

CH06_College_Scorecard_Python_Supplement

September 3, 2026

1 Chapter 6 — College Scorecard (Python supplement)

Condensed notebook for the **Python** portion of *Chapter 6: College Data*.

Target graphics:

- SAT math vs. verbal scatter with trendline and fit summary (**Figures 6.37–6.41**)
- Cost vs. acceptance rate: scatter, Indiana highlight, labels, and graph-objects version (**Figures 6.43–6.46**)

Data: Download institution-level data from the [College Scorecard](#) and save `Most-Recent-Cohorts-Institution_2022_23.csv` in `Data For Condensed Notebooks` (or update the path in the first `read_csv` cell). Figures **6.1–6.3** in the book show the download site and dictionary.

Dependencies:

- `pandas`
- `plotly`
- `numpy`
- `statsmodels` (for `trendline='ols'` in Plotly Express)

1.1 Imports and Plotly output

The chapter uses a PDF-capable default renderer (requires **Kaleido** if your environment uses that renderer).

```
[1]: import pandas as pd
import plotly.express as px
import numpy as np
```

```
[2]: # Set output options.
import plotly.io as pio
pio.renderers.default = "pdf+jupyterlab+notebook"
```

1.2 Reading and cleaning the data

Load the full Scorecard CSV, then restrict to the columns used in the chapter.

```
[3]: csc_all = pd.read_csv('../Data For Condensed Notebooks/
↳Most-Recent-Cohorts-Institution_2022_23.csv')
```

```

/var/folders/_x/nbw2t83x75d0lr3jn4pg_cx00000gn/T/ipykernel_26711/1372049275.py:1
: DtypeWarning: Columns (0: NPCURL, 1: INC_N, 2: DEP_STAT_N, 3: APPL_SCH_N, 4:
AGE_ENTRY, 5: AGEGE24, 6: FAMINC, 7: MD_FAMINC, 8: PCT_WHITE, 9: PCT_BLACK, 10:
PCT_ASIAN, 11: PCT_HISPANIC, 12: PCT_BA, 13: PCT_GRAD_PROF, 14: PCT_BORN_US, 15:
MEDIAN_HH_INC, 16: POVERTY_RATE, 17: UNEMP_RATE, 18: LN_MEDIAN_HH_INC, 19:
MN_EARN_WNE_P7, 20: GT_25K_P7, 21: PCT10_EARN_WNE_P8, 22: PCT90_EARN_WNE_P8, 23:
C150_L4_POOLED_SUPP, 24: C150_4_POOLED_SUPP, 25: C200_L4_POOLED_SUPP, 26:
C200_4_POOLED_SUPP, 27: ALIAS, 28: T4APPROVALDATE, 29: RET_FT4_POOLED_SUPP, 30:
RET_FTL4_POOLED_SUPP, 31: RET_PT4_POOLED_SUPP, 32: RET_PTL4_POOLED_SUPP, 33:
TRANS_4_POOLED_SUPP, 34: TRANS_L4_POOLED_SUPP, 35: C100_4_POOLED_SUPP, 36:
C100_L4_POOLED_SUPP, 37: OMAWDP6_FTFT_POOLED_SUPP, 38: OMAWDP8_FTFT_POOLED_SUPP,
39: OMENRYP8_FTFT_POOLED_SUPP, 40: OMENRAP8_FTFT_POOLED_SUPP, 41:
OMENRUP8_FTFT_POOLED_SUPP, 42: OMAWDP6_PTFT_POOLED_SUPP, 43:
OMAWDP8_PTFT_POOLED_SUPP, 44: OMENRYP8_PTFT_POOLED_SUPP, 45:
OMENRAP8_PTFT_POOLED_SUPP, 46: OMENRUP8_PTFT_POOLED_SUPP, 47:
OMAWDP6_FTNFT_POOLED_SUPP, 48: OMAWDP8_FTNFT_POOLED_SUPP, 49:
OMENRYP8_FTNFT_POOLED_SUPP, 50: OMENRAP8_FTNFT_POOLED_SUPP, 51:
OMENRUP8_FTNFT_POOLED_SUPP, 52: OMAWDP6_PTNFT_POOLED_SUPP, 53:
OMAWDP8_PTNFT_POOLED_SUPP, 54: OMENRYP8_PTNFT_POOLED_SUPP, 55:
OMENRAP8_PTNFT_POOLED_SUPP, 56: OMENRUP8_PTNFT_POOLED_SUPP, 57: CIPTITLE2, 58:
CIPTITLE3, 59: CIPTITLE4, 60: CIPTITLE5, 61: CIPTITLE6, 62:
OMENRYP_ALL_POOLED_SUPP, 63: OMENRAP_ALL_POOLED_SUPP, 64:
OMAWDP8_ALL_POOLED_SUPP, 65: OMENRUP_ALL_POOLED_SUPP, 66:
OMENRYP_FIRSTTIME_POOLED_SUPP, 67: OMENRAP_FIRSTTIME_POOLED_SUPP, 68:
OMAWDP8_FIRSTTIME_POOLED_SUPP, 69: OMENRUP_FIRSTTIME_POOLED_SUPP, 70:
OMENRYP_NOTFIRSTTIME_POOLED_SUPP, 71: OMENRAP_NOTFIRSTTIME_POOLED_SUPP, 72:
OMAWDP8_NOTFIRSTTIME_POOLED_SUPP, 73: OMENRUP_NOTFIRSTTIME_POOLED_SUPP, 74:
OMENRYP_FULLTIME_POOLED_SUPP, 75: OMENRAP_FULLTIME_POOLED_SUPP, 76:
OMAWDP8_FULLTIME_POOLED_SUPP, 77: OMENRUP_FULLTIME_POOLED_SUPP, 78:
OMENRYP_PARTTIME_POOLED_SUPP, 79: OMENRAP_PARTTIME_POOLED_SUPP, 80:
OMAWDP8_PARTTIME_POOLED_SUPP, 81: OMENRUP_PARTTIME_POOLED_SUPP, 82:
FTFTPCTPELL_POOLED_SUPP, 83: FTFTPCTFLOAN_POOLED_SUPP, 84: LPSTAFFORD_CNT, 85:
LPSTAFFORD_AMT, 86: C150_L4_PELL_POOLED_SUPP, 87: C150_4_PELL_POOLED_SUPP, 88:
OMAWDP8_PELL_FTFT_POOLED_SUPP, 89: OMENRYP8_PELL_FTFT_POOLED_SUPP, 90:
OMENRAP8_PELL_FTFT_POOLED_SUPP, 91: OMENRUP8_PELL_FTFT_POOLED_SUPP, 92:
OMAWDP8_PELL_PTFT_POOLED_SUPP, 93: OMENRYP8_PELL_PTFT_POOLED_SUPP, 94:
OMENRAP8_PELL_PTFT_POOLED_SUPP, 95: OMENRUP8_PELL_PTFT_POOLED_SUPP, 96:
OMAWDP8_PELL_FTNFT_POOLED_SUPP, 97: OMENRYP8_PELL_FTNFT_POOLED_SUPP, 98:
OMENRAP8_PELL_FTNFT_POOLED_SUPP, 99: OMENRUP8_PELL_FTNFT_POOLED_SUPP, 100:
OMAWDP8_PELL_PTNFT_POOLED_SUPP, 101: OMENRYP8_PELL_PTNFT_POOLED_SUPP, 102:
OMENRAP8_PELL_PTNFT_POOLED_SUPP, 103: OMENRUP8_PELL_PTNFT_POOLED_SUPP, 104:
OMENRYP_PELL_ALL_POOLED_SUPP, 105: OMENRAP_PELL_ALL_POOLED_SUPP, 106:
OMAWDP8_PELL_ALL_POOLED_SUPP, 107: OMENRUP_PELL_ALL_POOLED_SUPP, 108:
OMENRYP_PELL_FTT_POOLED_SUPP, 109: OMENRAP_PELL_FTT_POOLED_SUPP, 110:
OMAWDP8_PELL_FTT_POOLED_SUPP, 111: OMENRUP_PELL_FTT_POOLED_SUPP, 112:
OMENRYP_PELL_NFT_POOLED_SUPP, 113: OMENRAP_PELL_NFT_POOLED_SUPP, 114:
OMAWDP8_PELL_NFT_POOLED_SUPP, 115: OMENRUP_PELL_NFT_POOLED_SUPP, 116:
OMENRYP_PELL_FT_POOLED_SUPP, 117: OMENRAP_PELL_FT_POOLED_SUPP, 118:

```

OMAWDP8_PELL_FT_POOLED_SUPP, 119: OMENRUP_PELL_FT_POOLED_SUPP, 120:
 OMENRYP_PELL_PT_POOLED_SUPP, 121: OMENRAP_PELL_PT_POOLED_SUPP, 122:
 OMAWDP8_PELL_PT_POOLED_SUPP, 123: OMENRUP_PELL_PT_POOLED_SUPP, 124:
 GT_THRESHOLD_P6_SUPP, 125: ADM_RATE_SUPP, 126: ADDR, 127:
 PCTPELL_DCS_POOLED_SUPP, 128: PCTFLOAN_DCS_POOLED_SUPP) have mixed types.
 Specify dtype option on import or set low_memory=False.

```
csc_all = pd.read_csv('../Data For Condensed Notebooks/Most-Recent-Cohorts-
Institution_2022_23.csv')
```

You may see a `DtypeWarning` about mixed types: many columns contain strings such as NA or PS, so pandas cannot infer numeric dtypes for the whole file. The analyses below still work on the columns we keep; narrowing columns first is the usual fix. For Tableau, exporting clean columns (below) helps.

Check memory use of the full table.

```
[4]: csc_all.info()
```

```
<class 'pandas.DataFrame'>
RangeIndex: 6484 entries, 0 to 6483
Columns: 3305 entries, UNITID to MD_EARN_WNE_MALE1_P11
dtypes: float64(919), int64(14), object(79), str(2293)
memory usage: 163.5+ MB
```

Subset to OPEID, institution name, SAT 25th percentiles, admission rate, sticker price, city, and state.

```
[5]: csc = csc_all[['OPEID', 'INSTNM', 'SATMT25', 'SATVR25', 'ADM_RATE_ALL',
↳ 'COSTT4_A', 'CITY', 'STABBR']].copy()
```

Note: You could instead pass `usecols=[...]` to `read_csv` to read only these columns.

