Notas Test 2

- and y or se usan con un solo operador, es decir : & y |

-
```
many_airports <- flights %>%
  filter(dest == "SEA" | dest == "SFO" | dest == "PDX" |
         dest == "BTV" | dest == "BDL")
```
es igual a
```
many_airports <- flights %>%
  filter(dest %in% c("SEA", "SFO", "PDX", "BTV", "BDL"))
View(many_airports)
```

-
```
summary_temp <- weather %>%
  summarize(mean = mean(temp, na.rm = TRUE),
            std_dev = sd(temp, na.rm = TRUE))
summary_temp
```

- other summary functions we can use inside the `summarize()` verb to compute summary statistics:
    - `mean()`: the average
    - `sd()`: the standard deviation, which is a measure of spread
    - `min()` and `max()`: the minimum and maximum values, respectively
    - `IQR()`: interquartile range
    - `sum()`: the total amount when adding multiple numbers
    - `n()`: a count of the number of rows in each group.

- Say instead of a single mean temperature for the whole year, you would like 12 mean temperatures, one for each of the 12 months separately. We can do this by "grouping" temperature observations by the values of another variable, in this case by the 12 values of the variable `month`
- You are not limited to grouping by one variable
- To close out our discussion on the `mutate()` function to create new variables, note that we can create multiple new variables at once in the same `mutate()` code
- `arrange()` always returns rows sorted in ascending order by default. To switch the ordering to be in "descending" order instead, we use the `desc()` function as so:
```
freq_dest %>%
  arrange(desc(num_flights))
```
- Match primary keys:

```
flights_joined <- flights %>%
  inner_join(airlines, by = "carrier")
View(flights)
View(flights_joined)
```

- If primary key have different names:

```
flights_with_airport_names <- flights %>%
  inner_join(airports, by = c("dest" = "faa"))
View(flights_with_airport_names)
```

- join two data frames by *multiple key variables (multiple primary keys*

```
flights_weather_joined <- flights %>%
  inner_join(weather, by = c("year", "month", "day", "hour", "origin"))
View(flights_weather_joined)
```

- The *key* variable(s) that we base our joins on are often *identification variables* as we mentioned previously. This is an important property of what's known as *normal forms* of data. The process of decomposing data frames into less redundant tables without losing information is called *normalization*.
- Other dplyr verbs:
  - `glimpse()`: identify the names of these variables by running the `glimpse()` function
  - `select()` : This function makes it easier to explore large datasets since it allows us to limit the scope to only those variables we care most about. To deselect: `select(-year)`. Another way of selecting columns/variables is by specifying a range of columns:

    ```
    flight_arr_times <- flights %>% select(month:day, arr_time:sched_arr_time)
    flight_arr_times
    ```

    The `select()` function can also be used to reorder columns when used with the `everything()` helper function. `everything()` will pick up all remaining variables
  - Lastly, the helper functions `starts_with()`, `ends_with()`, and `contains()` can be used to select variables/columns that match those conditions.

    ```
    flights %>% select(starts_with("a"))
    flights %>% select(ends_with("delay"))
    flights %>% select(contains("time"))
    ```

- `rename()`, which as you may have guessed changes the name of variables

```
flights_time_new <- flights %>%
  select(dep_time, arr_time) %>%
  rename(departure_time = dep_time, arrival_time = arr_time)
glimpse(flights_time_new)
```

- We can also return the top `n` values of a variable using the `top_n()` function. For example, we can return a data frame of the top 10 destination airports. n specifies number and wt with the column…

```
named_dests  %>%
  top_n(n = 10, wt = num_flights) %>%
  arrange(desc(num_flights))
```

TABLE 3.2: Summary of data wrangling verbs

Verb	Data wrangling operation
`filter()`	Pick out a subset of rows
`summarize()`	Summarize many values to one using a summary statistic function like `mean()`, `median()`, etc.
`group_by()`	Add grouping structure to rows in data frame. Note this does not change values in data frame, rather only the meta-data
`mutate()`	Create new variables by mutating existing ones
`arrange()`	Arrange rows of a data variable in ascending (default) or `desc` ending order
`inner_join()`	Join/merge two data frames, matching rows by a key variable

Tuftes design principles:

- Maximize data-ink ratio: Data-ink ratio = data-ink / total ink used
- Erase non-data ink
- Erase redundant data ink
- Revise and edit

Tufte's key data visualization principles:

- Graphical integrity: Visual representations of data must tell the truth. This happens when the lie factor is greater than 1. Has six principles:

    - The representation of numbers, as physically measured on the surface of the graph itself, should be directly proportional to the numerical quantities represented
    - Clear, detailed and thorough labelling should be used to defeat graphical distortion and ambiguity. Write out explanations of the data on the graph itself. Label important events in the data.
    - Show data variation, not design variation.
    - In time-series displays of money, deflated and standardized units of monetary measurement are nearly always better than nominal units.
    - The number of information carrying (variable) dimensions depicted should not exceed the number of dimensions in the data. Graphics must not quote data out of context.
- Data Ink is the ink on a graph that represents data. Tufte claims that good graphical representations maximize data-ink and erase as much non-data-ink as possible. He put forward the data-ink ratio which is calculated by 1 minus the proportion of the graph that can be erased without loss of data-information.nHe puts forward the following 5 principles related to data ink :

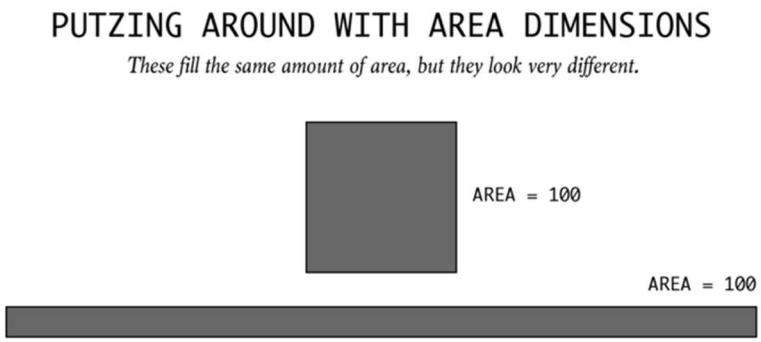
    1. Above all else show data.
    2. Maximize the data-ink ratio.
    3. Erase non-data-ink.
    4. Erase redundant data-ink.
    5. Revise and edit

- Chartjunk – the excessive and unnecessary use of graphical effects in graphs.
- Data Density - The data density of a graph is the proportion of the total size of the graph that is dedicated displaying data.
- Small multiples are series of the same small graph repeated in one visual. Tufte says that small multiples are a great tool to visualize large quantities of data and with a high number of dimensions.

Visualization Lies:

- Truncated Axis (doesn't start at zero) -> Disappearing baseline?
- Dual Axes that don't correspond
- Doesn't add up (piechart)
- Seeing only in absolutes
- Limited scope (cherrypicking)
- Odd choice of binning(more bins but not too complex)
- Area sized by single dimension (30 = 10*3 but most be linear and not by area if not will look much bigger)

- Area can be distributed weirdly



- Don't do 3d
- dimensions in data not matching dimensions in the graph

Violations of Graphical Integrity:

1. The Lie Factor
2. Scale Distortions and Disappearing Baseline
3. Design Distortions
4. Different Dimensions in Data and Graphic
5. Failing to Correct for Inflation
6. Lack of Context

-Lie factor:

LIE_effect = SIZE_effect_graphic / SIZE_effect_data

SIZE = (VALUE_one - VALUE_two) / VALUE_two

- Other errors:
    - Using a color palette that colorblind people can't see
    - Using colors that look too similar to each other
    - Not making scatterplots when you should (using something more complicated or more summarized, like boxplots)
    - Failing to take logs when data is very right-skewed
    - Using some standard graph instead of focusing on the point of the graph.
    - Lining up facets vertically when they should be horizontal – be sure to line up so the important comparison can be made..

How to better graphs:

1. Remove backgrounds
2. Remove duplicate labels
3. Remove borders in chart elements and background

4. Reduce colors
5. Remove special effects like shading and shadows
6. Remove bold text
7. Lighten Secondary Data Labels
8. Lighten or remove lines
9. Remove Y-Axis labels and label the

A small multiple (sometimes called trellis chart, lattice chart, grid chart, or panel chart) is a series of similar graphs or charts using the same scale and axes, allowing them to be easily compared. It uses multiple views to show different partitions of a dataset.

A sparkline is a very small line chart, typically drawn without axes or coordinates.

Some Strategies for Increasing Data-Ink Ratio

1. Avoid Chartjunk
2. Small Multiples
3. Sparklines
4. Simplify! (Erase non-data and redundant ink)

The best graphs are:

Data Dense
Uncluttered
Not Redundant
Not Misleading (e.g., correct for inflation)
Tell a Story..