Exercise 2: Tutorial

Due: Friday, Week 2

Here we learn to use ipynb notebooks and a few Python libraries that will be helpful for working with data.

In [1]:
```
%pip install "altair[all]"
%pip install "vegafusion[embed]"
```

```
Requirement already satisfied: altair[all] in /Users/chrislowzx/miniconda3/lib/python3.13/site-pack
ages (5.5.0)
Requirement already satisfied: jinja2 in /Users/chrislowzx/miniconda3/lib/python3.13/site-packages
(from altair[all]) (3.1.6)
Requirement already satisfied: jsonschema>=3.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site-
packages (from altair[all]) (4.25.0)
Requirement already satisfied: narwhals>=1.14.2 in /Users/chrislowzx/miniconda3/lib/python3.13/site
-packages (from altair[all]) (2.7.0)
Requirement already satisfied: packaging in /Users/chrislowzx/miniconda3/lib/python3.13/site-packag
es (from altair[all]) (25.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /Users/chrislowzx/miniconda3/lib/python
3.13/site-packages (from altair[all]) (4.15.0)
Requirement already satisfied: altair-tiles>=0.3.0 in /Users/chrislowzx/miniconda3/lib/python3.13/s
ite-packages (from altair[all]) (0.4.0)
Requirement already satisfied: anywidget>=0.9.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site
-packages (from altair[all]) (0.9.18)
Requirement already satisfied: numpy in /Users/chrislowzx/miniconda3/lib/python3.13/site-packages (
from altair[all]) (2.3.3)
Requirement already satisfied: pandas>=1.1.3 in /Users/chrislowzx/miniconda3/lib/python3.13/site-pa
ckages (from altair[all]) (2.2.2)
Requirement already satisfied: pyarrow>=11 in /Users/chrislowzx/miniconda3/lib/python3.13/site-pack
ages (from altair[all]) (21.0.0)
Requirement already satisfied: vega-datasets>=0.9.0 in /Users/chrislowzx/miniconda3/lib/python3.13/
site-packages (from altair[all]) (0.9.0)
Requirement already satisfied: vegafusion>=1.6.6 in /Users/chrislowzx/miniconda3/lib/python3.13/sit
e-packages (from vegafusion[embed]>=1.6.6; extra == "all"->altair[all]) (2.0.3)
```

```
Requirement already satisfied: vl-convert-python>=1.7.0 in /Users/chrislowzx/miniconda3/lib/python
3.13/site-packages (from altair[all]) (1.8.0)
Requirement already satisfied: mercantile in /Users/chrislowzx/miniconda3/lib/python3.13/site-packa
ges (from altair-tiles>=0.3.0->altair[all]) (1.2.1)
Requirement already satisfied: xyzservices in /Users/chrislowzx/miniconda3/lib/python3.13/site-pack
ages (from altair-tiles>=0.3.0->altair[all]) (2025.4.0)
Requirement already satisfied: ipywidgets>=7.6.0 in /Users/chrislowzx/miniconda3/lib/python3.13/sit
e-packages (from anywidget>=0.9.0->altair[all]) (8.1.5)
Requirement already satisfied: psygnal>=0.8.1 in /Users/chrislowzx/miniconda3/lib/python3.13/site-p
ackages (from anywidget>=0.9.0->altair[all]) (0.14.2)
Requirement already satisfied: comm>=0.1.3 in /Users/chrislowzx/miniconda3/lib/python3.13/site-pack
ages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.2.1)
Requirement already satisfied: ipython>=6.1.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site-p
ackages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (9.1.0)
Requirement already satisfied: traitlets>=4.3.1 in /Users/chrislowzx/miniconda3/lib/python3.13/site
-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (5.14.3)
Requirement already satisfied: widgetsnbextension~=4.0.12 in /Users/chrislowzx/miniconda3/lib/pytho
n3.13/site-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (4.0.13)
Requirement already satisfied: jupyterlab-widgets~=3.0.12 in /Users/chrislowzx/miniconda3/lib/pytho
n3.13/site-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (3.0.15)
Requirement already satisfied: decorator in /Users/chrislowzx/miniconda3/lib/python3.13/site-packag
es (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (5.2.1)
Requirement already satisfied: ipython-pygments-lexers in /Users/chrislowzx/miniconda3/lib/python3.
13/site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (1.1.1)
Requirement already satisfied: jedi>=0.16 in /Users/chrislowzx/miniconda3/lib/python3.13/site-packa
ges (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.19.2)
Requirement already satisfied: matplotlib-inline in /Users/chrislowzx/miniconda3/lib/python3.13/sit
e-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /Users/chrislowzx/miniconda3/lib/python3.13/site-pack
ages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (4.9.0)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in /Users/chrislowzx/miniconda3/lib/py
thon3.13/site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (3.
0.43)
Requirement already satisfied: pygments>=2.4.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site-
packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (2.19.1)
Requirement already satisfied: stack_data in /Users/chrislowzx/miniconda3/lib/python3.13/site-packa
ges (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.6.3)
Requirement already satisfied: wcwidth in /Users/chrislowzx/miniconda3/lib/python3.13/site-packages
(from prompt_toolkit<3.1.0,>=3.0.41->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[al
l]) (0.2.13)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /Users/chrislowzx/miniconda3/lib/python3.13/s
ite-packages (from jedi>=0.16->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (
0.8.4)
Requirement already satisfied: attrs>=22.2.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site-pa
ckages (from jsonschema>=3.0->altair[all]) (24.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /Users/chrislowzx/miniconda
3/lib/python3.13/site-packages (from jsonschema>=3.0->altair[all]) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in /Users/chrislowzx/miniconda3/lib/python3.13/s
ite-packages (from jsonschema>=3.0->altair[all]) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in /Users/chrislowzx/miniconda3/lib/python3.13/site-p
ackages (from jsonschema>=3.0->altair[all]) (0.22.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/chrislowzx/miniconda3/lib/python3.1
3/site-packages (from pandas>=1.1.3->altair[all]) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /Users/chrislowzx/miniconda3/lib/python3.13/site-pac
kages (from pandas>=1.1.3->altair[all]) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /Users/chrislowzx/miniconda3/lib/python3.13/site-p
ackages (from pandas>=1.1.3->altair[all]) (2025.2)
Requirement already satisfied: ptyprocess>=0.5 in /Users/chrislowzx/miniconda3/lib/python3.13/site-
packages (from pexpect>4.3->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.7.
0)
Requirement already satisfied: six>=1.5 in /Users/chrislowzx/miniconda3/lib/python3.13/site-package
s (from python-dateutil>=2.8.2->pandas>=1.1.3->altair[all]) (1.17.0)
Requirement already satisfied: arro3-core in /Users/chrislowzx/miniconda3/lib/python3.13/site-packa
ges (from vegafusion>=1.6.6->vegafusion[embed]>=1.6.6; extra == "all"->altair[all]) (0.6.3)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site-
packages (from jinja2->altair[all]) (3.0.2)
Requirement already satisfied: click>=3.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site-packa
ges (from mercantile->altair-tiles>=0.3.0->altair[all]) (8.3.0)
Requirement already satisfied: executing>=1.2.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site
-packages (from stack_data->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (2.2.
1)
Requirement already satisfied: asttokens>=2.1.0 in /Users/chrislowzx/miniconda3/lib/python3.13/site
-packages (from stack_data->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (3.0.
0)
Requirement already satisfied: pure_eval in /Users/chrislowzx/miniconda3/lib/python3.13/site-packag
es (from stack_data->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.2.3)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: vegafusion[embed] in /Users/chrislowzx/miniconda3/lib/python3.13/sit
```

```
e-packages (2.0.3)
Requirement already satisfied: arro3-core in /Users/chrislowzx/miniconda3/lib/python3.13/site-packa
ges (from vegafusion[embed]) (0.6.3)
Requirement already satisfied: packaging in /Users/chrislowzx/miniconda3/lib/python3.13/site-packag
es (from vegafusion[embed]) (25.0)
Requirement already satisfied: narwhals>=1.42 in /Users/chrislowzx/miniconda3/lib/python3.13/site-p
ackages (from vegafusion[embed]) (2.7.0)
Note: you may need to restart the kernel to use updated packages.
```

Computational notebooks

One nice affordance of computational notebooks is that we can interleave blocks of text (like this one) written in Markdown with chunks of `code` written in Python in our case. This style of coding is called **"literate programming"**. It facilitates process documentation, reporducibile analyses, and reflection on analysis choices.

Some advised practices for literate programming are:

1. Split up your code into distinct high level operations (e.g., load data).
2. Say why you are doing things. Why did you transform the data?
3. Interpret your charts in text. What patterns do you see? What do they mean for your analysis?
4. Comment your code if syntax isn't obvious.

The idea is that somebody unfamiliar with your work should be able to read this documentation of your work and repeat your analysis for themself, including your thought process. This is critical to doing data science.

