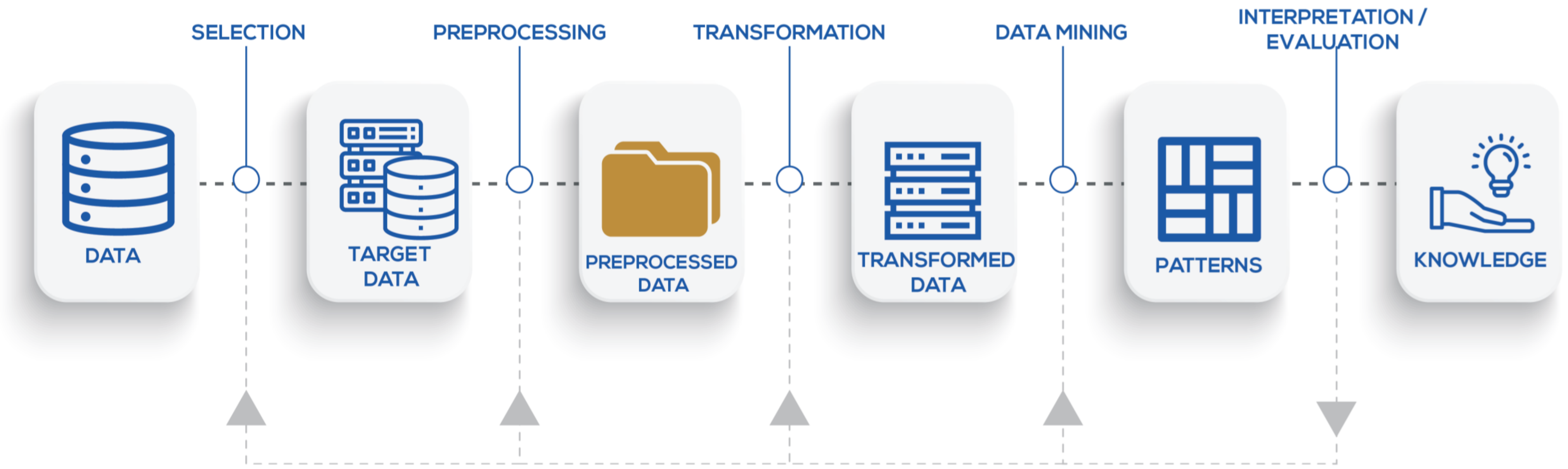
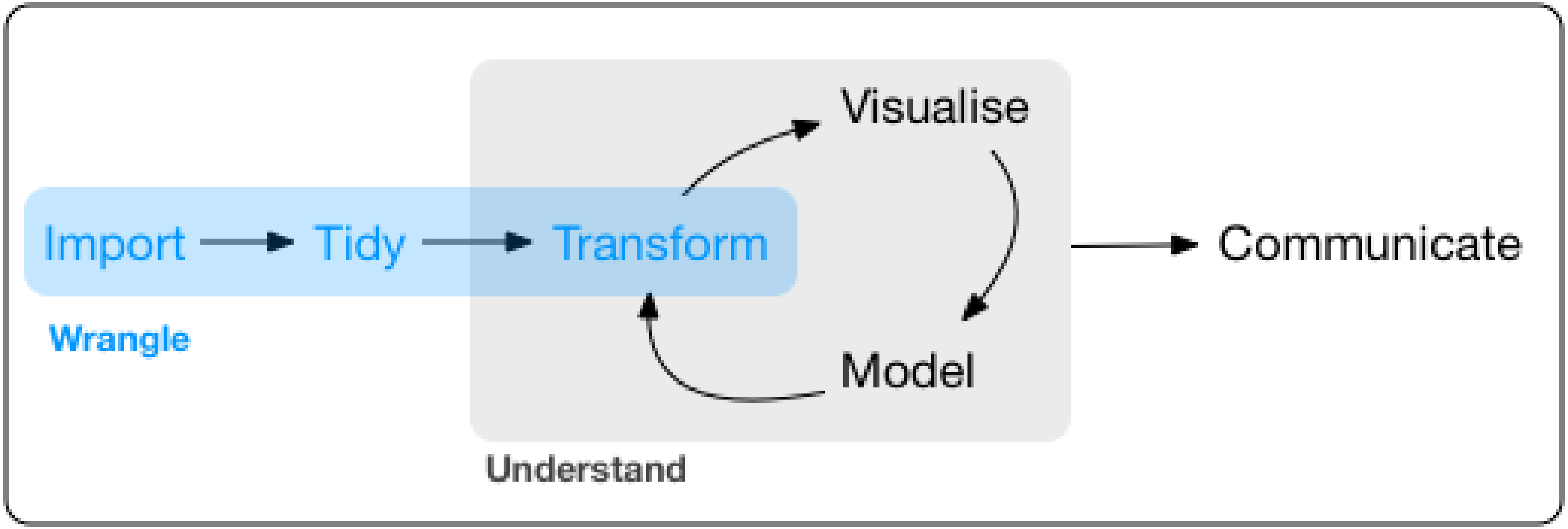
Module 4: R for Data Science

