Importing Data and Probability Distributions

EES 4891/5891

Probability & Statistics for Geosciences

Jonathan Gilligan

Class #9: Tuesday, February 04 2025

Learning Goals

Learning Goals

- Learn more about importing data from files
- Learn about combining data-frames by `binding` and `joining`
- Learn about several important probability distributions
  - Learn how to plot probability distributions in R

Getting Started

Getting Started

- Go to the GitHub Classroom and accept the “Practice with files” assignment:

  https://classroom.github.com/a/a83vTH7S

- Open RStudio and create a new project from version control, and give it the URL for the new assignment repository.
- This assignment is only for practice and there is nothing to turn in.

Importing Data

Importing Data

- File types:

Function	**Description**
`read_csv()`	Columns separated by commas
`read_csv2()`	Columns separated by semicolons
`read_tsv()`	Columns separated by tab characters
`read_table()`	Columns separated by any white-space
`read_delim()`	Columns separated by an arbitrary character
`read_fwf()`	Columns have fixed width

Using `read_csv()`, etc.

- `read_csv(‹filename›, ...)`
  - Optional arguments:

Argument	Description	Example
`col_names`	Names for the columns	`col_names = c(“year”, “month”, “precip”)`
`col_types`	Data types of each column	`col_types = cols(col_number(), col_character())`
`coL_select`	Only read certain columns	`col_select = starts_with(“cc_”)`
`na`	Cell contents to interpret as missing values	`na = c(“” , “NA”, “-99.99”)`
`comment`	Ignore everything after this character	`comment = “#”`
`skip`	Skip lines at the top	`skip = 9`
`name_repair`	Fix names of columns	`name_repair = “universal”`

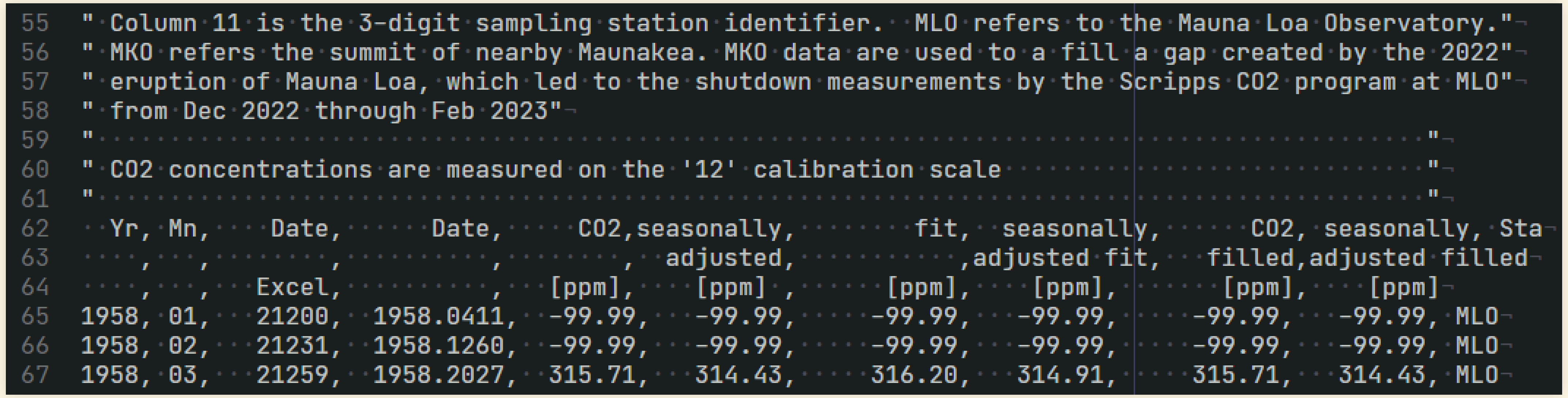
- I often like to load the package `janitor` and set `name_repair = make_clean_names` (no quotation marks).
- There are many other arguments. Look at the online help for `read_csv` for a complete listing.