Practice!

Let's practice literate programming by filling in the brief analysis below. Each time you see a codeblock with the comment `# PROMPT:...` or text *PROMPT:* ..., you should write code or text to complete the analysis.

For most of the quarter, you will write documents like this one yourself. However, you will first ease into literate programming by filling in this document.

Altair can take data from a variety tabular data formats

In fact, datasets for chart need to contain a collection of records (rows) and data fields (columns). A `Chart` can accept a dataset (as its first argument) in one of many forms, including

There are different ways of specifying the dataset, including:

- A pandas DataFrame
- An alt.Data object
- A url string pointing to a json or csv formatted text file
- A geopandas GeoDataFrame
- Shapely Geometries or GeoJSON Objects

Loading data

One of the first things we will want to do in many computational notebooks is load a dataset. In our case, you will load the `storms.csv` dataset.

In [2]:
```
# PROMPT: load the storms dataset based on its relative path
#          (you may have to change the path)
import pandas as pd

df = pd.read_csv('../exercise_data/storms.csv')
len(df)
```

Out[2]:
```
19066
```

Let's have a first look at this dataset.

In [3]:
```
# PROMPT: use the head method to see the data in a table
df.head(10)
```

Out[3]:

	Unnamed: 0	name	year	month	day	hour	lat	long	status	category	wind	pressure	tropicalstorm_
0	1	Amy	1975	6	27	0	27.5	-79.0	tropical depression	NaN	25	1013	
1	2	Amy	1975	6	27	6	28.5	-79.0	tropical depression	NaN	25	1013	
2	3	Amy	1975	6	27	12	29.5	-79.0	tropical depression	NaN	25	1013	
3	4	Amy	1975	6	27	18	30.5	-79.0	tropical depression	NaN	25	1013	
4	5	Amy	1975	6	28	0	31.5	-78.8	tropical depression	NaN	25	1012	
5	6	Amy	1975	6	28	6	32.4	-78.7	tropical depression	NaN	25	1012	
6	7	Amy	1975	6	28	12	33.3	-78.0	tropical depression	NaN	25	1011	
7	8	Amy	1975	6	28	18	34.0	-77.0	tropical depression	NaN	30	1006	
8	9	Amy	1975	6	29	0	34.4	-75.8	tropical storm	NaN	35	1004	
9	10	Amy	1975	6	29	6	34.0	-74.8	tropical storm	NaN	40	1002	

PROMPT: what do you notice about the column labels? What do you notice as you scan across rows of the data table?

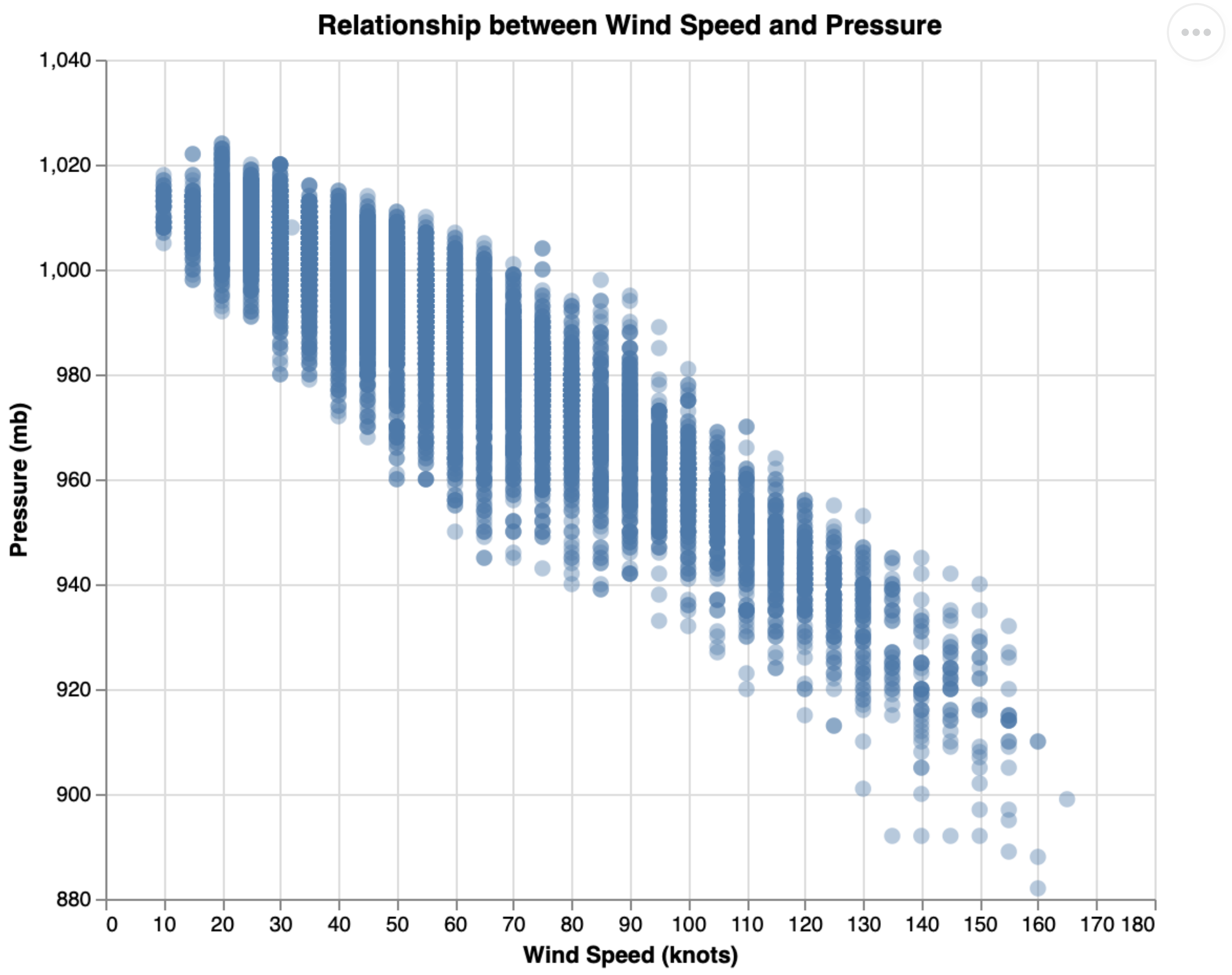
Visual exploration

Now we will practice making queries against this dataset and rendering views to show us specific combinations of variables or cross sections of the data. Creating some of these views will require a little bit of data wrangling.

(See `altair.X` documentation, e.g., to recall the syntax and check how the zero values are not included.)

In [4]:
```
# PROMPT: use the altair library to investigate the relationship between wind and pressure
import altair as alt
import vegafusion as vf

alt.data_transformers.enable("vegafusion", max_rows=50000)

df2 = df

# Build the scatter plot
storm_chart = alt.Chart(df2).mark_circle(size=60, opacity=0.4).encode(
    x=alt.X('wind:Q', title='Wind Speed (knots)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    tooltip=['name', 'year', 'wind', 'pressure']
).properties(
    width=500,
    height=400,
    title='Relationship between Wind Speed and Pressure'
)

storm_chart
```

Out[4]:



In [5]:

```
# # PROMPT: use the altair library to investigate the relationship between wind and pressure
# import altair as alt
# import vegafusion as vf

# # VegaFusion provides efficient Rust implementations of most of Altair’s data transformations.
# # Activating if often greatly reduces the size of the datasets that must be included in the final
# # In addition, VegaFusion automatically removes unused columns, which reduces dataset size even w

# alt.data_transformers.enable("vegafusion", max_rows=40000)
# # vf.enable(row_limit=50000)
# ### For some reason the MaxRowError is still raised...
# df2 = df.sample(5000, axis=0)

# storm_chart = alt.Chart(df2).mark_circle().encode(
#     x=alt.X('wind:Q', scale=alt.Scale(zero=False)),
#     y=alt.Y('pressure:Q', scale=alt.Scale(zero=False))
# )

# vf.transformed_data(storm_chart).head()

# storm_chart
```

PROMPT: what do you notice about the chart you just made?

Answer: The chart shows a clear negative relationship between wind speed and pressure. As wind speeds rise, pressure drops, forming a tight downward trend. Most points cluster around lower wind speeds with higher pressures, likely weaker storms, while stronger storms appear at the lower end of the pressure range. Also, as wind speeds increase, the vertical spread in pressure becomes a bit wider, suggesting more variability in storm behavior at higher intensities. There aren't any obvious outliers, and the overall pattern is smooth and consistent.