Inspect dtypes for the subset.

```
[6]: csc.dtypes
```

```
[6]: OPEID          float64
INSTNM           str
SATMT25          float64
SATVR25          float64
ADM_RATE_ALL     float64
COSTT4_A         float64
CITY             str
STABBR           str
dtype: object
```

Numeric columns are floats despite NA/PS strings elsewhere in the file. Inspect unique values for one SAT column.

```
[7]: csc['SATMT25'].unique()
```

```
[7]: array([400., 590., nan, 613., 399., 560., 500., 610., 480., 460., 450.,
          530., 493., 475., 570., 490., 510., 380., 423., 458., 553., 470.,
          545., 445., 540., 600., 730., 568., 760., 650., 680., 635., 750.,
          550., 740., 580., 640., 620., 440., 660., 410., 630., 382., 710.,
          430., 505., 420., 520., 700., 425., 525., 398., 670., 563., 495.,
          390., 593., 448., 562., 555., 443., 469., 473., 455., 558., 483.,
          438., 478., 585., 485., 690., 720., 488., 533., 780., 477., 528.,
          790., 515., 518., 482., 532., 225., 537., 603., 465., 535., 770.,
          310., 598., 360., 565., 733., 340., 543., 350., 538., 463., 408.,
          435., 618., 523., 583., 503., 393., 715., 648., 513., 370., 428.,
          368., 422.]])
```

`read_csv` maps the literal string "NA" to NaN for these columns. Cast ID and text fields to pandas string dtype.

```
[8]: for c in ['OPEID', 'INSTNM', 'CITY', 'STABBR']:
      csc[c] = csc[c].astype('string')

csc.dtypes
```

```
[8]: OPEID          string
INSTNM          string
SATMT25        float64
SATVR25        float64
ADM_RATE_ALL    float64
COSTT4_A        float64
CITY            string
STABBR          string
dtype: object
```

Write a small CSV for Tableau (**Figure 6.18** / clean data workflow in the chapter).

```
[9]: csc.to_csv('../Data For Condensed Notebooks/csc_small.csv', index=False)
```

1.3 SAT scores

Compare math and verbal SAT (25th percentile) across institutions. **Figure 6.37** — quick pandas scatter; **6.38–6.41** — interactive Plotly, trendline, and regression summary.

Preview the cleaned dataframe.

```
[10]: csc
```

```
[10]:
```

	OPEID	INSTNM	SATMT25	\
0	100200.0	Alabama A & M University	400.0	
1	105200.0	University of Alabama at Birmingham	590.0	
2	2503400.0	Amridge University	NaN	
3	105500.0	University of Alabama in Huntsville	613.0	
4	100500.0	Alabama State University	399.0	

```

...
6479 4270802.0 Wilton Simpson Technical College NaN
6480 2609404.0 Valley College - Fairlawn - School of Nursing NaN
6481 4247201.0 Western Maricopa Education Center - Southwest ... NaN
6482 4247202.0 Western Maricopa Education Center - Northeast ... NaN
6483 4285601.0 Burlington County Institute of Technology - Ad... NaN

      SATVR25  ADM_RATE_ALL  COSTT4_A  CITY STABBR
0      430.0      0.683956  23167.0    Normal    AL
1      610.0      0.866794  26257.0  Birmingham    AL
2       NaN       NaN       NaN  Montgomery    AL
3      613.0      0.781043  25777.0  Huntsville    AL
4      429.0      0.965978  21900.0  Montgomery    AL
...
6479    NaN    NaN    NaN  Brooksville    FL
6480    NaN    NaN    NaN   Fairlawn    OH
6481    NaN    NaN    NaN   Buckeye    AZ
6482    NaN    NaN    NaN   Phoenix    AZ
6483    NaN    NaN    NaN   Medford    NJ

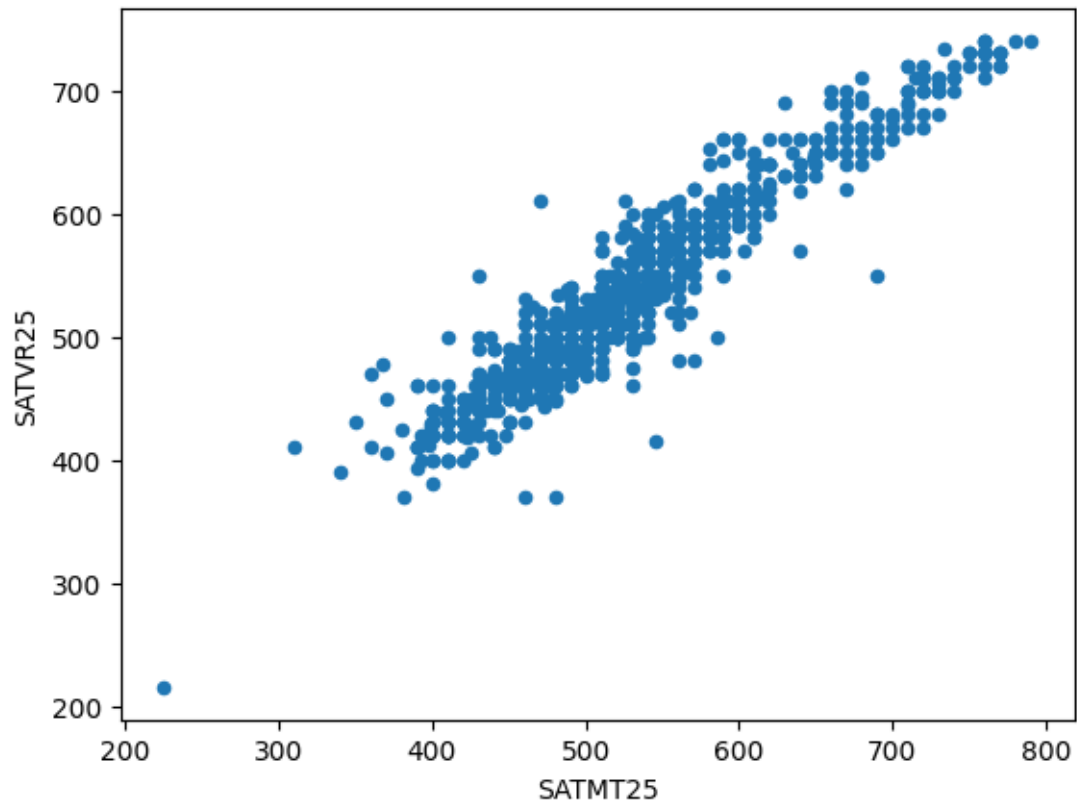
```