R Exercise:

- In the `practice-with-files` project, open the file `read_co2.R`
- There should be a file in your project directory called `monthly_in_situ_co2_mlo.csv`

Read the File

- Open the file in RStudio:



```
55 " Column 11 is the 3-digit sampling station identifier.  MLO refers to the Mauna Loa Observatory."
56 " MKO refers the summit of nearby Maunakea. MKO data are used to a fill a gap created by the 2022"
57 " eruption of Mauna Loa, which led to the shutdown measurements by the Scripps CO2 program at MLO"
58 " from Dec 2022 through Feb 2023"
59 "                                                                                           "
60 " CO2 concentrations are measured on the '12' calibration scale                             "
61 "                                                                                           "
62   Yr, Mn,    Date,      Date,     CO2,seasonally,        fit,  seasonally,      CO2, seasonally, Sta
63     ,   ,        ,          ,        ,  adjusted,           ,adjusted fit,   filled,adjusted filled
64     ,   ,   Excel,          ,   [ppm],    [ppm] ,      [ppm],    [ppm],       [ppm],    [ppm]
65 1958, 01,   21200,  1958.0411,  -99.99,   -99.99,     -99.99,   -99.99,     -99.99,   -99.99, MLO
66 1958, 02,   21231,  1958.1260,  -99.99,   -99.99,     -99.99,   -99.99,     -99.99,   -99.99, MLO
67 1958, 03,   21259,  1958.2027,  315.71,   314.43,     316.20,   314.91,     315.71,   314.43, MLO
```

  - It begins with 61 lines of comments
  - The column names are spread across 3 rows

- Skip the first 64 rows and manually supply the column names:

Read the File into R

- Skip the first 64 rows and manually supply the column names:

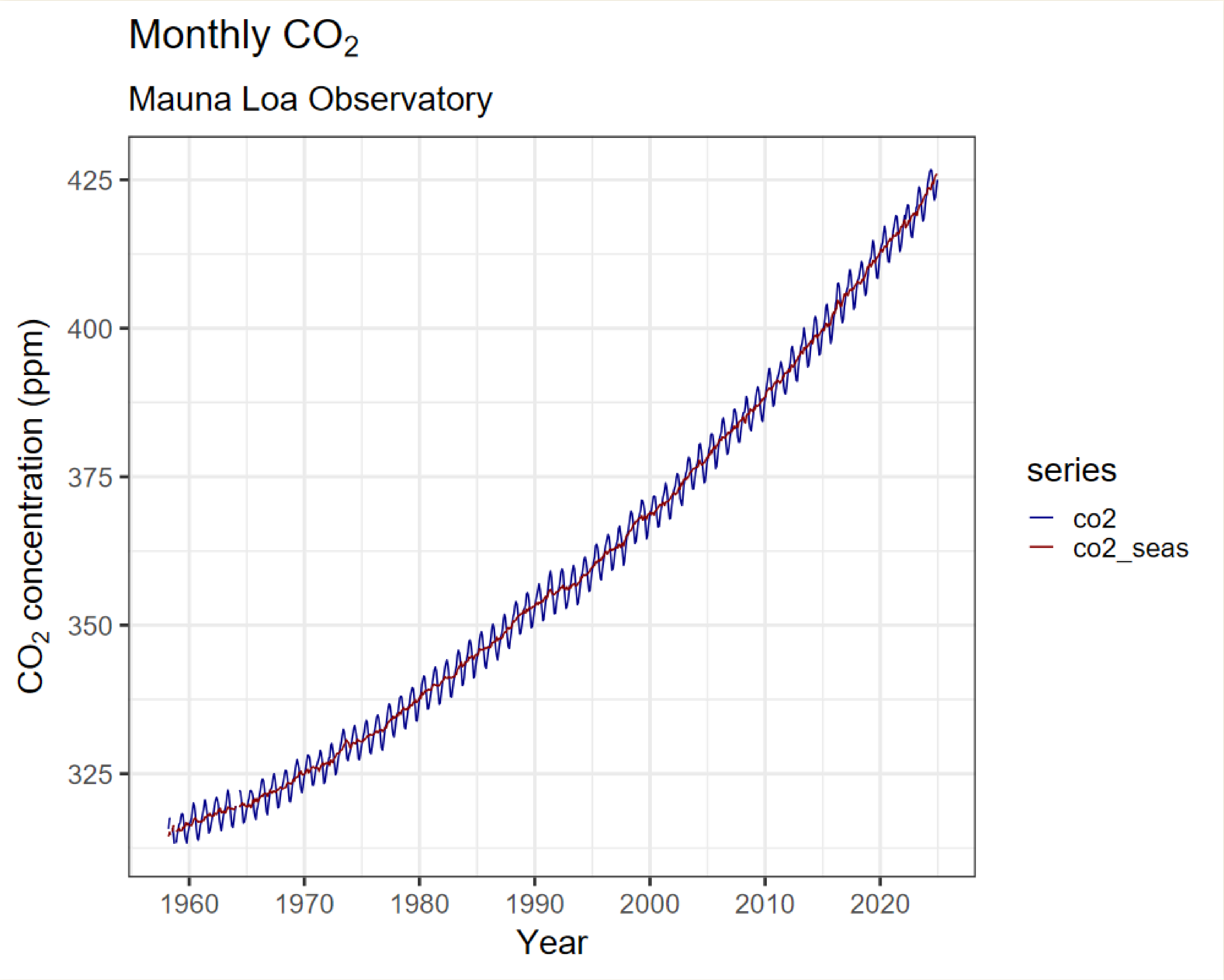
```r
co2 <- read_csv("monthly_in_situ_co2_mlo.csv", skip = 64,
                col_names = c("year", "month", "date_excel", "date",
                              "co2", "co2_seas",
                              "co2_fit", "co2_fit_seas",
                              "co2_filled", "co2_filled_seas",
                              "station"),
                col_select = c("year", "month", "date",
                               "co2", "co2_seas",
                               "station"),
                na = "-99.99")

head(co2)
```

```
## # A tibble: 6 × 6
##    year month  date   co2 co2_seas station
##   <dbl> <chr> <dbl> <dbl>    <dbl> <chr>
## 1  1958 01    1958.    NA        NA  MLO
## 2  1958 02    1958.    NA        NA  MLO
## 3  1958 03    1958.   316.     314. MLO
## 4  1958 04    1958.   317.     315. MLO
## 5  1958 05    1958.   318.     315. MLO
## 6  1958 06    1958.    NA        NA  MLO
```

