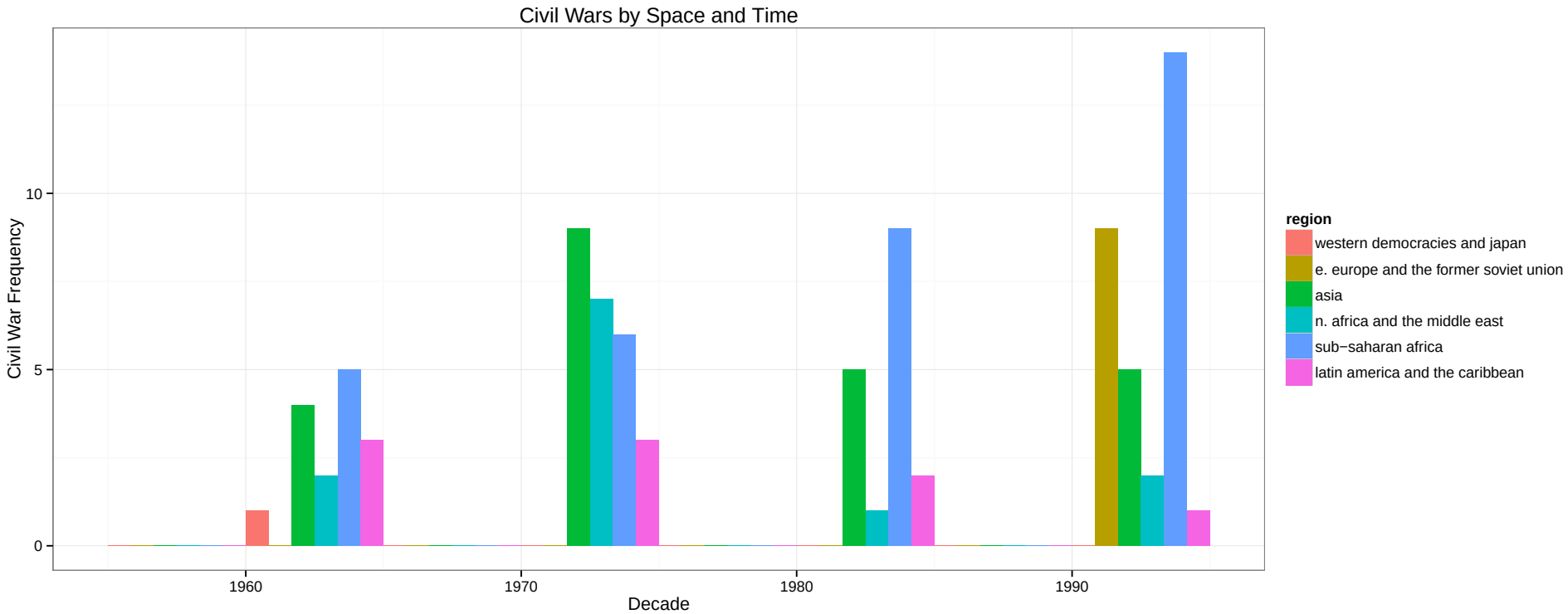
```
p = p + xlab("Decade") +
  ylab("Civil War Frequency")
p
```



ggplot Syntax: Theme

Change Theme.

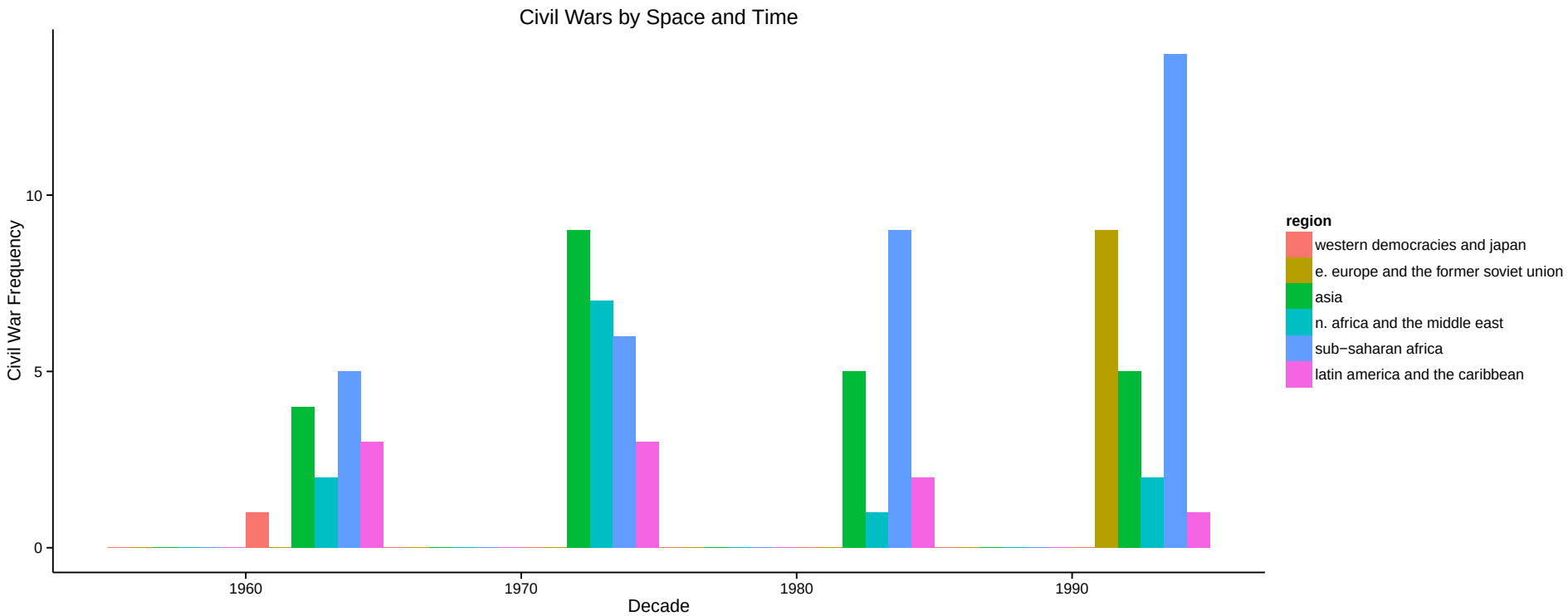
```
p = p + theme_bw()
p
```



ggplot Syntax: Theme

Change Theme.

```
p = p + theme_classic() #looks like plot!
p
```



ggplot Syntax: Theme

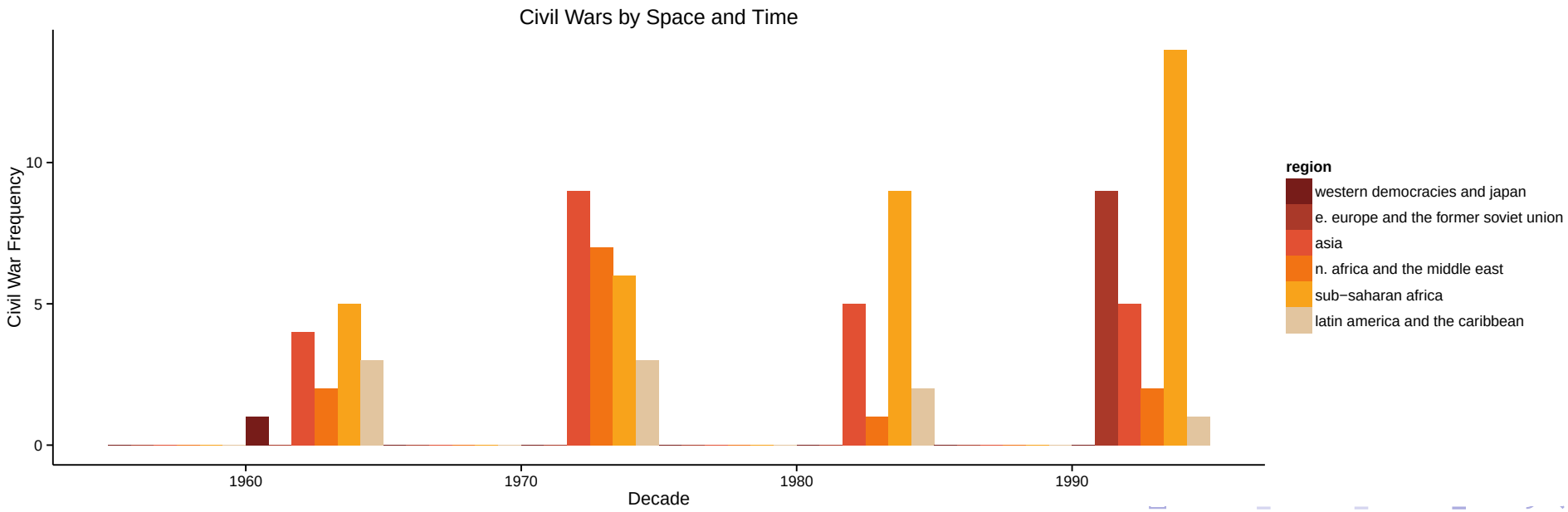
What's great about ggplot is you can customize your own! This is what's going on, under the hood, for theme_classic for example:

```
p = p + theme(panel.grid.major = element_blank(), #grid lines
      panel.grid.minor = element_blank() ,
      panel.border = element_blank(), #border
      panel.background = element_blank(), #background
      #change axes line
      axis.line = element_line(colour = "black"),
      axis.text.x=element_text(colour="black"),
      axis.text.y=element_text(colour="black"))
```