[6484 rows x 8 columns]

Figure 6.37 — Built-in pandas scatter (non-interactive).

```
[11]: csc.plot.scatter('SATMT25', 'SATVR25')
```

```
[11]: <Axes: xlabel='SATMT25', ylabel='SATVR25'>
```



Identify the low corner outlier with `idxmin` (cf. **Figure 6.10** in the Excel walkthrough).

```
[12]: min_index = csc['SATMT25'].idxmin()
      csc.loc[min_index]
```

```
[12]: OPEID          243300.0
      INSTNM         Rust College
      SATMT25         225.0
      SATVR25         215.0
      ADM_RATE_ALL    0.787596
      COSTT4_A        16700.0
      CITY           Holly Springs
      STABBR          MS
      Name: 1685, dtype: object
```

Figure 6.38 — Interactive `plotly.express` scatter with institution names on hover.

```
[13]: sat_fig = px.scatter(csc,
                          x='SATMT25',
                          y='SATVR25',
                          hover_data='INSTNM')
      sat_fig
```

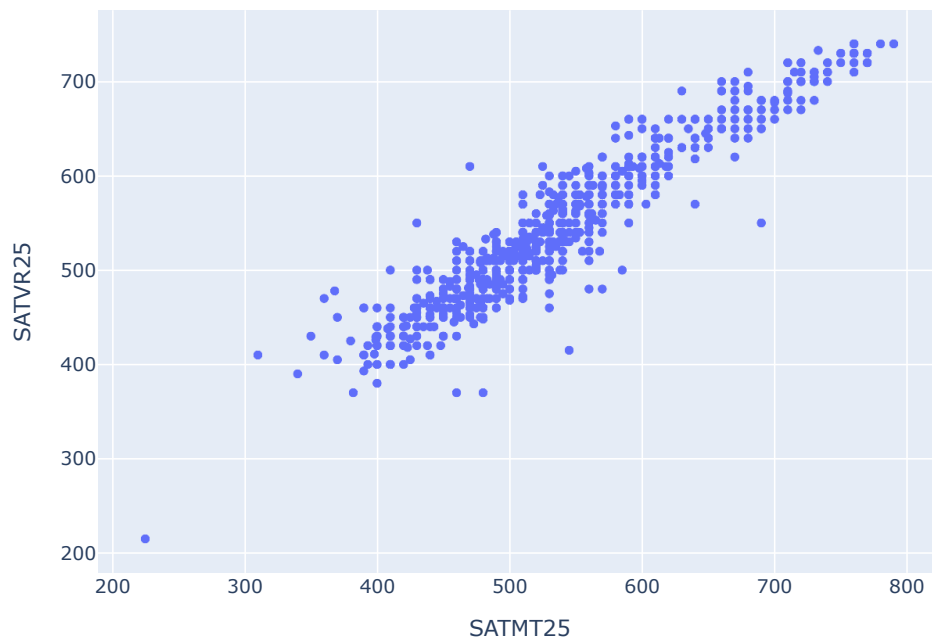


Figure 6.39 — OLS trendline (`trendline='ols'`). Requires `statsmodels` (uv add statsmodels or conda install -c conda-forge statsmodels); restart kernel after install if needed.

```
[14]: sat_fig = px.scatter(csc,  
                        x='SATMT25',  
                        y='SATVR25',  
                        hover_data='INSTNM',  
                        trendline='ols')  
  
sat_fig
```

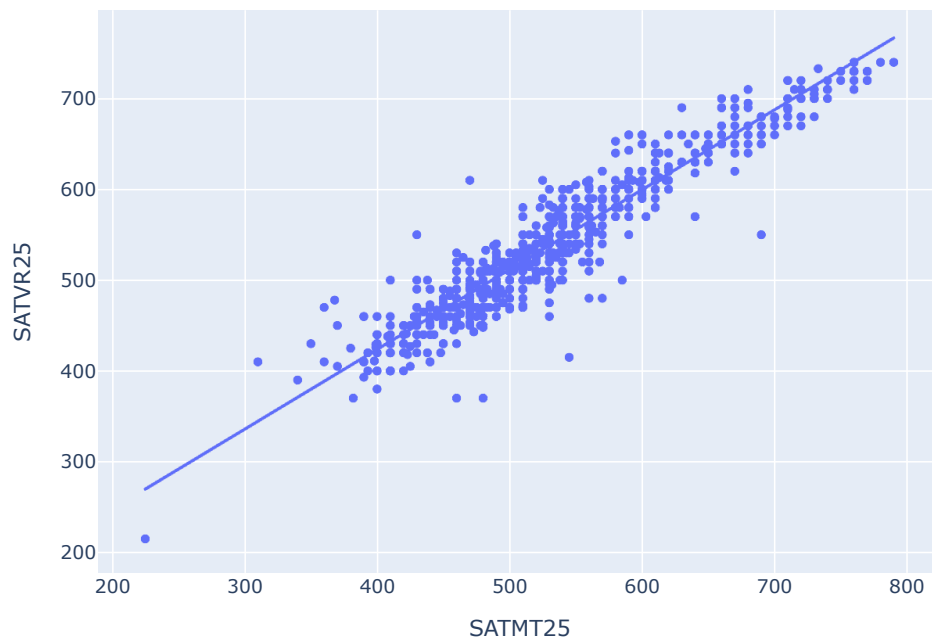


Figure 6.40 — Trendline in a contrasting color.

```
[15]: sat_fig = px.scatter(csc,  
                           x='SATMT25',  
                           y='SATVR25',  
                           hover_data='INSTNM',  
                           trendline='ols',  
                           trendline_color_override='black')  
  
sat_fig
```