BeVera

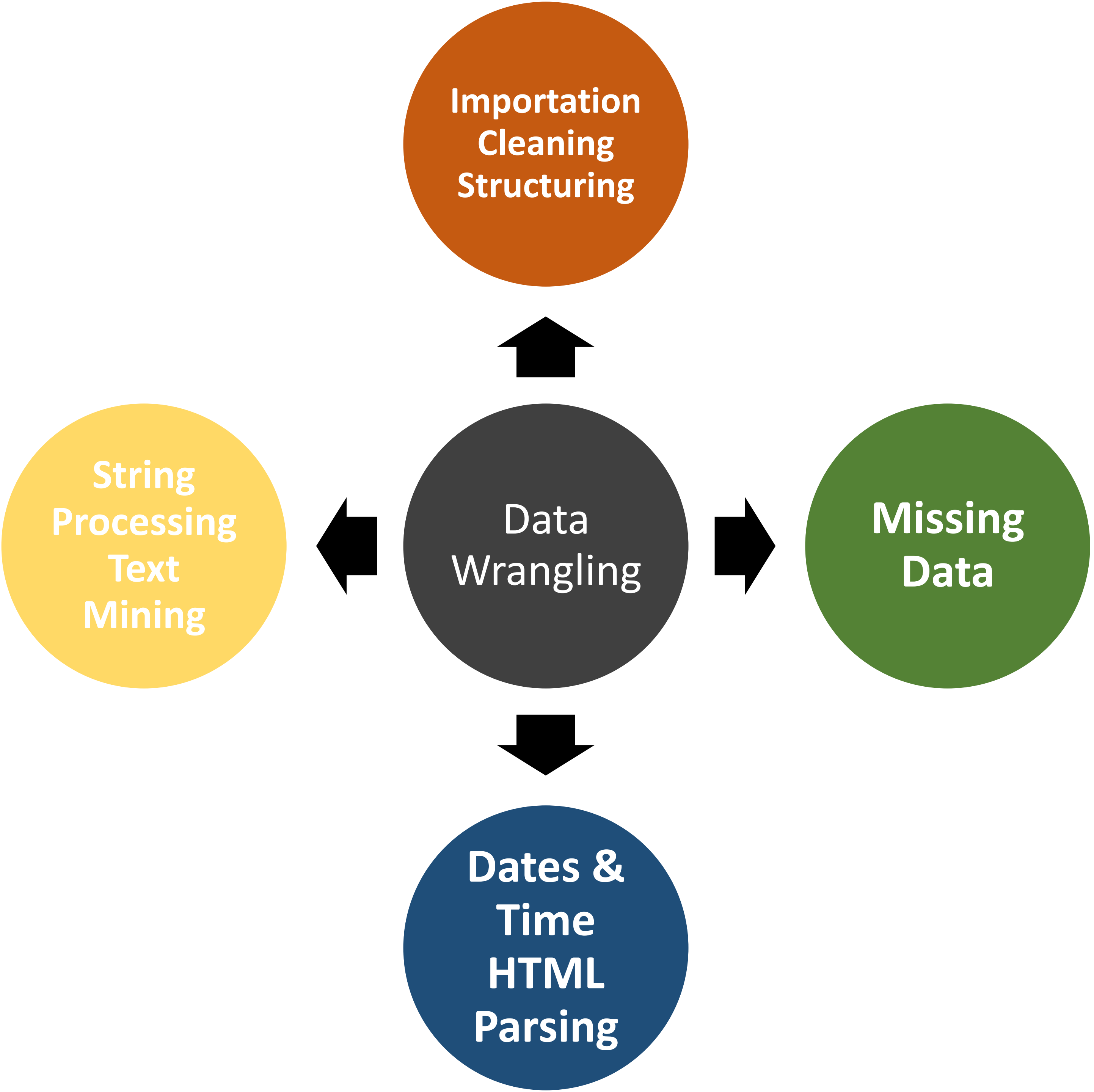
Knowledge Discovery Process



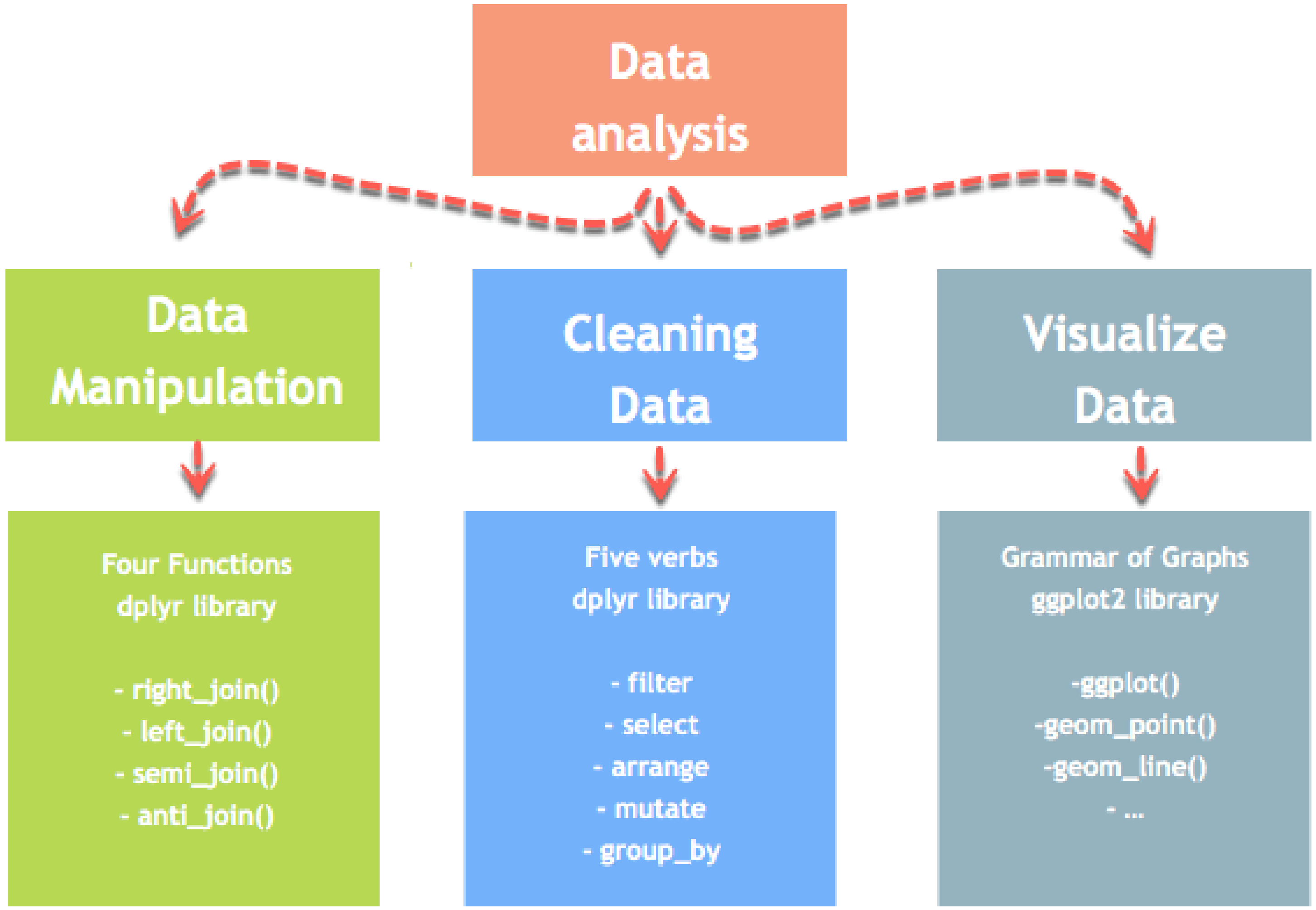

Visualise
Import
Tidy
Transform
Communicate
Model
Wrangle
Understand

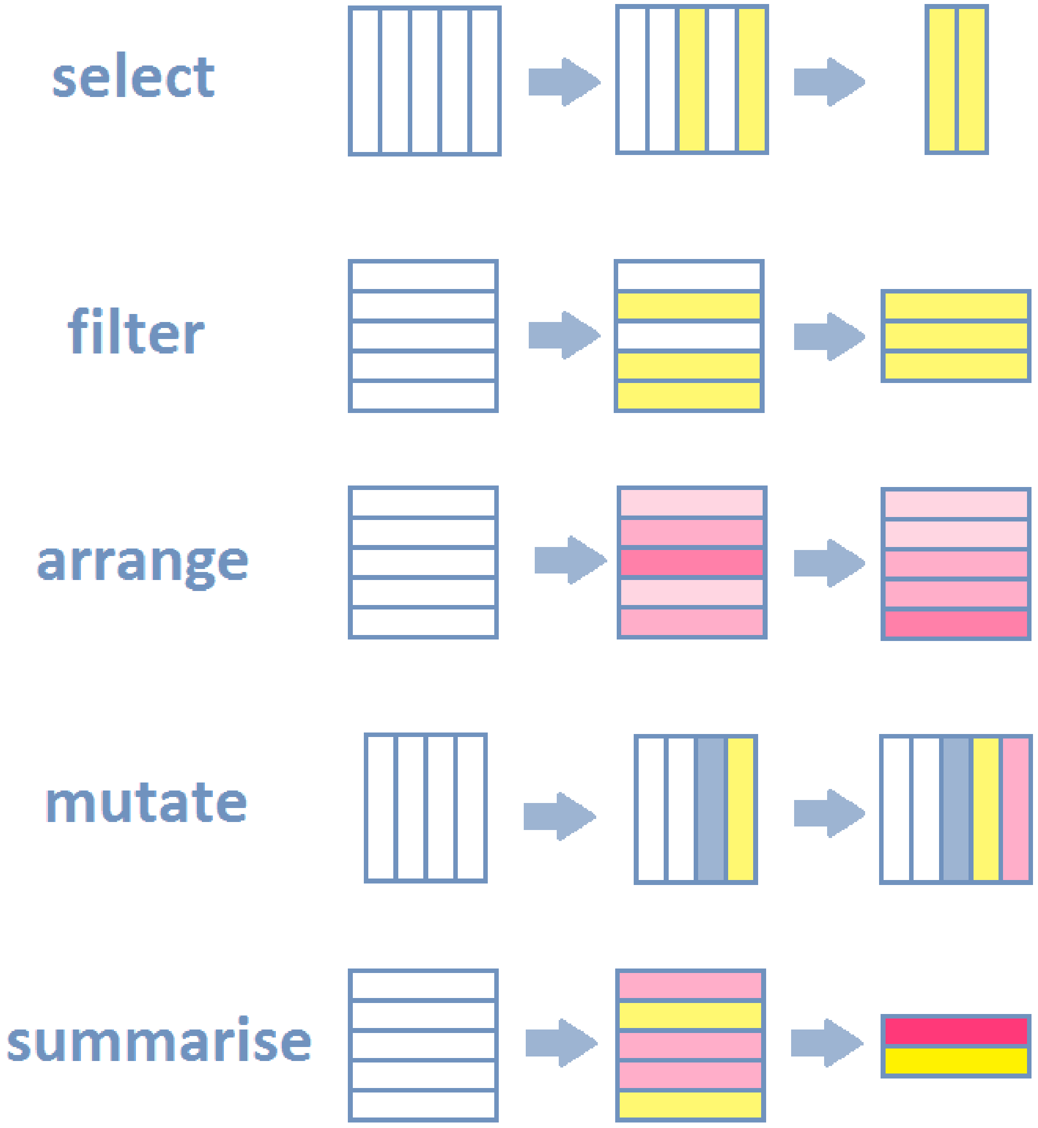
Learning objectives of this module:

- Introduction to Data Analysis
- Learn the basic vocabulary of dplyr
- Exercise commands
- Translating questions into data manipulation statements
- Visit the tidyr package
- Learn the why and how of Exploratory Data Analysis (EDA).


Importation
Cleaning
Structuring
String
Processing
Text
Mining
Data
Wrangling
Missing
Data
Dates &
Time
HTML
Parsing

Data Wrangling – Dplyr Package

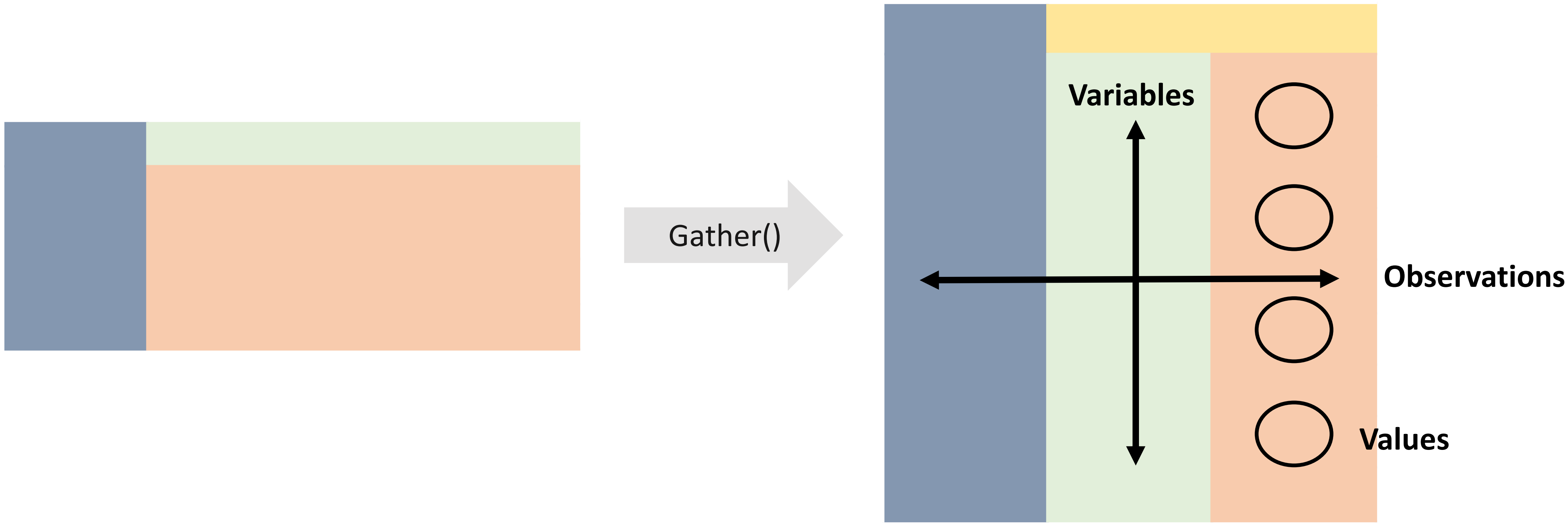




Source: http://perso.ens-lyon.fr/lise.vaudor/dplyr/

- Inspect your tibble (glimpse())
- Select specific columns (select())
- Filter out a subset of rows (filter())
- Change or add columns (mutate())
- Group observations by a grouping variable (group_by())
- Get a summary (in particular per group) (summarise())
- Join two distinct tibbles by a common column (left_join(), right_join() and full_join())

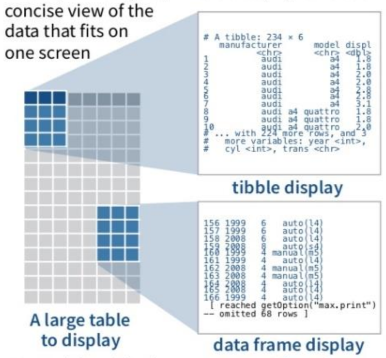
Tame Data   Tidy Data



Tibbles - an enhanced data frame

The **tibble** package provides a new S3 class for storing tabular data, the tibble. Tibbles inherit the data frame class, but improve three behaviors:

- **Subsetting** - [always returns a new tibble, [[and $ always return a vector.
- **No partial matching** - You must use full column names when subsetting
- **Display** - When you print a tibble, R provides a concise view of the data that fits on one screen



- Control the default appearance with options:
  options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf**)**
- View full data set with **View()** or **glimpse()**
- Revert to data frame with **as.data.frame()**

CONSTRUCT A TIBBLE IN TWO WAYS

tibble(…) Construct by columns.
tibble(x = 1:3, y = c("a", "b", "c"))

tribble(…) Construct by rows.
tribble(~x, ~y, 1, "a", 2, "b", 3, "c")

Both make this tibble

```
A tibble: 3 × 2
      x     y
  <int> <chr>
1     1     a
2     2     b
3     3     c
```

as_tibble(x, …) Convert data frame to tibble.

enframe(x, name = "name", value = "value") Convert named vector to a tibble

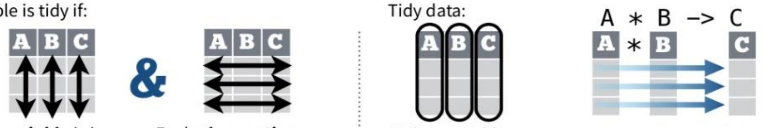
is_tibble(x) Test whether x is a tibble.

Tidy Data with tidyr

Tidy data is a way to organize tabular data. It provides a consistent data structure across packages.

A table is tidy if: Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

Tidy data: Makes variables easy to access as vectors; Preserves cases during vectorized operations



Reshape Data - change the layout of values in a table

Use **gather()** and **spread()** to reorganize the values of a table into a new layout.

gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE**)**

gather() moves column names into a **key** column, gathering the column values into a single **value** column.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

gather(table4a, `1999`, `2000`, key = "year", value = "cases")

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL**)**

spread() moves the unique values of a **key** column into the column names, spreading the values of a **value** column across the new columns.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

key value

spread(table2, type, count)

Handle Missing Values

drop_na(data, ...**)** Drop rows containing NA's in … columns.

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

drop_na(x, x2)

fill(data, ..., .direction = c("down", "up")**)** Fill in NA's in … columns with most recent non-NA values.

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

fill(x, x2)

replace_na(data, replace = list(), ...**)** Replace NA's by column.

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

replace_na(x, list(x2 = 2))

Expand Tables - quickly create tables with combinations of values

complete(data, ..., fill = list()**)** Adds to the data missing combinations of the values of the variables listed in …
complete(mtcars, cyl, gear, carb)

expand(data, ...**)** Create new tibble with all possible combinations of the values of the variables listed in …
expand(mtcars, cyl, gear, carb)

Split Cells

Use these functions to split or combine cells into individual, isolated values.