ggplot Syntax: Color.

Change Colors. Reds! Use `scale_fill_manual` for bins, `scale_colour_manual` for lines, points, and `scale_linetype_manual` otherwise.

```
rhg_cols = c("#771C19","#AA3929","#E25033","#F27314",
             "#F8A31B","#E2C59F","#556670","#000000")
p = p + scale_fill_manual(values = rhg_cols)
p
```



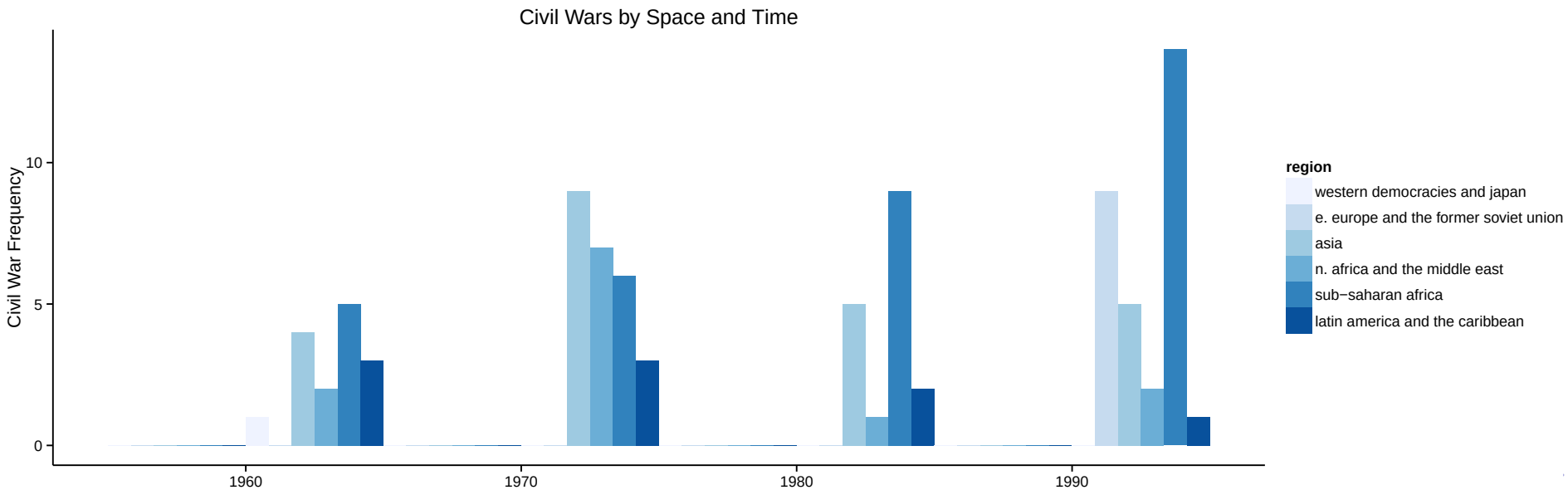
ggplot Syntax: Color.

Or choose others! Blues!

```
#default brewer colors
p = p +
    scale_fill_brewer()
```

```
## Scale for 'fill' is already present. Adding another scale f
```

```
p
```

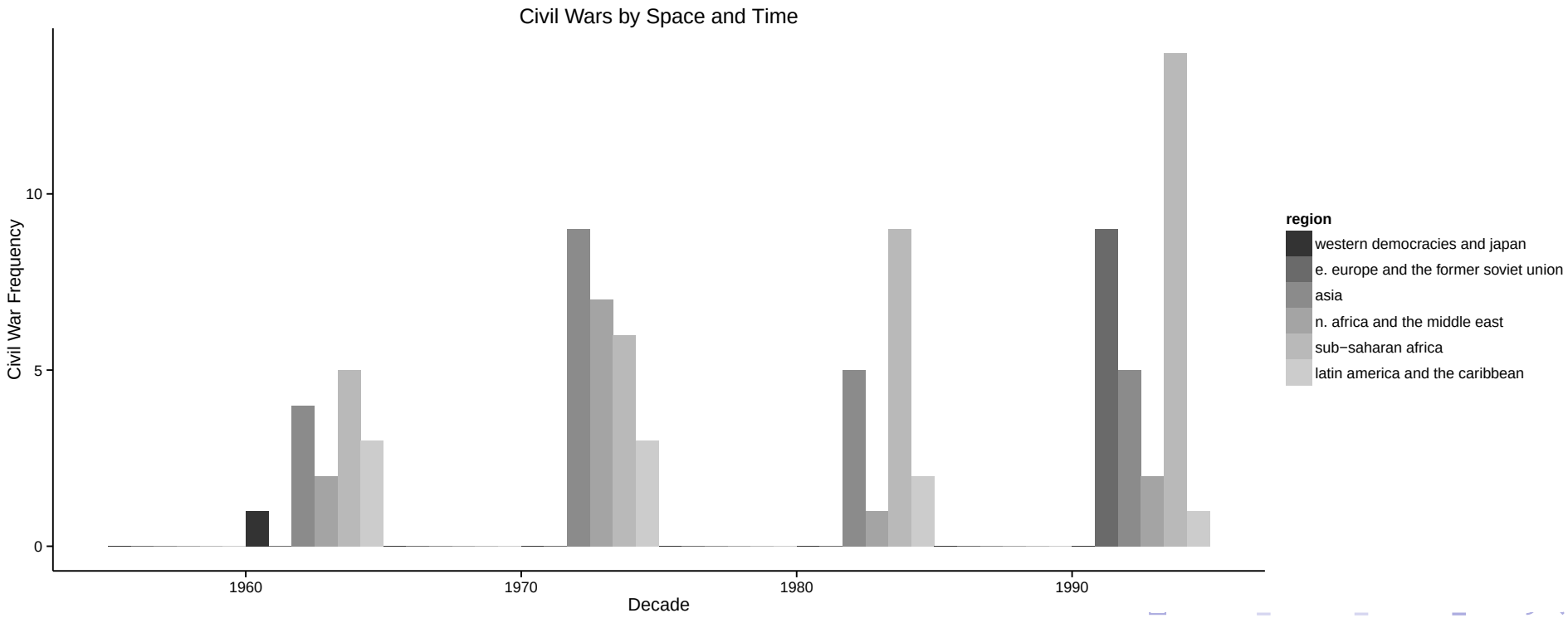


ggplot Syntax: Color.

```
#default grey
p = p + scale_fill_grey()
```

```
## Scale for 'fill' is already present. Adding another scale f
```

```
p
```



ggplot Syntax: Legend.

Change Legend Labels (or Colors)

```
p = p +
  scale_fill_manual(name ="My Super Awesome Legend Title",
    values=c("darkblue", "darkred", "darkgreen",
             "grey", "darkorange", "purple"),
labels=c("Western Dems and Japan", "Former USSR", "Asia",
   "North Africa and \n Middle East", "Sub-Saharan Africa",
    "Latin America"))
```

```
## Scale for 'fill' is already present. Adding another scale f
```

ggplot Syntax: Legend.

Why define fill (or color or linetype) inside aes? Keep track of variables!