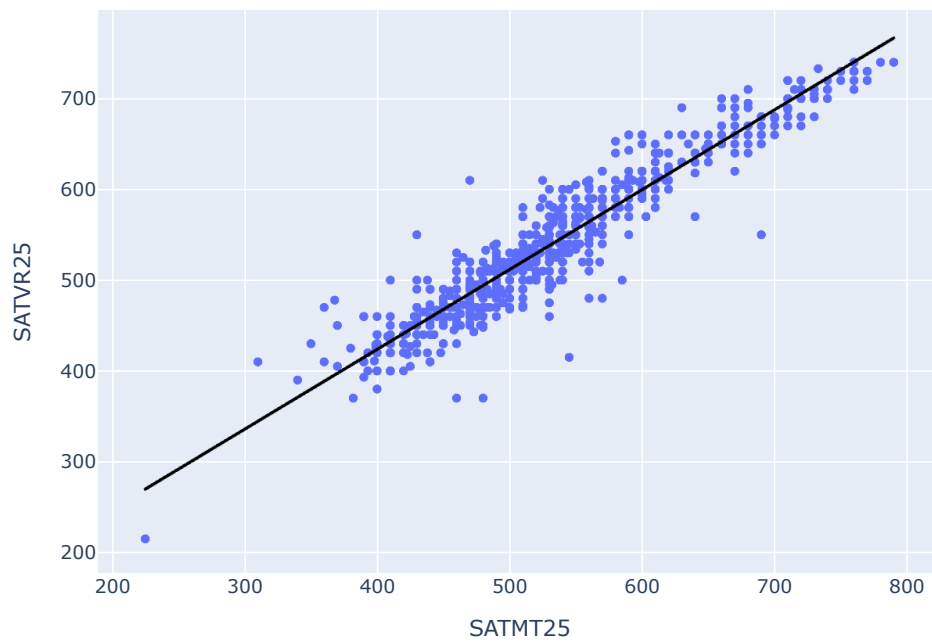
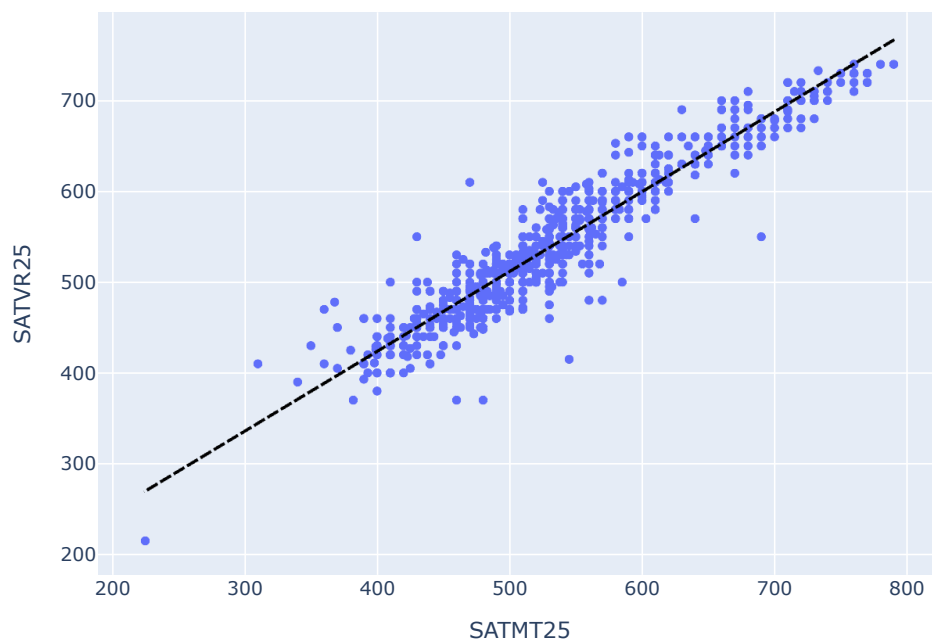


Figure 6.41 — Dashed trendline via the figure’s trace objects.

```
[16]: # Change the line style of the trendline.  
sat_fig.data[1].line.dash = 'dash'  
  
sat_fig
```



Fit summary from `px.get_trendline_results` (see [Plotly docs](#)). The chapter discusses $R^2 \approx 0.90$ for this relationship.

```
[17]: ols_info = px.get_trendline_results(sat_fig)
      ols_info
```

```
[17]:                                px_fit_results
      0  <statsmodels.regression.linear_model.Regressio...
```

Access the underlying statsmodels summary for the first (only) trendline.

```
[18]: ols_info.px_fit_results.iloc[0].summary()
```

```
[18]:
```

Dep. Variable:	y	R-squared:	0.903
Model:	OLS	Adj. R-squared:	0.902
Method:	Least Squares	F-statistic:	9322.
Date:	Thu, 03 Sep 2026	Prob (F-statistic):	0.00
Time:	15:20:56	Log-Likelihood:	-4636.6
No. Observations:	1008	AIC:	9277.
Df Residuals:	1006	BIC:	9287.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	71.8545	4.883	14.714	0.000	62.272	81.437
x1	0.8796	0.009	96.550	0.000	0.862	0.898
Omnibus:		108.921	Durbin-Watson:		1.955	
Prob(Omnibus):		0.000	Jarque-Bera (JB):		704.797	
Skew:		-0.229	Prob(JB):		9.02e-154	
Kurtosis:		7.071	Cond. No.		3.45e+03	

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 3.45e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Strong linear association between math and verbal 25th percentiles; most table output is standard regression diagnostics.

