Lecture 9: Data Visualization (base R)

Harris Coding Camp – Standard Track

Summer 2022

Data Visualization: Motivation

Suppose we want to know the following info:

- What is the annual housing sales in Texas over time?
- Can we learn about the relevant housing prices of one city to another in Texas?

We probably want info about the *average* of housing price, how much the price *varies* over time, and info about the *extremes*

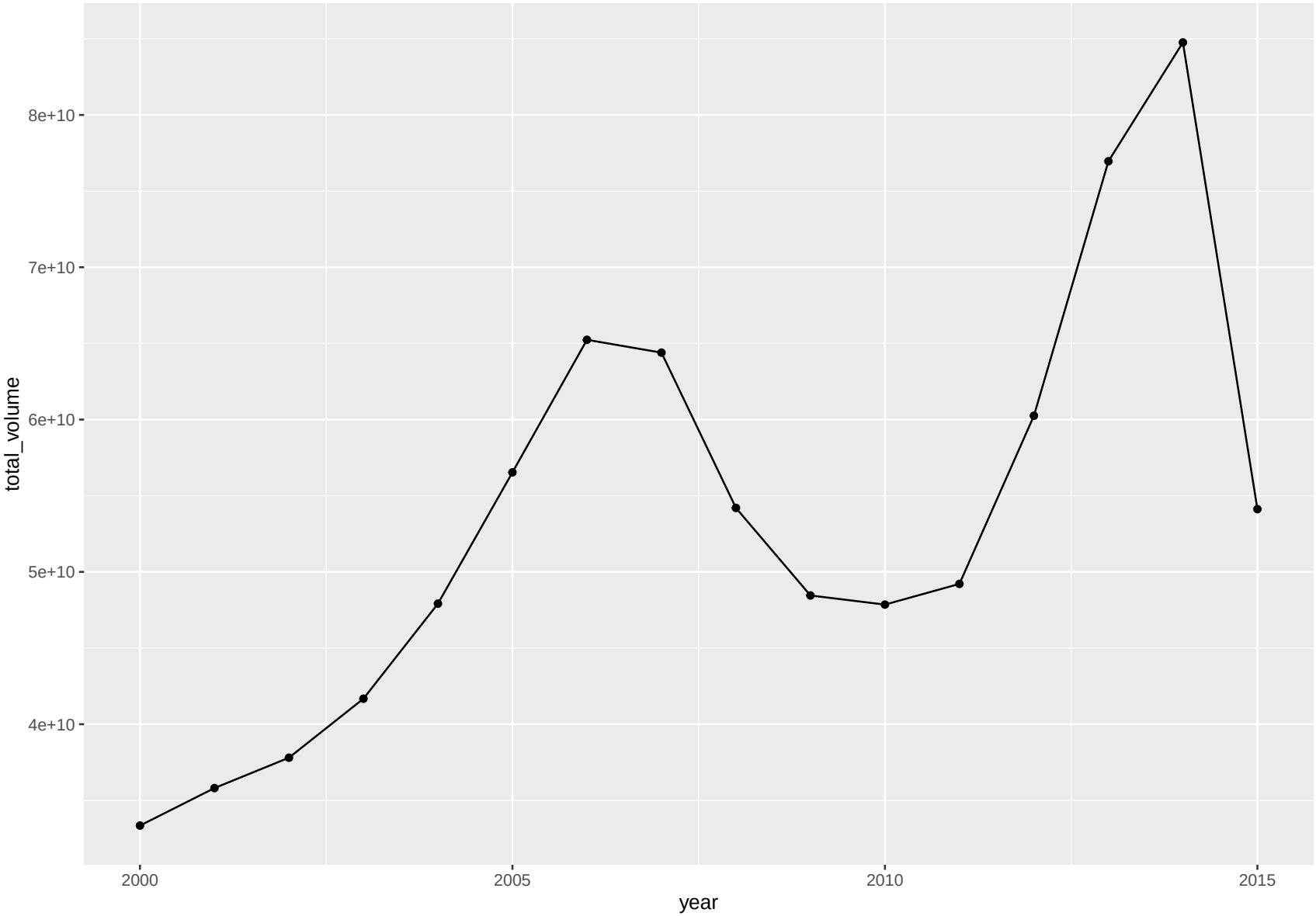
Data Visualization: Motivation

Is simply looking at all these data helpful?

	city	year	month	sales	volume	median	listings	inventory	date
171	Abilene	2014	3	165	21615979	112400	929	5.1	2014.167
172	Abilene	2014	4	178	25070931	124500	965	5.4	2014.250
173	Abilene	2014	5	191	26615130	129300	1001	5.7	2014.333
174	Abilene	2014	6	230	34398506	135200	1023	5.8	2014.417
175	Abilene	2014	7	231	35861350	145800	1033	5.8	2014.500
176	Abilene	2014	8	203	29671329	129200	1024	5.8	2014.583
177	Abilene	2014	9	201	30904319	135900	1012	5.8	2014.667
178	Abilene	2014	10	190	25431388	122400	987	5.6	2014.750
179	Abilene	2014	11	158	24120554	132200	941	5.3	2014.833
180	Abilene	2014	12	165	23959468	130000	823	4.6	2014.917
181	Abilene	2015	1	158	23486998	134100	801	4.4	2015.000
182	Abilene	2015	2	151	19834263	126500	767	4.1	2015.083
183	Abilene	2015	3	198	31869437	136800	821	4.4	2015.167
184	Abilene	2015	4	201	28301159	129600	891	4.7	2015.250
185	Abilene	2015	5	199	31385757	144700	919	4.8	2015.333
186	Abilene	2015	6	260	41396230	141500	965	5.0	2015.417
187	Abilene	2015	7	268	45845730	148700	986	5.0	2015.500
188	Amarillo	2000	1	102	8860000	80000	972	5.3	2000.000
189	Amarillo	2000	2	147	13875000	78300	937	5.1	2000.083
190	Amarillo	2000	3	201	17930000	74800	995	5.5	2000.167
191	Amarillo	2000	4	176	16955000	87700	985	5.5	2000.250
192	Amarillo	2000	5	198	19420000	81100	1020	5.6	2000.333
193	Amarillo	2000	6	206	20035000	84100	1055	5.9	2000.417
194	Amarillo	2000	7	190	20430000	94300	1193	6.8	2000.500
195	Amarillo	2000	8	242	25995000	95300	1208	6.6	2000.583
196	Amarillo	2000	9	158	16000000	91000	1248	6.9	2000.667
197	Amarillo	2000	10	165	16810000	89000	1191	6.7	2000.750
198	Amarillo	2000	11	133	13955000	87300	1174	6.7	2000.833
199	Amarillo	2000	12	158	15800000	88300	1092	6.3	2000.917
200	Amarillo	2001	1	134	14445000	89000	1087	6.2	2001.000
201	Amarillo	2001	2	157	15345000	87100	1037	5.9	2001.083
202	Amarillo	2001	3	187	19260000	86300	1030	5.9	2001.167
203	Amarillo	2001	4	199	21740000	93200	1055	6.0	2001.250
204	Amarillo	2001	5	225	22225000	88300	1064	5.9	2001.333
205	Amarillo	2001	6	242	25135000	91200	1092	6.0	2001.417

Data Visualization: Motivation

What if we make a plot of annual housing sales over time. . .



Data Visualization: Introduction

- Be able to plot in base R
- Know what types of figures you want
- Use plots to gain information
  - graphs, including axes, must be labeled properly to allow readers to understand the visualization with ease
  - variable scales should be portrayed accurately
  - graphs should be as intuitive as possible

Data Visualization: Introduction

Two main frameworks which create graphics in R: The *base R* framework, and the *tidyverse/ggplot* framework

- base R: consists of about 30 packages that are always loaded automatically when we open R
- It is the oldest way to generate visualizations in R
- *base R* can and is still used to create visualizations, though most visualizations are now generated using the *tidyverse/ggplot* framework (more details in Lecture 10)

Getting started

First, let's load the data set and call it `pdat`[1]. The data set contains:

- `Date`: the year when a text was written
- `Genre`: the genre of the text
- `Text`: the name of the text
- `Prepositions`: the relative frequency of prepositions in the text
- `Region`: the region in which the text was written
- `GenreRedux` collapses the existing genres into five main categories (*Conversational*, *Religious*, *Legal*, *Fiction*, and *NonFiction*)
- `DateRedux` collapses the dates when the texts were composed into five periods (1150-1499, 1500-1599, 1600-1699, 1700-1799, and 1800-1913)

```
# load online data and call it 'pdat'
pdat <- base::readRDS(url("https://slcladal.github.io/data/pvd.rda",
                          "rb"))
```

[1] For more details, see https://ladal.edu.au/introviz.html

Getting started

Let's look at the data (first 15 rows):

Date	Genre	Text	Prepositions	Region	GenreRedux	DateRedux
1,736	Science	albin	166.01	North	NonFiction	1700-1799
1,711	Education	anon	139.86	North	NonFiction	1700-1799
1,808	PrivateLetter	austen	130.78	North	Conversational	1800-1913
1,878	Education	bain	151.29	North	NonFiction	1800-1913
1,743	Education	barclay	145.72	North	NonFiction	1700-1799
1,908	Education	benson	120.77	North	NonFiction	1800-1913
1,906	Diary	benson	119.17	North	Conversational	1800-1913
1,897	Philosophy	boethja	132.96	North	NonFiction	1800-1913
1,785	Philosophy	boethri	130.49	North	NonFiction	1700-1799
1,776	Diary	boswell	135.94	North	Conversational	1700-1799
1,905	Travel	bradley	154.20	North	NonFiction	1800-1913
1,711	Education	brightland	149.14	North	NonFiction	1700-1799
1,762	Sermon	burton	159.71	North	Religious	1700-1799
1,726	Sermon	butler	157.49	North	Religious	1700-1799
1,835	PrivateLetter	carlyle	124.16	North	Conversational	1800-1913