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...**)**

Separate each cell in a column to make several columns.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174
C	1999	212K	1T
C	2000	213K	1T

separate(table3, rate, into = c("cases", "pop"))

separate_rows(data, ..., sep = "[^[:alnum:].]+", convert = FALSE**)**

Separate each cell in a column to make several rows. Also **separate_rows_()**.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M
C	1999	212K
C	1999	1T
C	2000	213K
C	2000	1T

separate_rows(table3, rate)

unite(data, col, ..., sep = "_", remove = TRUE**)**

Collapse cells across several columns to make a single column.

table5

country	century	year
Afghan	19	99
Afghan	20	0
Brazil	19	99
Brazil	20	0
China	19	99
China	20	0

→

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

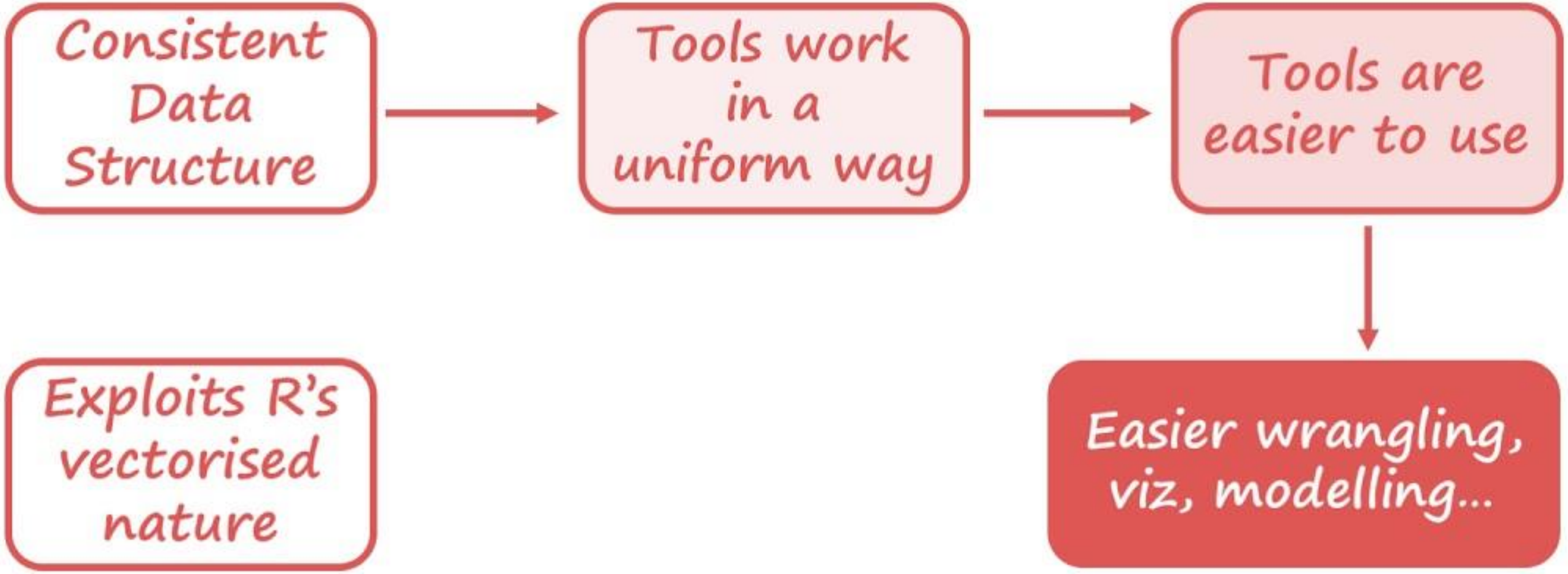
unite(table5, century, year, col = "year", sep = "")

RStudio

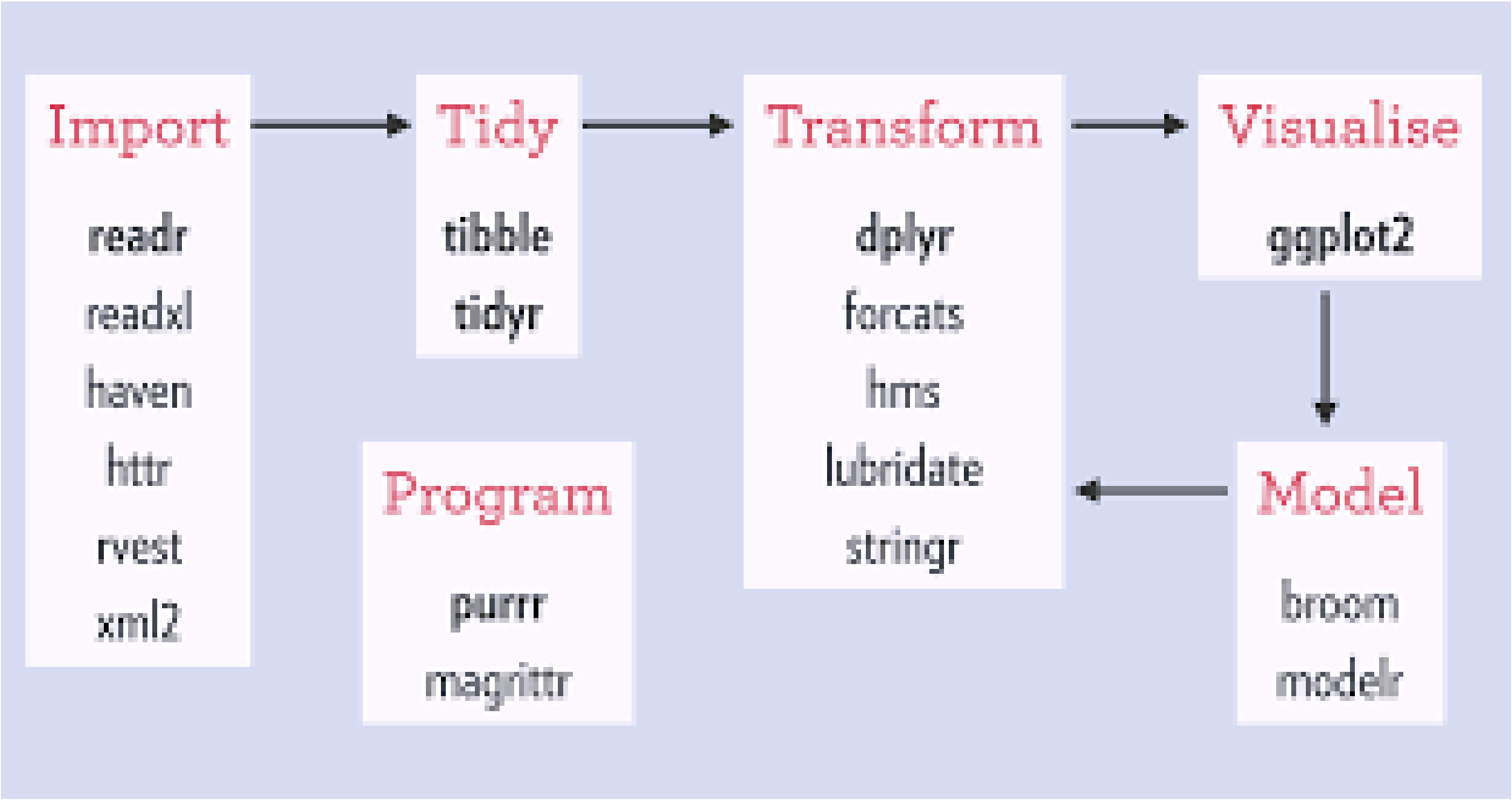
RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at tidyverse.org • readr 1.1.0 • tibble 1.2.12 • tidyr 0.6.0 • Updated: 2017-01

http://tidyr.tidyverse.org/ vignette("tidy-data")

Why tidy data?



https://www.tidyverse.org/




```
install.packages("tidyverse")

library("tidyverse")
```

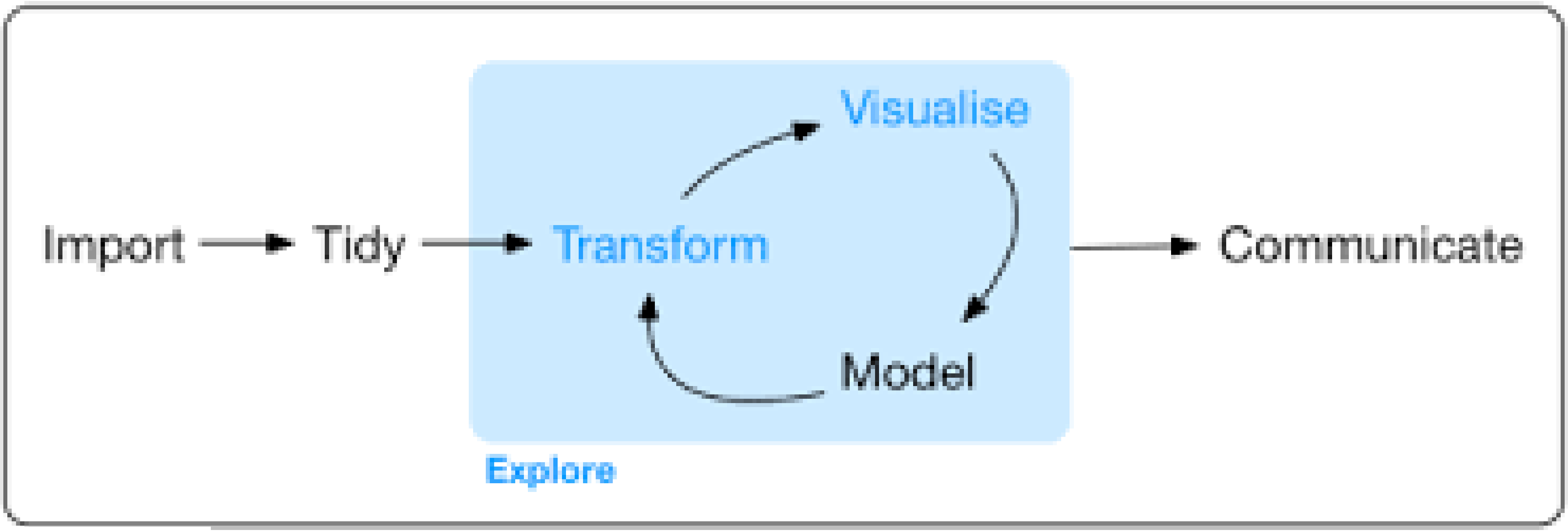
https://github.com/rstudio/master-the-tidyverse/archive/master.zip

Tidy Data

See the paper Tidy Data by Hadley Wickham in Journal of Statistical Software (2014)

https://github.com/rstudio/master-the-tidyverse/archive/master.zip

All material is permissively licensed under CC-BY-SA 4.0.


Import
Tidy
Transform
Visualise
Model
Communicate
Explore

EXPLORATORY DATA ANALYSIS
1
Collect the data and gain domain knowledge
2
Confirm data types and their probabilities
3
Measures of Central Tendency
a. Mean
b. Median
C. Mode
4
Measures of Dispersion
a. Variance
b. Std. Dev.
c. Range
5
Skewness right & left
Kurtosis
Thinner peck
Wider peck
6
Graphical Representation
a. Histogram
b. Boxplot
c. Dotplot
d. Stem and Leaf

“Get to Know" the dataset

- Doing so upfront will make the rest of the project much smoother, in 3 main ways:
    1. You’ll gain valuable hints for Data Cleaning.
    2. You’ll think of ideas for Feature Engineering.
    3. You’ll get a "feel" for the dataset, which will help you communicate results and deliver greater impact.

- EDA should be **quick, efficient, and decisive**... not long and drawn out!
- You see, there are infinite possible plots, charts, and tables, but you only need a **handful** to "get to know" the data well enough to work with it.