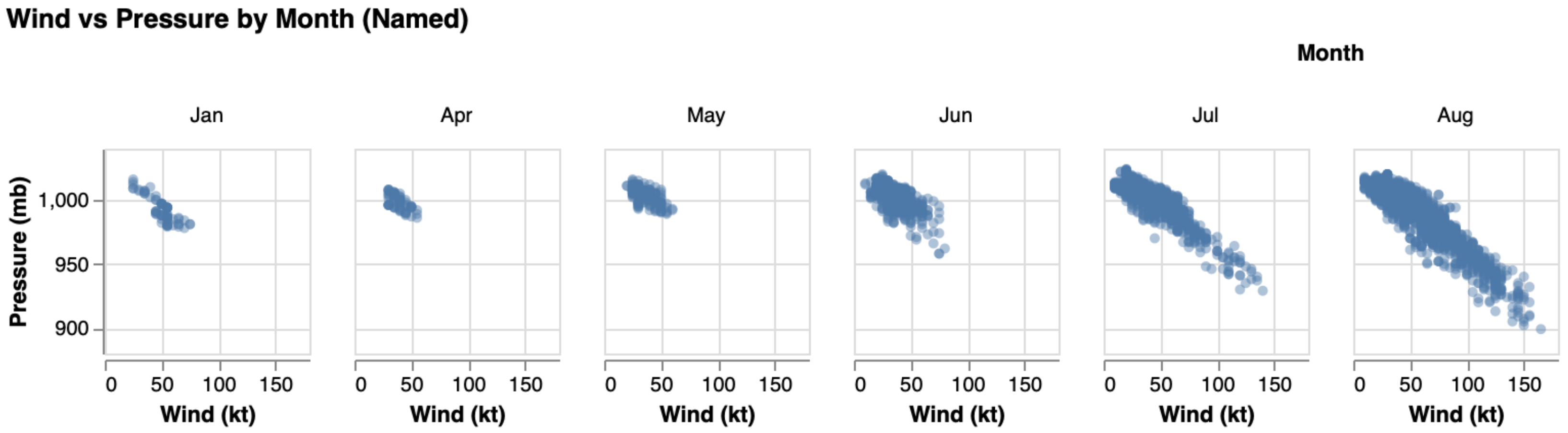
Now, let's see how wind and pressure are related to other variables.

PROMPT: create three charts below showing the relationship between wind and pressure and other variables in the dataset.
For each chart you create, add a cell below interpreting it.

In [6]:
```
# What would wind/pressure look like for each month separately?
month_names = {
    1: "Jan", 2: "Feb", 3: "Mar", 4: "Apr",
    5: "May", 6: "Jun", 7: "Jul", 8: "Aug",
    9: "Sep", 10: "Oct", 11: "Nov", 12: "Dec"
}

df2 = df.copy()
df2['month_name'] = df2['month'].map(month_names)
```

```
alt.Chart(df2).mark_circle(size=25, opacity=0.45).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    column=alt.Column('month_name:N', title='Month', sort=list(month_names.values())),
    tooltip=['name','year','month_name','status','wind','pressure']
).properties(
    width=100,
    height=100,
    title='Wind vs Pressure by Month (Named)'
).resolve_scale(
    x='shared',
    y='shared'
)
```

Out[6]:



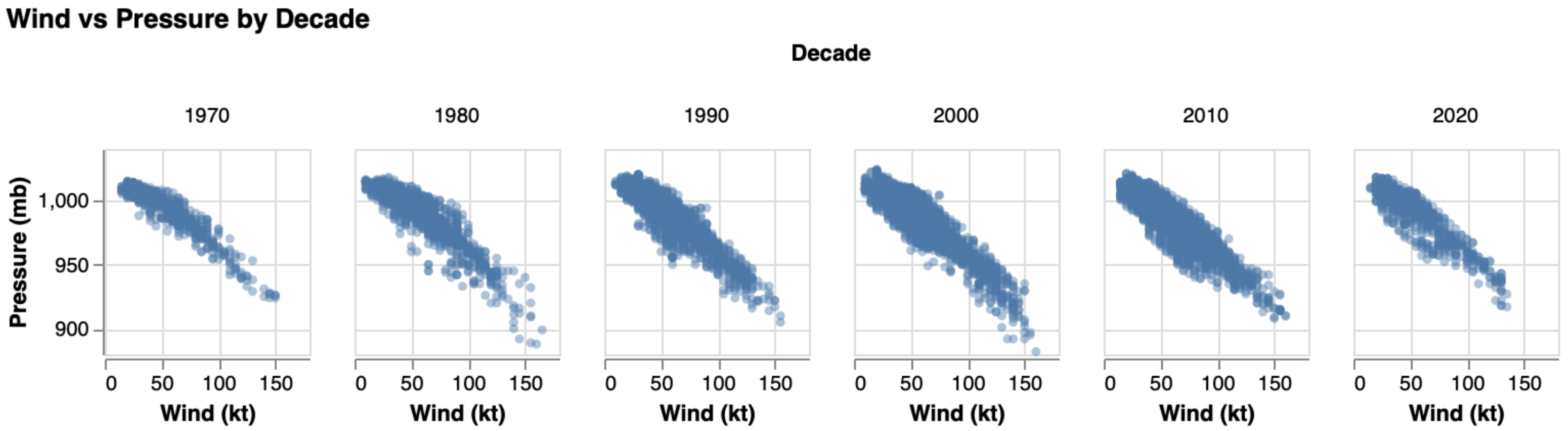
Interpretation 1: The wind–pressure relationship looks similar across months (inverse wind-pressure), but the number of storms changes a lot. Activity is low in the early months, with mostly weaker systems, and picks up sharply from June through October. We also see storms with larger wind speeds (> 100 kt) appear from around July to November. During peak months like August and September, there's a wider range of wind speeds and lower pressures, reflecting stronger storms. This matches the typical Atlantic hurricane season pattern.

In [7]:

```
# Try another vaiable here: By Decade

# # I plotted this by decades, `floor(datum.year/10)*10` creates a new field 'decade' by flooring t
```

```
by_decade = alt.Chart(df).transform_calculate(
    decade="floor(datum.year/10)*10"
).mark_circle(size=20, opacity=0.45).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    column=alt.Column('decade:O', title='Decade'),
    tooltip=['name','year','status','wind','pressure']
).properties(
    width=100,
    height=100,
    title='Wind vs Pressure by Decade'
).resolve_scale(x='shared', y='shared')

by_decade
```

Out[7]:

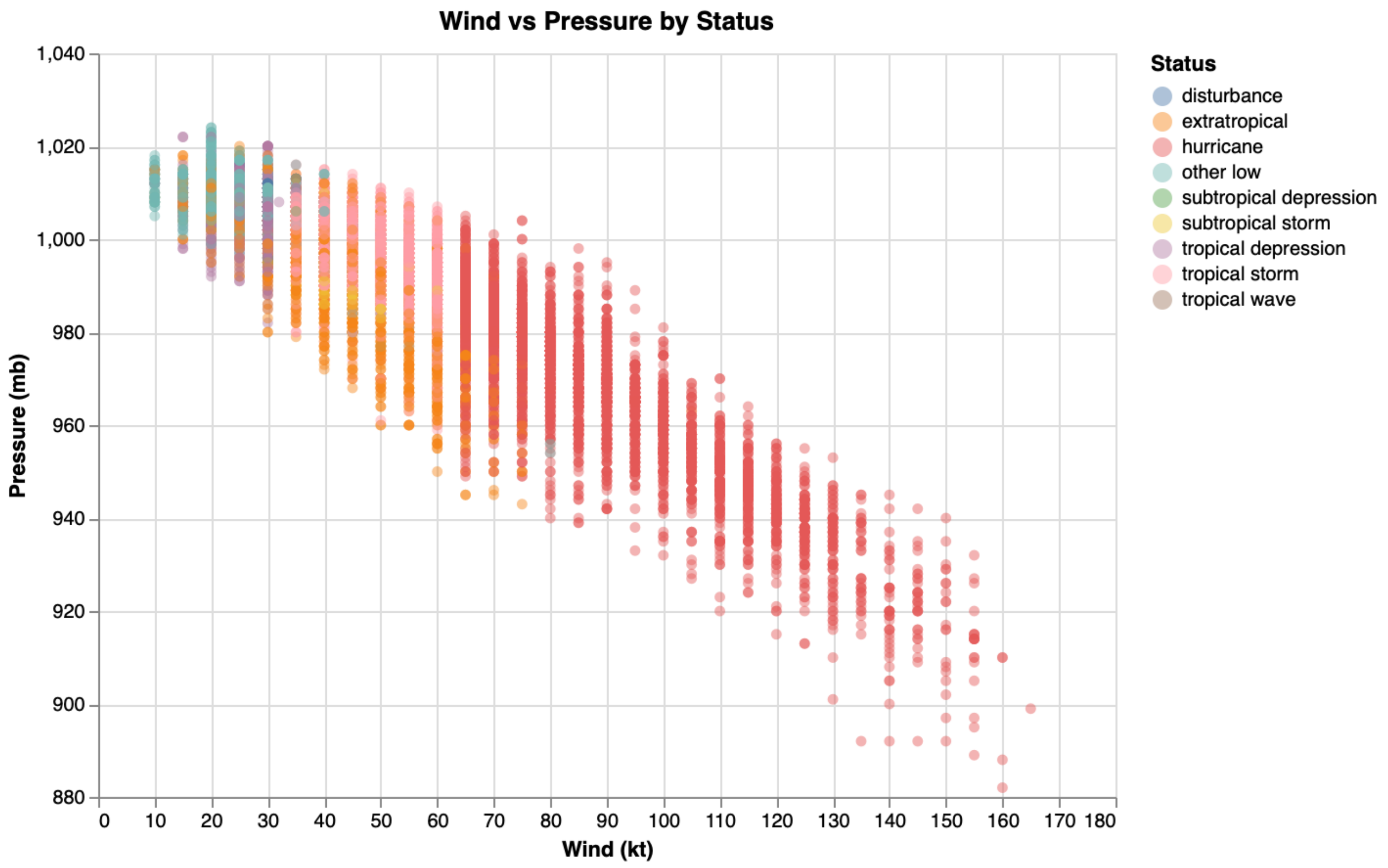


Interpreation 2: The negative relationship between wind and pressure remains strong and consistent across all decades, from 1970s to 2020s.