Recall:

```
ggplot(data = NULL, aes(x = year)) +
    geom_line(aes(y = numdems, colour = "numdems")) +
    geom_line(aes(y = numauts, colour = "numauts")) +
    scale_color_manual(name = "Regime Type",
    labels = c(numdems="Democracies", numauts = "Autocracies")
    values=c(numdems="blue",numauts="red"))
```

ggplot Syntax: Legend.

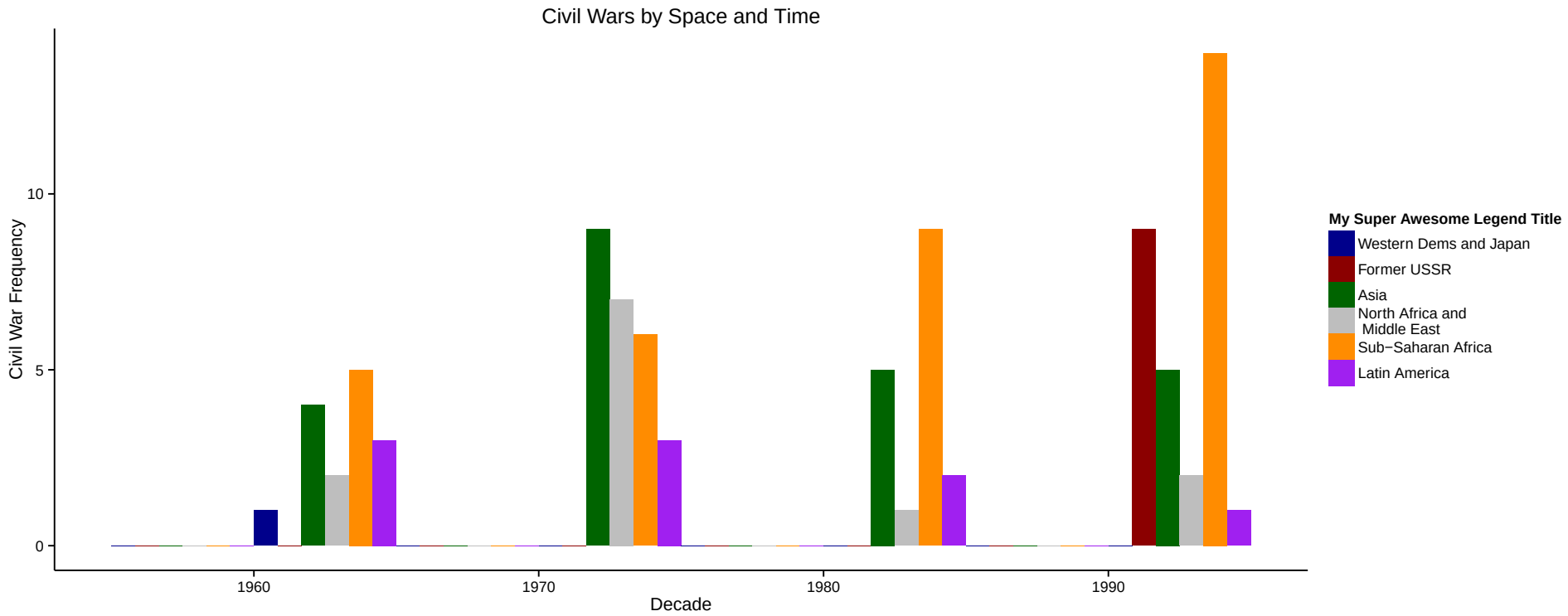
Note: p is ggplot object with 2 components in aes: x = decade and fill = region

```
p = p +
  scale_fill_manual(name ="My Super Awesome Legend Title",
    values=c("darkblue", "darkred", "darkgreen",
             "grey", "darkorange", "purple"),
labels=c("Western Dems and Japan", "Former USSR", "Asia",
   "North Africa and \n Middle East", "Sub-Saharan Africa",
    "Latin America"))
```

```
## Scale for 'fill' is already present. Adding another scale f
```

ggplot Syntax: Legend

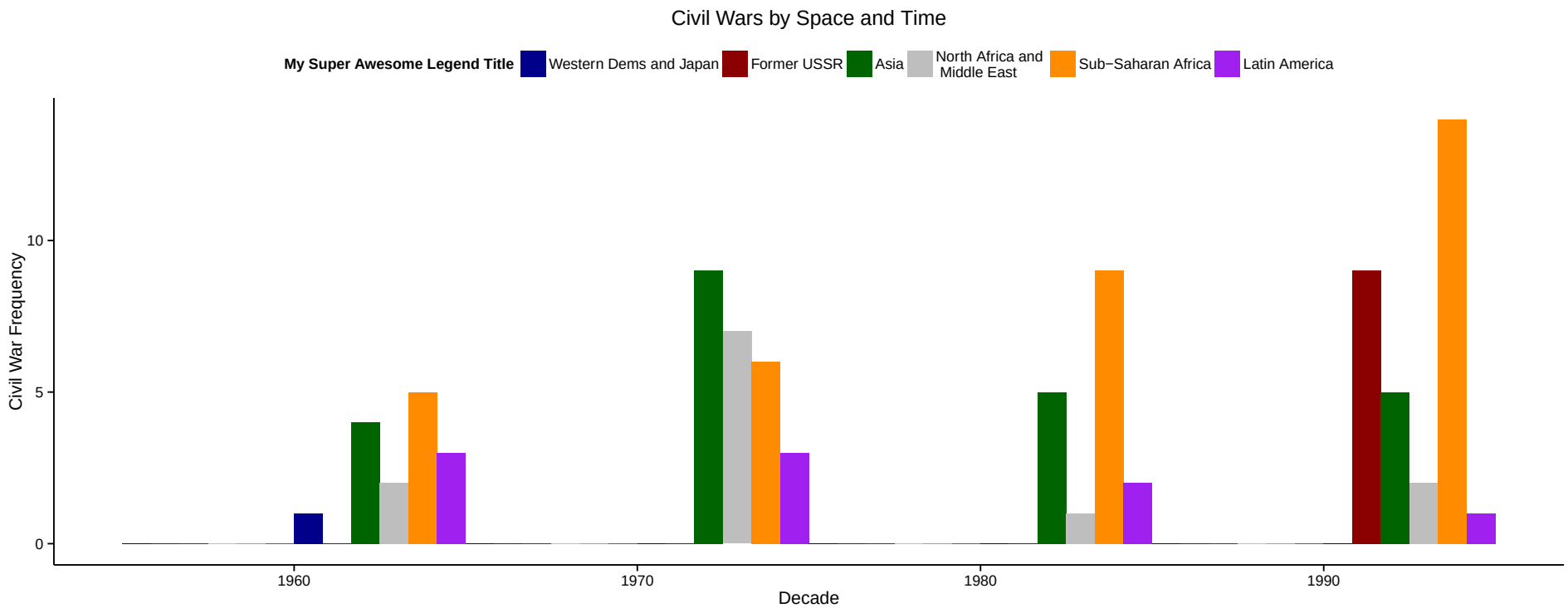
```
p
```



ggplot Syntax: Legend.

Change the position of the legend.