Simplest `plot` code structure

```
plot(y ~ x,                      # plot y by x
     type = "[type]",            # type (points, lines, ...)
     data = [df_name],           # data from [df_name]
     ylab = "[y variable]",      # add y-axis label
     xlab = "[x variable]",      # add x-axis label
     main = "[title]"            # add title
     )
```

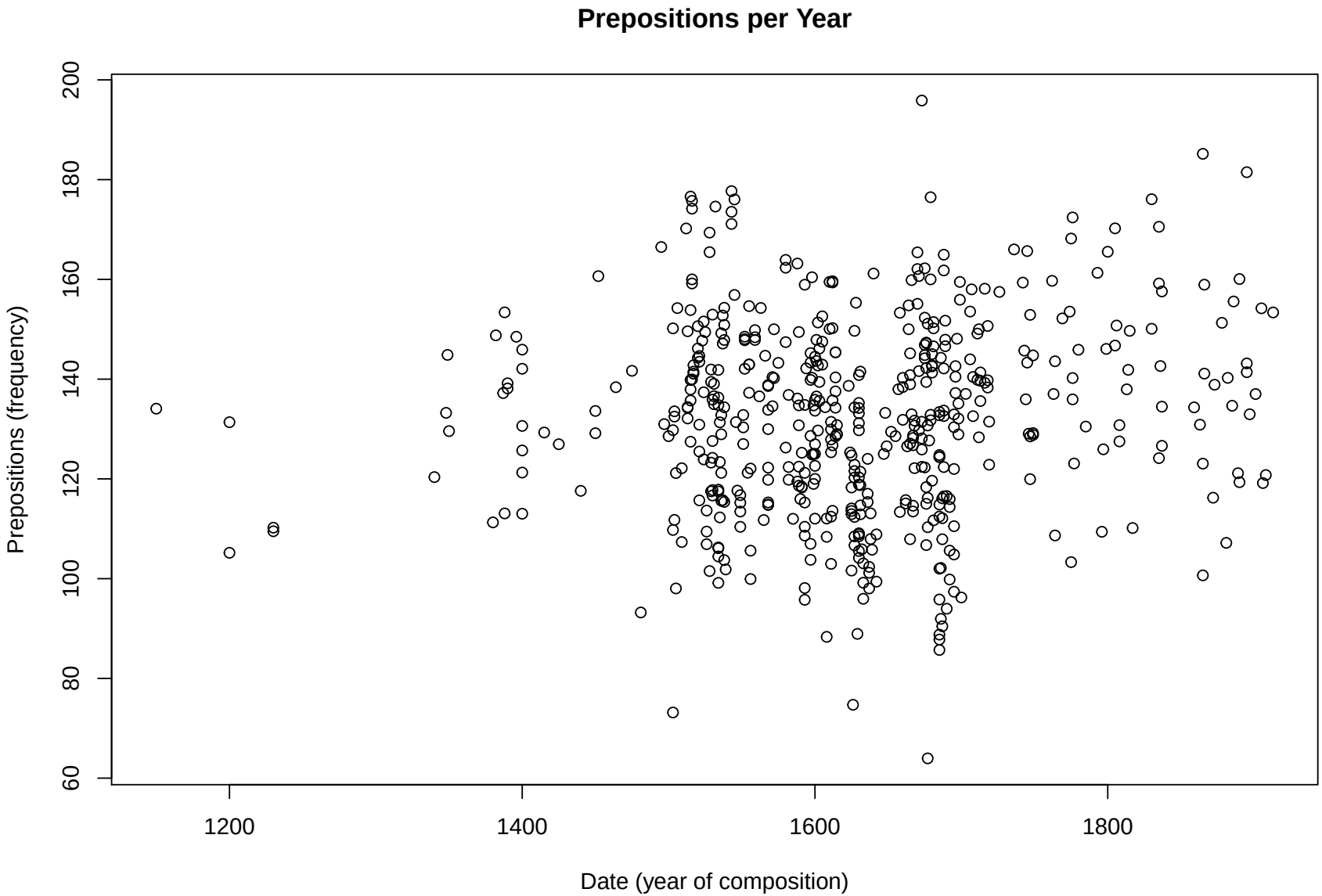
Understanding `plot()`: Scatter plot

Let's get started! `plot()` tells R to prepare to make a plot.

The simplest graph is a *scatter* or *dot plot*. Scatter plots are used when the graph is set up to display the relationship between two variables.

```
plot(Prepositions ~ Date,    # plot Prepositions by Date
     type = "p",             # plot type "p" (points)
     data = pdat,            # data from pdat
     ylab = "Prepositions (Frequency)",     # add y-axis label
     xlab = "Date (year of composition)",   # add x-axis label
     main = "Prepositions per Year"         # add title
     )
```

Understanding `plot()`: Scatter plot



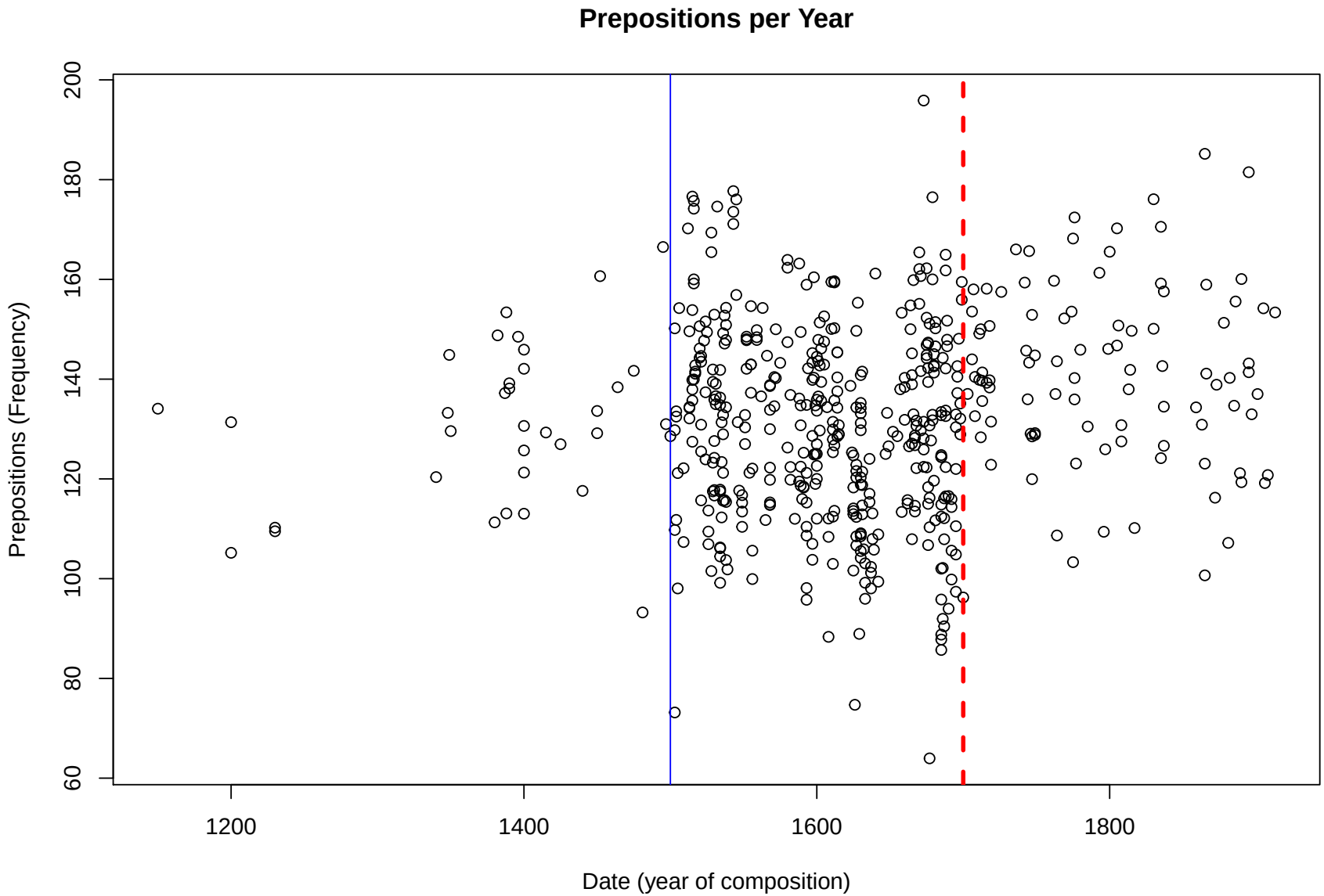
Understanding `plot()`: Scatter plot + Vertical lines

Often, we may include more info by adding some vertical lines to a plot:

Syntax: `abline(v = [number], <other options>)`

```
plot(Prepositions ~ Date,    # plot Prepositions by Date
     type = "p",             # plot type "p" (points)
     data = pdat,            # data from pdat
     ylab = "Prepositions (frequency)",     # add y-axis label
     xlab = "Date (year of composition)",   # add x-axis label
     main = "Prepositions per Year"         # add title
     )
abline(                      # add a line
  v = c(1500,1700),          # draw two v (vertical) lines
  col=c("blue", "red"),      # define line colors
  lty=c(1,2),                # define line types
  lwd=c(1,3)                 # define line widths
  )
```