What changes over time is the density of points — the 2000s show many more observations than the 1970s–1990s, probably reflecting better coverage of monitoring and reporting, or simply that there are more storms in that period. There's also slightly more spread in wind speeds and pressure in later years (i.e. a wider range of both wind and pressure values), reflecting more recorded intense storms. Overall, the pattern itself hasn't changed, but data coverage has improved over time.

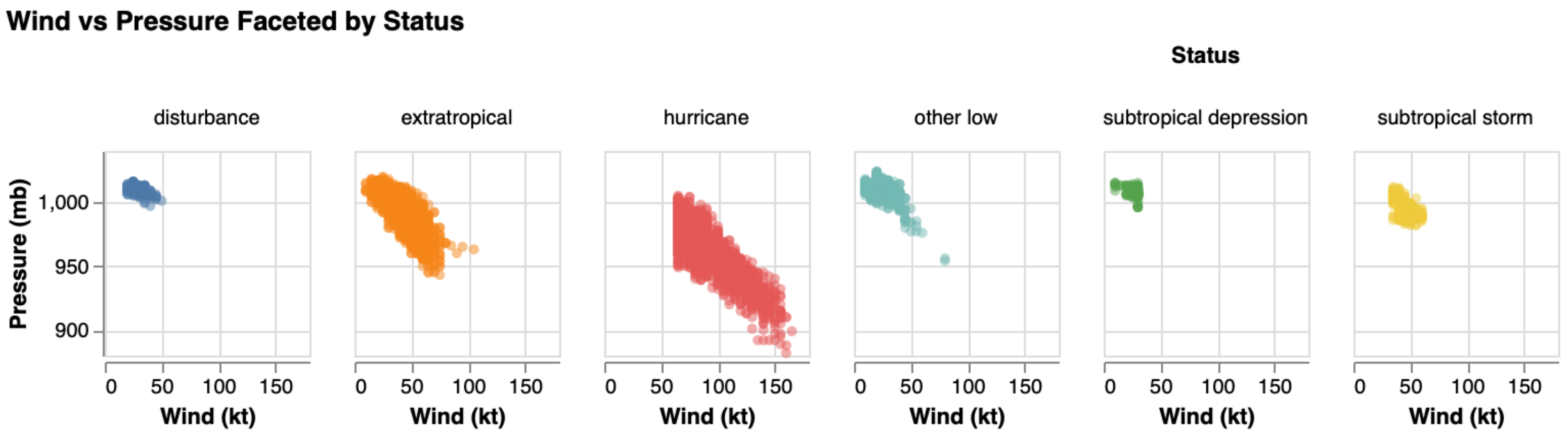
In [8]:
```
# # PROMPT: Color by status to show category separation
alt.Chart(df).mark_circle(size=30, opacity=0.45).encode(
    x=alt.X('wind:Q', scale=alt.Scale(zero=False), title='Wind (kt)'),
    y=alt.Y('pressure:Q', scale=alt.Scale(zero=False), title='Pressure (mb)'),
    color=alt.Color('status:N', title='Status'),
    tooltip=['name','year','status','wind','pressure']
).properties(width=520, height=380, title='Wind vs Pressure by Status')
```

Out[8]:



In [9]:
```
# This just shows it more clearly by faceting it
alt.Chart(df).mark_circle(size=25, opacity=0.5).encode(
    x=alt.X('wind:Q', scale=alt.Scale(zero=False), title='Wind (kt)'),
    y=alt.Y('pressure:Q', scale=alt.Scale(zero=False), title='Pressure (mb)'),
    column=alt.Column('status:N', title='Status'),
    color=alt.Color('status:N', legend=None),
    tooltip=['name','year','status','wind','pressure']
).properties(
    width=100,
    height=100,
    title='Wind vs Pressure Faceted by Status'
).resolve_scale(
    x='shared',
    y='shared'
)
```

Out[9]:



Interpretation 3: These charts show how the wind–pressure relationship differs by storm status.

- Tropical and subtropical depressions, storms, and waves cluster at lower wind speeds (below ~60 kt) and higher pressures (around 1000 mb). Hurricanes sit toward the bottom right — higher winds and lower pressures.
- Other categories like disturbances and extratropical systems stay mostly at the weaker end.

You can clearly see the progression in intensity from left to right: as storms strengthen, wind speeds rise and pressure drops. The colors just help distinguish those categories visually.

One thing we notice when exploring the data is the status column.
Let's see unique values of the `status` variable.

In [10]:
```
df[["status"]].drop_duplicates()
```

Out[10]:

	status
0	tropical depression
8	tropical storm
30	extratropical
44	hurricane
84	subtropical storm
224	subtropical depression
1035	disturbance
2653	other low
7737	tropical wave

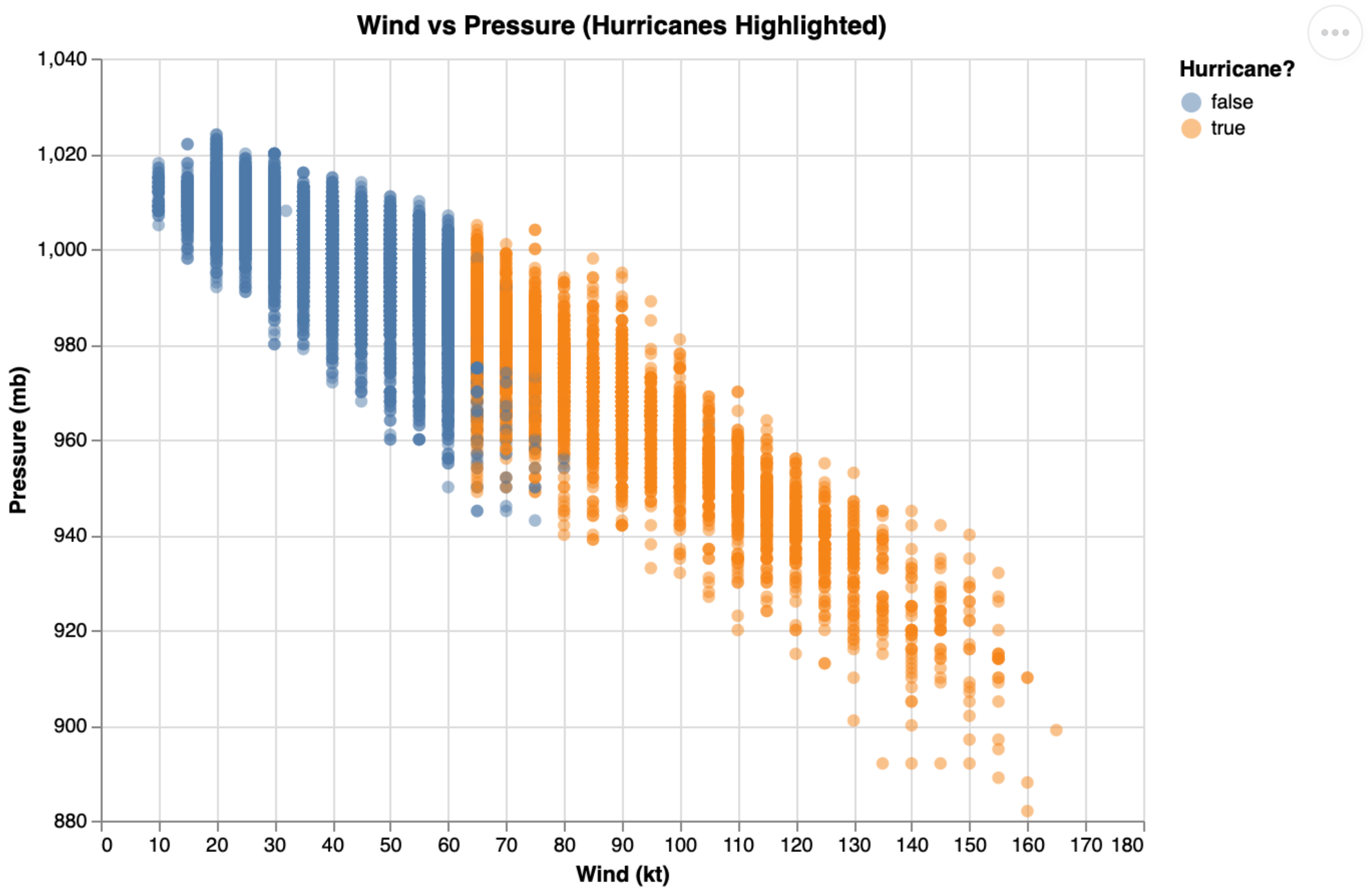
Let's create an indicator variable `is_hurricane` to differentiate hurricanes from other storms in the dataset.