What is EDA?

An approach for data analysis that employs a variety of techniques

1. Maximize insight into a data set
2. Uncover underlying structure
3. Extract important variables
4. Detect outliers and anomalies
5. Test underlying assumptions
6. Develop parsimonious models and
7. Determine optimal factor settings

EDA is a data approach.

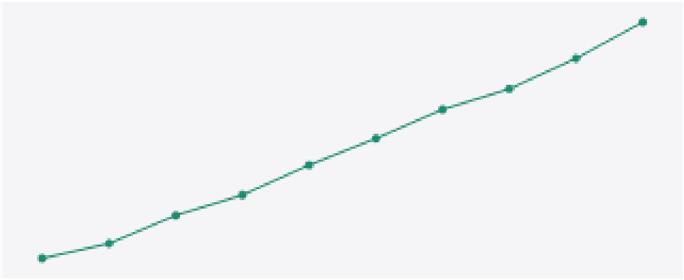
The EDA sequence is:

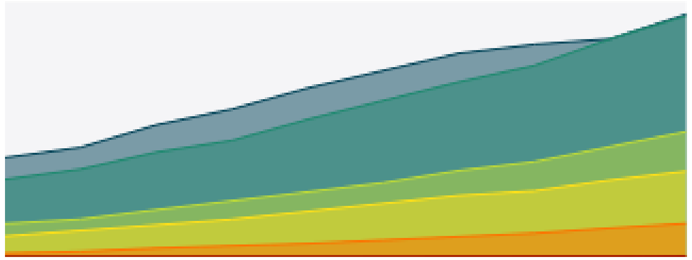
Problem => Data => Analysis=> Model=> Conclusions

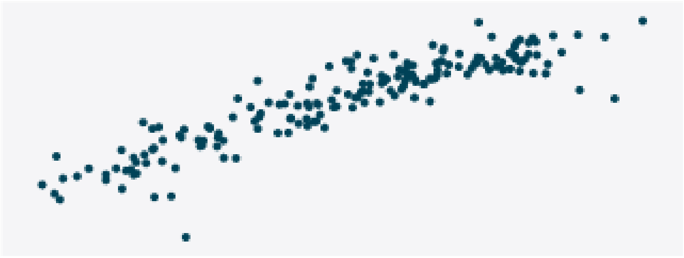
As opposed for a classical approach:

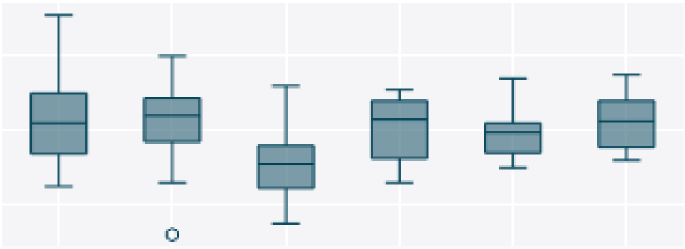
Problem => Data => Model=> Analysis=> Conclusions

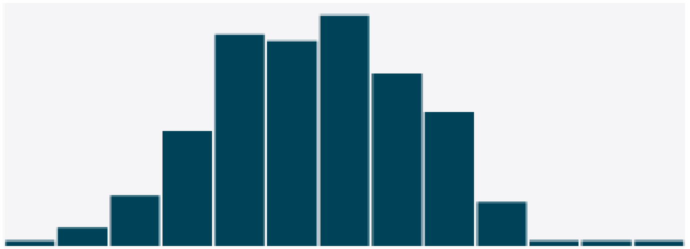
EDA Techniques are generally graphical











EDA is majorly performed using the following methods:

Univariate visualization – provides summary statistics for each field in the raw data set

Bivariate visualization – is performed to find the relationship between each variable in the dataset and the target variable of interest

Multivariate visualization – is performed to understand interactions between different fields in the dataset

Dimensionality reduction – helps to understand the fields in the data that account for the most variance between observations and allow for the processing of a reduced volume of data.

Useful Packages

```
## Needed libs for EDA
## Some may need to install these packages if this is their
## first time using R

library(dplyr)
library(ggplot2)
library(gapminder)
library(tidyr)
library(readr)
library(openintro)
options(scipen=999,digits=3)
```

EDA of Categorical Data

Example: Coronvirus

- https://www.kaggle.com/xordux/india-corona-severity-zones
- The zones are:
  1. Green Zone: Least impacted zone, A district will be considered under green zone if there has been no confirmed cases of COVID-19 so far or there is no reported case since last 21 days in the district.
  2. Orange Zone: Districts that do not have enough confirmed cases to meet the ‘red zone’, but are being seen as potential hotspots, are part of the ‘orange zone’. A Red Zone can be categorised as a Orange Zone if no new confirmed case is reported there for 14 consecutive days.
  3. Red Zone: Districts reporting a large number of cases or high growth rates. Inclusion criteria for Red Zone:
     1. Highest case-load districts contributing to over 80 percent of cases in India, or
     2. Highest case-load districts contributing to more than 80 percent of cases for each state in the country, or
     3. Districts with doubling rate at less than four days (calculated every Monday for last seven days, to be determined by the state government).

Upload Data

```
## Download data and see what the set is composed of
## Make sure you download data in the working directory
## TO CHANGE: Toolbar: Session > Set Working Directory > Choose Directory > (select
df <- read.csv("covid_zones.csv")
glimpse(df)
```

```
> glimpse(df)
Rows: 733
Columns: 4
$ No       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,...
$ District <chr> " South Andamans ", " Nicobars ", " North And Middle Andaman ", " ...
$ State    <chr> " Andaman And Nicobar Islands ", " Andaman And Nicobar Islands ", ...
$ Zone     <chr> " Red Zone", " Green Zone", " Green Zone", " Red Zone", " Red Zone...
```

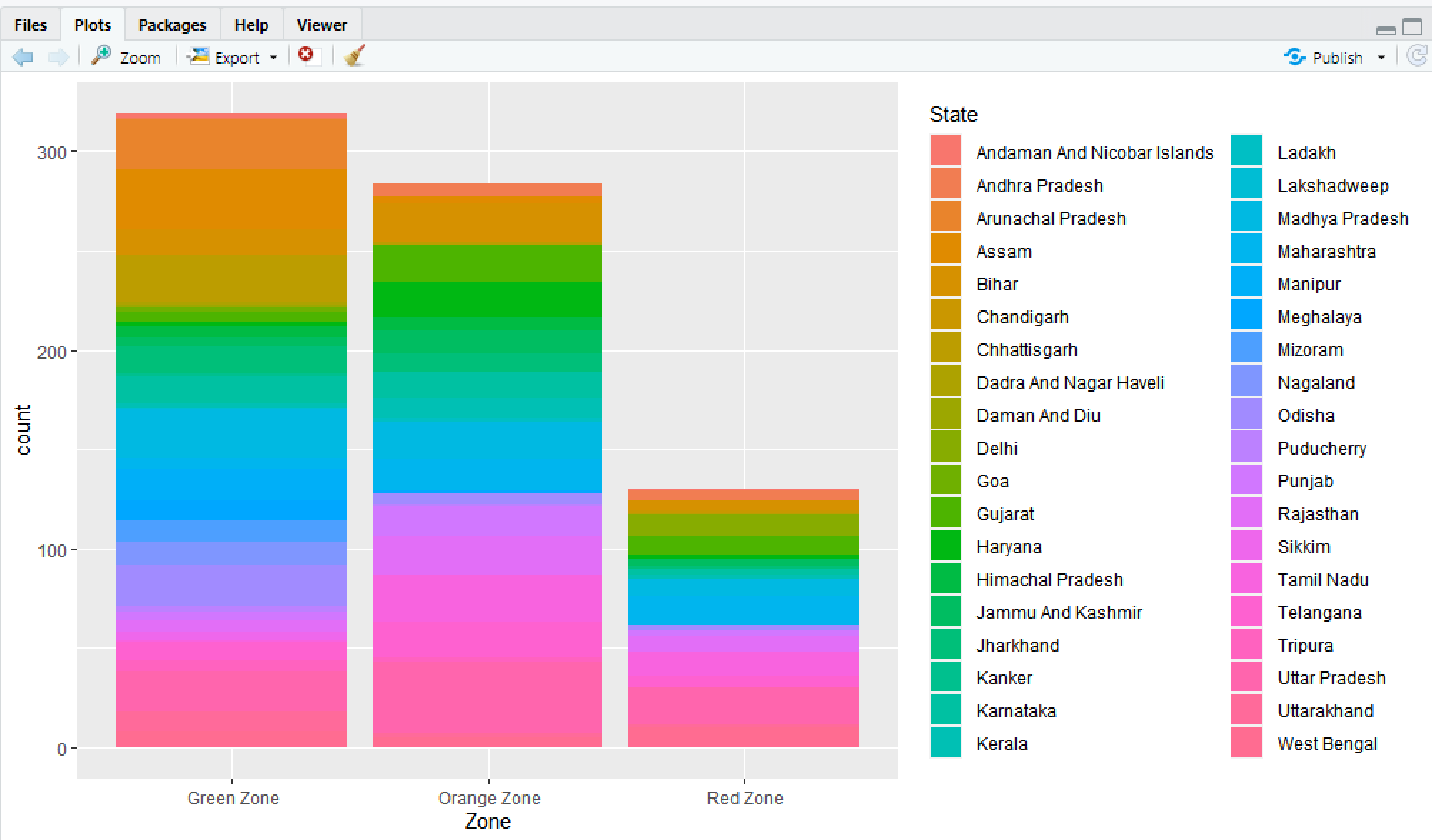
Contingency Table

```
## Contingency table: To get a frequency distribution between 2 factors variables

table(df$Zone,df$State)

## A ggplot always needs three basic inputs - 1) dataset 2) variables on axes
## 3) layer to be used. For 2 categorical variables, a stack bar chart is good.
## In this case, one categorical variable goes on x axis, in each bar,
## the other categorical variable is filled using the color.

ggplot(df,aes(x=Zone,fill=State)) + geom_bar()
```