Understanding `plot()`: Scatter plot + Vertical lines



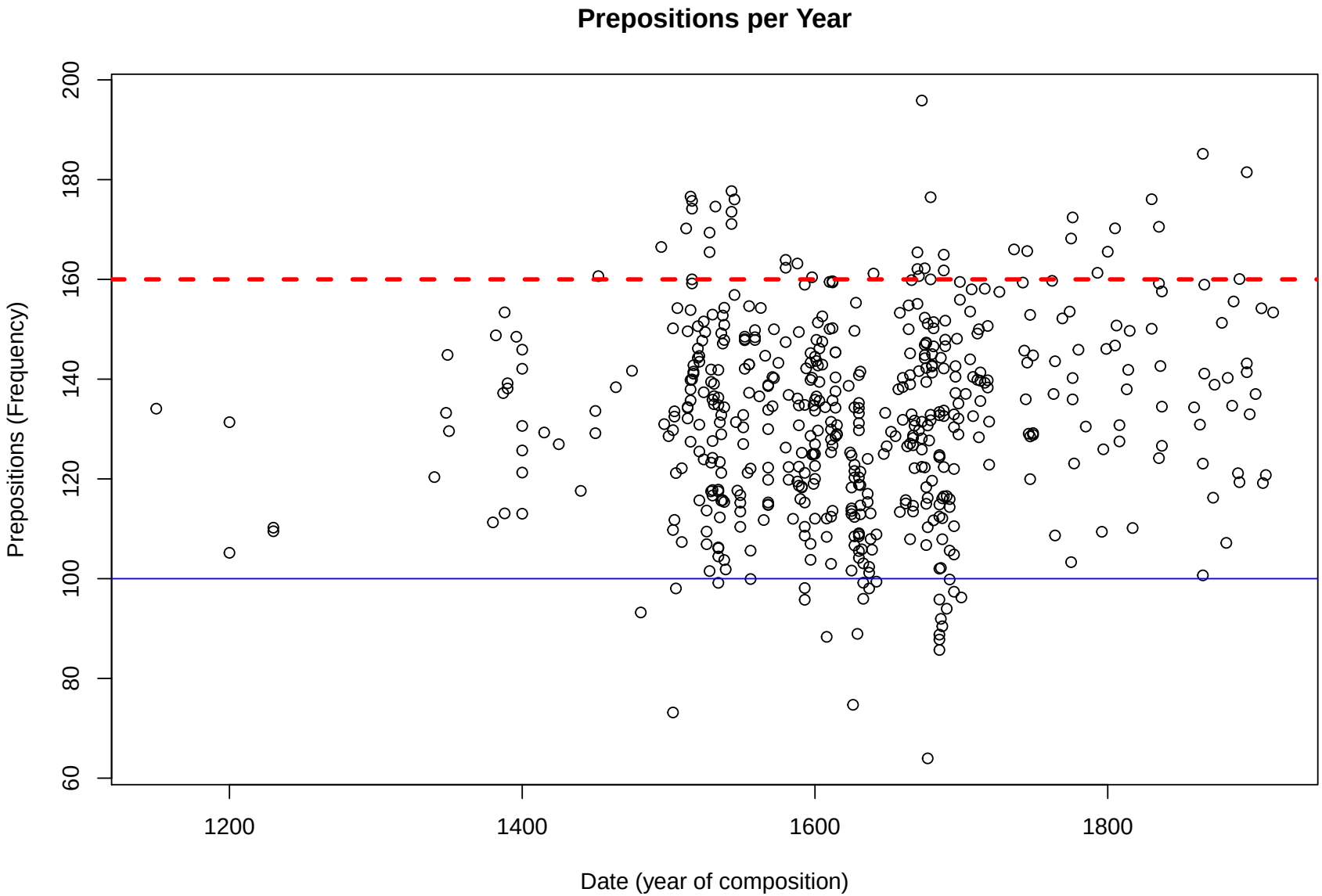
Understanding `plot()`: Scatter plot + Horizontal lines

. . . or by adding some horizontal lines to a plot:

Syntax: `abline(h = [number], <other options>)`

```
plot(Prepositions ~ Date,    # plot Prepositions by Date
     type = "p",             # plot type "p" (points)
     data = pdat,            # data from pdat
     ylab = "Prepositions (frequency)",     # add y-axis label
     xlab = "Date (year of composition)",   # add x-axis label
     main = "Prepositions per Year"         # add title
     )
abline(                      # add a line
  h = c(100,160),            # draw two h (horizontal) lines
  col=c("blue", "red"),      # define line colors
  lty=c(1,2),                # define line types
  lwd=c(1,3)                 # define line widths
  )
```

Understanding `plot()`: Scatter plot + Horizontal lines



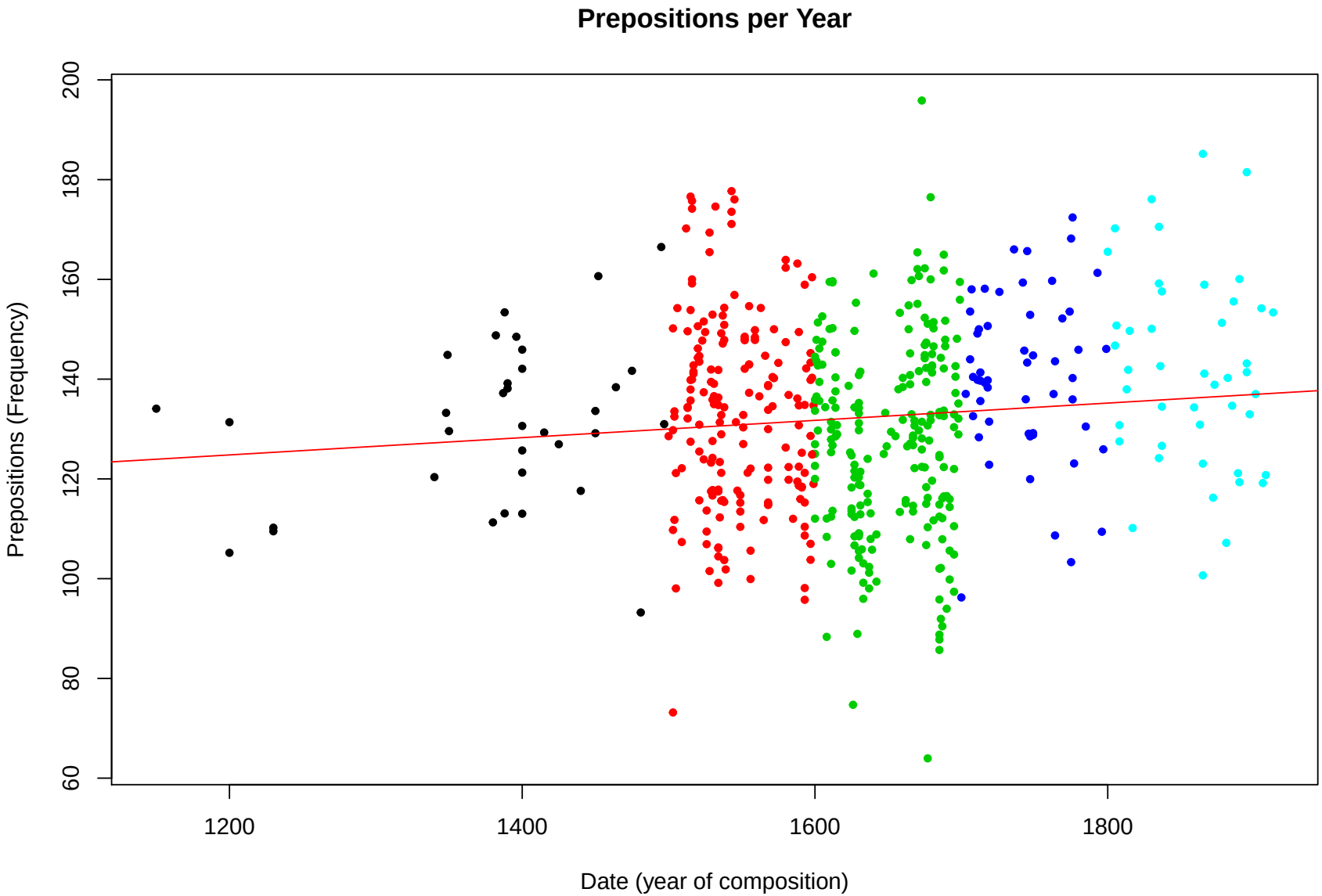
Understanding `plot()`: Scatter plot + tendency line

We may consider add a linear (regression) line to the scatter plot to see the tendency, and change the color of the points:

Syntax: `abline(lm(df_name$y ~ df_name$x), <other options>)`

```
plot(Prepositions ~ Date,    # plot Prepositions by Date
     type = "p",             # plot type p (points)
     data = pdat,            # data from pdat
     ylab = "Prepositions (Frequency)",     # add y-axis label
     xlab = "Date (year of composition)",   # add x-axis label
     main = "Prepositions per Year",        # add title
     pch = 20,               # use point symbol 20 (filled circles)
     col = DateRedux         # define colors by DateRedux
     )
abline(                                     # add a line
  lm(pdat$Prepositions ~ pdat$Date),        # draw line of linear model (lm)
  col="red"                                 # define line color as red
  )
```

Understanding `plot()`: Scatter plot + tendency line



Try it yourself

1. Use the code below to load the data set and call it `dta9_1`.

```
dta9_1 <-
  base::readRDS(url("https://slcladal.github.io/data/d03.rda","rb"))
```

2. Take a look at `dta9_1` – how many variables (columns) and observations (rows) are included?
3. Find the mean and standard deviation of `Variable1` and `Variable2`.
4. Then, create a scatter plot showing `Variable1` on the x-axis and `Variable2` on the y-axis. You may change the color of the points.
5. Finally, add a linear tendency line. By eyeballing the plot, can you determine the linear relationship between the two variables?