In [11]:
```
# PROMPT: add a hurricane indicator varible to the dataframe
df2["is_hurricane"] = (df2["status"] == "hurricane")
```

Now let's see where the hurricanes end up in our distribution.

In [12]:
```
# PROMPT: plot the distribution of wind and pressure highlighting hurricanes in color
alt.Chart(df2).mark_circle(size=40, opacity=0.5).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    color=alt.Color('is_hurricane:N', title='Hurricane?'),
    tooltip=['name', 'year', 'status', 'wind', 'pressure']
).properties(
    width=520,
    height=380,
    title='Wind vs Pressure (Hurricanes Highlighted)'
)
```

Out[12]:



Answer: We see how hurricanes end up at the bottom half of the distribution (i.e. higher wind speeds, lower pressrues).

Let's take a look at where hurricanes end up in different geographic regions.

We do this by **binning** `lat` and `long` coordinates and using these bins to facet our preferred visualization of `wind`, `pressure`, and `is_hurricane`.

PROMPT: Bin the lat and long variables to create a 4-by-4 grid of plots showing the relationship between wind, pressure, and is_hurricane across different geographic zones measured in the data

In [ ]:
```
# import numpy as np

# q_splits = [0, 0.25, 0.5, 0.75, 1]
# lat_q = [np.quantile(df2["lat"], q) for q in q_splits]
# print('Latitude quantiles:', lat_q)

# df2["lat_zone"] = pd.cut(df2["lat"], lat_q)
# df2["lat_zone"] = df2["lat_zone"].apply(str) # we want the zone names as strings

# long_q = [np.quantile(df["long"], q) for q in q_splits]
# print('Longitude quantiles:', long_q)

# df2["long_zone"] = pd.cut(df2["long"], long_q)
# df2["long_zone"] = df2["long_zone"].apply(str)


import numpy as np
df2 = df2.dropna(subset=["lat", "long"]).copy()

df2["lat_zone"]  = pd.qcut(df2["lat"],  4, duplicates="drop")
df2["long_zone"] = pd.qcut(df2["long"], 4, duplicates="drop")

df2 = df2.dropna(subset=["lat_zone", "long_zone"])

df2["lat_zone"]  = df2["lat_zone"].astype(str)
df2["long_zone"] = df2["long_zone"].astype(str)
lat_order  = sorted(df2["lat_zone"].unique(),  key=lambda s: float(s.split(',')[0][1:]))
long_order = sorted(df2["long_zone"].unique(), key=lambda s: float(s.split(',')[0][1:]))
```

In [14]:
```
display(df2[["lat_zone"]].drop_duplicates())
display(df2[["long_zone"]].drop_duplicates())

df2 = df2[df2["lat_zone"].notna()]

display(df2[["lat_zone"]].drop_duplicates())
```

	lat_zone
0	(26.6, 33.7]
7	(33.7, 70.7]
31	(18.4, 26.6]
113	(6.999, 18.4]

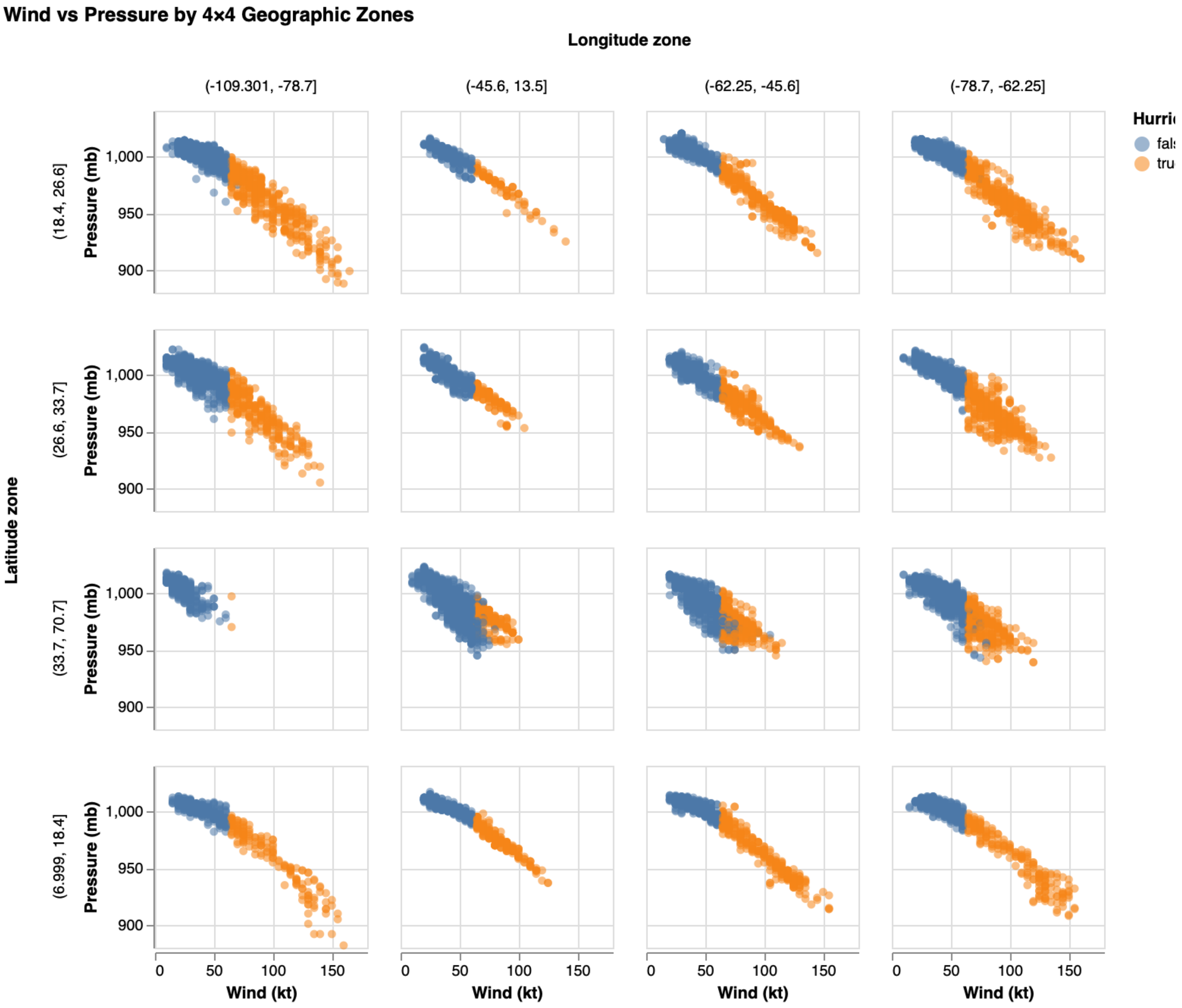
	long_zone
0	(-109.301, -78.7]
6	(-78.7, -62.25]
25	(-62.25, -45.6]
96	(-45.6, 13.5]

	lat_zone
0	(26.6, 33.7]
7	(33.7, 70.7]
31	(18.4, 26.6]
113	(6.999, 18.4]

Now make the required 4x4 grid of plots (and remember to color the hurricanes):

In [15]:
```python
# keep only 4x4
alt.Chart(df2).mark_circle(size=30, opacity=0.55).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    color=alt.Color('is_hurricane:N', title='Hurricane?'),
    row=alt.Row('lat_zone:N',
                title='Latitude zone',
                sort='ascending'),
    column=alt.Column('long_zone:N',
                      title='Longitude zone',
                      sort='ascending'),
    tooltip=['name','year','lat','long','wind','pressure','is_hurricane']
).properties(
    width=140,
    height=120,
    title='Wind vs Pressure by 4×4 Geographic Zones'
).resolve_scale(
    x='shared',
    y='shared'
)
```