Plot CO_2

```
co2 |> pivot_longer(c(co2, co2_seas), names_to = "series",
                    values_to = "co2") |>
  ggplot(aes(x = date, y = co2, color = series)) +
  geom_line(line_thickness = 1) +
  scale_color_manual(values = c(co2 = "darkblue",
                                co2_seas = "darkred")) +
  scale_x_continuous(breaks = seq(1950, 2025, 10)) +
  labs(x = "Year",
       y = expression(paste(CO[2], " concentration
         (ppm)")),
       title = expression(paste("Monthly ", CO[2])),
       subtitle = "Mauna Loa Observatory")
```



- You could use

  ```
  labs(x = "Year", y = "CO2 concentration (ppm)",
       title = "Monthly CO2",
       subtitle = "Mauna Loa Observatory")
  ```

  but this example illustrates how to use mathematical notation, such as subscripts, in plot captions

Other Kinds of Files

- R also has its own efficient format for reading and writing files:
  - `read_rds()` and `write_rds()` read and write a single R object to a `.Rds` file.
- The `readxl` and `writexl` packages have functions for reading and writing Excel `.xls` and `.xlsx` spreadsheet files.
- The `rjson` and `jsonlite` packages have functions `toJSON()` and `fromJSON()` to read and write JSON files
- The `sf` package has the functions `read_sf()` and `write_sf()` to read and write GIS shapefiles
- The `rvest` package lets you automate scraping data off web pages
- The `DBI` and `dbplyr` packages have functions to read and write from common databases:
  - SQLite
  - MySQL
  - Postgres
  - …
  - See chapter 21 for more details.
- And lots more…
- For this course, we'll focus on reading and writing text files (`csv`, etc.)

Combining Data From Multiple Sources

Starting up

- Open the script `joins.R` in RStudio

Binding Rows and Columns

- bind_rows() combines multiple data frames, row by row

```
df_1 <- tibble(num = 1:5, letter = letters[num])
df_2 <- tibble(num = 15:20, letter = letters[num])

bind_rows(df_1, df_2)
```

```
## # A tibble: 11 × 2
##      num letter
##    <int> <chr>
##  1     1 a
##  2     2 b
##  3     3 c
##  4     4 d
##  5     5 e
##  6    15 o
##  7    16 p
##  8    17 q
##  9    18 r
## 10    19 s
## 11    20 t
```

- bind_cols() combines column by column
  - Rows must be in the same order in both data frames

```
df_3 <- tibble(num = 1:10)
df_4 <- tibble(letter = letters[1:10])

bind_cols(df_3, df_4)
```

```
## # A tibble: 10 × 2
##      num letter
##    <int> <chr>
##  1     1 a
##  2     2 b
##  3     3 c
##  4     4 d
##  5     5 e
##  6     6 f
##  7     7 g
##  8     8 h
##  9     9 i
## 10    10 j
```