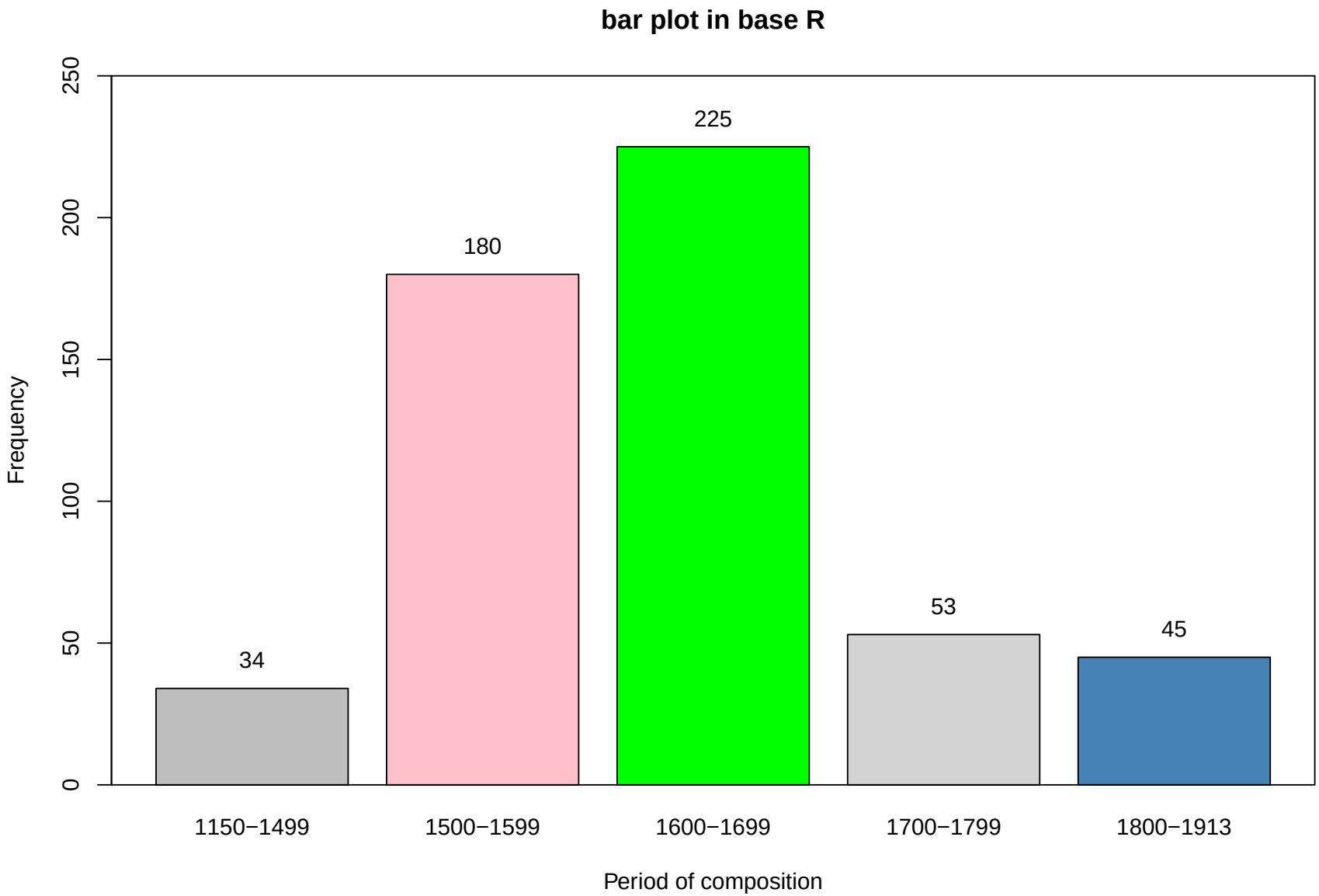
barplot(): Bar plot

Bar plot displays frequency information across categorical variable levels.

`barplot()` takes a table with frequency counts as its main argument. As before, we can also specify axes labels and a title. We specify text, grids, and boxes separately after `barplot()`.

```
barplot(table(pdat$DateRedux),          # plot Frequency by DateRedux
     ylab = "Frequency",                # add y-axis label
     xlab = "Period of composition",    # add x-axis label
     main = "bar plot in base R",       # add title
     col = c("grey", "pink", "green", "lightgrey", "steelblue"),
                                        # define color of each bar
     ylim = c(0, 250)                   # define y-axis limits
     )
text(seq(0.7, 5.5, 1.2),                # add label positions (x-axis)
     table(pdat$DateRedux)+10,          # add label positions (y-axis)
     table(pdat$DateRedux))             # add labels
box()                                   # add box
```

barplot(): Bar plot

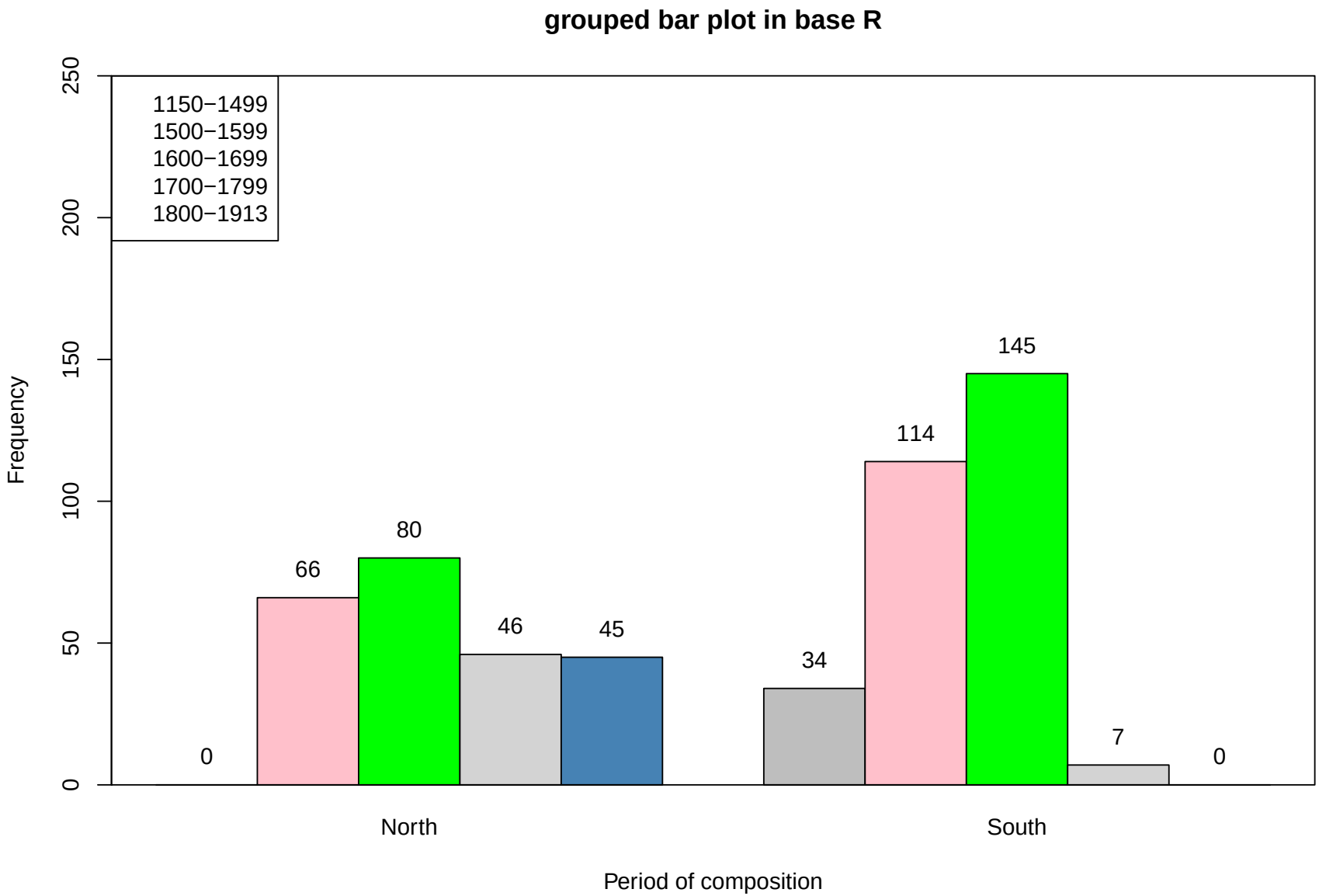


barplot(): Grouped bar plot

To create grouped bar plots, we tabulate the variables that we are interested in (e.g. Region)

```
barplot(table(pdat$DateRedux, pdat$Region),  # plot Frequency
                                             # by DateRedux and Region
        beside = T,                          # bars beside each other
        ylab = "Frequency",                  # add y-axis label
        xlab = "Period of composition",      # add x-axis label
        main = "grouped bar plot in base R", # add title
        col = c("grey", "pink", "green", "lightgrey", "steelblue"),
                                             # define color of each bar
        ylim = c(0, 250)                     # define y-axis limits
        )
text(c(seq(1.5, 5.5, 1.0), seq(7.5, 11.5, 1.0)), # add label positions
     table(pdat$DateRedux, pdat$Region)+10,      # add label positions
     table(pdat$DateRedux, pdat$Region))         # add labels
legend("topleft", names(table(pdat$DateRedux)))  # add legend
box()                                            # add box
```