Output for Contingency Table

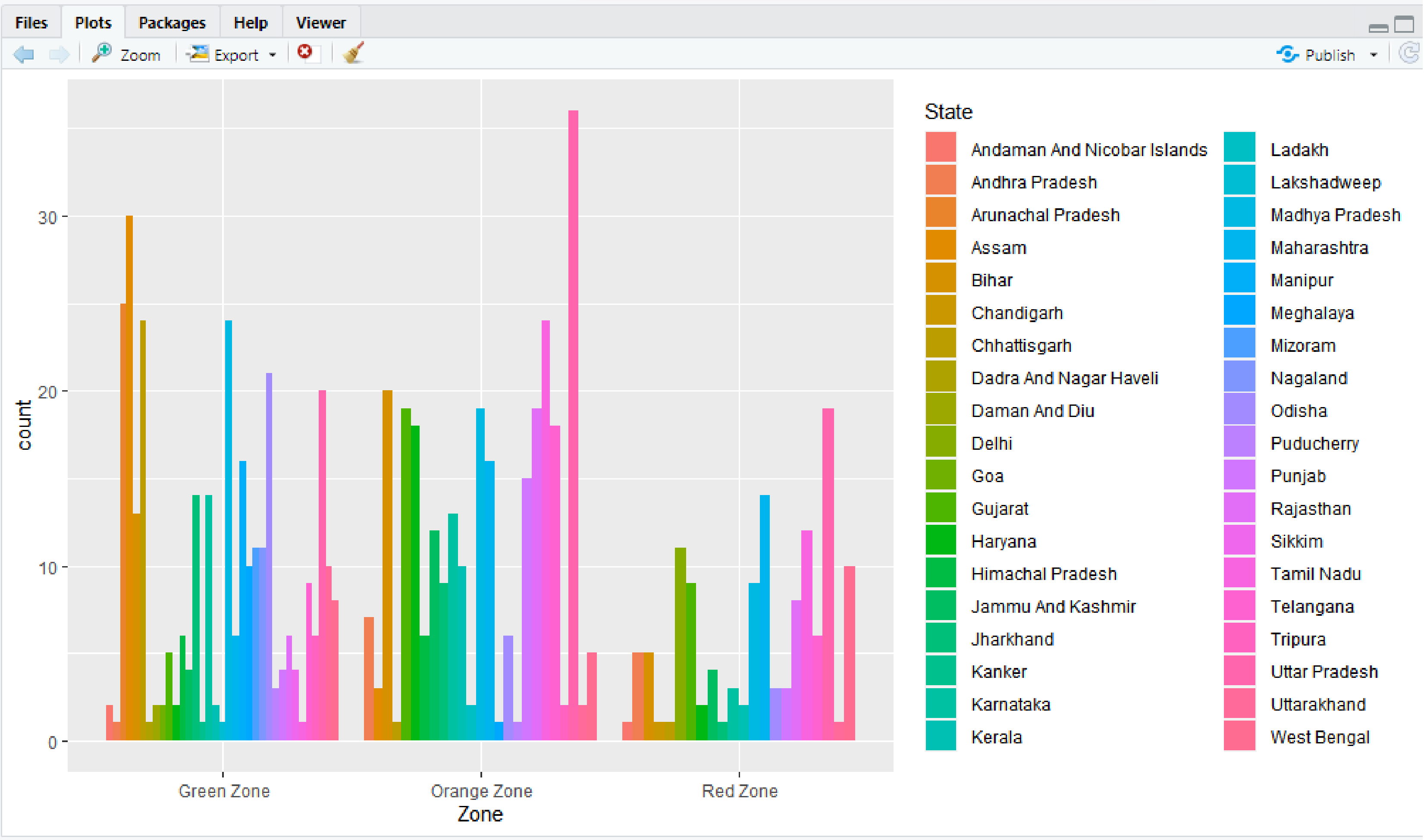
```
> table(df$Zone,df$State)

               Andaman And Nicobar Islands   Andhra Pradesh   Arunachal Pradesh
  Green Zone                             2                1                  25
  Orange Zone                            0                7                   0
  Red Zone                               1                5                   0

               Assam   Bihar   Chandigarh   Chhattisgarh   Dadra And Nagar Haveli
  Green Zone      30      13            0             24                        1
  Orange Zone      3      20            0              1                        0
  Red Zone         0       5            1              1                        0

               Daman And Diu   Delhi   Goa   Gujarat   Haryana   Himachal Pradesh
  Green Zone               2       0     2         5         2                  6
  Orange Zone              0       0     0        19        18                  6
  Red Zone                 0      11     0         9         2                  0

               Jammu And Kashmir   Jharkhand   Kanker   Karnataka   Kerala   Ladakh
  Green Zone                   4          14        1          14        2        0
  Orange Zone                 12           9        0          13       10        2
  Red Zone                     4           1        0           3        2        0

               Lakshadweep   Madhya Pradesh   Maharashtra   Manipur   Meghalaya
  Green Zone             1               24             6        16          10
  Orange Zone            0               19            16         0           1
  Red Zone               0                9            14         0           0

               Mizoram   Nagaland   Odisha   Puducherry   Punjab   Rajasthan   Sikkim
  Green Zone        11         11       21            3        4           6        4
  Orange Zone        0          0        6            1       15          19        0
  Red Zone           0          0        3            0        3           8        0

               Tamil Nadu   Telangana   Tripura   Uttar Pradesh   Uttarakhand
  Green Zone            1           9         6              20            10
  Orange Zone          24          18         2              36             2
  Red Zone             12           6         0              19             1

               West Bengal
  Green Zone             8
  Orange Zone            5
  Red Zone              10
> 
```

Output for ggplot()



Side-by-Side?

```
## When you dont want the information to be stacked, but want them side-by-side,
## use the **position=“dodge"** arguement in the geom_bar().

ggplot(df,aes(x=Zone,fill=State)) + geom_bar(position="dodge")
```



Proportions

```r
## Sometimes the count isn't what is important. We want proportions, so the
## arguement **prop.table()** will change the contingency table to where the
## values are percentages

tab_cnt <- table(df$Zone,df$State)
prop.table(tab_cnt)
```

Output

```
> prop.table(tab_cnt)

             Andaman And Nicobar Islands Andhra Pradesh Arunachal Pradesh
  Green Zone                     0.00273        0.00136           0.03411
  Orange Zone                    0.00000        0.00955           0.00000
  Red Zone                       0.00136        0.00682           0.00000

             Assam   Bihar   Chandigarh Chhattisgarh Dadra And Nagar Haveli
  Green Zone 0.04093 0.01774    0.00000      0.03274                0.00136
  Orange Zone 0.00409 0.02729   0.00000      0.00136                0.00000
  Red Zone   0.00000 0.00682    0.00136      0.00136                0.00000

             Daman And Diu   Delhi     Goa Gujarat Haryana Himachal Pradesh
  Green Zone       0.00273 0.00000 0.00273 0.00682 0.00273          0.00819
  Orange Zone      0.00000 0.00000 0.00000 0.02592 0.02456          0.00819
  Red Zone         0.00000 0.01501 0.00000 0.01228 0.00273          0.00000

             Jammu And Kashmir Jharkhand  Kanker Karnataka  Kerala  Ladakh
  Green Zone           0.00546   0.01910 0.00136   0.01910 0.00273 0.00000
  Orange Zone          0.01637   0.01228 0.00000   0.01774 0.01364 0.00273
  Red Zone             0.00546   0.00136 0.00000   0.00409 0.00273 0.00000

             Lakshadweep Madhya Pradesh Maharashtra Manipur Meghalaya
  Green Zone     0.00136        0.03274     0.00819 0.02183   0.01364
  Orange Zone    0.00000        0.02592     0.02183 0.00000   0.00136
  Red Zone       0.00000        0.01228     0.01910 0.00000   0.00000

             Mizoram Nagaland  Odisha Puducherry  Punjab Rajasthan  Sikkim
  Green Zone 0.01501  0.01501 0.02865    0.00409 0.00546   0.00819 0.00546
  Orange Zone 0.00000 0.00000 0.00819    0.00136 0.02046   0.02592 0.00000
  Red Zone   0.00000  0.00000 0.00409    0.00000 0.00409   0.01091 0.00000

             Tamil Nadu Telangana Tripura Uttar Pradesh Uttarakhand
  Green Zone    0.00136   0.01228 0.00819       0.02729     0.01364
  Orange Zone   0.03274   0.02456 0.00273       0.04911     0.00273
  Red Zone      0.01637   0.00819 0.00000       0.02592     0.00136

             West Bengal
  Green Zone     0.01091
  Orange Zone    0.00682
  Red Zone       0.01364
>
```

Proportions (con't)

```
## This forces the rows to be to add to give 1
prop.table(tab_cnt,1)
```

```
> ## This forces the rows to be to add to give 1
> prop.table(tab_cnt,1)

             Andaman And Nicobar Islands  Andhra Pradesh  Arunachal Pradesh
  Green Zone                     0.00627         0.00313            0.07837
  Orange Zone                    0.00000         0.02465            0.00000
  Red Zone                       0.00769         0.03846            0.00000

             Assam   Bihar   Chandigarh  Chhattisgarh  Dadra And Nagar Haveli
  Green Zone  0.09404 0.04075    0.00000       0.07524                 0.00313
  Orange Zone 0.01056 0.07042    0.00000       0.00352                 0.00000
  Red Zone    0.00000 0.03846    0.00769       0.00769                 0.00000

             Daman And Diu   Delhi     Goa  Gujarat  Haryana  Himachal Pradesh
  Green Zone        0.00627 0.00000 0.00627  0.01567  0.00627           0.01881
  Orange Zone       0.00000 0.00000 0.00000  0.06690  0.06338           0.02113
  Red Zone          0.00000 0.08462 0.00000  0.06923  0.01538           0.00000

             Jammu And Kashmir  Jharkhand  Kanker  Karnataka  Kerala  Ladakh
  Green Zone           0.01254    0.04389 0.00313    0.04389 0.00627 0.00000
  Orange Zone          0.04225    0.03169 0.00000    0.04577 0.03521 0.00704
  Red Zone             0.03077    0.00769 0.00000    0.02308 0.01538 0.00000

             Lakshadweep  Madhya Pradesh  Maharashtra  Manipur  Meghalaya
  Green Zone      0.00313         0.07524      0.01881  0.05016    0.03135
  Orange Zone     0.00000         0.06690      0.05634  0.00000    0.00352
  Red Zone        0.00000         0.06923      0.10769  0.00000    0.00000

             Mizoram  Nagaland  Odisha  Puducherry  Punjab  Rajasthan  Sikkim
  Green Zone  0.03448   0.03448 0.06583     0.00940 0.01254    0.01881 0.01254
  Orange Zone 0.00000   0.00000 0.02113     0.00352 0.05282    0.06690 0.00000
  Red Zone    0.00000   0.00000 0.02308     0.00000 0.02308    0.06154 0.00000

             Tamil Nadu  Telangana  Tripura  Uttar Pradesh  Uttarakhand
  Green Zone     0.00313    0.02821  0.01881        0.06270      0.03135
  Orange Zone    0.08451    0.06338  0.00704        0.12676      0.00704
  Red Zone       0.09231    0.04615  0.00000        0.14615      0.00769

             West Bengal
  Green Zone     0.02508
  Orange Zone    0.01761
  Red Zone       0.07692
>
```

