Simulating Discrete random variables

- X - random variable — $F(x) = \mathbb{P}(X \le x)$

$$Y = F(X) \in [0,1]$$

$$\mathbb{P}(Y \le y) = \mathbb{P}(F(X) \le y)$$

$$\overset{(Ex.)}{=} \mathbb{P}(X \le F^{-1}(y))$$

$$= F(F^{-1}(y))$$

$$= y$$

$0 \le y \le 1$

- $Y \sim \text{Uniform}[0,1]$

Simulating Uniform Random Variables

```
> set.seed(2021)
> runif(10)
 [1] 0.4512674 0.7837798 0.7096822 0.3817443 0.6363238
 [6] 0.7013460 0.6404389 0.2666797 0.8154215 0.9829869
> set.seed(2021)
> runif(10)
 [1] 0.4512674 0.7837798 0.7096822 0.3817443 0.6363238
 [6] 0.7013460 0.6404389 0.2666797 0.8154215 0.9829869
> runif(10)
 [1] 0.02726706 0.83749040 0.60324073 0.56745337 0.82005281
 [6] 0.25157128 0.50549403 0.86753810 0.95818157 0.54569770
```

Pseudo Random Numbers

- runif(1)

generates a sample from U(0,1)

Either from $\mathbb{R}$

or

otherwise

Generate

$U \sim U(0,1)$

Generate samples from Discrete Distributions

- $X \overset{d}{=} \text{Bernoulli}(p)$, $0 < p < 1$ $\begin{cases} \text{Range}(X) = \{0,1\} \\ \mathbb{P}(X=1) = p. \end{cases}$

(i) Generate $u \sim U(0,1)$ from R.

(ii) Define $X := \begin{cases} 1 & \text{if } u < p \\ 0 & \text{if } u \geq p \end{cases}$

$\mathbb{P}(X=1) = \mathbb{P}(u < p) = p$!

Generate samples from Discrete Distributions

$X \overset{d}{=} \text{Binomial}(n, p)$ $\rightsquigarrow$ $X \overset{d}{=} \sum_{i=1}^{n} X_i$

$X : \Omega \to S = \{0, 1, 2, \ldots n\}$

$X_i \overset{d}{=} \text{Bernoulli}(p)$

(independent)

(i) R - generate $u_1, u_2, \ldots, u_n$ independent

$U(0,1)$ samples

(ii) $X := \#\{i : 1 \le i \le n \ \& \ u_i < p\}$

Ex: $X \overset{d}{=} \text{Binomial}(n, p)$

Generate samples from Discrete Distributions

$(\Omega, \mathcal{F}, \mathbb{P})$ – Probability space

$X: \Omega \to S$ (S – Countable set)

- Range $(X) = S = \{x_i\}_{i=1}^{N}$ $\quad N \in \mathbb{N} \cup \{\infty\}$

$\mathbb{P}(X = x_i) = p_i$ $\quad 0 \le p_i \le 1$ & $\sum_{i=1}^{N} p_i = 1$ $\quad \to p_i < p_{i+1}$

(i) Generate $u \overset{d}{=} U(0,1)$ from R.