1.4 Cost vs. acceptance rate

Figures 6.43–6.46 — Exploratory summaries, scatter, Indiana highlight, labels, and a graph-objects version.

```
[19]: csc['COSTT4_A'].describe()
```

```
[19]: count      3316.000000
      mean      29298.406514
      std       17702.594870
      min       5500.000000
      25%       15092.750000
      50%       23940.500000
      75%       38900.500000
      max       86964.000000
      Name: COSTT4_A, dtype: float64
```

```
[20]: csc['ADM_RATE_ALL'].describe()
```

```
[20]: count      2215.000000
      mean        0.715795
      std         0.234499
      min         0.000000
      25%         0.596295
      50%         0.769961
      75%         0.888959
      max         1.000000
      Name: ADM_RATE_ALL, dtype: float64
```

Counts differ between cost and admission columns—summary stats describe different sets of non-missing schools.

Most expensive institution by sticker price (`idxmax`).

```
[21]: csc.loc[csc['COSTT4_A'].idxmax()][['INSTNM', 'COSTT4_A']]
```

```
[21]: INSTNM      Jewish Theological Seminary of America
COSTT4_A      86964.0
Name: 2093, dtype: object
```

Search institution names containing “Ohio” to see naming conventions.

```
[22]: csc[csc['INSTNM'].str.contains('Ohio')][['INSTNM', 'CITY']]
```

```
[22]:
```

	INSTNM	CITY
2407	Athenaeum of Ohio	Cincinnati
2422	Central Ohio Technical College	Newark
2433	Ohio Christian University	Circleville
2482	Ohio Business College-Sheffield	Sheffield Village
2483	Ohio Business College-Sandusky	Sandusky
2490	Mercy College of Ohio	Toledo
2491	Methodist Theological School in Ohio	Delaware
2509	Northeast Ohio Medical University	Rootstown
2510	University of Northwestern Ohio	Lima
2512	Ohio Technical College	Cleveland
2513	Ohio Dominican University	Columbus
2514	Ohio Northern University	Ada
2515	Ohio State University Agricultural Technical I...	Wooster
2516	Ohio State University-Lima Campus	Lima
2517	Ohio State University-Mansfield Campus	Mansfield
2518	Ohio State University-Marion Campus	Marion
2519	Ohio State University-Newark Campus	Newark
2520	Ohio State Beauty Academy	Lima
2521	Ohio State College of Barber Styling	Columbus
2523	Ohio State School of Cosmetology-Canal Winchester	CANAL WINCHESTER
2524	Ohio State University-Main Campus	Columbus
2525	Ohio University-Eastern Campus	Saint Clairsville
2526	Ohio University-Chillicothe Campus	Chillicothe
2527	Ohio University-Southern Campus	Ironton
2528	Ohio University-Lancaster Campus	Lancaster
2529	Ohio University-Main Campus	Athens
2530	Ohio University-Zanesville Campus	Zanesville
2531	East Ohio College	East Liverpool
2533	Ohio Wesleyan University	Delaware
3965	Ohio State School of Cosmetology-Heath	Heath
4013	Ohio Media School-Valley View	Valley View
4083	Ohio Media School-Cincinnati	Norwood
4710	Ohio Media School-Columbus	Columbus
4711	Ohio Medical Career College	Dayton
4714	Chamberlain University-Ohio	Columbus
5286	DeVry University-Ohio	Columbus
5322	Ohio Institute of Allied Health	Huber Heights

Query a specific campus (quote style matters inside query).

```
[23]: csc.query('INSTNM == "Ohio State University-Main Campus"')[['INSTNM',
    ↪ 'COSTT4_A', 'ADM_RATE_ALL']]
```

```
[23]:
```

	INSTNM	COSTT4_A	ADM_RATE_ALL
2524	Ohio State University-Main Campus	28133.0	0.527236

Figure 6.43 — Scatter of sticker price vs. acceptance rate.

```
[24]: adm_cost_fig = px.scatter(csc,
    x='ADM_RATE_ALL',
    y='COSTT4_A',
    hover_data='INSTNM')
adm_cost_fig
```