Joining Data Frames

- `full_join()`, `right_join()`, `full_join()`, `inner_join()` combine data frames by matching corresponding columns

```
state_pop <- tibble(state = c("AL", "GA", "MS", "TN"),
                    pop = c(5157699, 11180878, 2943045,
        7227750))

state_gdp <- tibble(state = c("AL", "TN", "MS", "GA"),
                    gdp = c(318080, 545695, 156026,
        877746))

full_join(state_pop, state_gdp)
```

```
## # A tibble: 4 × 3
##   state      pop    gdp
##   <chr>    <dbl>  <dbl>
## 1 AL     5157699 318080
## 2 GA    11180878 877746
## 3 MS     2943045 156026
## 4 TN     7227750 545695
```

- If the `by` column is the same in all data frames, `full_`, `left_`, `_right_`, and `innter_` are the same
  - If the `by` column is different, different joins keep different sets of rows.

Full & Inner Joins

```
df_left
```

```
## # A tibble: 5 × 2
##   month  days
##   <chr> <dbl>
## 1 Jan      31
## 2 Feb      28
## 3 Mar      31
## 4 Apr      30
## 5 May      31
```

```
df_right
```

```
## # A tibble: 4 × 2
##   month order
##   <chr> <dbl>
## 1 Jan       1
## 2 Feb       2
## 3 Jun       6
## 4 Jul       7
```

```
full_join(df_left, df_right, by = "month")
```

```
## # A tibble: 7 × 3
##   month  days order
##   <chr> <dbl> <dbl>
## 1 Jan      31     1
## 2 Feb      28     2
## 3 Mar      31    NA
## 4 Apr      30    NA
## 5 May      31    NA
## 6 Jun      NA     6
## 7 Jul      NA     7
```

```
inner_join(df_left, df_right, by = "month")
```

```
## # A tibble: 2 × 3
##   month  days order
##   <chr> <dbl> <dbl>
## 1 Jan      31     1
## 2 Feb      28     2
```

Left & Right Joins

```
df_left
```

```
## # A tibble: 5 × 2
##   month  days
##   <chr> <dbl>
## 1 Jan      31
## 2 Feb      28
## 3 Mar      31
## 4 Apr      30
## 5 May      31
```

```
df_right
```

```
## # A tibble: 4 × 2
##   month order
##   <chr> <dbl>
## 1 Jan       1
## 2 Feb       2
## 3 Jun       6
## 4 Jul       7
```

```
left_join(df_left, df_right, by = "month")
```

```
## # A tibble: 5 × 3
##   month  days order
##   <chr> <dbl> <dbl>
## 1 Jan      31     1
## 2 Feb      28     2
## 3 Mar      31    NA
## 4 Apr      30    NA
## 5 May      31    NA
```

```
right_join(df_left, df_right, by = "month")
```

```
## # A tibble: 4 × 3
##   month  days order
##   <chr> <dbl> <dbl>
## 1 Jan      31     1
## 2 Feb      28     2
## 3 Jun      NA     6
## 4 Jul      NA     7
```

Probability Distributions

Getting Started

- Open the file `prob_dist.R` in RStudio

Common Probability Distributions

- Discrete Distributions:
  - **Binomial:** Tossing coins
  - **Poisson:** Total counts over a long time
  - **Geometric:** How long until something happens?

- Continuous Distributions:
  - **Normal:** Focus for Thursday
    - **Log-Normal:** For numbers that must be ≥ 0
  - **Gamma:** For numbers that must be ≥ 0. Very flexible
    - **Exponential:** Special case of Gamma
    - **Chi-Squared:** Another special case
  - **Weibull:** Extreme values (hurricanes, floods, earthquakes, etc.)

Probability Distributions in R

- Probability distribution functions in R:
  - Many families of distributions
  - Consistent organization:
    - `rnorm(n, mean sd)`: **sample** `n` random numbers from a normal distribution
    - `dnorm(x, mean, sd)`: get the **probability density** for a normal distribution at `x`
    - `qnorm(p, mean, sd)`: get the **quantile** for probability `p`: what value of x has cumulative probability p?
    - `pnorm(q, mean, sd)`: get the **cumulative probability** at q.
    - `pnorm` and `qnorm` are inverses: `pnorm(qnorm(x)) = x`, for $0 < x < 1$, and `qnorm(pnorm(x)) = x` as long as x is not ridiculously large.