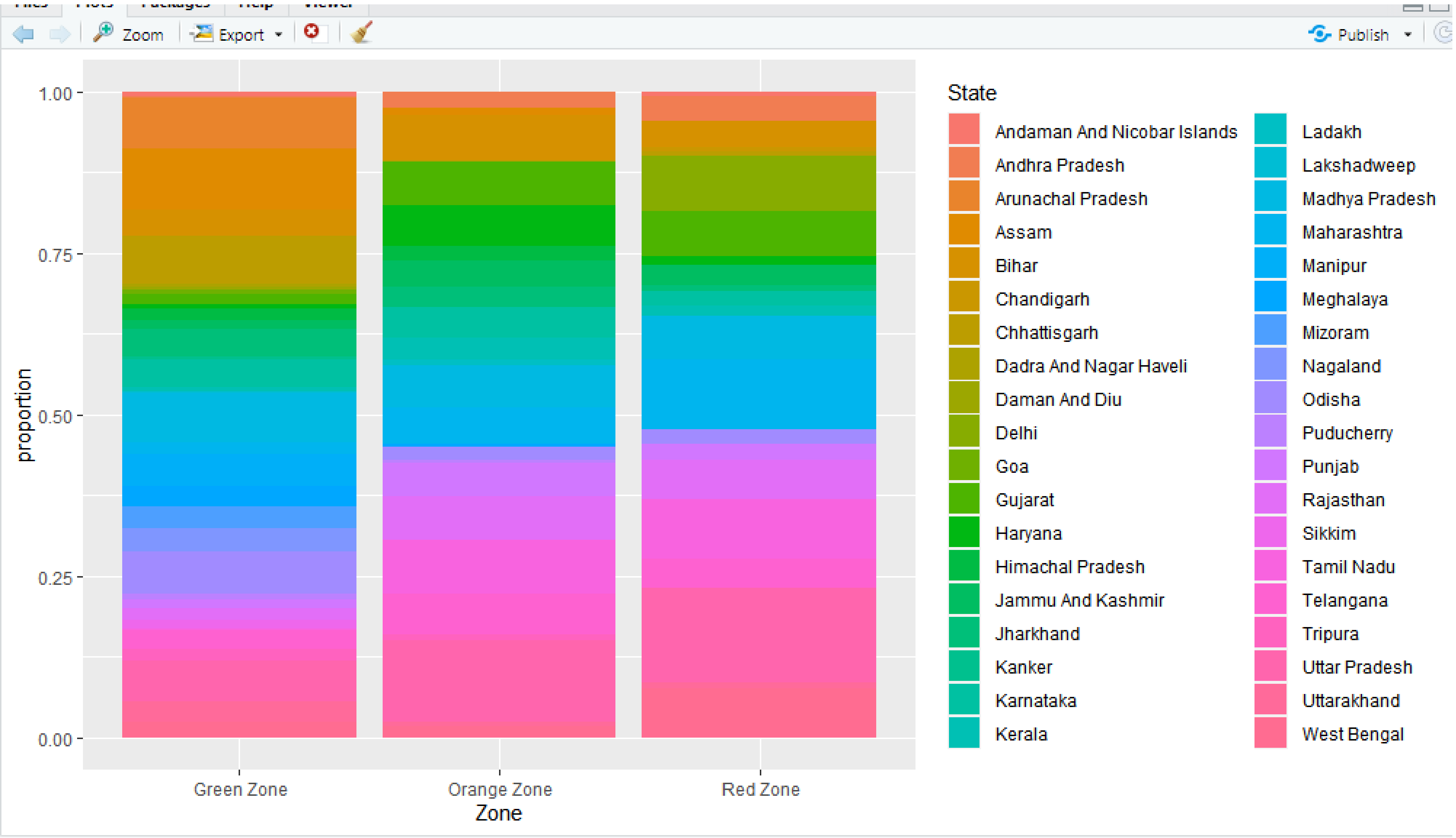
Proportions (con't)

```r
## This forces the columns to be to add to give 1
prop.table(tab_cnt,2)
```

```
> ## This forces the columns to be to add to give 1
> prop.table(tab_cnt,2)

              Andaman And Nicobar Islands Andhra Pradesh Arunachal Pradesh
  Green Zone                       0.6667         0.0769            1.0000
  Orange Zone                      0.0000         0.5385            0.0000
  Red Zone                         0.3333         0.3846            0.0000

              Assam  Bihar  Chandigarh Chhattisgarh Dadra And Nagar Haveli
  Green Zone  0.9091 0.3421     0.0000       0.9231                 1.0000
  Orange Zone 0.0909 0.5263     0.0000       0.0385                 0.0000
  Red Zone    0.0000 0.1316     1.0000       0.0385                 0.0000

              Daman And Diu  Delhi    Goa  Gujarat Haryana Himachal Pradesh
  Green Zone         1.0000 0.0000 1.0000   0.1515  0.0909           0.5000
  Orange Zone        0.0000 0.0000 0.0000   0.5758  0.8182           0.5000
  Red Zone           0.0000 1.0000 0.0000   0.2727  0.0909           0.0000

              Jammu And Kashmir Jharkhand Kanker Karnataka Kerala Ladakh
  Green Zone             0.2000    0.5833 1.0000    0.4667 0.1429 0.0000
  Orange Zone            0.6000    0.3750 0.0000    0.4333 0.7143 1.0000
  Red Zone               0.2000    0.0417 0.0000    0.1000 0.1429 0.0000

              Lakshadweep Madhya Pradesh Maharashtra Manipur Meghalaya
  Green Zone       1.0000         0.4615      0.1667  1.0000    0.9091
  Orange Zone      0.0000         0.3654      0.4444  0.0000    0.0909
  Red Zone         0.0000         0.1731      0.3889  0.0000    0.0000

              Mizoram Nagaland Odisha Puducherry Punjab Rajasthan Sikkim
  Green Zone   1.0000   1.0000 0.7000     0.7500 0.1818    0.1818 1.0000
  Orange Zone  0.0000   0.0000 0.2000     0.2500 0.6818    0.5758 0.0000
  Red Zone     0.0000   0.0000 0.1000     0.0000 0.1364    0.2424 0.0000

              Tamil Nadu Telangana Tripura Uttar Pradesh Uttarakhand
  Green Zone      0.0270    0.2727  0.7500        0.2667      0.7692
  Orange Zone     0.6486    0.5455  0.2500        0.4800      0.1538
  Red Zone        0.3243    0.1818  0.0000        0.2533      0.0769

              West Bengal
  Green Zone       0.3478
  Orange Zone      0.2174
  Red Zone         0.4348
```

100% stack chart, conditioned on Zone

```r
## Stacked 100% bar chart. This is called 100% stack chart, conditioned on Zone
ggplot(df,aes(x=Zone,fill=State)) + geom_bar(position="fill") + ylab("proportion")
```

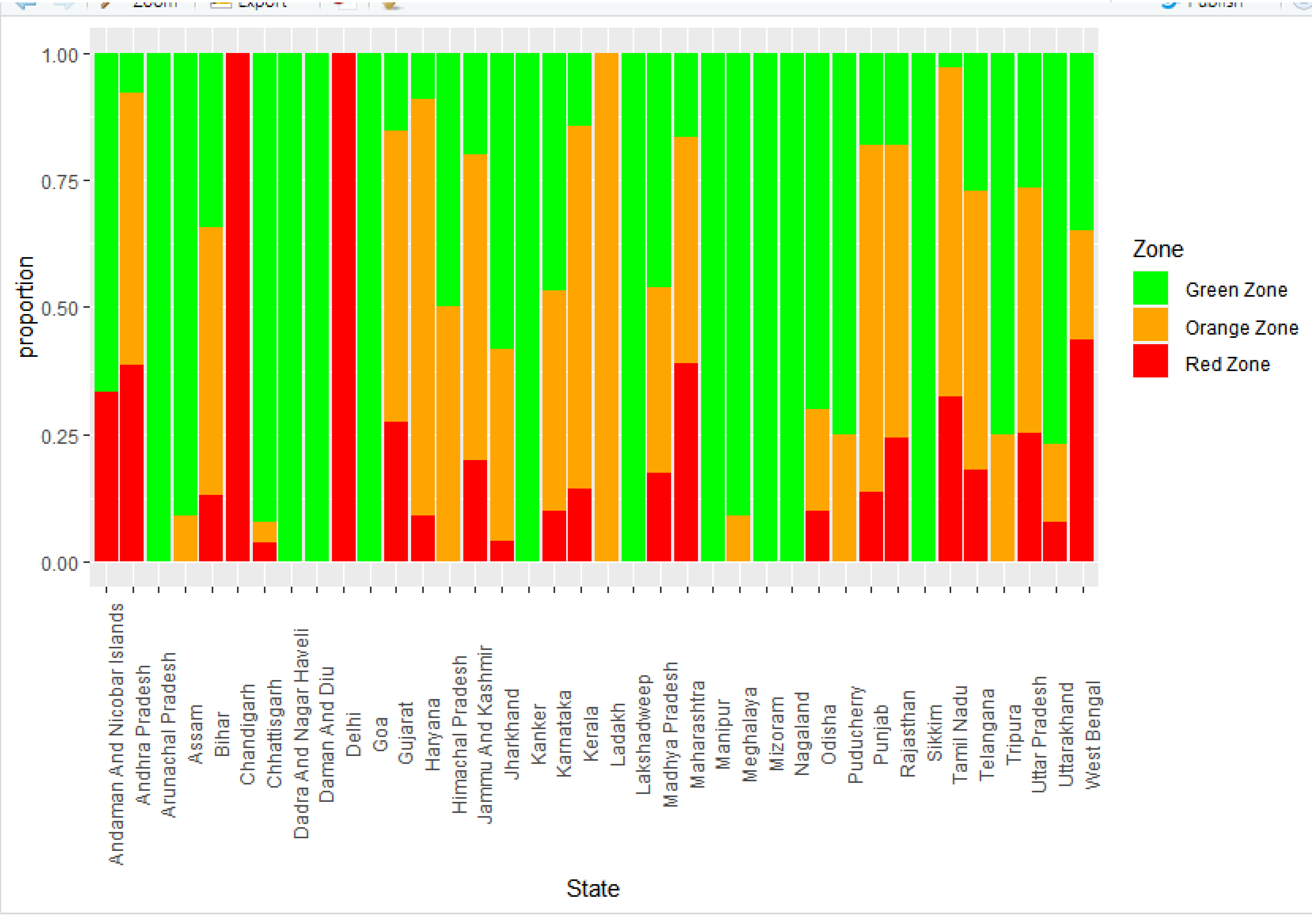


100% stacked bar chart, conditioned on State

```
## 100% stacked bar chart, conditioned on State
ggplot(df,aes(x=State,fill=Zone)) + geom_bar(position="fill") + ylab("proportion")+ scale_fill_manual(values = c("Green", "Orange", "Red"))
```

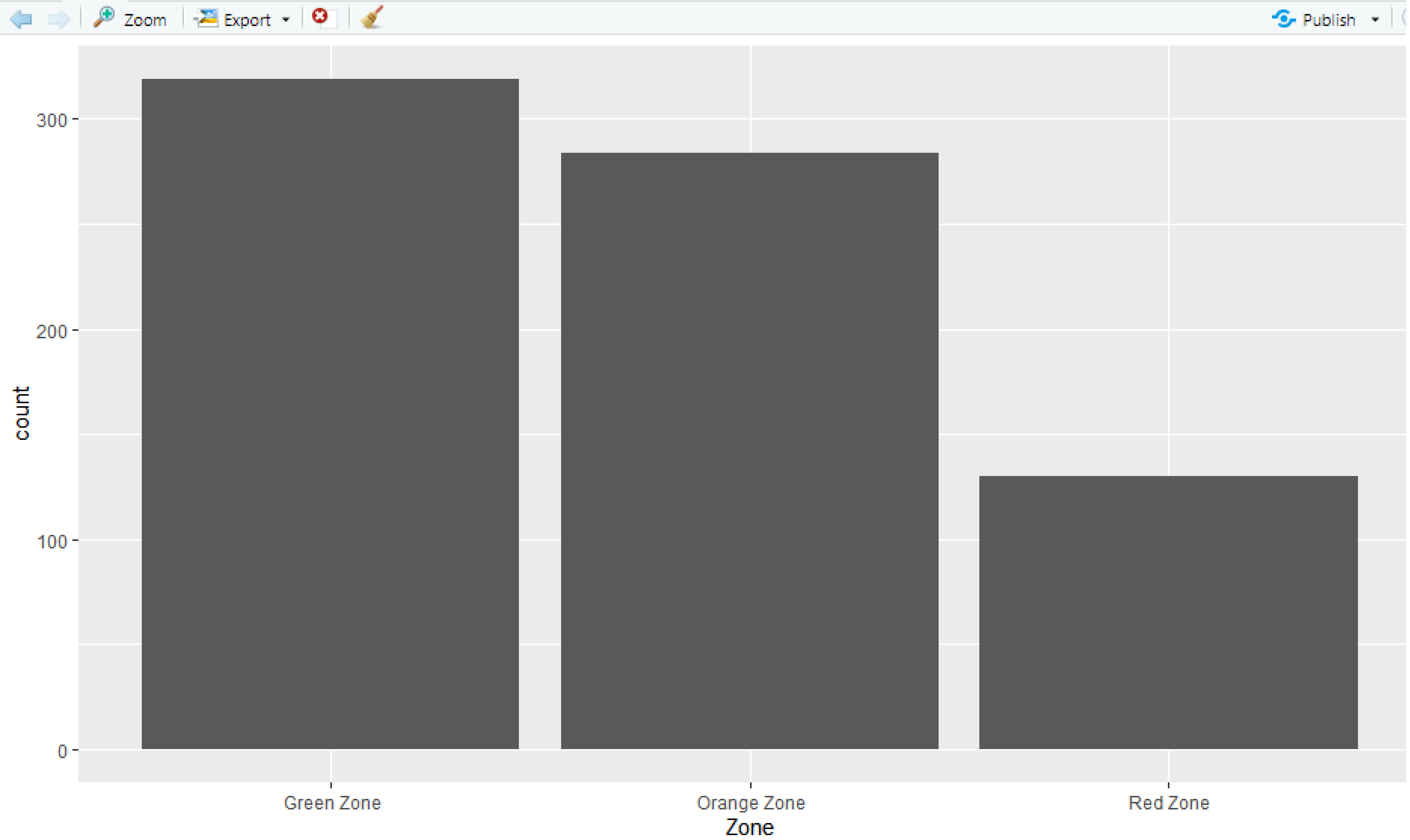
Let's Clean the labels up!

```
## Rotates x labels by a 90 degree angle.
####****** They cannot space/enter here. otherwise an error will occur..
####****** It must all stay on one line.
ggplot(df,aes(x=State,fill=Zone)) + geom_bar(position="fill") + ylab("proportion")+
  scale_fill_manual(values = c("Green", "Orange", "Red")) + theme(axis.text.x = element_text(angle=90))
```



Marginal Distribution

```r
## Contingency table of marginal distributions.
table(df$Zone)
ggplot(df,aes(x=Zone)) + geom_bar()
```



Distribution against one variable

```r
## If concerned with analyzing one variable's distribution against only one value
## of another variable.
ggplot(df,aes(x=Zone)) + geom_bar() + facet_wrap(~State)+ theme(axis.text.x = element_text(angle=90))

```

EDA of Numeric Data

Example: Air Quality

- https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/airquality.html
- Daily air quality measurements in New York, May to September 1973.
- Daily readings of the following air quality values for May 1, 1973 (a Tuesday) to September 30, 1973.
    - Ozone: Mean ozone in parts per billion from 1300 to 1500 hours at Roosevelt Island
    - Solar.R: Solar radiation in Langleys in the frequency band 4000–7700 Angstroms from 0800 to 1200 hours at Central Park
    - Wind: Average wind speed in miles per hour at 0700 and 1000 hours at LaGuardia Airport
    - Temp: Maximum daily temperature in degrees Fahrenheit at La Guardia Airport.
    - Month: Numeric value between 1-12
    - Day: Numeric value between 1-31

Upload Numeric Data

```r
## Download data from R using data() and see what the set is composed of
## Make sure you download data in the working directory

data("airquality")
str(airquality)
```

```
> str(airquality)
'data.frame':	153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
> 
```

Data Cleaning

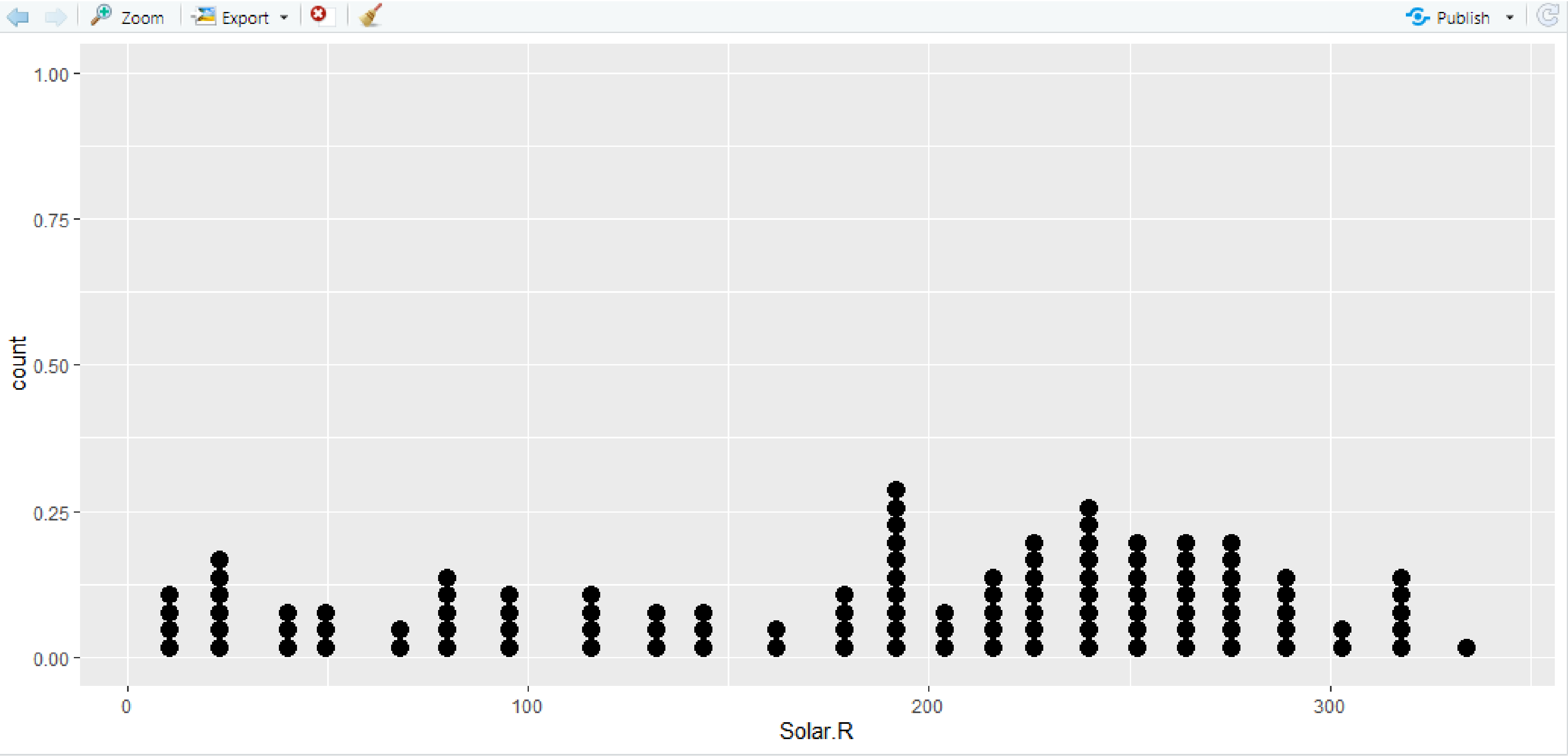
```r
## To remove NA values, we use complete.cases() which will assign all NA as False,
## else, True.
complete.cases(airquality)

## To drop values option 1:
x <- airquality[complete.cases(airquality), ]
str(x)

## To drop values option 2:
y <- na.omit(airquality)
str(y)
```

Output

```
> str(x)
'data.frame':   111 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 23 19 8 16 11 14 ...
 $ Solar.R: int  190 118 149 313 299 99 19 256 290 274 ...
 $ Wind   : num  7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...
 $ Temp   : int  67 72 74 62 65 59 61 69 66 68 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 7 8 9 12 13 14 ...
> ## To drop values option 2:
> y <- na.omit(airquality)
> str(y)
'data.frame':   111 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 23 19 8 16 11 14 ...
 $ Solar.R: int  190 118 149 313 299 99 19 256 290 274 ...
 $ Wind   : num  7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...
 $ Temp   : int  67 72 74 62 65 59 61 69 66 68 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 7 8 9 12 13 14 ...
 - attr(*, "na.action")= 'omit' Named int [1:42] 5 6 10 11 25 26 27 32 33 34 ...
  ..- attr(*, "names")= chr [1:42] "5" "6" "10" "11" ...
> 
```

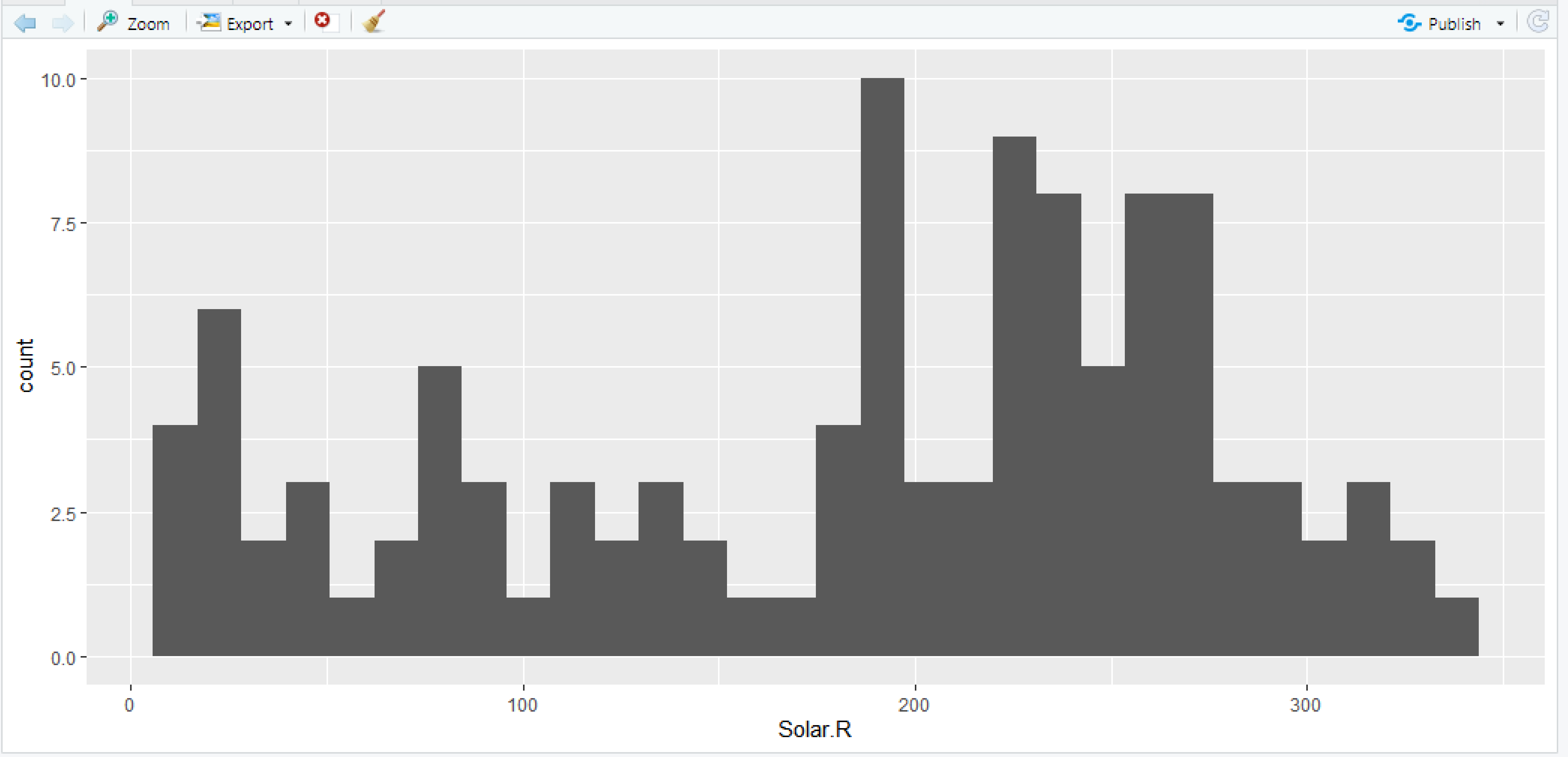
Dotplot

```
## Making a dotplot to show numerical data. It's like a bar chart,
## but with points stacked on top of each other
ggplot(y,aes(x=Solar.R)) + geom_dotplot(dotsize=0.4)
```