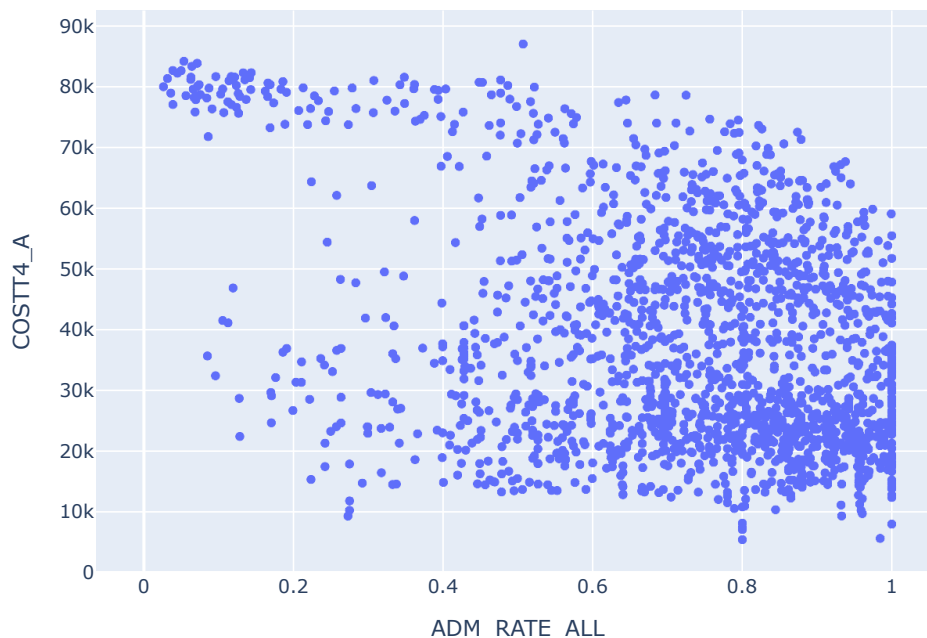


Figure 6.44 — Boolean mask for Indiana (STABBR == 'IN') with color and symbol encoding.

```
[25]: # Create a boolean DataFrame to identify Indiana schools.
IN_bool = csc['STABBR'] == 'IN'
```

```
[26]: adm_cost_fig = px.scatter(csc,
                                x='ADM_RATE_ALL',
                                y='COSTT4_A',
                                hover_data='INSTNM',
                                color=IN_bool,
                                symbol_sequence=['x-thin', 'circle'],
                                symbol=IN_bool
                                )

adm_cost_fig
```

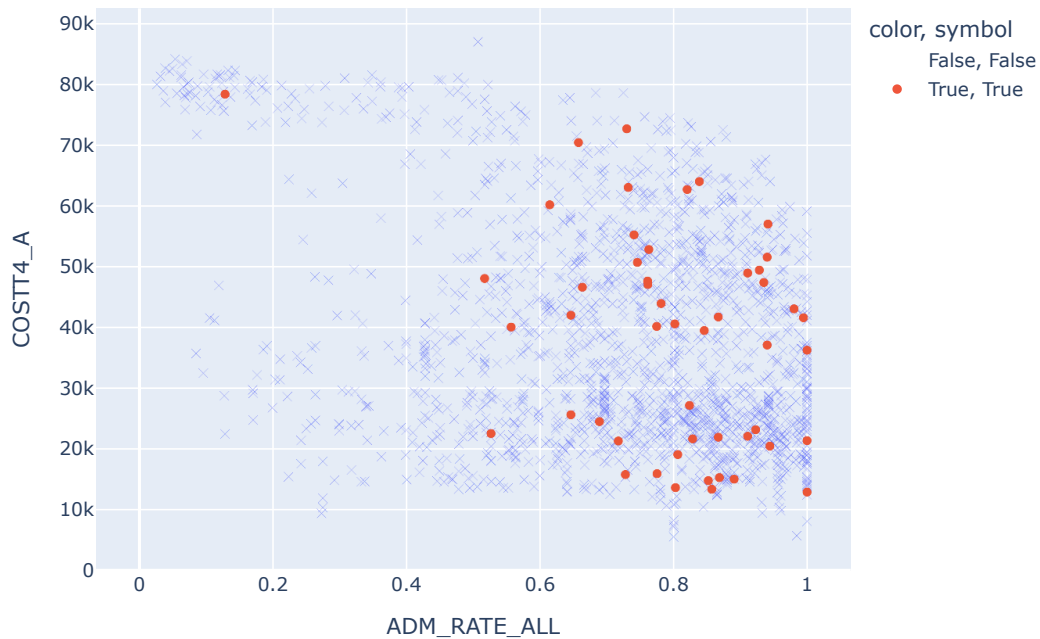


Figure 6.45 — Label only selected Indiana schools; widen the x-axis and position text so labels are readable.

```
[27]: # Make a list of the institutions we want to label.
schools_to_label = ['Valparaiso University',
                    'University of Notre Dame',
                    'Purdue University-Main Campus',
                    'Veritas Baptist College',
                    'DePauw University',
                    'Earlham College',
                    'Indiana University-Bloomington',
                    'University of Indianapolis']
```

Build a labels column: empty except for chosen names (isin).

```
[28]: # Get the indexes of the rows for the schools we
# want to label.
idx = csc['INSTNM'].isin(schools_to_label)

# Create a new column with empty labels.
csc['labels'] = ''

# Replace the blank labels with school names
# for the chosen schools.
csc.loc[idx, 'labels'] = csc.loc[idx, 'INSTNM']

# Check that the labels column has the correct values in it.
csc['labels'].value_counts()
```

```
[28]: labels
                                     6476
DePauw University                   1
Earlham College                     1
University of Indianapolis           1
Indiana University-Bloomington      1
University of Notre Dame            1
Valparaiso University               1
Purdue University-Main Campus        1
Veritas Baptist College              1
Name: count, dtype: int64
```

Express version with text labels and axis titles.

```
[29]: # Create the scatter plot.
adm_cost_fig = px.scatter(csc,
                           x='ADM_RATE_ALL',
                           y='COSTT4_A',
                           hover_data='INSTNM',
                           color=IN_bool,
                           symbol_sequence=['x-thin', 'circle'],
                           symbol=IN_bool,
                           text='labels',
                           labels={'ADM_RATE_ALL': 'Acceptance Rate',
                                   'COSTT4_A': 'Cost to Attend'},
                           height=600
                           )

# Remove the legend.
adm_cost_fig.update_layout(showlegend=False)

# Make sure none of the labels get cut off.
```