Probability Distrbutions in R

Name	R functions
Normal	`rnorm,dnorm,pnorm,qnorm`
Lognormal	`rlnorm,dlnorm,plnorm,qlnorm`
Beta	`rbeta,dbeta,pbeta,qbeta`
Cauchy	`rcauchy,dcauchy,pcauchy,qcauchy`
Chi Squared	`rchisq,dchisq,pchisq,qchisq`
Exponential	`rexp,dexp,pexp,qexp`
Gamma	`rgamma,dgamma,pgamma,qgamma`
Uniform	`runif,dunif,punif,qunif`
Weibull	`rweibull,dweibull,pweibull,qweibull`

Name	R functions
Binomial	`rbinom,dbinom,pbinom,qbinom`
Poisson	`rpois,dpois,ppois,qpois`
Geometric	`rgeom,dgeom,pgeom,qgeom`

- These are common distributions. There are many others as well.

Binomial Distribution

- Number of heads for tossing a coin n times, with probability p of coming up heads on any toss.

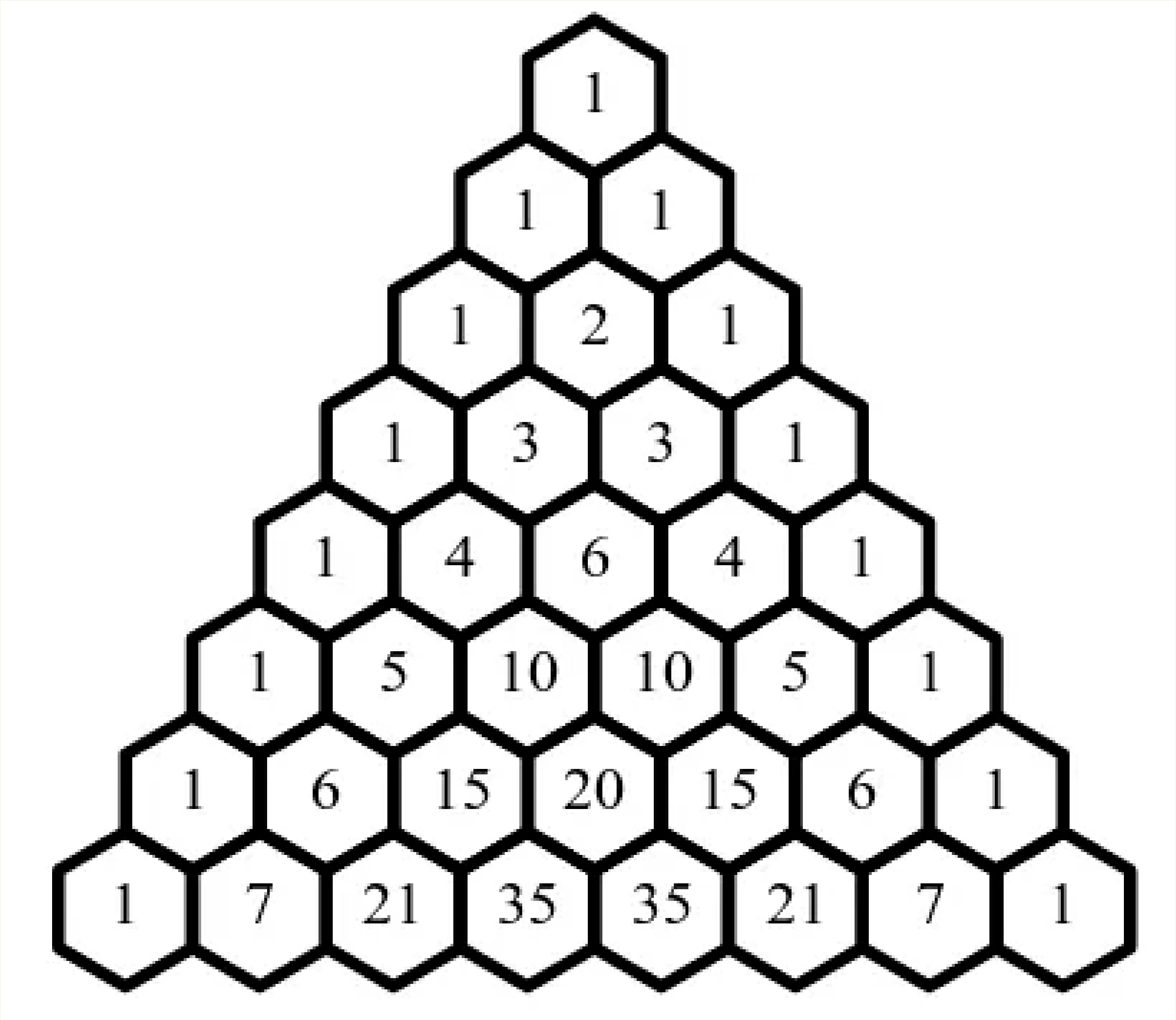
$$X \sim \mathcal{B}(n, p)$$

- Probability of k heads in n tosses:

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k},$$

where the *binomial coefficient*

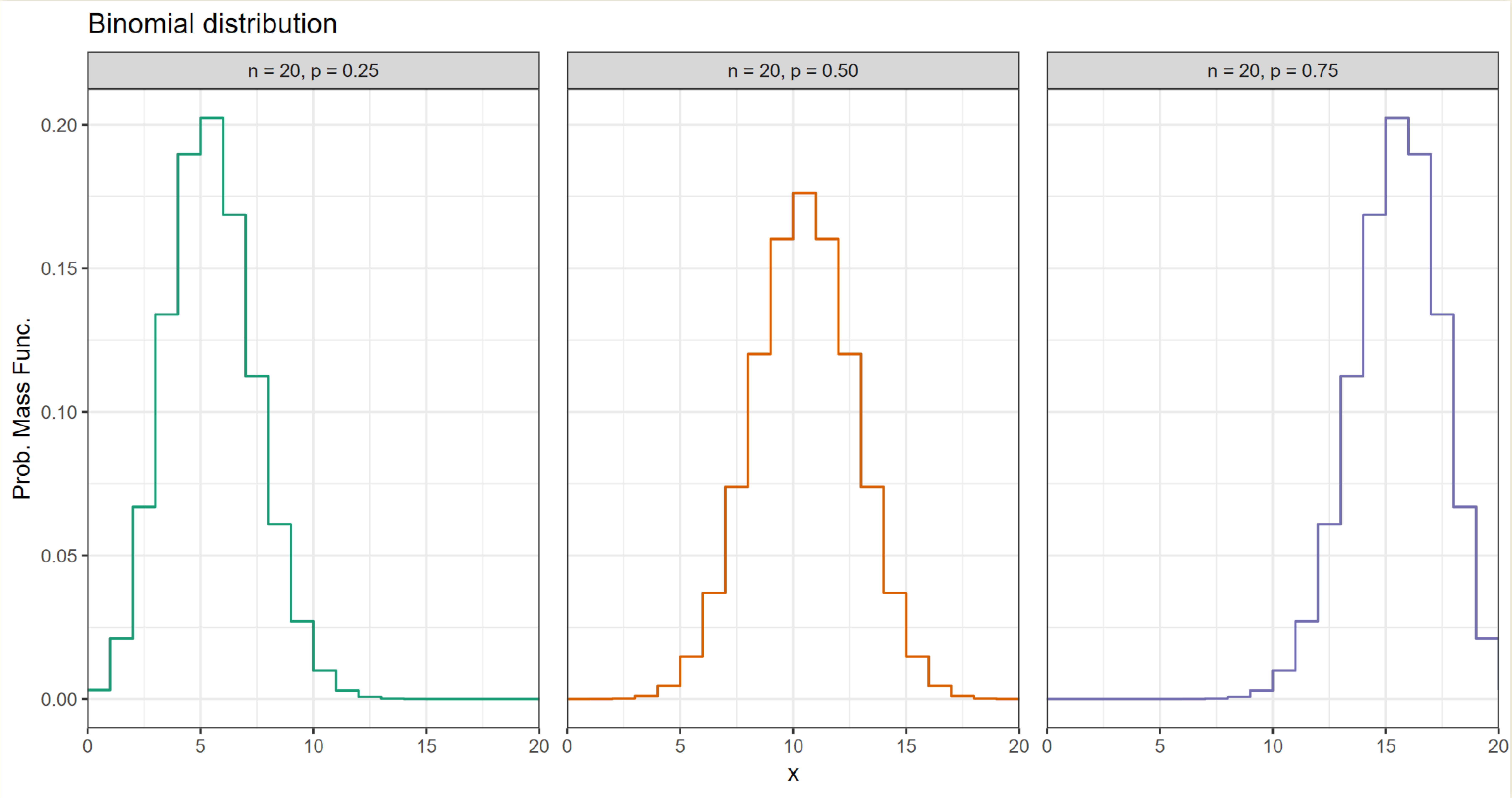
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$



1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

$\binom{n}{k}$ is the number of different ways to get k heads in n tosses.