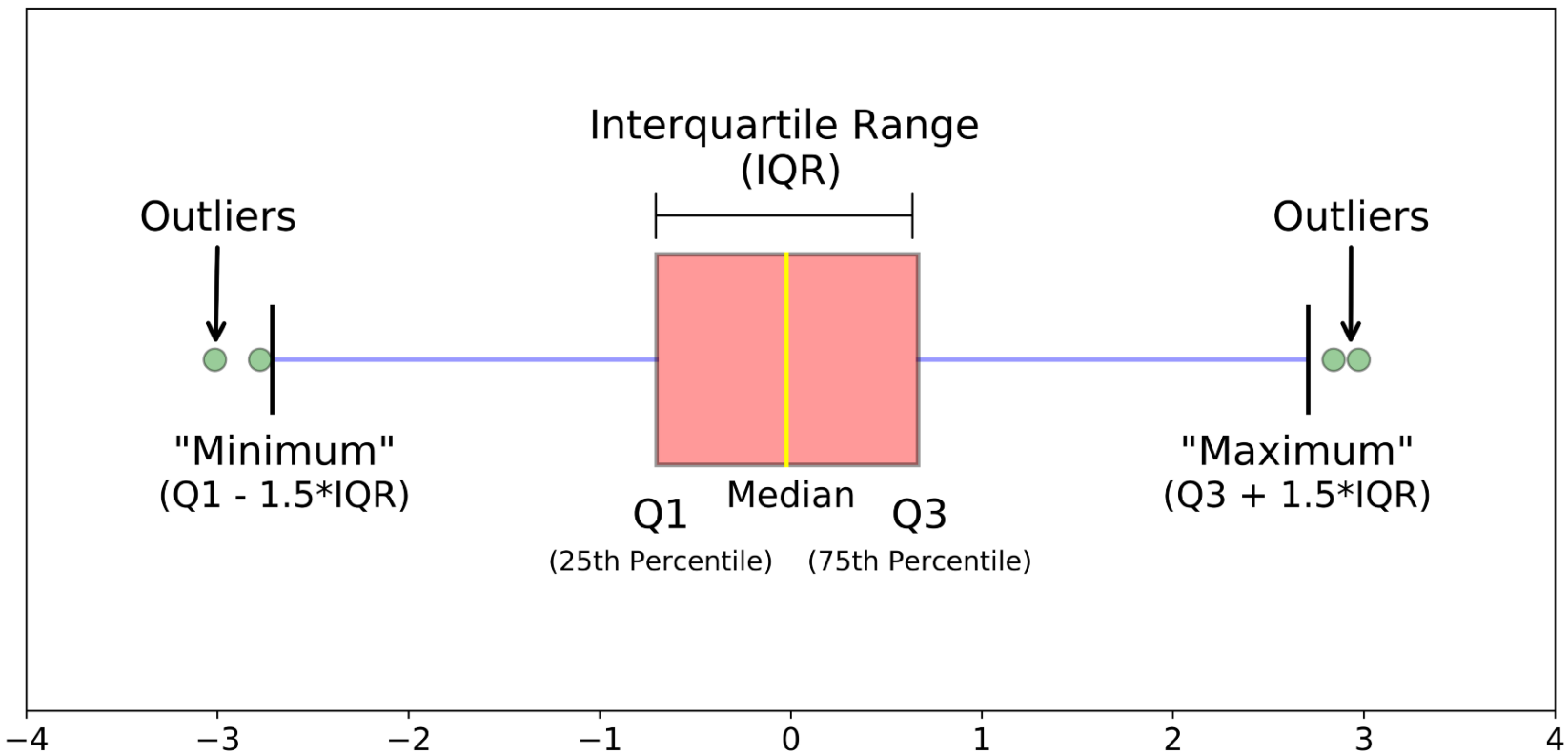
barplot(): Grouped bar plot



boxplot(): box plot

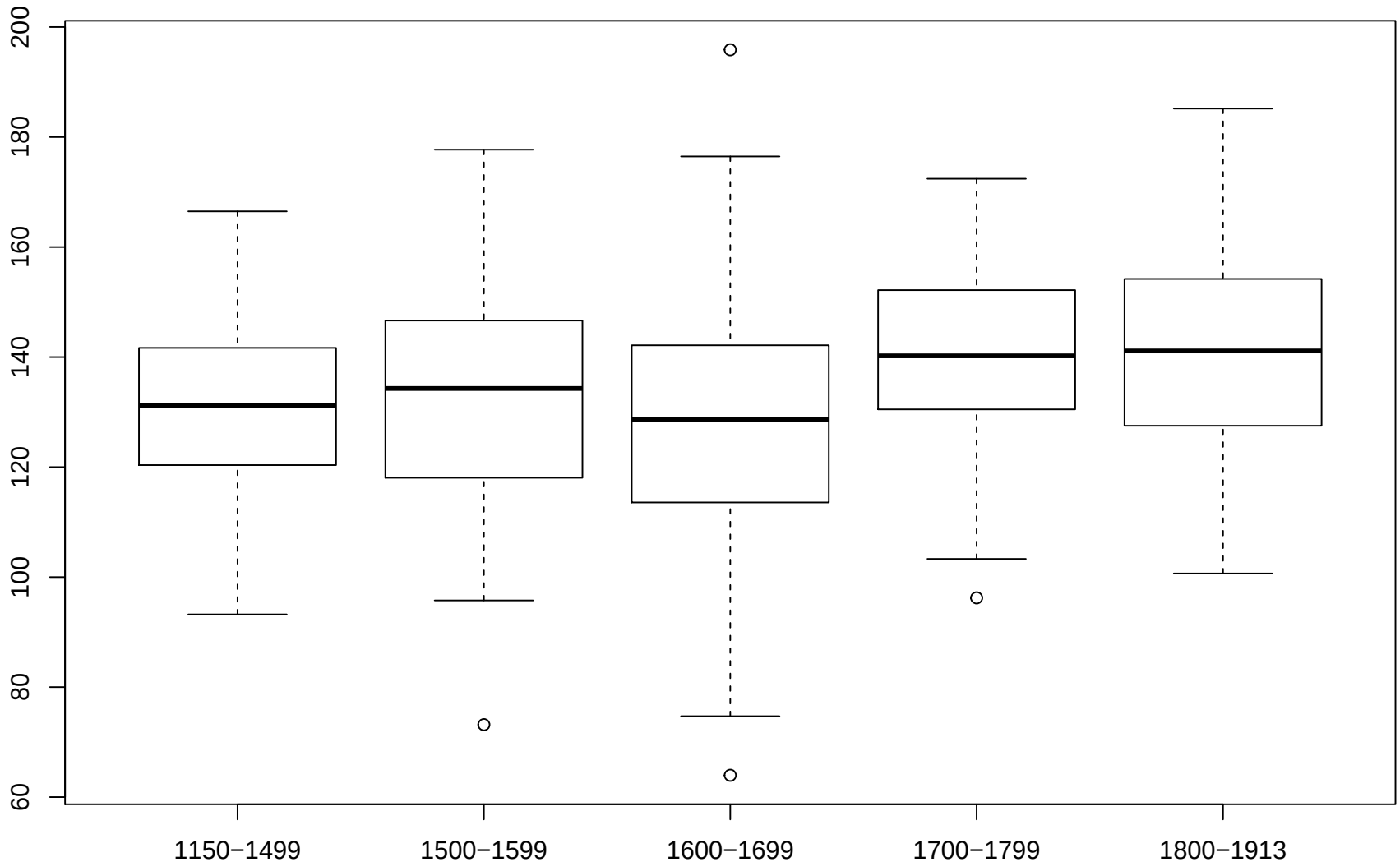
Boxplots show the relationships between categorical and numeric variables.