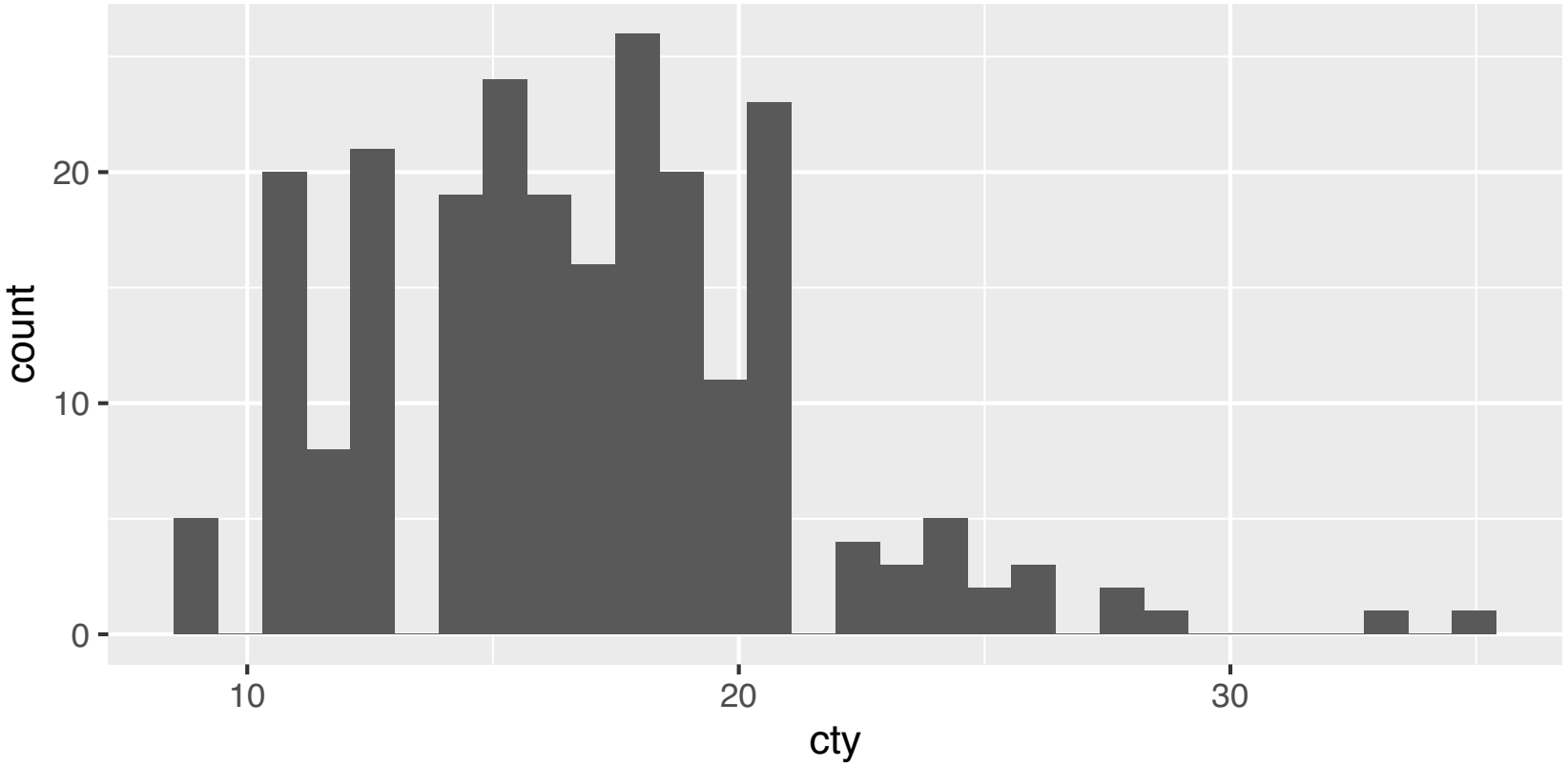
(ii) $$X = \begin{cases} x_1 & \text{if } u < p_1 \\ x_k & \text{if } \sum_{i=1}^{k-1} p_i < u < \sum_{i=1}^{k-1} p_i + p_k \end{cases}$$

Histogram

- First specifies a sequence of points, called breaks.
- It counts the number of observation between the breaks, called bins.
- Places a bar in each bin with
  - base being the length of the bin and
  - height being either the frequency or proportion of observations in the bin.

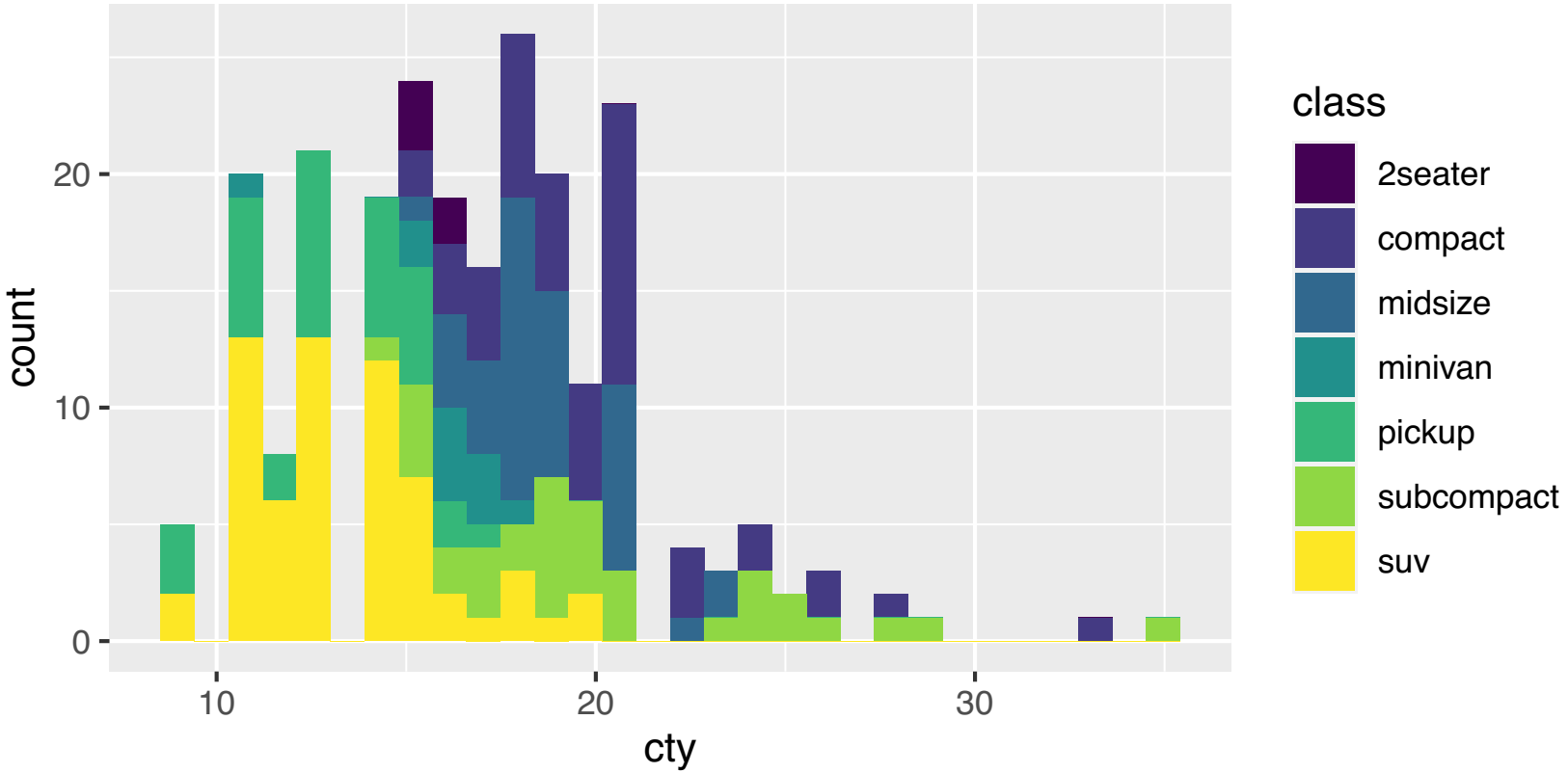
Histogram

```
> library(tidyverse)
> ggplot(data = mpg,  mapping = aes(x = cty)) +
+   geom_histogram()
```



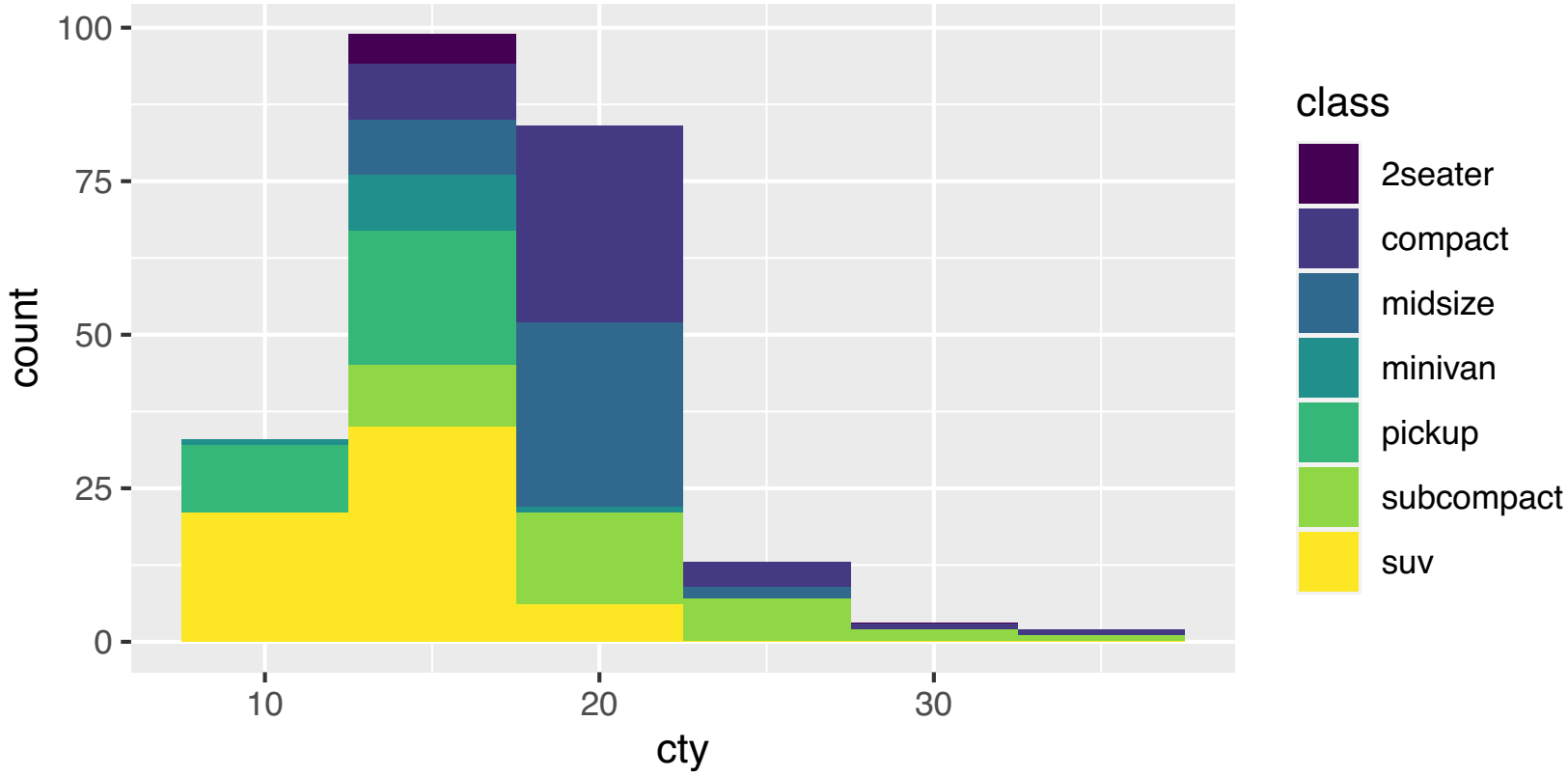
Histogram

```
> library(viridis)
> ggplot(data = mpg,  mapping = aes(x = cty,fill=class)) +
+   geom_histogram() +
+   scale_fill_viridis(discrete = TRUE)
```



Histogram

```
> ggplot(data = mpg,  mapping = aes(x = cty,fill=class)) +
+   geom_histogram(binwidth = 5) +
+   scale_fill_viridis(discrete = TRUE)
```



R- Categorical Data –Factors and Levels

- Categorical Variables- Qualitative variables, i.e. those which cannot meaningfully be expressed in numbers. E.g.: Clothes Colour.
  - `Base`-R deals with them through the use of `factors`.
  - `factors` are useful in Statistical Modeling and Plotting data.
- Packages - `tidyverse`- `dplyr`, `tidyr`, `forcats`, `readr` help in dealing with factors.
- Many get frustrated with factors and use these to avoid them!

- storing data as factors will ensure modeling / plot functions treat such data correctly.

- factor() – function that is used to create a factor.

– Used to work with Categorical data

R- Categorical Data – Factors and Levels

```
> xc = c("June", "July", "August", "September",
+        "August", "July","July","August" )
> factorxc  = factor(xc)
> factorxc
[1] June      July      August    September August
[6] July      July      August
Levels: August July June September
```

- x - vector of data.
- factor(x) in R is stored as a vector of integers, but correspond to a character string for display.
- Levels - the unique set of values taken by as.chararcter(x)

```
> xn = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
> factorxn  = factor(xn)
> factorxn
 [1] 1 2 2 3 1 2 3 3 1 2 3 3 1
Levels: 1 2 3
```

factor(x)

- store as vector of integers
- displayed as characters
- Levels :- assigned by default are those given by as.character(x)

- store both

xc – character data

xn – numeric

as factors.

- factor levels are always characters

R- Categorical Data – Factors and Levels

```
> xn = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
> factorxn  = factor(xn)
> factorxn
 [1] 1 2 2 3 1 2 3 3 1 2 3 3 1
Levels: 1 2 3

> mean(xn)
[1] 2.076923
> mean(factorxn)
[1] NA
> mean(as.numeric(levels(factorxn)[factorxn]))
[1] 2.076923
```

Doing Computations with numeric

{ argument is not numeric
so will return NA

- use levels function to convert them to original numeric value

R- Categorical Data– Factors and Levels

Specify ordering of Levels

```
> table(factorxc)
factorxc
   August      July      June September
        3         3         1         1
```

```
> months = factor(xc,levels=c("Garbage", "January","February",
+     "March","April","May","June","July","August",
+     "September","October","November","December"),
+     ordered=TRUE)
> months
[1] June      July      August    September August
[6] July      July      August
13 Levels: Garbage < January < February < ... < December
> months[3] < months[4]
[1] TRUE
```

- in our specification levels contain elements not present in xc

Previous - slide - defined

```
> xc = c("June", "July", "August", "September",
+        "August", "July","July","August" )
> factorxc  = factor(xc)
```

```
> factorxc
[1] June      July      August    September August
[6] July      July      August
Levels: August July June September
```

- ordering is w.r.t to as.character(.) and not related to order in months.

- Specify Levels needed
- order can be specified.

R- Categorical Data– Factors and Levels

```
> table(months)

months
  Garbage   January  February     March     April       May
        0         0         0         0         0         0
     June      July    August September   October  November
        1         3         3         1         0         0
 December
        0
```

```
> table(factor(months))

     June      July    August September
        1         3         3         1
```

Previous slide

```
> months = factor(xc,levels=c("Garbage", "January","February"
+     "March","April","May","June","July","August",
+     "September","October","November","December"),
+     ordered=TRUE)
```

- When factor is created, all its levels are stored along with the factor
- Display. All levels specified and counts
- occur when subsetting a factor.

- factor (months)
  - retains only the levels present
  - maintains the ordering.