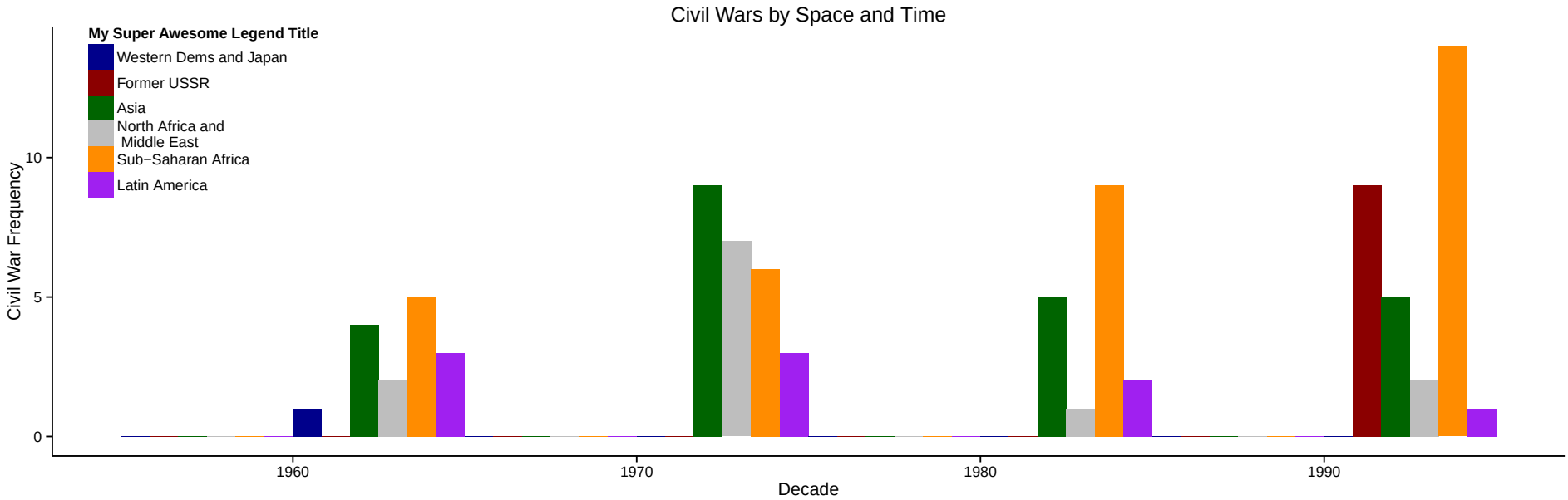
```
p = p + theme(legend.position="top")
p
```



ggplot Syntax: Legend.

Change the position of the legend.

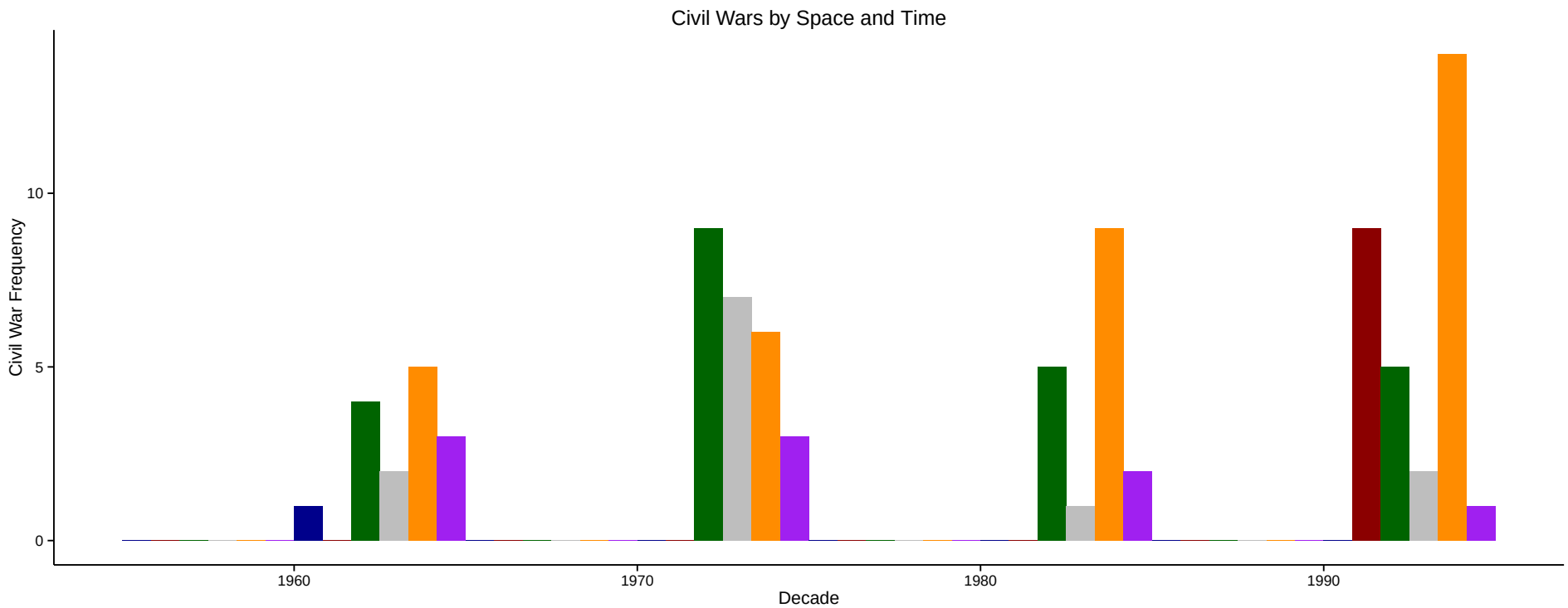
```
# Position legend in graph, where x,y is 0,0 (bottom left)
# to 1,1 (top right)
p = p + theme(legend.position=c(0.10, .8))
p
```



ggplot Syntax: Legend.

Remove the legend.

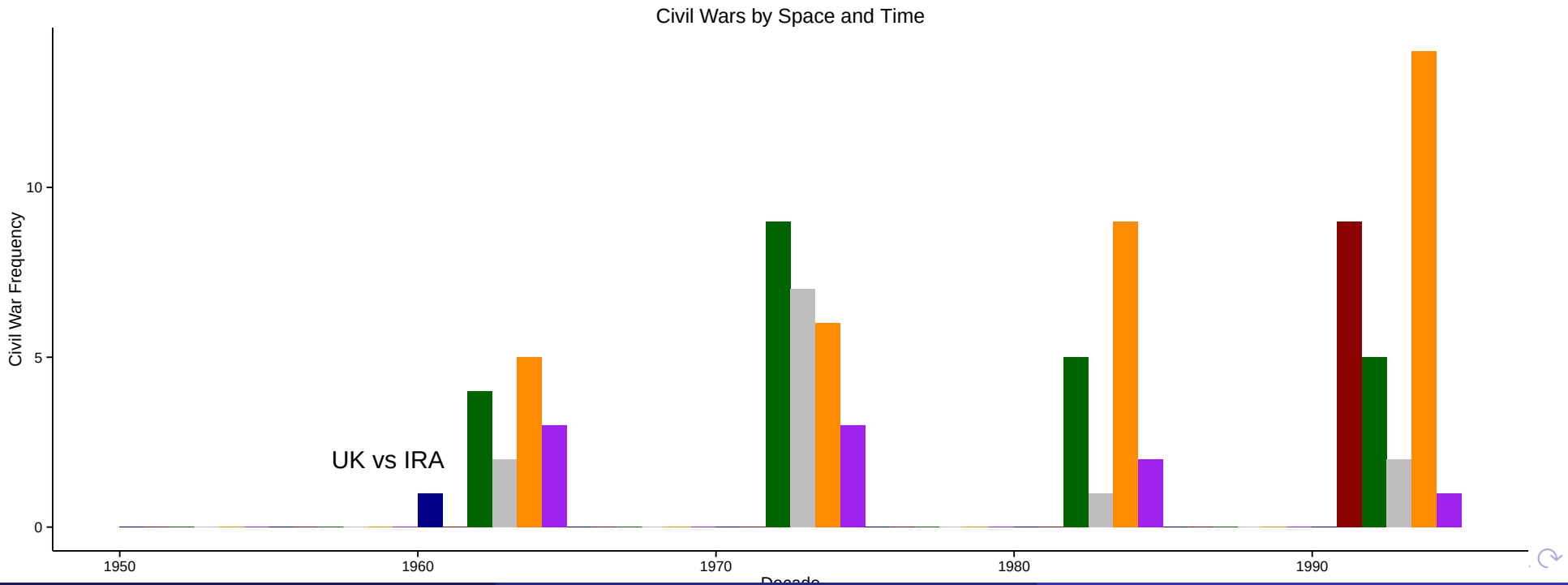
```
p = p + theme(legend.position="none")
p
```



ggplot Syntax: Annotation.

Suppose you want to add a label. For example, what's the one western democracy with a civil war in the 1960s?