They are very useful because they not only provide measures of central tendency (e.g. median) but also offer info about the distribution (e.g. 1st and 3rd quartile, min, max) of the data.



boxplot(): box plot

```
boxplot(Prepositions ~ DateRedux, data = pdat)
```

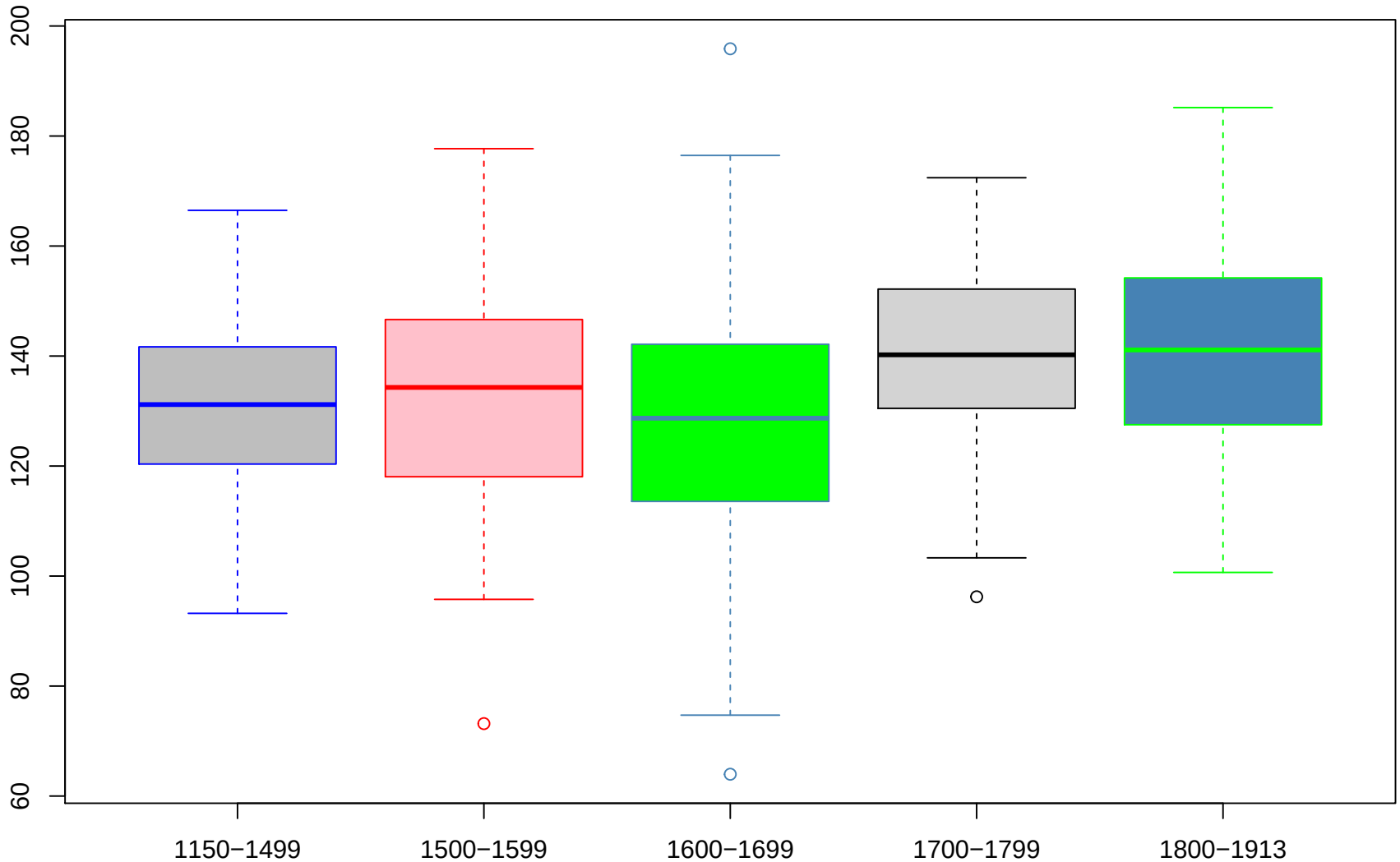


boxplot(): box plot

We can change the color of the borders and/or fill colors in each box:

```
boxplot(Prepositions ~ DateRedux, data = pdat,
        border = c("blue", "red", "steelblue", "black", "green"),
        col = c("grey", "pink", "green", "lightgrey", "steelblue"))
```

boxplot(): box plot



Try it yourself: Plot 1

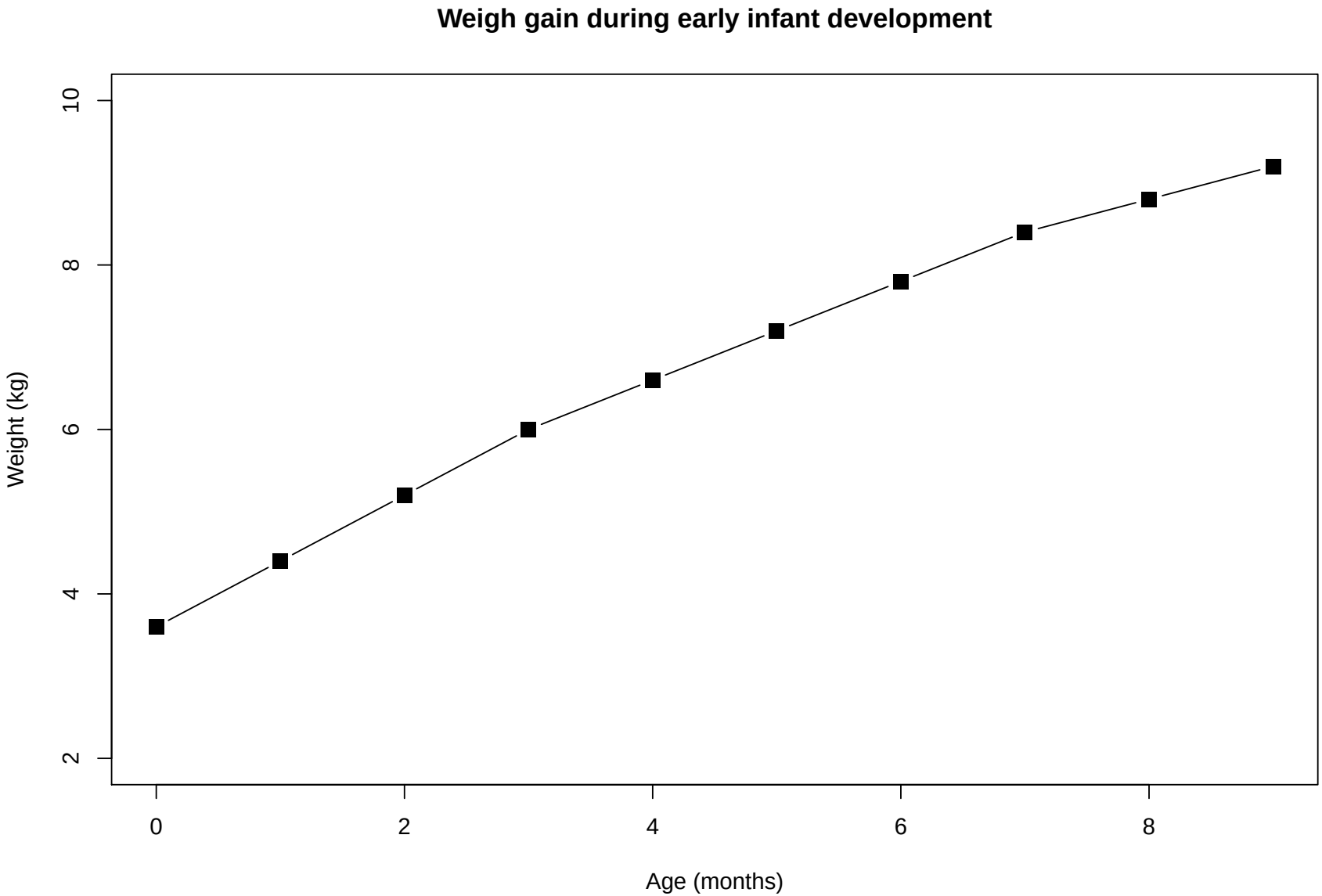
1. First, load the dataset `weight_chart.txt` and give it a name (e.g. weight_chart). Don't forget to change the working directory if needed! Then, take a look at the dataset.

```
# read the space separated dataframe
weight_chart <- read.delim("weight_chart.txt")
weight_chart
```

2. Then, try to replicate the plot in the next slide by modifying the following code. Hint: When choosing type, use `"b"` (which means *both* line and points).

```
plot(??? ~ ???,                  # fill in appropriate X and Y
     type = "b",                 # plot type "b" (both line and points)
     pch = 15,                   # filled square
     cex = 1.5,                  # size of pch symbols
     data = ???,                 # data source
     ylim = c(?,?),              # fill in the y-axis limits
     ylab = "???",
     xlab = "???",
     main = "???"
     )
```

Try it yourself: Plot 1



Try it yourself: Plot 2

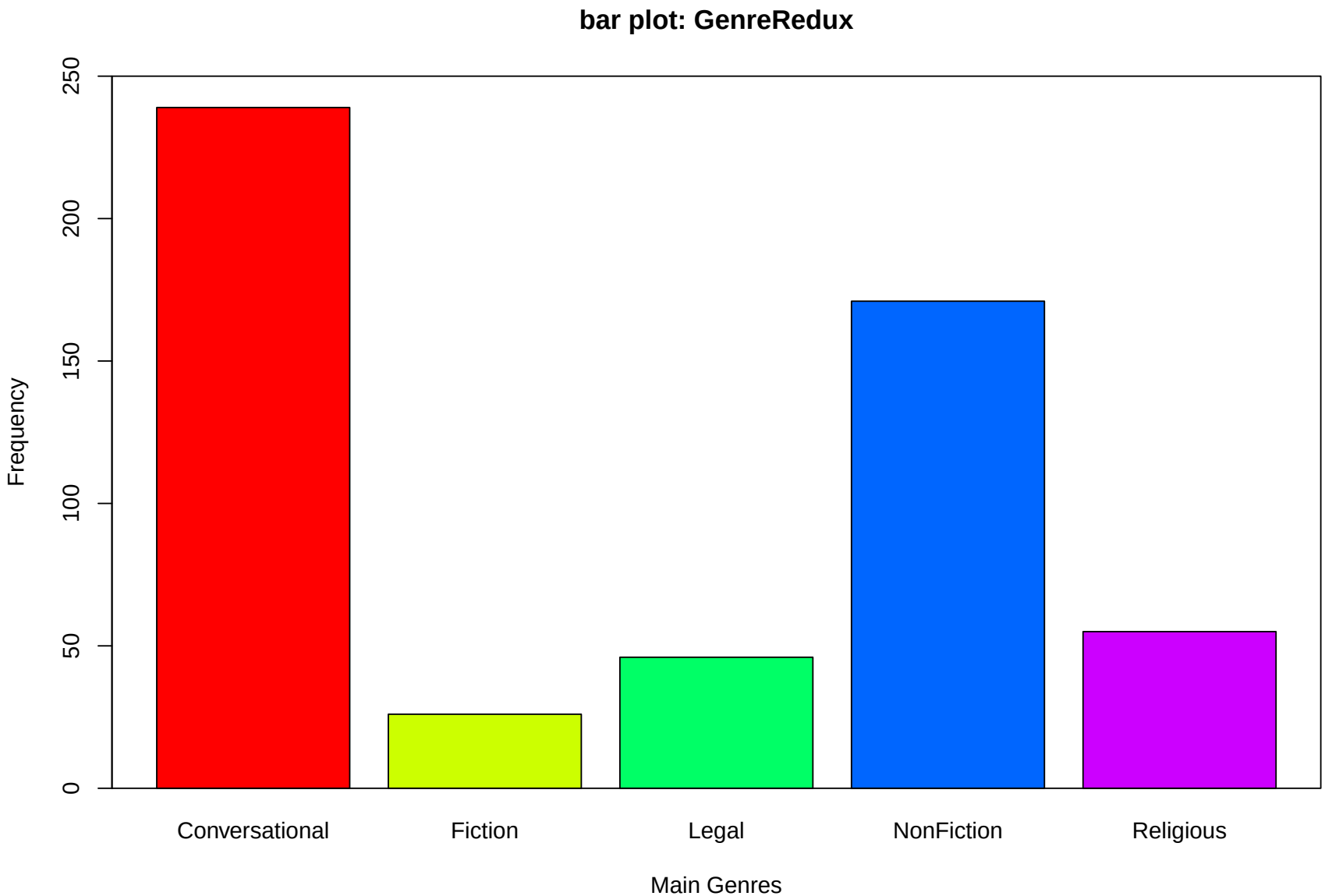
1. First, load the dataset and give it a name (e.g. pdat).

```
# load data and call it 'pdat'
pdat <- base::readRDS(url("https://slcladal.github.io/data/pvd.rda",
                          "rb"))
```

2. Then, try to replicate the plot in the next slide by modifying the following code.

```
barplot(table("_fill in_"),
     ylab = "[_fill in_]",
     xlab = "[_fill in_]",
     main = "[_fill in_]",
     ylim = c(?, ?),      # fill in the y-axis limits
     col = rainbow(5)     # a built-in color palettes which can be used
                          # to quickly generate color vectors
     )
box()
```

Try it yourself: Plot 2

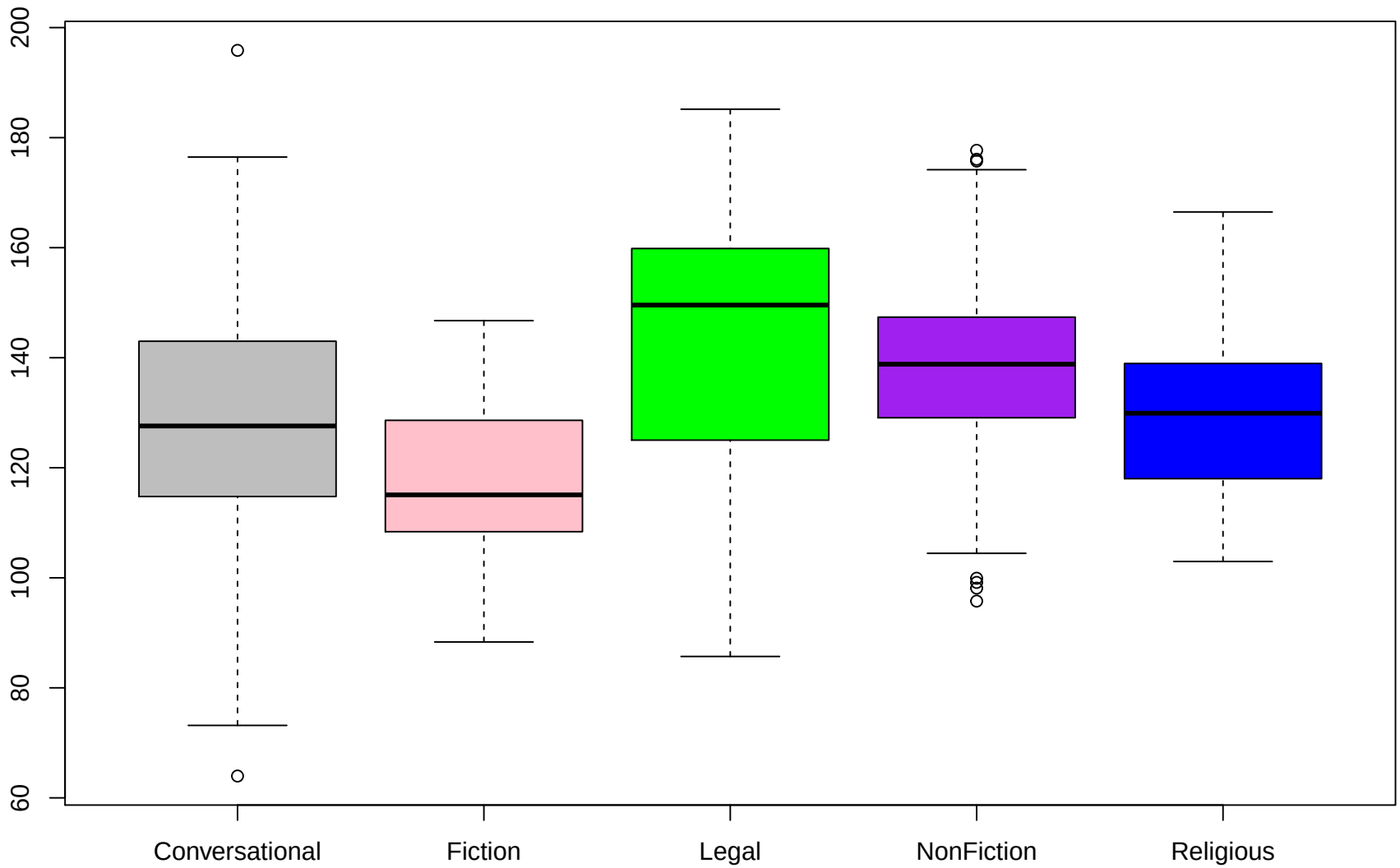


Try it yourself: Plot 3

1. Use the dataset `pdat` and try to replicate the plot in the next slide by modifying the following code. For the color of each box, you can choose whichever you want.

```
boxplot(Prepositions ~ ???, data = ???,
        col = ???
```

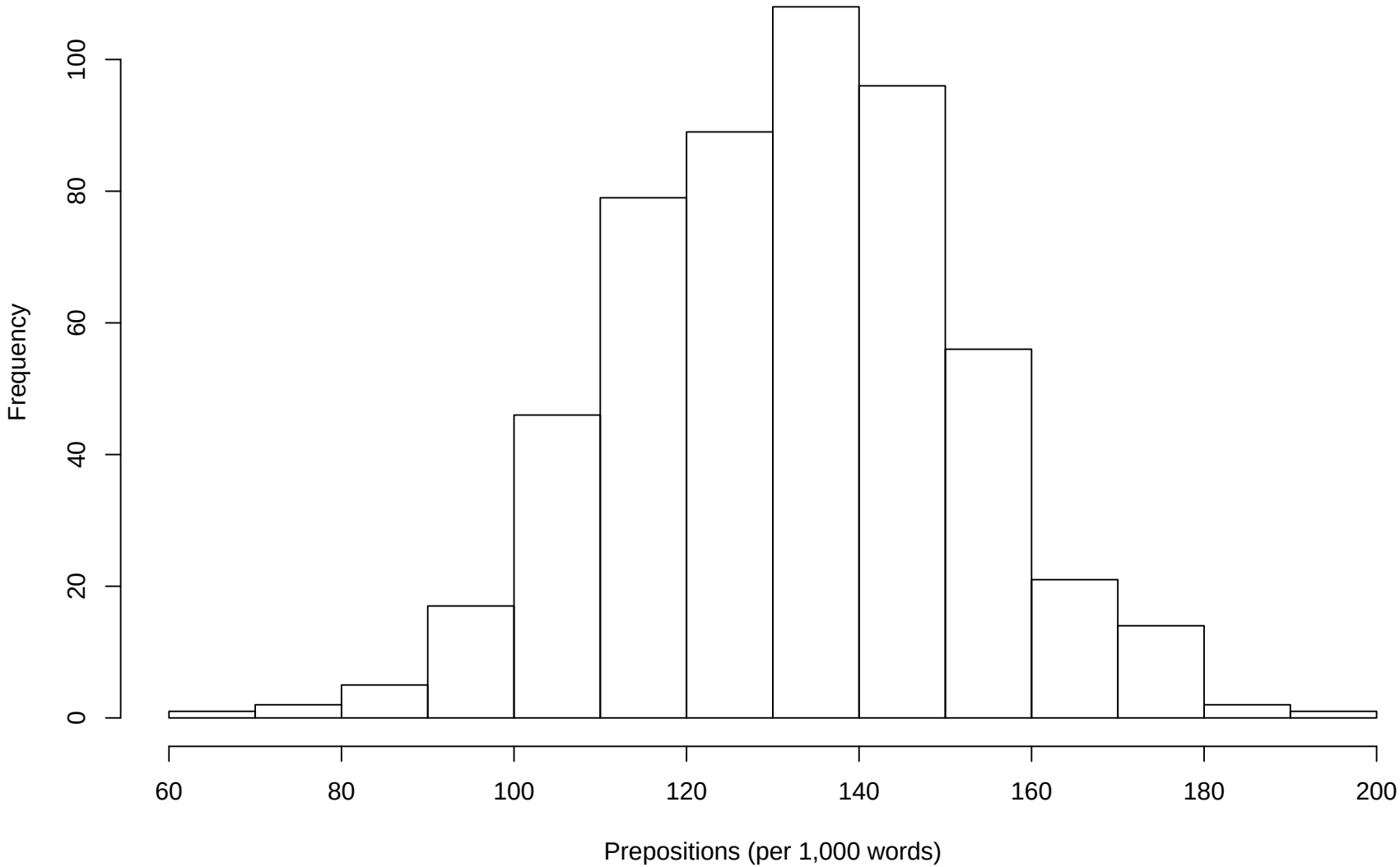
Try it yourself: Plot 3



Appendix: Histogram

It summarizes numeric variables by showing their distribution across bins.

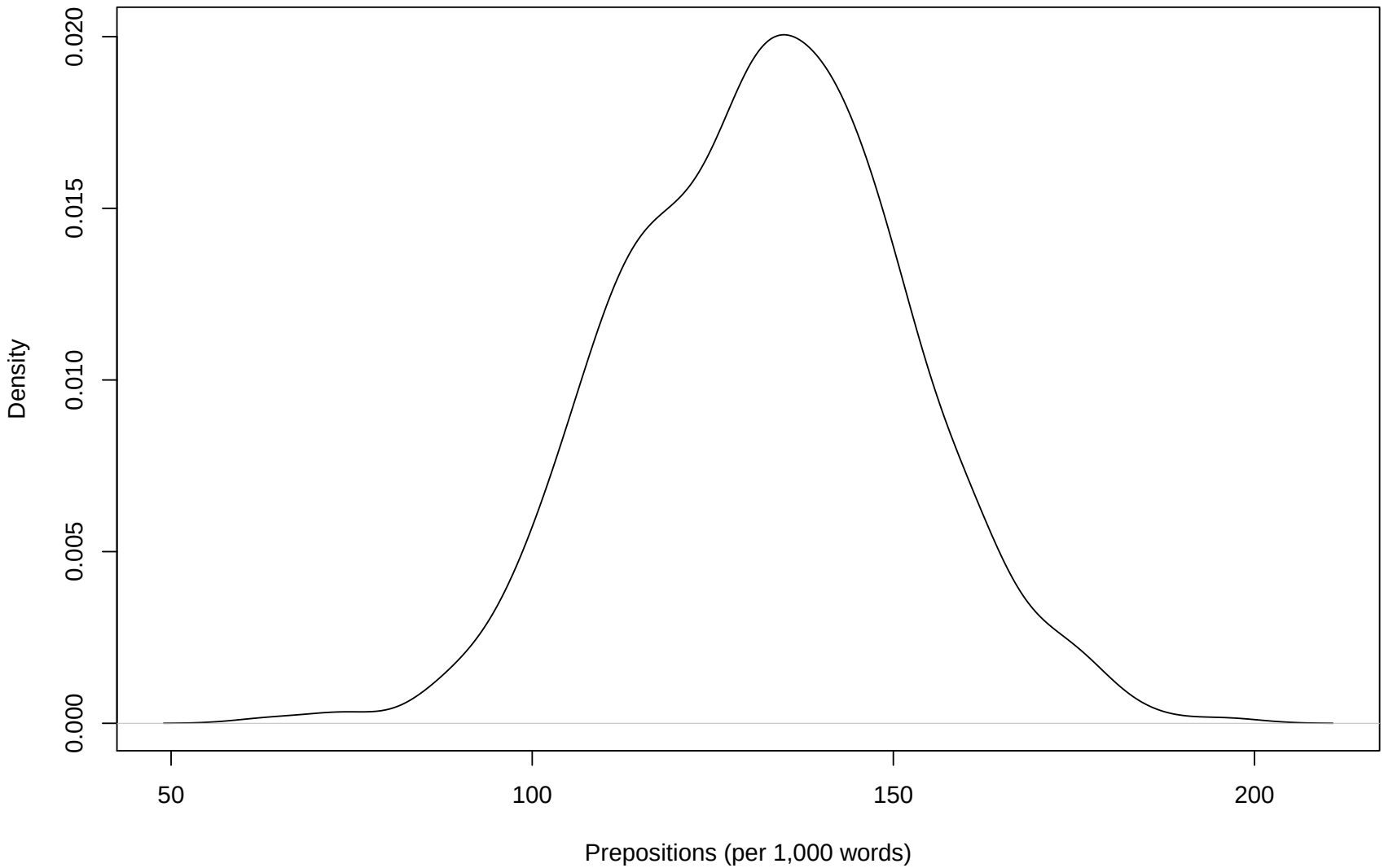
```
hist(pdat$Prepositions,
     xlab = "Prepositions (per 1,000 words)", main = "")
```



Appendix: Density Plot

It is a smoothed version of the histogram and is used in the same concept.

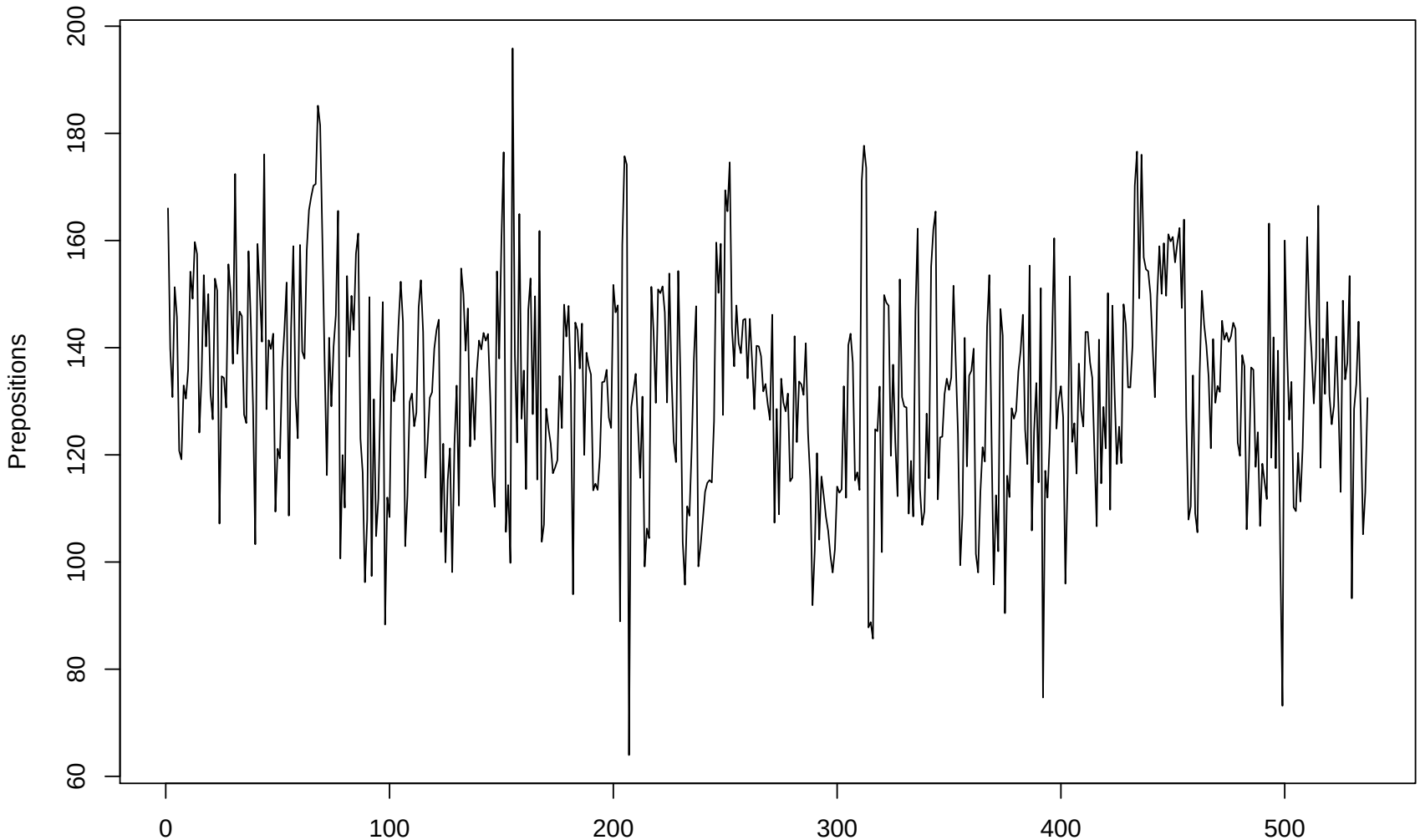
```
plot(density(pdat$Prepositions),
     xlab = "Prepositions (per 1,000 words)", main = "")
```



Appendix: Line Plot

It is particularly useful in time series or finance.

```
plot(pdat$Prepositions,
     type = "l",
     ylab = "Prepositions", main = "")
```



Recap

- Visualizing our data can help lead to powerful insights between variable relationships
  - Making quick plots helps us understand data and makes us aware of data issues
- We've learned how to make commonly used plots, including:
  - scatter plots (with auxiliary lines)
  - bar plots
  - box plots
- There are many ways you can visualize your data
  - We can even use `ggplot()` to generate similar plots more conveniently! (more details in Lecture 10)