R- Categorical Data– cut

```
> x = round(1000*runif(10))
> x
 [1] 707  26 211 929   2 549 225 135 191 383

> xfactor= cut(x,4)
> xfactor
 [1] (697,930]  (1.07,234] (1.07,234] (697,930]  (1.07,234]
 [6] (466,697]  (1.07,234] (1.07,234] (1.07,234] (234,466]
Levels: (1.07,234] (234,466] (466,697] (697,930]
> table(xfactor)
xfactor
(1.07,234]  (234,466]  (466,697]  (697,930]
         6          1          1          2
```

- cut - function is used to convert a numeric variable into a factor.
- cut(____ , ____)
  - ____ ← numeric data
  - ____ ← breaks

how range of numbers will be converted to factor values

Exercise:

labels - Specify levels of factors

```
cut (x, 3, labels = c("L","M","H"))
```

R- Categorical Data– cut

```
> xpfactor= cut(x,pretty(x,4))
> xpfactor
 [1] (600,800]   (0,200]     (200,400]   (800,1e+03]
 [5] (0,200]     (400,600]   (200,400]   (0,200]
 [9] (0,200]     (200,400]
5 Levels: (0,200] (200,400] (400,600] ... (800,1e+03]
> table(xpfactor)
xpfactor
   (0,200]   (200,400]   (400,600]   (600,800] (800,1e+03]
         4           3           1           1           1
```

- nicer set of labels
- but may provide more levels than specified.

```
> xqfactor= cut(x,quantile(x, probs=seq(0,1,0.25)))
> table(xqfactor)
xqfactor
  (2,149] (149,218] (218,508] (508,929]
        2         2         2         3
```

- produce factors based on percentiles of your data.
- insure "equal" number of observations in each level

R – Factors and Levels

```
> everyday = seq(from=as.Date('2021-1-1'),
+                to=as.Date('2021-12-31'), by='day')
> cmonth = format(everyday,'%b')
> head(cmonth,3)
[1] "Jan" "Jan" "Jan"
> df= as.data.frame(table(cmonth))
> names(df)=c("Month", "Freq")
> df
   Month Freq
1    Apr   30
2    Aug   31
3    Dec   31
4    Feb   28
5    Jan   31
6    Jul   31
7    Jun   30
8    Mar   31
9    May   31
10   Nov   30
11   Oct   31
12   Sep   30
```

- Create factors from dates/times
  - strptime
  - strftime

  } extract information

- extract month from each day
- table – tabulates values in each month
- ordering is alphabetical

R – Factors and Levels

```
> everyday = seq(from=as.Date('2021-1-1'),
+                to=as.Date('2021-12-31'), by='day')
> cmonth = format(everyday,'%b')
> months = factor(cmonth,levels=unique(cmonth),ordered=TRUE)
> head(months,3)
[1] Jan Jan Jan
12 Levels: Jan < Feb < Mar < Apr < May < Jun < ... < Dec
> df2= as.data.frame(table(months))
> names(df2)= c("Month", "Freq")
> df2
   Month Freq
1    Jan   31
2    Feb   28
3    Mar   31
4    Apr   30
5    May   31
6    Jun   30
7    Jul   31
8    Aug   31
9    Sep   30
10   Oct   31
11   Nov   30
12   Dec   31
```

format() : can be used to extract months using %b

unique() :- returns the unique values in the order that they are encountered

- stored months as factors
- specified levels & ordering using the unique()

R – Factors and Levels

```
> decdf= read.csv(file="  , Master.csv", header=TRUE)
> decdf$Month= months(as.Date(decdf$MB.Date))
> data = as.data.frame(table(decdf$Month))
> names(data) = c("Month", "val")
> data
```

	Month	val
1	April	2974
2	August	4108
3	December	430
4	February	483
5	January	843
6	July	3561
7	June	6049
8	March	239
9	May	13599
10	November	737
11	October	2593
12	September	3643