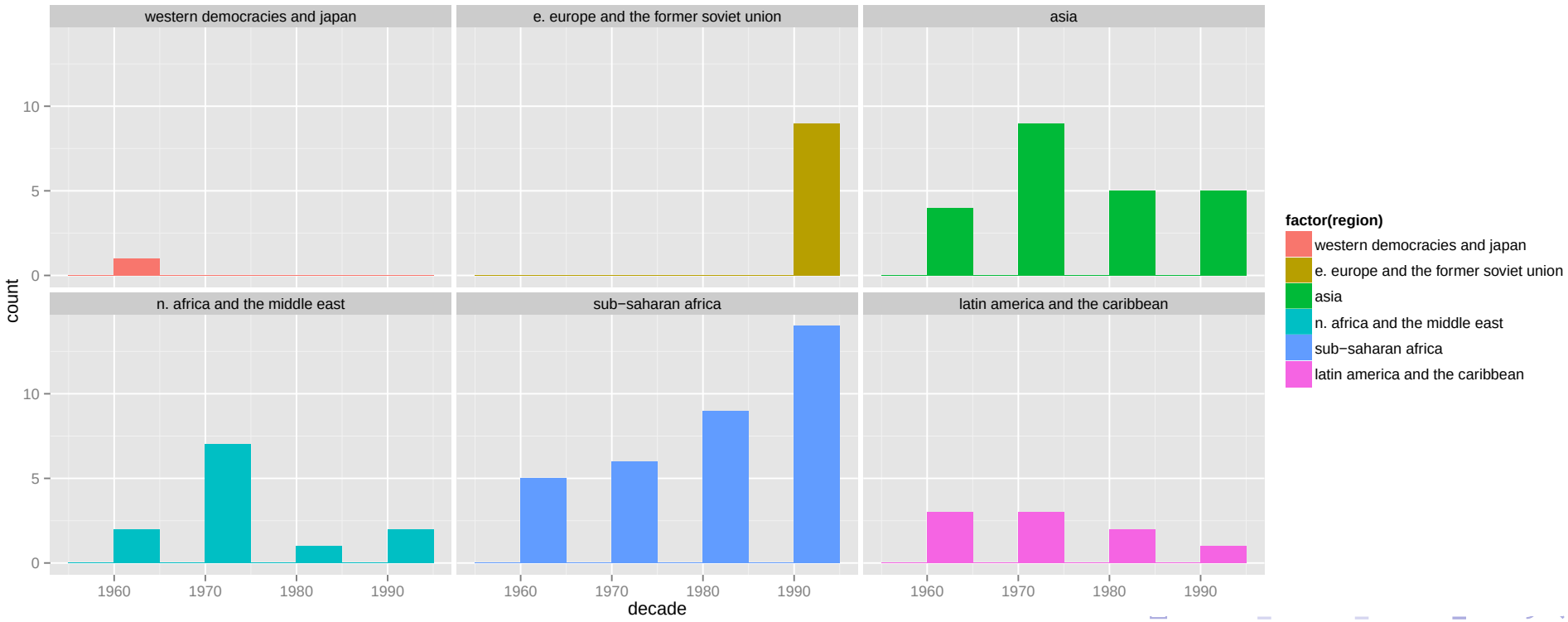
```
#it's the UK vs the IRA
p = p + annotate("text", label = "UK vs IRA",
  x = 1959, y = 2, size = 6, colour = "black")
p
```



ggplot Syntax: Multiple plots.

Suppose you want a separate barplot for every decade.

```
ggplot(data = dfonset, aes(x=decade, group = region,
                           fill = factor(region))) +
  geom_bar(stat="bin", position = "dodge", binwidth=5) +
  facet_wrap(~region) #var you want separate plots by
```



ggplot Syntax: Saving results.

Option 1.

```
pdf("nameoffile.pdf", width=12, height = 5)
p
dev.off()
```

```
## pdf
##   2
```

Option 2.

```
ggsave(p, file="nameoffile.pdf", width=12, height=5)
```

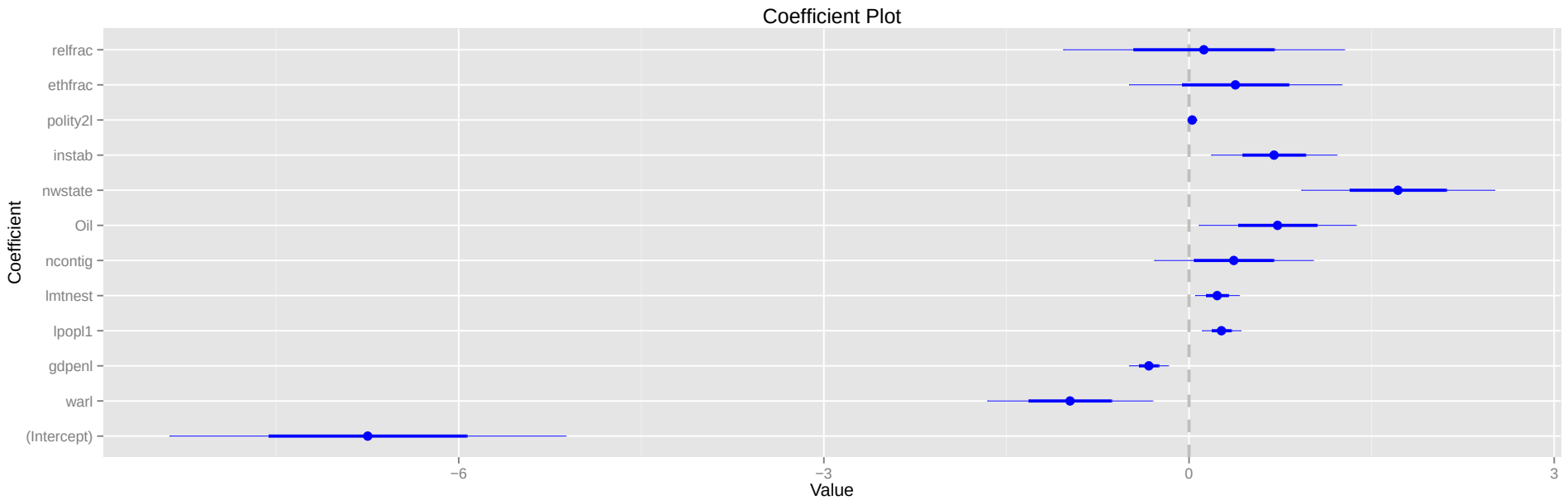
Visualizing Regressions

Visualizing Regression Results

- Coef Plots
- Not covered here: marginal effects, predicted outcomes

Coef Plot: Canned Function

```
m1 = glm(onset ~ warl + gdpenl + lpopl1 + lmtnest + ncontig +
library(coefplot)
coefplot(m1)
```



Coefplot: Our Own Function

Adapted from Stat Bandit

```
#Format the data
coefplot.gg = function(model, data){
    # data is a data frame with 4 columns
    # data$names gives variable names
    # data$modelcoef gives center point
    # data$ylo gives lower limits
    # data$yhi gives upper limits
  modelcoef = summary(model)$coefficients[1:length(model$coeff
  modelse = summary(model)$coefficients[1:length(model$coeffic
  ylo = modelcoef - qt(.975, nrow(data))*(modelse)
  yhi = modelcoef + qt(.975, nrow(data))*(modelse)
  names = names(m1$coefficients)
  dfplot = data.frame(names, modelcoef, modelse, ylo, yhi)
# ...
}
```