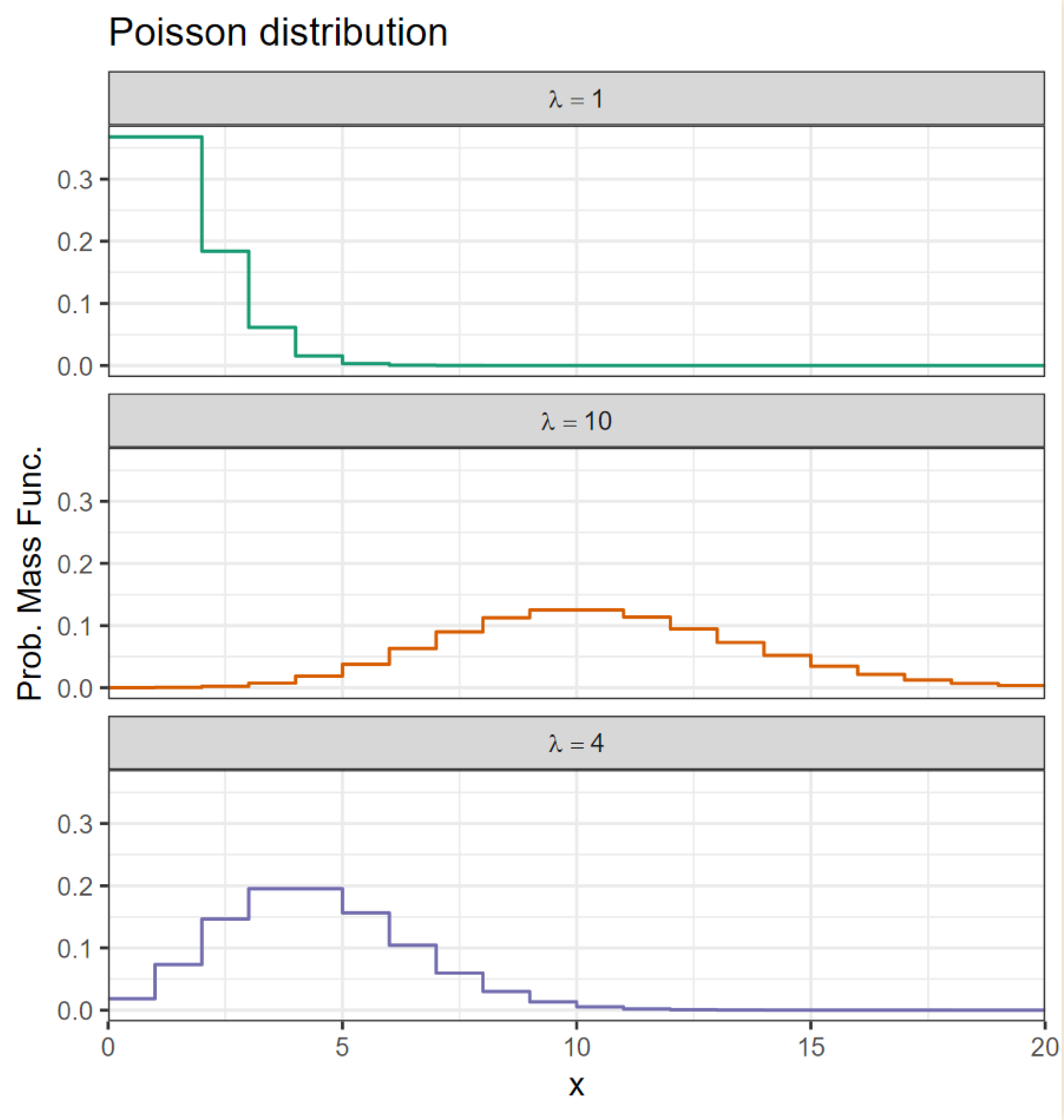
Binomial Distribution



Poisson Distribution

- Suppose you're tossing a coin where $p \ll 1$ and n is large.
- $\lambda = p \times n$ is the average number of heads you expect, but you'll often see more or fewer.
  - The Poisson distribution describes the probability of k heads when $n \times p = \lambda$