Master.csv

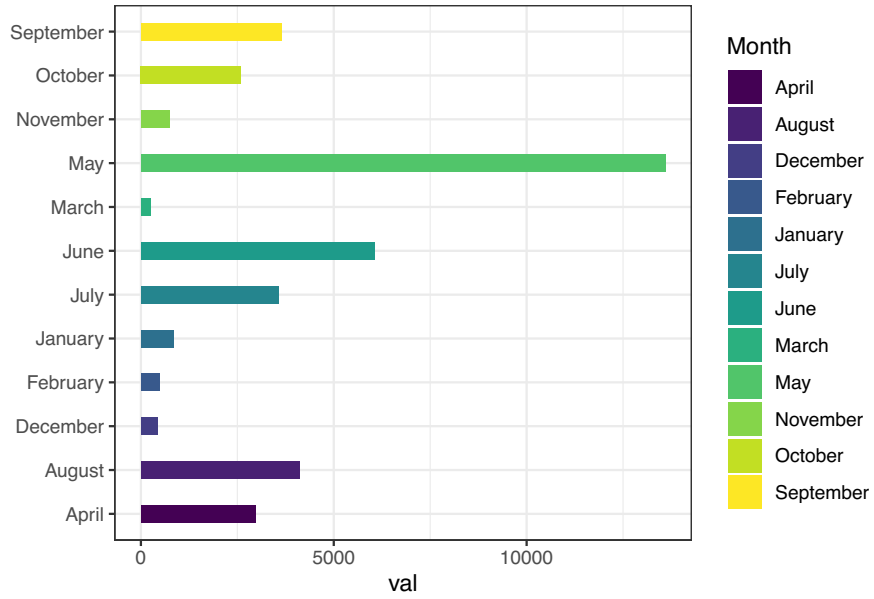
- Deceased data from KA – COVID-19 bulletins.

- Use months( ) to extract month of reporting date

- Use table() to compute reported case across months

R – Factors and Levels

```
> library(tidyverse)
> ggplot(data=data, aes(x=Month, y=val,  fill=Month)) +
+   geom_bar(stat="identity", width=.4) +
+   coord_flip() +
+     scale_fill_viridis_d()+
+   xlab("") +
+   theme_bw()
```



Previous slide

```
> decdf= read.csv(file="  , Master.csv", header=TRUE)
> decdf$Month= months(as.Date(decdf$MB.Date))
> data = as.data.frame(table(decdf$Month))
> names(data) = c("Month", "val")
> data
```

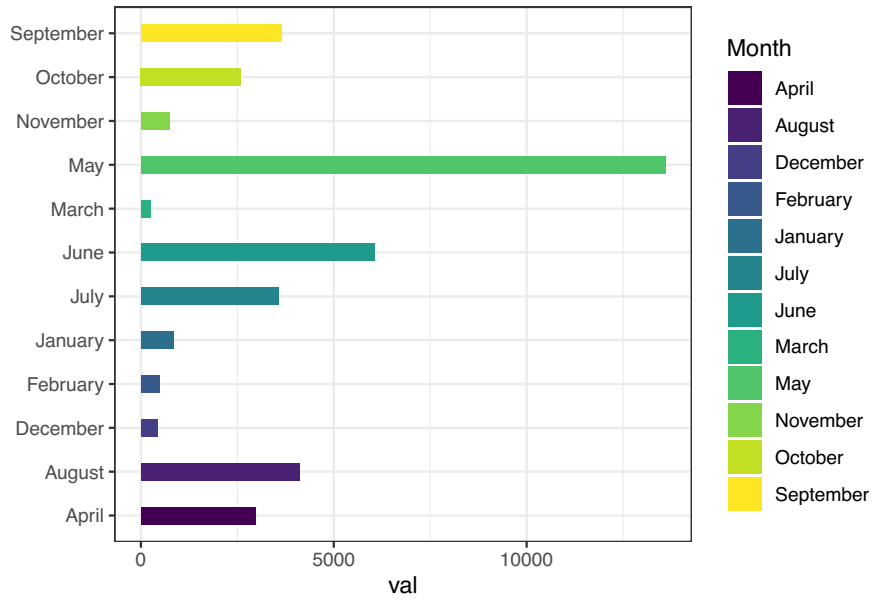
- Notice :-

- ggplot uses ordering alphabetically

- treats Month as factor and default level ordering

R – Factors and Levels

```
> library(tidyverse)
> data  = arrange(data, val)
> ggplot(data=data, aes(x=Month, y=val,  fill=Month)) +
+   geom_bar(stat="identity", width=.4) +
+   scale_fill_viridis_d()+
+   coord_flip() +
+   xlab("") +
+   theme_bw()
```