Coefplot

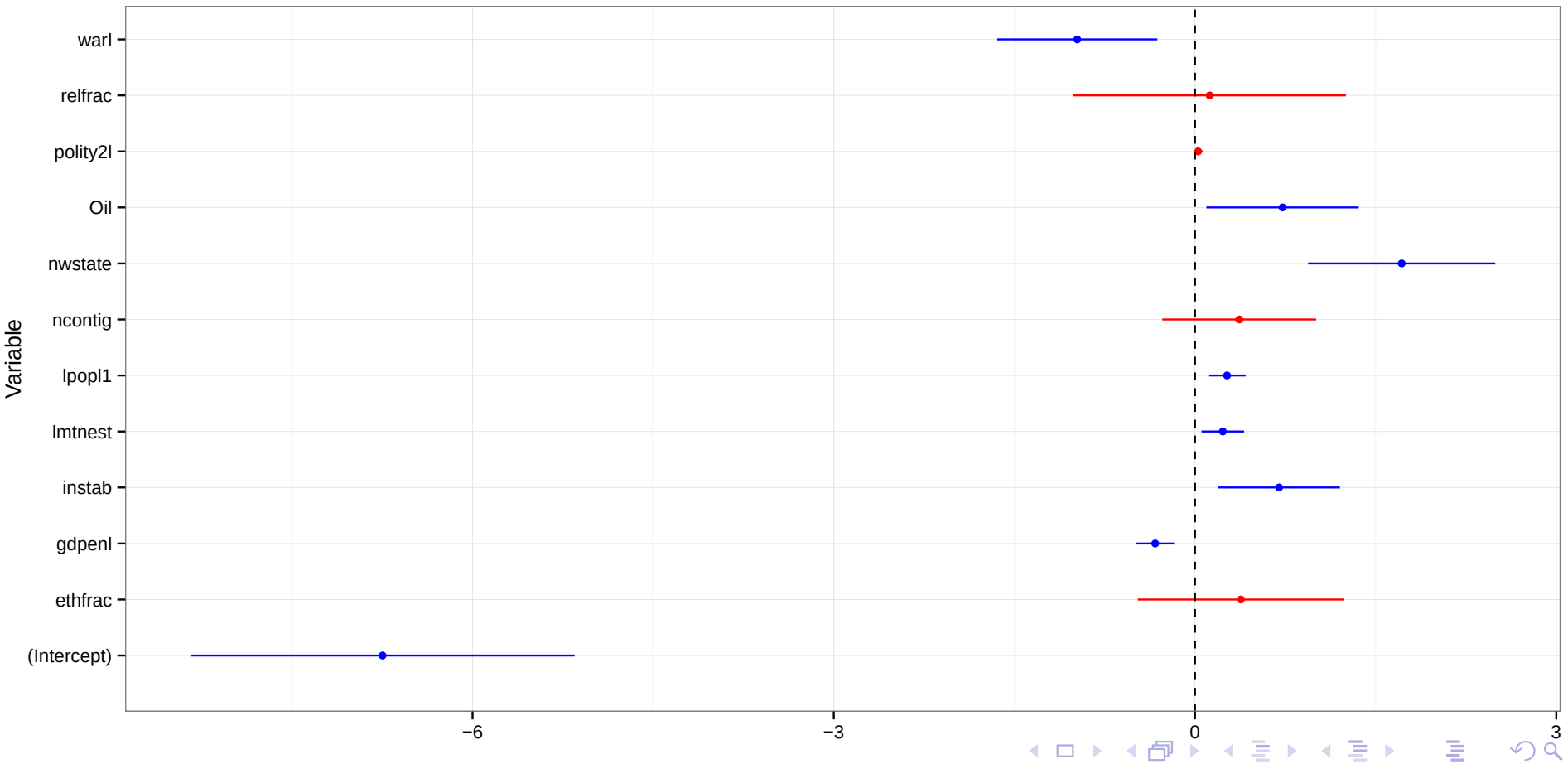
Define the plot

```
coefplot.gg = function(model, data){
  # ...
  #define plot
  library(ggplot2)
    p = ggplot(dfplot, aes(x=names,
                           y=modelcoef,
                           ymin=ylo, ymax=yhi))
    + geom_pointrange(colour=ifelse(ylo < 0 & yhi > 0,
          "red", "blue"))
    + theme_bw()  + coord_flip()
    + geom_hline(aes(x=0), lty=2)
    + xlab('Variable') + ylab('')
    return(p)
}
```

Coefplot

Evaluate the function

```
coefplot.gg(m1, df)
```



Spatial Visualization

Packages:

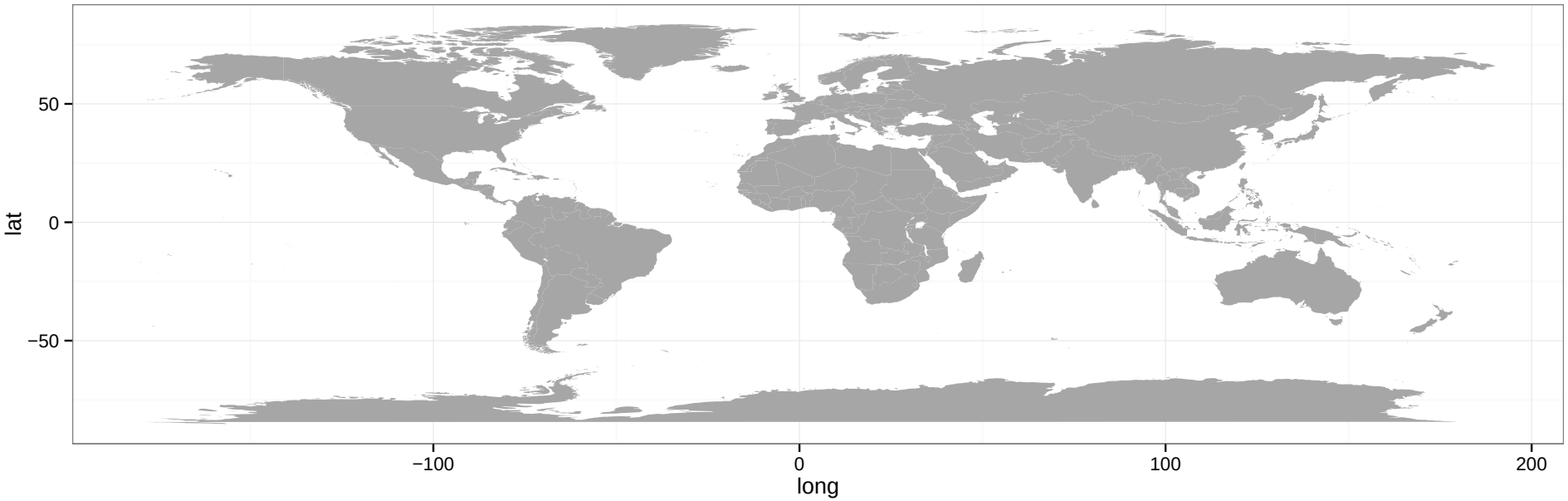
- rworldmap
- maps
- ggmap

```
suppressMessages(library(maps))
suppressMessages(library(ggmap))
suppressMessages(library(mapproj))
```

maps

Adapted from Mahbubul Majumder.

```
dfworldmap = map_data("world")
ggplot() + geom_polygon(aes(x=long,y=lat, group=group),
                        fill="grey65",
                    data=dfworldmap) + theme_bw()
```



Chloropleth maps

Suppose we want to map different levels of a variable by some unit like a state or country. For a toy example, we'll map 1973 murder rates by state using the USArrests data.

Step 1: Format data

```
suppressMessages(library(dplyr))
us = map_data("state")
head(us)
```

```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

Chloropleth maps

```
head(USArrests)
```

```
##             Murder Assault UrbanPop Rape
## Alabama       13.2     236       58 21.2
## Alaska        10.0     263       48 44.5
## Arizona        8.1     294       80 31.0
## Arkansas       8.8     190       50 19.5
## California     9.0     276       91 40.6
## Colorado       7.9     204       78 38.7
```

```
#mismatch between region and state
# need to add var to arrest data to match 1:1
```

Chloropleth maps

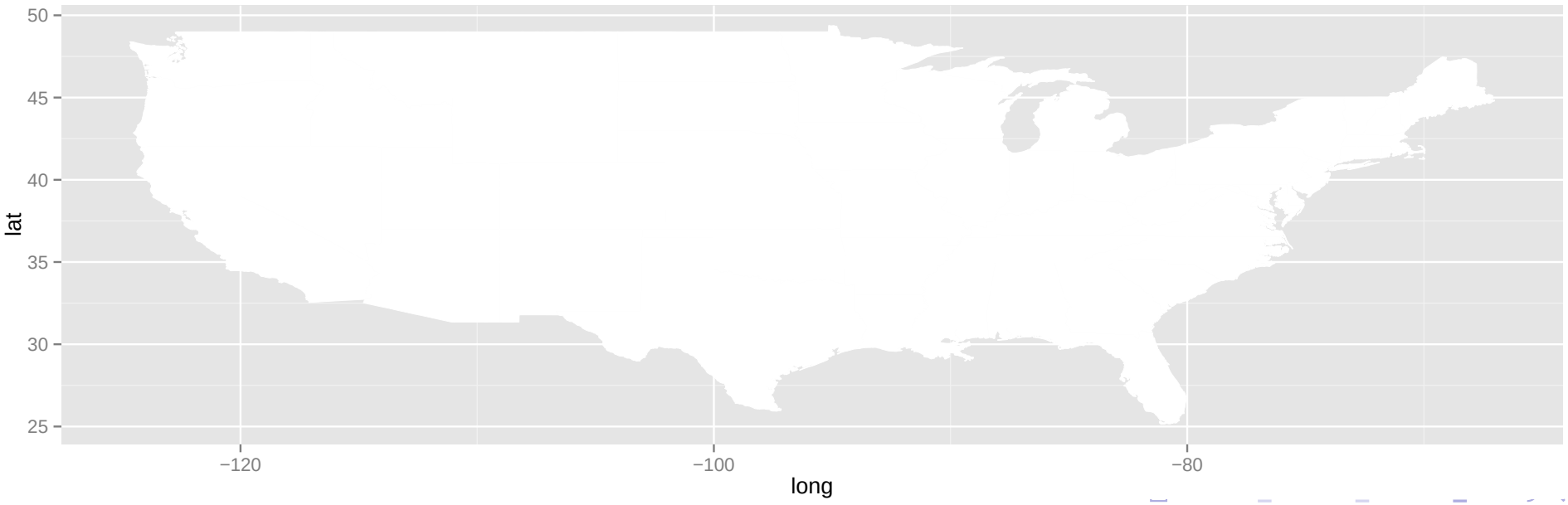
```
arrest = USArrests %>%
  add_rownames("region") %>%
  #use mutate function from plyr
  mutate(region=tolower(region)) #make it all lowercase
#format to work with map
head(arrest)
```