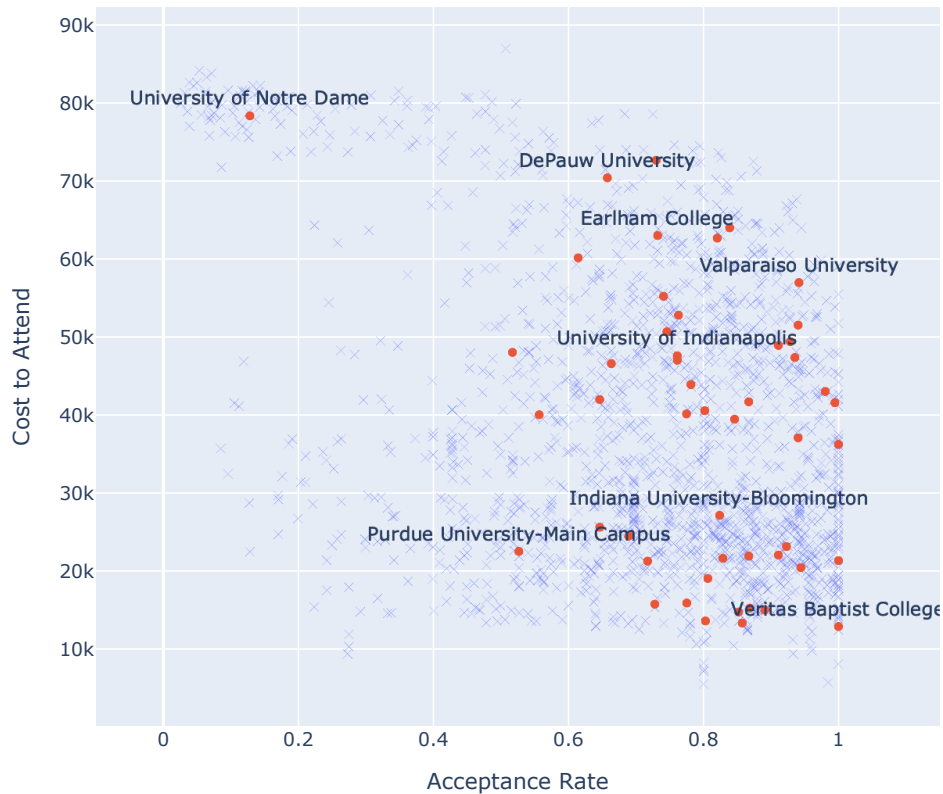
```

adm_cost_fig.update_xaxes(range=[-0.1, 1.15])

# Set where the labels appear relative to the points.
adm_cost_fig.update_traces(textposition='top center')

adm_cost_fig

```



1.5 Graph objects approach

Figure 6.46 — Two traces (non-Indiana vs. Indiana), explicit colors, opacity, and legend. Trace order puts Indiana on top.

```

[30]: # Import the graph objects part of Plotly.
import plotly.graph_objects as go
import numpy as np

```

Split the dataframe with `IN_bool` and `np.invert`.

```
[31]: # Pick out just the schools in Indiana.
IN_csc = csc[IN_bool].copy()

# Create a DataFrame of all other schools (not in Indiana).
non_IN_csc = csc[np.invert(IN_bool)].copy()
```

Add the same labels column on the Indiana subset only.

```
[32]: # Get the indexes of the rows for the schools we
# want to label.
idx = IN_csc['INSTNM'].isin(schools_to_label)

# Create a new column with empty labels.
IN_csc['labels'] = ''

# Replace the blank labels with school names
# for the chosen schools.
IN_csc.loc[idx, 'labels'] = IN_csc.loc[idx, 'INSTNM']

# Check that the labels column has the correct values in it.
IN_csc['labels'].value_counts()
```

```
[32]: labels
134
DePauw University      1
Earlham College        1
University of Indianapolis  1
Indiana University-Bloomington  1
University of Notre Dame  1
Valparaiso University  1
Purdue University-Main Campus  1
Veritas Baptist College  1
Name: count, dtype: int64
```

go.Scatter: red markers + text for Indiana, translucent green × for others. If a label fails to appear, `texttemplate='%{text}'` can help (see chapter note).

```
[33]: # Make red dots for the Indiana schools.
# Add text labels for selected schools.
IN_trace = go.Scatter(x=IN_csc['ADM_RATE_ALL'],
                      y=IN_csc['COSTT4_A'],
                      mode='markers+text',
                      marker_color='red',
                      text=IN_csc['labels'],
                      textposition='top center',
                      texttemplate='%{text}',
                      name='Schools in Indiana',
                      hovertext=IN_csc['INSTNM'])
```

```

    )
# Make translucent green x's for non-Indiana schools.
non_IN_trace = go.Scatter(x=non_IN_csc['ADM_RATE_ALL'],
                          y=non_IN_csc['COSTT4_A'],
                          mode='markers',
                          marker_symbol='x',
                          marker_color='green',
                          marker_opacity=0.2,
                          name='Schools Outside Indiana',
                          hovertext=non_IN_csc['INSTNM'])
)
# Create a figure with those traces, making sure IN schools
# are on top.
adm_cost_fig = go.Figure(data=[non_IN_trace, IN_trace])

# Add some formatting.
adm_cost_fig.update_layout(height=600, width=900)
adm_cost_fig.update_xaxes(range=[-0.1, 1.15], title='Acceptance Rate')
adm_cost_fig.update_yaxes(range=[8000, 85000], title='Cost to Attend')
adm_cost_fig.update_traces(textposition='top center')

adm_cost_fig

```



Graph objects give direct control over styling and the legend compared to the express shortcut

above.

Caveat: “Cost to attend” here is sticker price; acceptance rates are hard to compare across institutions and over time. See the closing discussion in the chapter.