Note: ggplot() takes into account ordering from the factor given by its levels

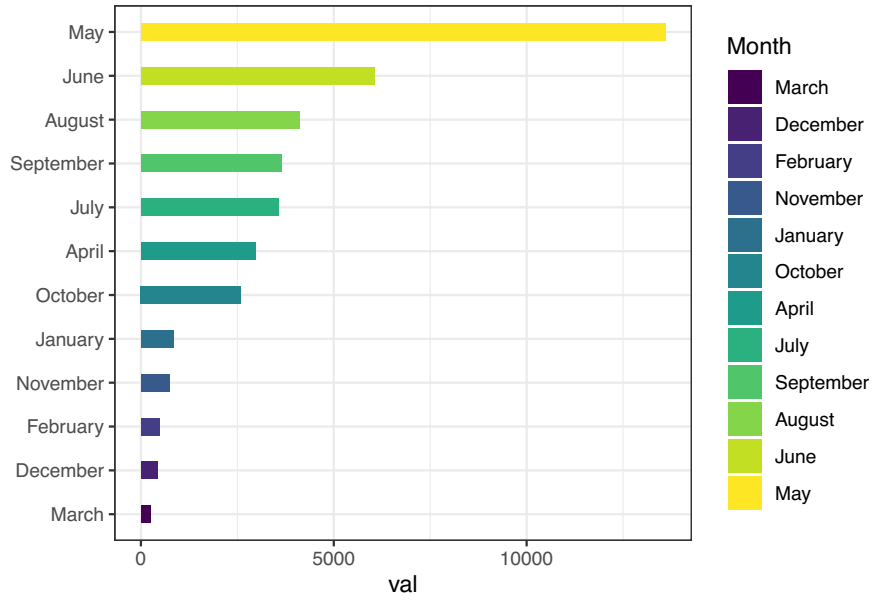
and

NOT

as you see it in the data frame

R – Factors and Levels

```
> library(tidyverse)
> data  = arrange(data, val)
> data$Month= factor(data$Month, levels=data$Month)
> ggplot(data=data, aes(x=Month, y=val, fill=Month)) +
+   geom_bar(stat="identity", width=.4) +
+   coord_flip() +
+   scale_fill_viridis_d()+
+   xlab("") +
+   theme_bw()
```



Re order the levels according to the values

- ggplot() will then oblige

R - With Dplyr — Order months according to reported deaths

```
> decdf= read.csv(file="../Master.csv", header=TRUE)
> names(decdf) = c("Sno","District","Pid","Age","Sex", "Description","Symptoms",
+                  "CMB", "DOA","DOD","MB.Date","Notes")
> decdf$Month= months(as.Date(decdf$MB.Date))
> ggplot(decdf, aes(x = fct_infreq(Month), fill=cut(Age,pretty(Age,4)))) +labs(fill='Age')+
+ geom_bar(stat="count", width=.4) + scale_fill_viridis_d()+
+   coord_flip()+ theme_bw() +ggtitle("Deceased Month Count- Age Stacked")+ ylab("Count of Deceased") +
+   xlab("Month") + theme_bw()+ theme(plot.title = element_text( color="#e95462",size=14, face="bold",hjust = 0.5))+
+   theme(axis.title.x = element_text( color="#e95462", size=14, vjust = 0.5,face="bold"),
+         axis.title.y = element_text(color="#e95462", size=14, face="bold") )
```

Use forcats package
fct_infreq

cut - add Age bin layer.

format labels