```
## Source: local data frame [6 x 5]
##
##       region Murder Assault UrbanPop  Rape
##        (chr)  (dbl)   (int)    (int) (dbl)
## 1     alabama   13.2     236       58  21.2
## 2      alaska   10.0     263       48  44.5
## 3     arizona    8.1     294       80  31.0
## 4    arkansas    8.8     190       50  19.5
## 5  california    9.0     276       91  40.6
## 6    colorado    7.9     204       78  38.7
```

Chloropleth maps

Step 2: Plot the base map layer

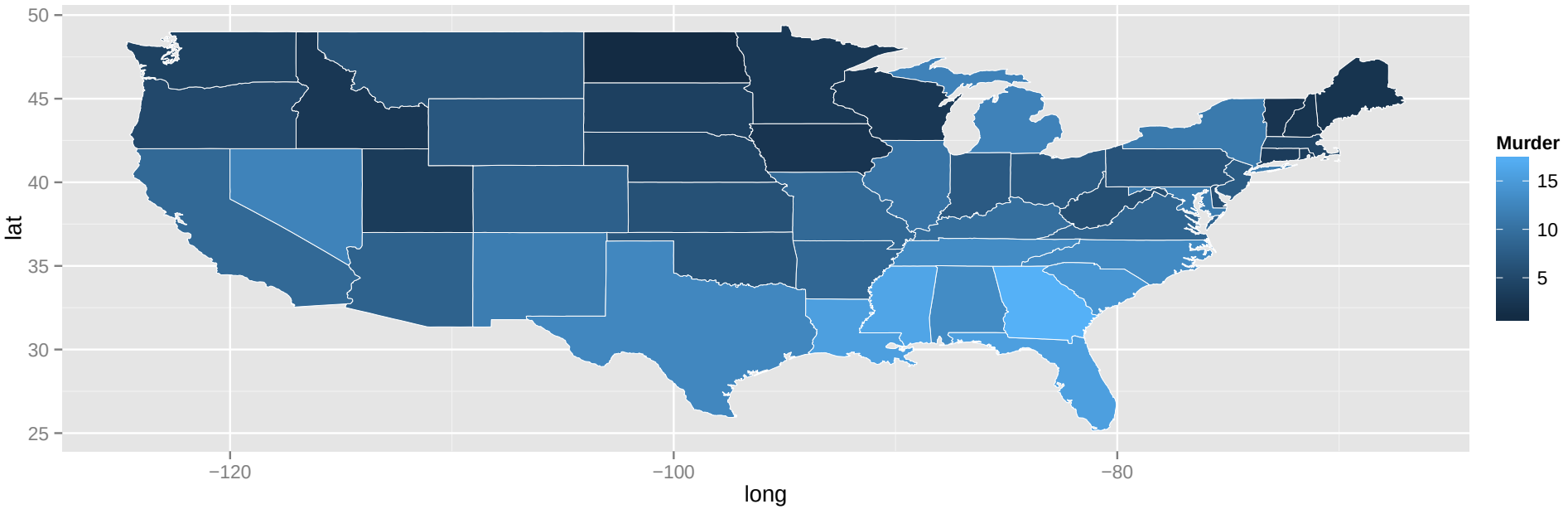
```
g = ggplot()
#must define map first
g = g + geom_map(data=us, map=us,
                 aes(x=long, y=lat, map_id=region),
                 fill="#ffffff", color="#ffffff", size=0.15
g
```



Chloropleth maps

Step 3: Add our arrest data

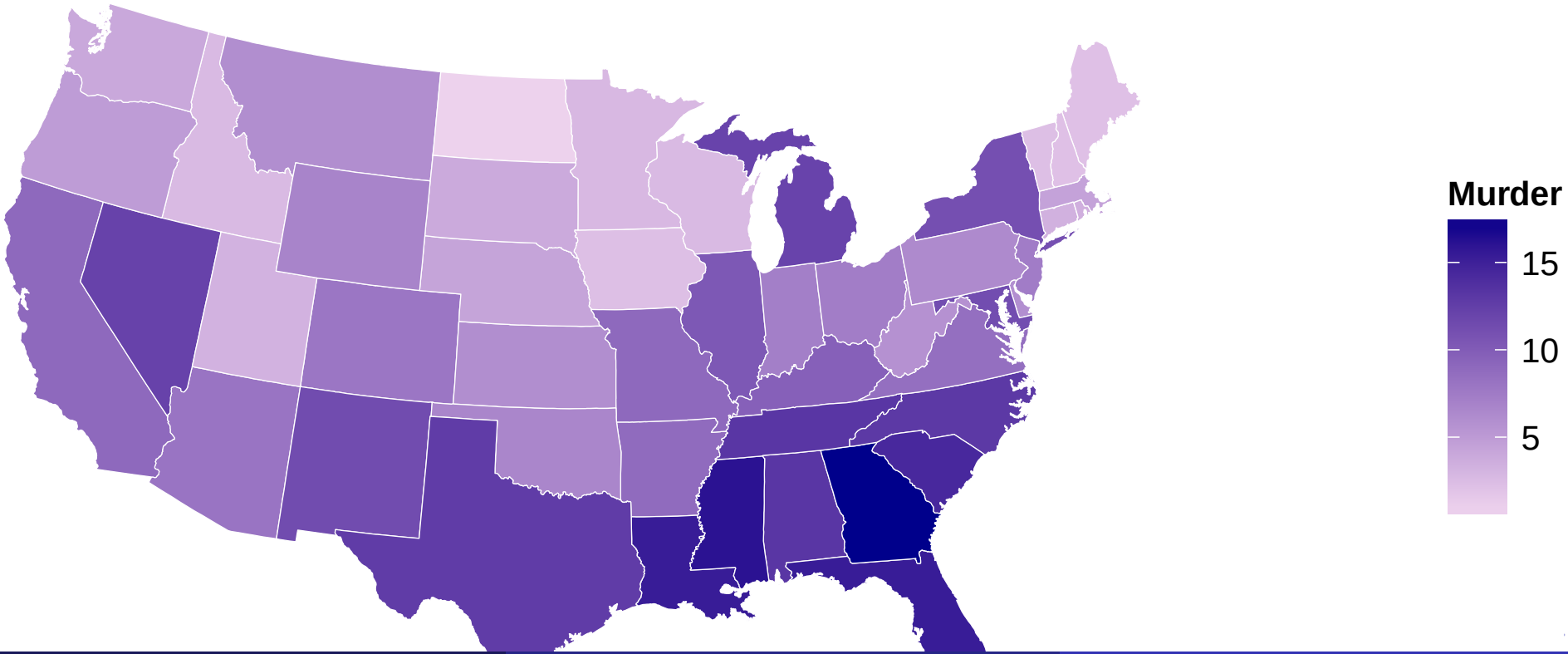
```
g = g + geom_map(data=arrest, map=us,
                 aes(fill=Murder, map_id=region),
                 color="#ffffff", size=0.15)
g
```



Chloropleth maps

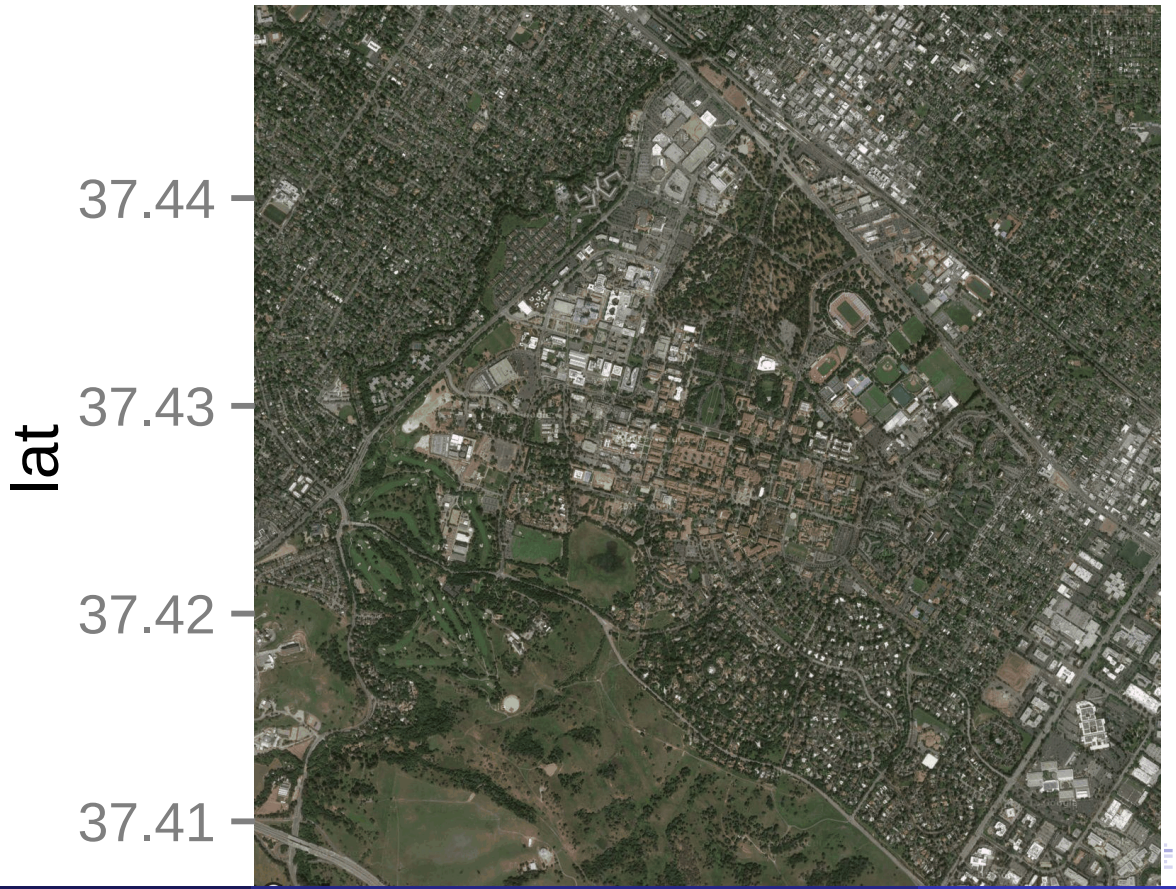
Step 4: Make it look pretty.