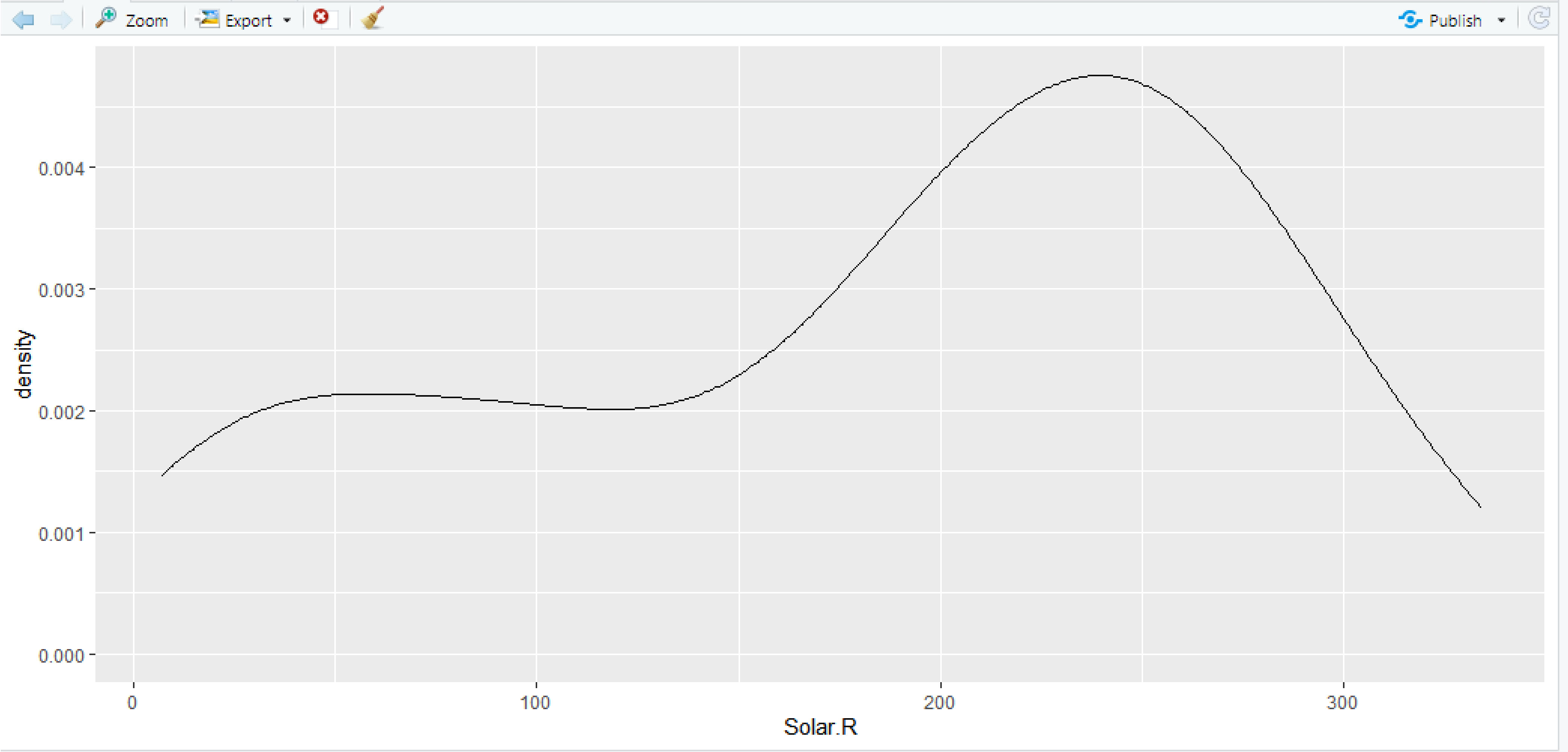
Histogram

```
## Histogram combines the dots, and the y axis now shows the actual count
ggplot(y,aes(x=Solar.R)) + geom_histogram()
```



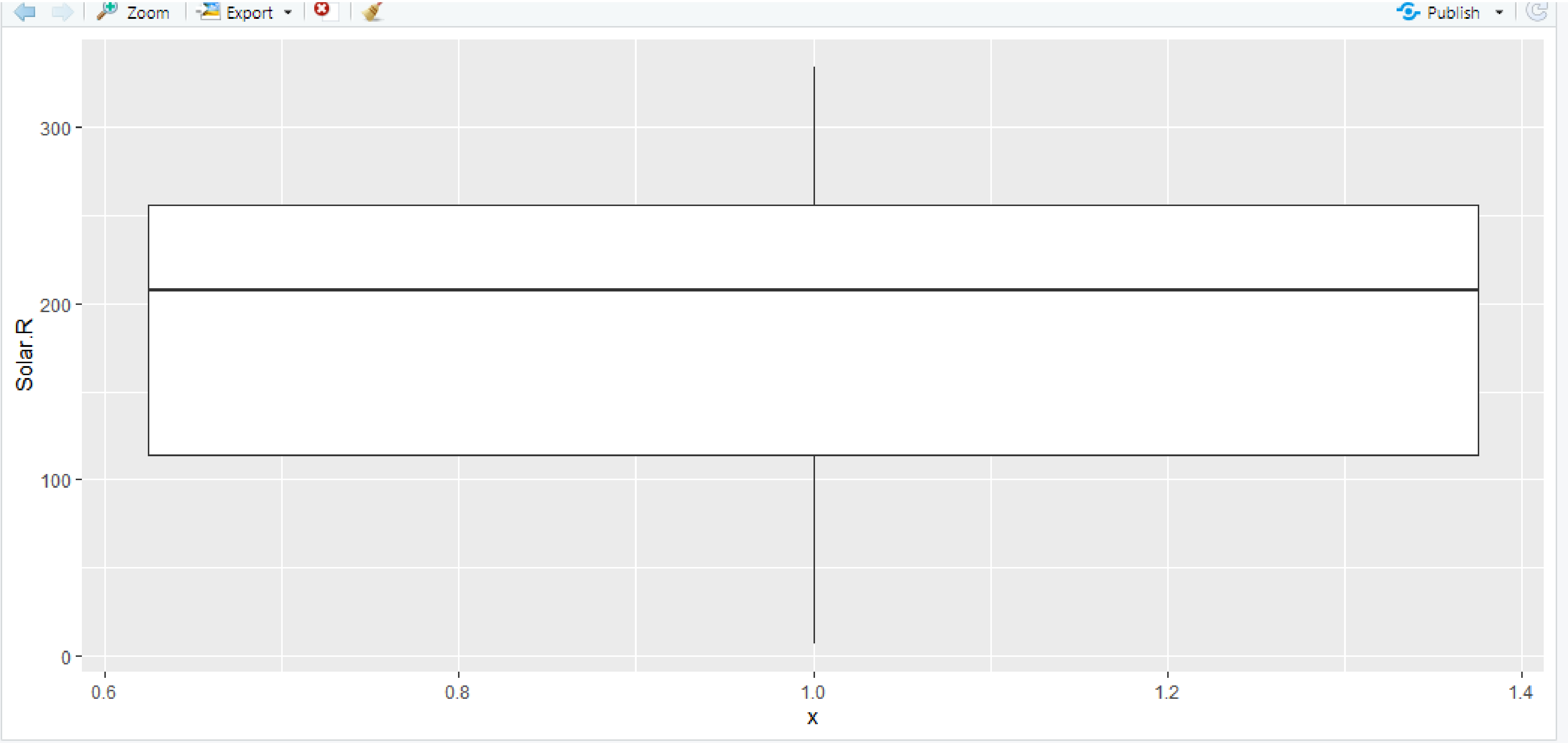
Density plot

```
## The shape of the distribution can be better represented with a density plot,
## without the stepwise nature of a histogram
ggplot(y,aes(x=Solar.R)) + geom_density()
```



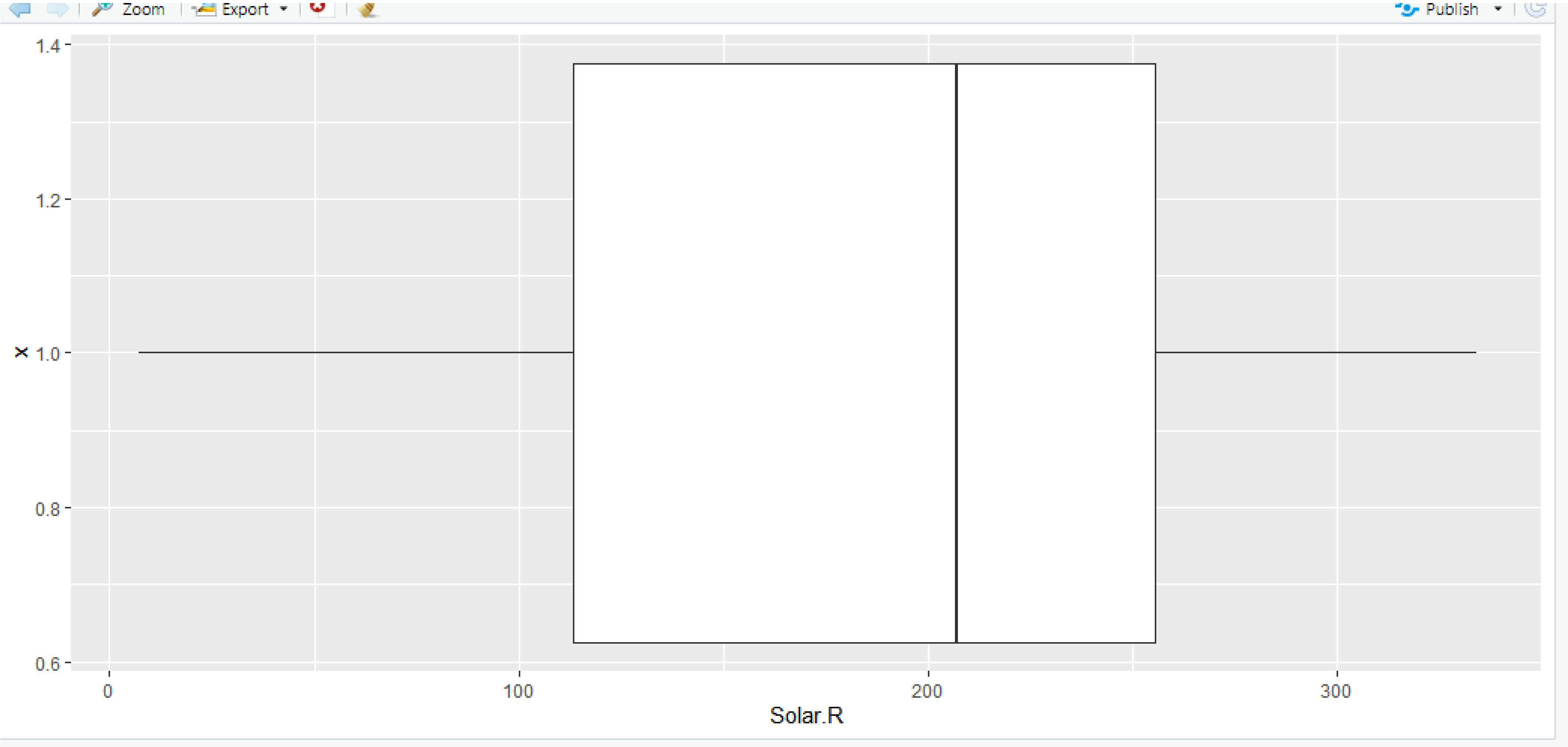
Boxplot

```
## Another view of distribution where you use a boxplot
ggplot(y,aes(x=1,y=Solar.R)) + geom_boxplot()
```



Boxplot (coord_flipped)

```
ggplot(y,aes(x=1,y=Solar.R)) + geom_boxplot() + coord_flip()
```



Faceted plots

```
## Temperature faceted by wind speeds
ggplot(y,aes(x=Temp)) + geom_histogram() + facet_wrap(~Wind)
```