$$\mathbb{P}(X = k) = e^{-\lambda} \frac{\lambda^k}{k!}$$



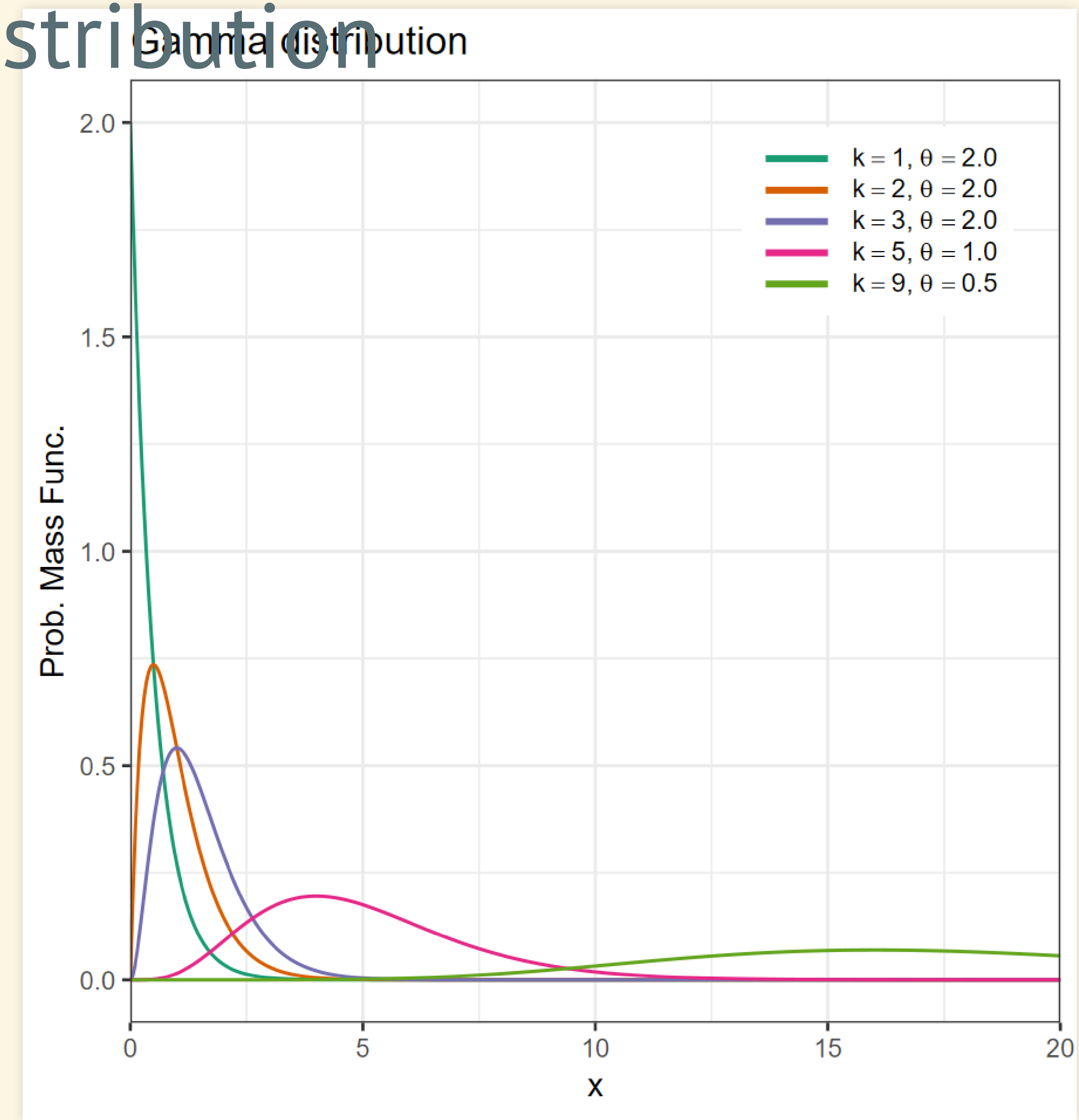
Normal Distribution

- We’ll spend Thursday talking about this one…

Gamma Distribution

Weibull Distribution

- Characterized by 2 numbers:
  - $k = shape$
  - $\theta = scale$
    - or $1/\theta = rate$
  - **mean** $= k\theta$
- $\mathcal{P}(x, k, \theta) > 0$ only for $x \geq 0$
- Two parameters make this very flexible
- Good for things that are ≥ 0:
  - Rainfall
- Special cases:
  - $k = 1$: exponential distribution
  - $\theta = 2$: Chi-squared distribution


Gamma distribution
Prob. Mass Func.
x
k = 1, θ = 2.0
k = 2, θ = 2.0
k = 3, θ = 2.0
k = 5, θ = 1.0
k = 9, θ = 0.5