```
g = g + scale_fill_continuous(low='thistle2', high='darkblue',
                              guide='colorbar') +  xlab("")
g = g + theme(panel.border = element_blank()) + theme(panel.ba
g
```



ggmap

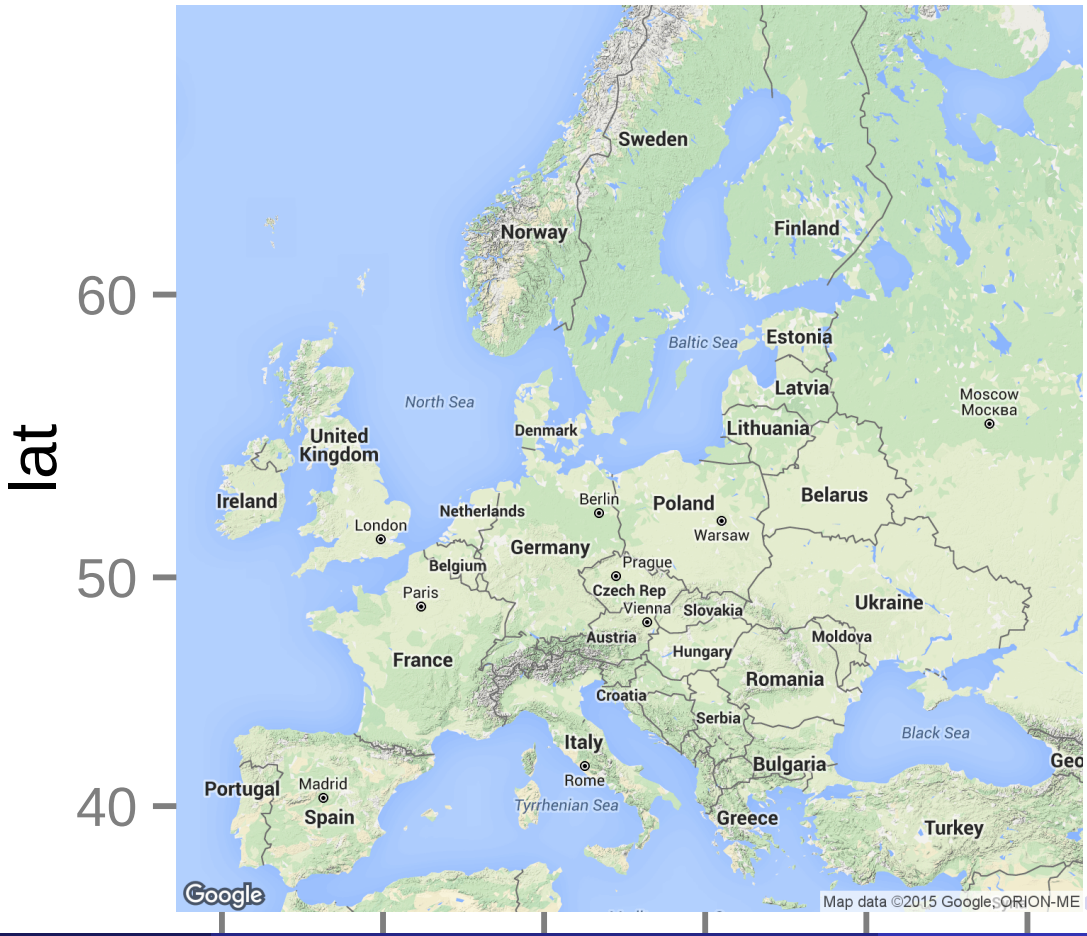
```
map1 = suppressMessages(get_map(
  location = 'Stanford University', zoom = 14, #zoom-in level
  maptype="satellite")) #map type
ggmap(map1)
```



ggmap

Step 1: Pull a location you want to plot from Google maps.

```
map = suppressMessages(get_map(location = 'Europe', zoom = 4))
ggmap(map)
```



ggmap

Step 2: Get geocoordinates for points or locations you're interested in.

```
europegps = suppressMessages(geocode(c(
                          "Lisbon, Portugal",
                           "Eiffel Tower",
                          "Berlin, Germany",
                           "Crimea, Ukraine"), source="google"))
europegps
```

```
##         lon      lat
## 1 -9.139337 38.72225
## 2  2.294481 48.85837
## 3 13.404954 52.52001
## 4 34.102417 44.95212
```

ggmap

Step 3: Add geocoordinates to ggmap! It's just like working with another ggplot object.

```
ggmap(map) + geom_point(aes(x=europegps$lon, y = europegps$lat
    lwd = 4, colour = "red") + ggtitle("Place I would like to
```

Summary

- ggplot is super powerful

Summary

- ggplot is super powerful but kind of annoying to learn

Summary

- ggplot is super powerful but kind of annoying to learn
- Ability to make complicated graphs awesome

Summary

- ggplot is super powerful but kind of annoying to learn
- Ability to make complicated graphs awesome
- Benefits outweigh the start-up costs

Summary

- ggplot is super powerful but kind of annoying to learn
- Ability to make complicated graphs awesome
- Benefits outweigh the start-up costs