Out[15]:


Wind vs Pressure by 4×4 Geographic Zones
Longitude zone
(-109.301, -78.7]
(-45.6, 13.5]
(-62.25, -45.6]
(-78.7, -62.25]
Latitude zone
(18.4, 26.6]
(26.6, 33.7]
(33.7, 70.7]
(6.999, 18.4]
Pressure (mb)
1,000
950
900
Wind (kt)
0
50
100
150
Hurri
fal
tru

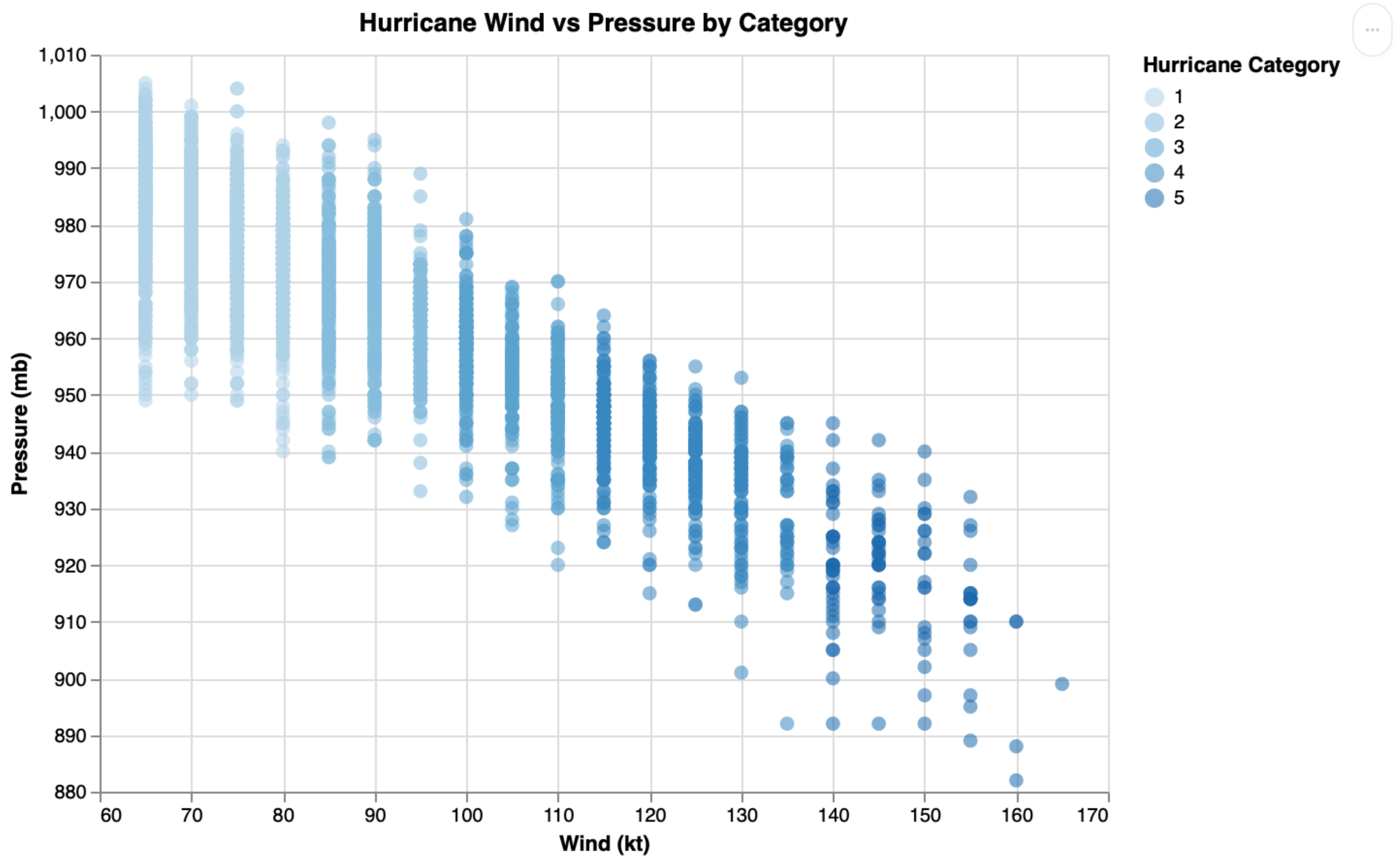
Now let's look only at the storms that are hurricanes.

In [16]:
```
# PROMPT: create a filtered version of the dataframe containing only the storms that were designate
h_df = df2[df2["status"] == "hurricane"]
```

Let's take a look at storm wind and pressure grouped by hurricane category.

In [17]:
```
# PROMPT: using the filtered data, plot wind and pressure and color code by hurricane category

alt.Chart(h_df).mark_circle(size=55, opacity=0.55).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    color=alt.Color('category:O', title='Hurricane Category'),
    tooltip=['name', 'year', 'category', 'wind', 'pressure']
).properties(
    width=520,
    height=380,
    title='Hurricane Wind vs Pressure by Category'
)
```

Out[17]:



PROMPT: what do you notice about the chart you just made?

Hurricanes begin around 60 knots, and as wind speeds increase, pressure consistently drops. Weaker hurricanes (Category 1–2) cluster at lower wind speeds and higher pressures, while stronger ones (Category 4–5) appear in the high wind, low pressure range. This shows the physical relationship between storm intensity, wind speed, and central pressure.

Now, let's create a version of this chart where we superimpose lines representing the trend between wind and pressure within each category.

We'll start by creating the line chart showing mean pressure by wind and category.

In [18]:
```python
# The trend of pressures as a function of winds (for each category) can be calculated by
# finding the mean pressure for each [wind x category] combination:
h_df.groupby(['wind', 'category'])[['wind', 'category', 'pressure']].mean()
```

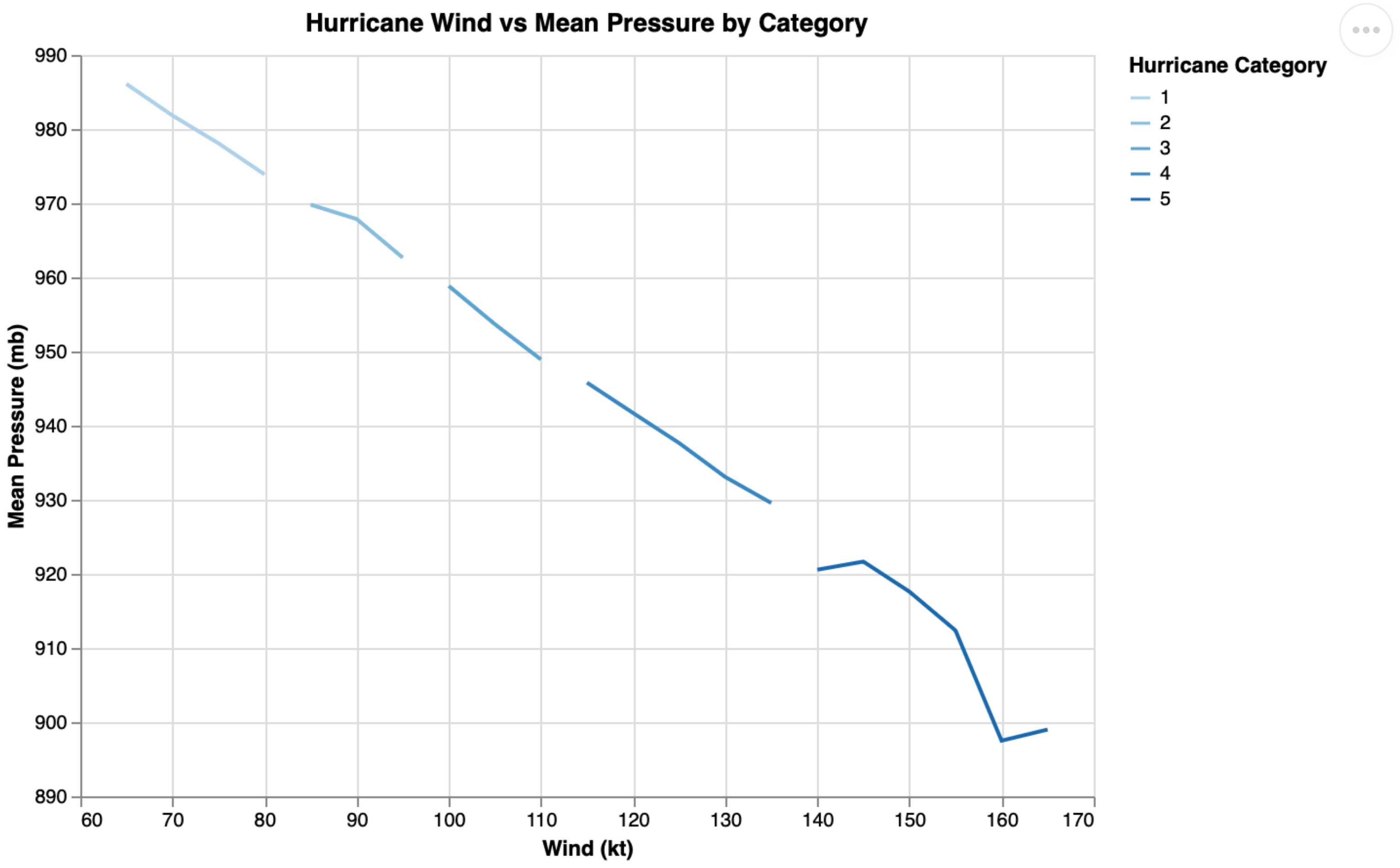
Out[18]:

		wind	category	pressure
wind	**category**			
65	**1.0**	65.0	1.0	986.094092
70	**1.0**	70.0	1.0	981.835570
75	**1.0**	75.0	1.0	978.084095
80	**1.0**	80.0	1.0	973.888361
85	**2.0**	85.0	2.0	969.801802
90	**2.0**	90.0	2.0	967.865526
95	**2.0**	95.0	2.0	962.658009
100	**3.0**	100.0	3.0	958.838057
105	**3.0**	105.0	3.0	953.682927
110	**3.0**	110.0	3.0	948.928571
115	**4.0**	115.0	4.0	945.800000
120	**4.0**	120.0	4.0	941.705426
125	**4.0**	125.0	4.0	937.651786
130	**4.0**	130.0	4.0	933.075949
135	**4.0**	135.0	4.0	929.564103
140	**5.0**	140.0	5.0	920.564103
145	**5.0**	145.0	5.0	921.656250

150	**5.0**	150.0	5.0	917.600000
155	**5.0**	155.0	5.0	912.368421
160	**5.0**	160.0	5.0	897.500000
165	**5.0**	165.0	5.0	899.000000

In [19]:
```
# PROMPT: using the filtered data, plot the trend of wind x pressure within categories.
#          Trends
h_summary = h_df.groupby(['wind', 'category'])[['wind', 'category', 'pressure']].mean()

# Altair frowns upon hierarchical indices
h_summary.index = h_summary.index.to_flat_index()

lines = alt.Chart(h_summary).mark_line(size=2).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Mean Pressure (mb)', scale=alt.Scale(zero=False)),
    color=alt.Color('category:O', title='Hurricane Category')
).properties(width=520, height=380, title='Hurricane Wind vs Mean Pressure by Category')
lines
```

Out[19]:



PROMPT: what do you notice about the chart you just made?

As wind speed goes up, the average pressure drops pretty quickly. Stronger hurricanes show up on the right side of the chart and have much lower pressures. This makes sense as more intense storms tend to have faster winds and lower central pressure.

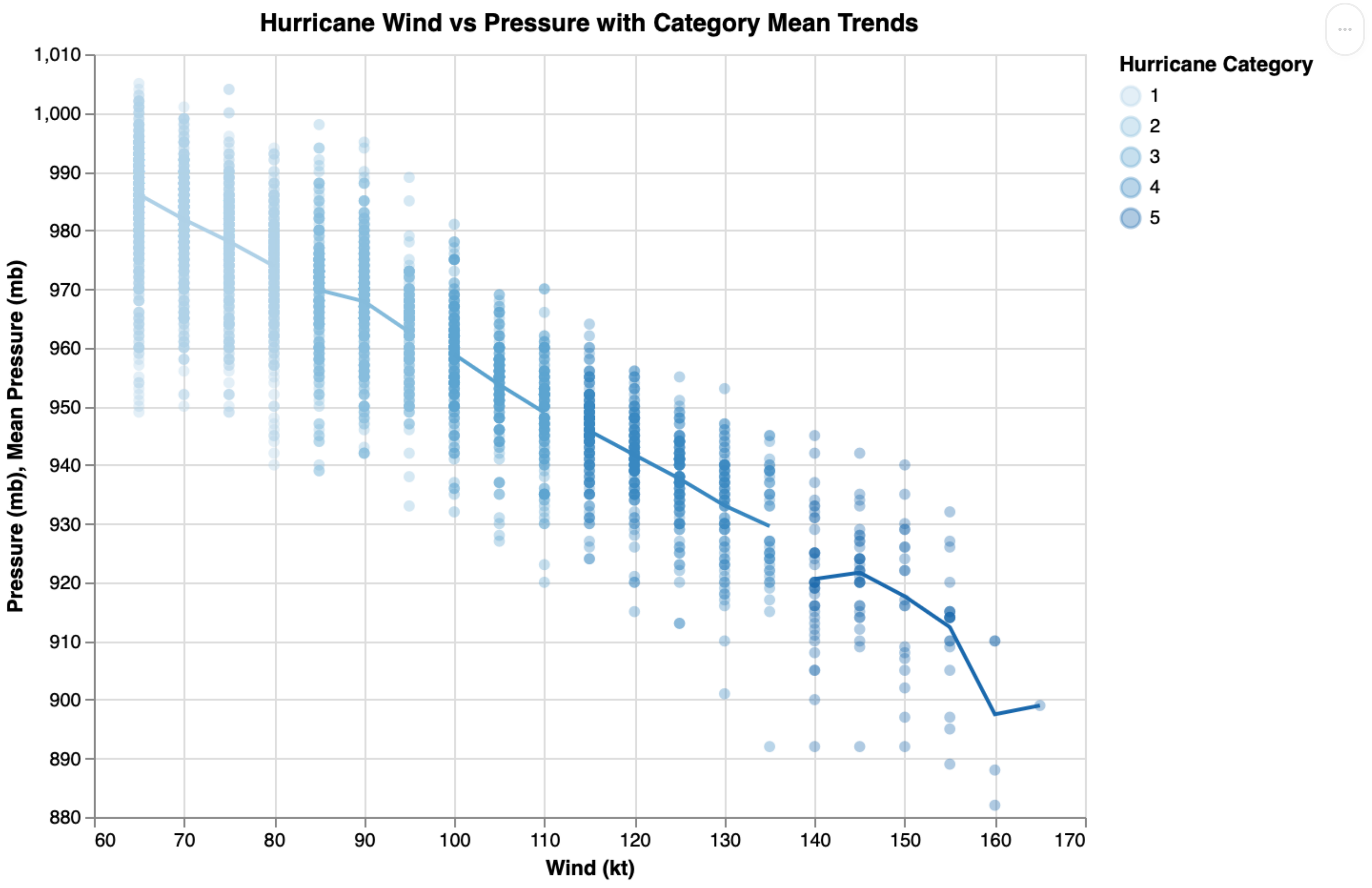
Now, let's layer our line chart on top of the chart we created earlier comparing wind, pressure, and hurricane category.

In [20]:

```
# PROMPT: using the filtered data, plot the trend of wind x pressure within categories,
```

```python
# superimposing mean lines by layering charts with the `+` operator

points = alt.Chart(h_df).mark_circle(size=35, opacity=0.35).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    color=alt.Color('category:O', title='Hurricane Category'),
    tooltip=['name','year','category','wind','pressure']
)

lines = alt.Chart(h_summary).mark_line(size=2).encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Mean Pressure (mb)', scale=alt.Scale(zero=False)),
    color=alt.Color('category:O', title='Hurricane Category')
)

(points + lines).properties(
    width=520, height=400, title='Hurricane Wind vs Pressure with Category Mean Trends'
)
```

Out[20]:



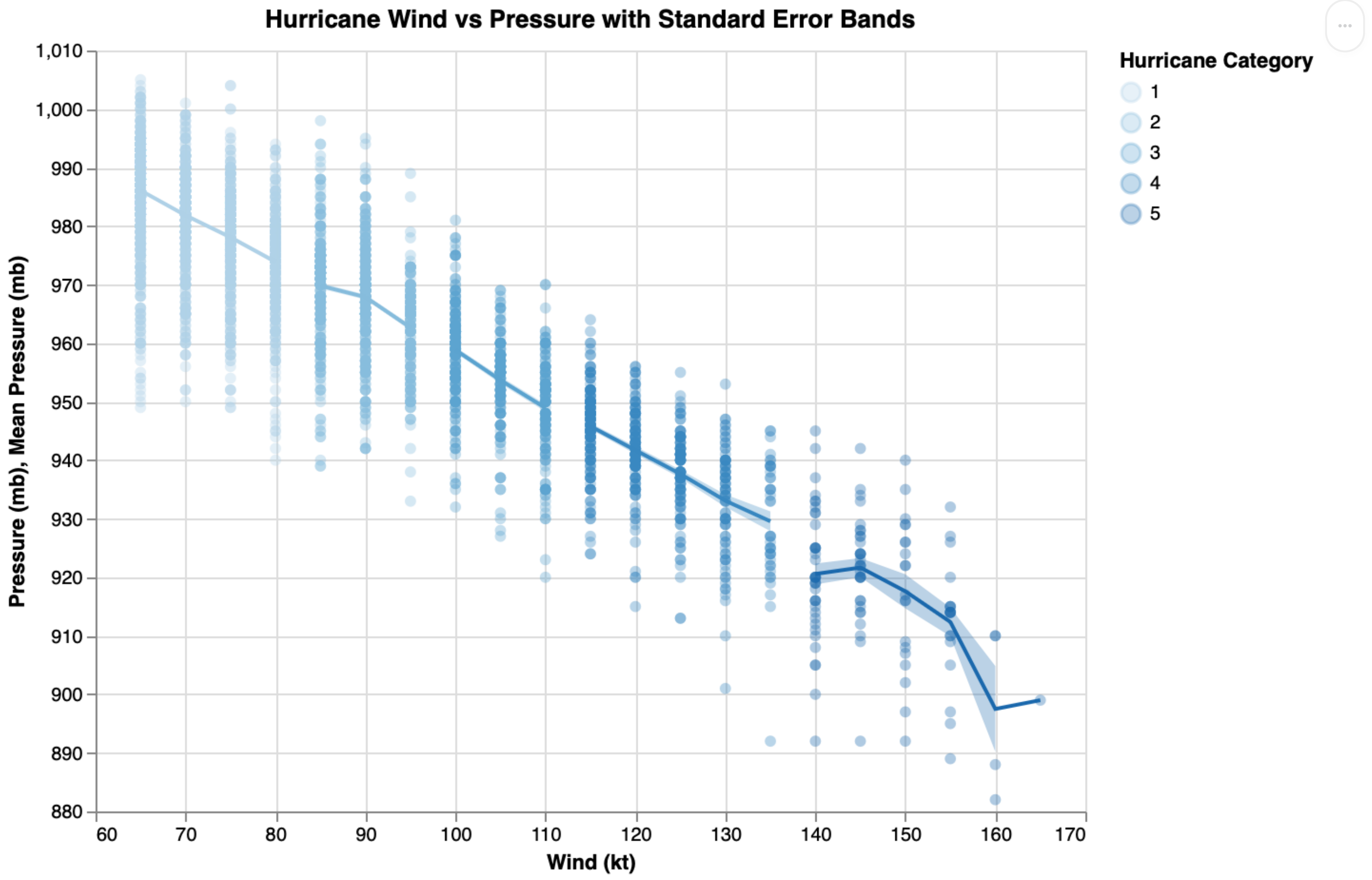
As a final challenge, let's add an additional layer showing one standard error around our lines as an interval.

In [21]:

```
# PROMPT: add an interval representing the standard error of the mean to the scatterplot with lines

interval = alt.Chart(h_df).mark_errorband(extent='stderr').encode(
    x=alt.X('wind:Q', title='Wind (kt)', scale=alt.Scale(zero=False)),
    y=alt.Y('pressure:Q', title='Pressure (mb)', scale=alt.Scale(zero=False)),
    color=alt.Color('category:O', title='Hurricane Category')
)
```

```
(interval + points + lines).properties(
    width=520, height=400,
    title='Hurricane Wind vs Pressure with Standard Error Bands'
)
```

Out[21]:



PROMPT: what do you notice about the chart you just made?

The chart shows a **clear downward trend**—as wind speed increases, the mean pressure drops steadily. The shaded error bands are fairly narrow at lower wind speeds but widen at higher speeds, meaning there's more **variability** in pressure among the strongest hurricanes. This makes sense because intense storms can have different structures even at similar wind speeds.

In [22]:
```
import os
os.chdir("/Users/chrislowzx/data227/data-viz/exercise/exercise_folder")
print(os.getcwd())
```

```
/Users/chrislowzx/data227/data-viz/exercise/exercise_folder
```

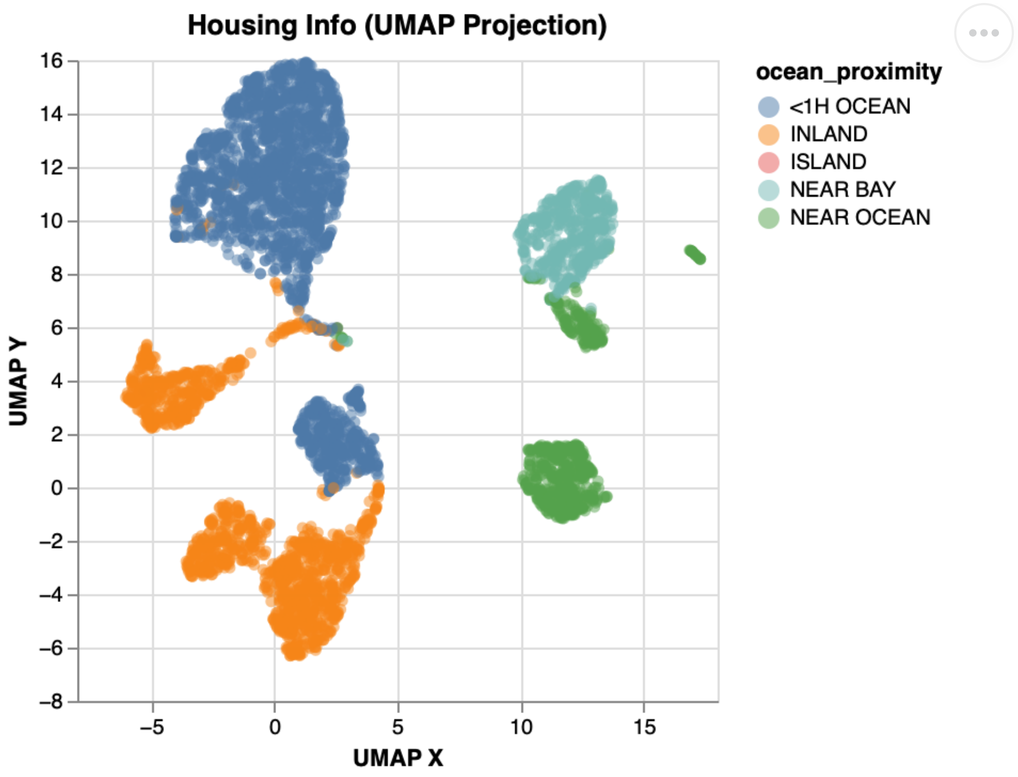
In [23]:
```
import pandas as pd

df = pd.read_csv("../exercise_data/ca-housing-umap.csv")
df.head()
```

Out[23]:

	Unnamed: 0	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	r
0	0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	
1	1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	
2	2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	
3	3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	
4	4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	

In [24]:
```
# Check out the tooltip!

# FYI, UMAP (Uniform Manifold Approximation and Projection) for Dimension Reduction
# is similar to t-SNE. It arranges high-dimensional data in low-dimensional space.
# UMAP constructs a high dimensional graph representation of the data then optimizes
# a low-dimensional graph to be as structurally similar as possible.
# The goal is to preserve local structures as well as complex (non-linear) relationships.

# import altair as alt
# import vegafusion as vf
# vf.enable(row_limit=50000)

# ### Same issue as above...
# df2 = df.sample(5000, axis=0)

# alt.Chart(df2).mark_circle().encode(
#     x=alt.X("x:Q", title="UMAP X"),
#     y=alt.Y("y:Q", title="UMAP Y"),
#     opacity=alt.value(0.5),
#     tooltip=['longitude', 'latitude', 'housing_median_age', 'total_rooms',
#              'total_bedrooms', 'population', 'households', 'median_income',
#              'median_house_value', 'ocean_proximity'],
#     color="ocean_proximity"
# ).properties(
#     title="Housing Info"
# )


import altair as alt
import vegafusion as vf

# Correct way to set row limit
alt.data_transformers.enable("vegafusion", max_rows=50000)

# Sample to avoid overload (optional)
df2 = df.sample(5000, axis=0)

alt.Chart(df2).mark_circle().encode(
    x=alt.X("x:Q", title="UMAP X"),
    y=alt.Y("y:Q", title="UMAP Y"),
    opacity=alt.value(0.5),
    tooltip=['longitude', 'latitude', 'housing_median_age', 'total_rooms',
             'total_bedrooms', 'population', 'households', 'median_income',
             'median_house_value', 'ocean_proximity'],
    color="ocean_proximity"
).properties(
    title="Housing Info (UMAP Projection)"
)
```

Out[24]:


Housing Info (UMAP Projection)
UMAP Y
UMAP X
16
14
12
10
8
6
4
2
0
−2
−4
−6
−8
−5
0
5
10
15
ocean_proximity
<1H OCEAN
INLAND
ISLAND
NEAR BAY
NEAR OCEAN

In [ ]: