Supporting Information: Analyzing open science data in style: A reproducible repoducible research report report

Thomas P. Urbach
Kutas Lab
Cognitive Science Department
University of California
San Diego

Contents

Summary

Additional information can go here and be formated to APA 6th guidelines or something else. The supporting information and main document can use the same research_report.bib file so the references match. With `\usepackage[xr,user,titleref]{zref}`, you can cross-reference back and forth between documents . . . the main report and this SI. For instance here is a reference to Figure 2 in the main document. The reference is via the label, i.e., `\zlabel{ms:multipanel}` so if the figure is moved to a different page or its number changes because of additions or deletions, this reference by number will update automatically. The following sections show the source files that generated the plots, figures, manuscript and si .pdfs.

APA 7th docs http://ctan.math.washington.edu/tex-archive/macros/latex/contrib/apa7/apa7.pdf APA 6th docs http://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/biblatex-contrib/biblatex-apa6/biblatex-apa6.pdf

System setup

Installing conda environments

If you already use conda environments in a recent linux operating system, you can install a minimal conda environment to run the notebooks like so and follow the prompt (or omit -y to the end of the command to install the packages without prompting).

```
conda create -n apa67_report pandas pyarrow matplotlib jupyter firefox -y
$ activate apa67_report
$ jupyter notebooks
```

If you are not yet set up to use conda environments, you can follow the instructions to download and install a minimal conda installer, miniconda3 (). This provides just enough infrastructure to create a conda environemnt and install packages as shown in the example above. If you want to create conda environments and install packages faster, then install the ‘mamba‘ conda package ().

If you are not yet set up to use conda environments and don't want to be then you are on your own. You can run pipeline_1.ipynb if you have numpy, pandas, matplotlib and jupyter. You need the spudtr package to run pipeline_2.ipynb. Older versions are available via pip install, but there is no assurance it is compatible with the versions of packages you already have installed.

Installing LaTeX

Linux Installation via network

You do not need to be root or admin to install TeX Live over the networks and best practices are to install your copy in your directory. That way you control the version and packages you use. First read through the quick installation instructions here. Then, (summarizing from https://www.tug.org/texlive/acquire-netinstall.html):

1. Download install-tl-unx.tar.gz to some scratch/working directory, unpack the archive, change to the new directory it made, i.e., install–tl–YEARMONTHDAY for whatever version, and run the installer.

   ```
   $ tar -xf install-tl-unx.tar.gz
   $ cd install-tl-20200814
   $ perl install-tl
   ```

   Follow the prompts, make sure you are happy with and have write permissions in the default installation directory, and press “i” to install.

2. Update your /.bashrc file with the path to the new TeX Live installation.

```
PATH=/home/turbach/texlive/2020/bin/x86_64-linux:$PATH
INFOPATH=/home/turbach/2020/texmf-dist/doc/info:$INFOPATH
MANPATH=/home/turbach/2020/texmf-dist/doc/man:$MANPATH
```

That's it, you have a complete functioning installation of LaTeX with the latest packages, TeX Live 2020 as of this writing.

The installation probably has everything you need including the apa6 and apa7 styles used for this report.

If there is a new package or update you want and you want to manage the TeX packages with the TeX Live GUI you also need to install perl/tk. There is a conda package for this, you can install into any compatible conda env.

```
$ conda activate some_general_purpose_env
$ conda install perl-tk -c BioBuilds -y
```

OSX Installation

See instructions for MacTeX here: https://www.tug.org/mactex/

Windows

See Quick Install insructions here
and Windows installer instructions here.

Source: author_analysis.ipynb

The pdf of the notebook is generated by `jupyter convert ... -to pdf`. The LaTeX package `pdfpages` is used to slurp it into the SI pdf.

apa_analysis

September 5, 2021

1 Reproducible results for LaTeX manuscripts

- arbitrary narrative text and results
- pandas LaTeX table generation
- custom APA-style table generation
- APA-style graphics styled with matplotlib style sheets

WARNING: Running this code the first time downloads an 87MB EEG data file to your disk from Zenodo.

The package dependencies are python, numpy, pandas, pyarrow, matplotlib, jupyter

2 The reproducible data analysis

Set up Python packages for data analysis and visualization

Guard the conda environment and EEG file MD5 checksum

[1]:

```python
import os
import re
import copy
import hashlib
import warnings
from pathlib import Path
import pprint as pp
import platform
import numpy as np
import pandas as pd


# matplotlib and packages for plot tuning
import matplotlib as mpl
from matplotlib import pyplot as plt
from matplotlib import cycler
from matplotlib import cm

# guard conda environment
conda_env = os.environ["CONDA_DEFAULT_ENV"] if "CONDA_DEFAULT_ENV" in os.
 ↪environ.keys() else None
```

```
if conda_env and not conda_env == "apa67_report_090421":
    msg = (
        f"unknown conda env {conda_env}, to reproduce the report on linux␣
↪create run these:\n\n"
        f"    conda create -n apa67_report_090421 --files environment.txt\n"
        f"    conda activate \n\n"
    )
    warnings.warn(msg)

# fetch the EEG recording from Zenodo if it isn't found locally
ARCHIVE = r"https://zenodo.org/record/3968485/files/"
DATA_F = "sub000p3.ms1500.epochs.feather"
if not Path(DATA_F).exists():
    print(f"downloading {DATA_F} from Zenodo ... please wait")
    pd.read_feather(ARCHIVE + DATA_F).to_feather(DATA_F)
    print("ok")

# guard the data file MD5
with open(DATA_F, 'rb') as _f:
    checksum = hashlib.md5(_f.read()).hexdigest()
    if not checksum == "9ce6af68c74ab0fca41bd1da3414533d":
        raise ValueError(f"bad md5 checksum {DATA_F}")

print(f"{DATA_F} ok")

for pkg in [np, pd, mpl]:
    print(pkg.__name__, pkg.__version__)
```

```
sub000p3.ms1500.epochs.feather ok
numpy 1.21.2
pandas 1.3.2
matplotlib 3.4.3
```

3 Experiment parameters

3.1 Electrode and fiducial landmark locations

[2]:

```
# ------------------------------------------------------------
# scalp electrodes, EOG, mastoids, ground
import io
sph26_txt = io.StringIO("""
channel  phi   theta  ch_type
MiPf  90.0   90.0   eeg
LLPf  90.0  126.0   eeg
LLFr  90.0  162.0   eeg
LLTe  90.0  198.0   eeg
LLOc  90.0  234.0   eeg
MiOc  90.0  270.0   eeg
RLOc  90.0  306.0   eeg
RLTe  90.0  342.0   eeg
RLFr  90.0   18.0   eeg
RLPf  90.0   54.0   eeg
LMPf  59.0  108.0   eeg
LDFr  59.0  144.0   eeg
LDCe  59.0  180.0   eeg
LDPa  59.0  216.0   eeg
LMOc  59.0  252.0   eeg
RMOc  59.0  288.0   eeg
RDPa  59.0  324.0   eeg
RDCe  59.0    0.0   eeg
RDFr  59.0   36.0   eeg
RMPf  59.0   72.0   eeg
LMFr  26.0  126.0   eeg
LMCe  26.0  198.0   eeg
MiPa  26.0  270.0   eeg
RMCe  26.0  342.0   eeg
RMFr  26.0   54.0   eeg
MiCe   0.0    0.0   eeg
A1    130.0  205.0  ref
A2    130.0  335.0  ref
lle   140.0  120.0  eog
rle   140.0   60.0  eog
lhz   108.0  130.0  eog
rhz   108.0   50.0  eog
nasion 108.0  90.0  fid
lpa    108.0 180.0  fid
rpa    108.0   0.0  fid
gnd     72.0   90.0 gnd
""")

# parse lcoations into a data frame
SPH_LOCS = pd.read_csv(sph26_txt, sep="\s+")
SPH_LOCS.insert(3, "r", np.sin(SPH_LOCS["phi"]))
SPH_LOCS

def sph2cart(row):
    """convert spherical coordinates to 2-D cartesian"""
    row = row.copy()
    label, phi, theta, r, ch_type = [*row]

    deg2rad = 2.0 * np.pi / 360.0
    phi *= deg2rad
    theta *= deg2rad

    x = np.cos(theta) * np.sin(phi)
    y = np.sin(theta) * np.sin(phi)
    z = np.cos(phi)

    # lambert projection
    lambert_x = x * np.sqrt(1 / (1 + z))
    lambert_y = y * np.sqrt(1 / (1 + z))

    row['x'], row['y'], row['z'] = x, y, z
    row['x_lambert'], row['y_lambert'] = lambert_x, lambert_y

    return row

SPH_CART_LOCS = SPH_LOCS.apply(lambda row: sph2cart(row), axis=1)
```

3.2 Data columns and indexes

[3]:

```
INDEXES = ["epoch_id", "time_ms"]
EEG_MIDLINE = ["MiPf", "MiCe", "MiPa", "MiOc"]
EXPT_VARS =  ["bin", "tone", "stimulus", "accuracy"]

EEG_COLUMNS = SPH_LOCS.query("ch_type == 'eeg'")["channel"].tolist()
COI = INDEXES + EXPT_VARS + EEG_COLUMNS # EEG_MIDLINE
```

3.3 Groom the recordings for analysis

[4]:

```
data = pd.read_feather("sub000p3.ms1500.epochs.feather")
data.rename(columns={"match_time": "time_ms"}, inplace=True)
data["epoch_id"] = data["epoch_id"].astype(int)
data.rename(columns={"stim": "stimulus"}, inplace=True)

# data QC screening
display(len(data.epoch_id.unique()))
good_epoch_ids = data.query("time_ms==0 and log_flags==0").epoch_id
data = data.query("epoch_id in @good_epoch_ids")
print(data.columns)

good_epochs = []
absmax = 125
for epoch_id, epoch in data.groupby("epoch_id"):
    vals = epoch[EEG_COLUMNS].to_numpy().flatten()
    if vals.max() - vals.min() <= absmax:

        # center EEG on mean amplitude 200 - 0 ms prestimulus
        epoch[EEG_COLUMNS] = (
            epoch[EEG_COLUMNS]
            - epoch.query("time_ms >= -200 and time_ms < 0")[EEG_COLUMNS].mean()
        )
        good_epochs.append(epoch)

p3_eeg = pd.concat(good_epochs, axis=0)


# save
p3_eeg[COI].reset_index(drop=True).to_feather("p3_eeg.fthr")
```

```
600
```

```
Index(['epoch_id', 'data_group', 'dblock_path', 'dblock_tick_idx',
       'dblock_ticks', 'crw_ticks', 'raw_evcodes', 'log_evcodes', 'log_ccodes',
       'log_flags', 'epoch_match_tick_delta', 'epoch_ticks', 'dblock_srate',
       'match_group', 'idx', 'dlim', 'anchor_str', 'match_str', 'anchor_code',
       'match_code', 'anchor_tick', 'match_tick', 'anchor_tick_delta',
       'is_anchor', 'regexp', 'ccode', 'instrument', 'bin', 'tone', 'stimulus',
       'accuracy', 'acc_type', 'time_ms', 'anchor_time', 'anchor_time_delta',
       'pygarv', 'lle', 'lhz', 'MiPf', 'LLPf', 'RLPf', 'LMPf', 'RMPf', 'LDFr',
       'RDFr', 'LLFr', 'RLFr', 'LMFr', 'RMFr', 'LMCe', 'RMCe', 'MiCe', 'MiPa',
       'LDCe', 'RDCe', 'LDPa', 'RDPa', 'LMOc', 'RMOc', 'LLTe', 'RLTe', 'LLOc',
       'RLOc', 'MiOc', 'A2', 'HEOG', 'rle', 'rhz'],
      dtype='object')
```

3.4 Load the groomed EEG data

[5]:

```
p3_df = pd.read_feather("p3_eeg.fthr")
p3_events = p3_df.query("time_ms == 0 and stimulus != 'cal'")[INDEXES +␣
 ↪EXPT_VARS]

display(len(p3_df.epoch_id.unique()))
display(p3_events.shape)
```

```
447
```

```
(239, 6)
```

3.5 Tabulate stimulus event counts by experimental condition

[6]:

```
event_table = pd.crosstab(p3_events.stimulus, p3_events.tone, margins=True)

# event_table.columns = [col for col in event_table.columns]
event_table.reset_index(inplace=True)

# event_table["stimulus"] = event_table["stimulus"].str.capitalize()
# event_table.columns = event_table.columns.str.capitalize()

event_table.set_index("stimulus", inplace=True)
display(event_table)
```

```
tone       hi   lo  All
stimulus
standard  107   94  201
target     14   24   38
All       121  118  239
```

4 Example: Linking data and arbitrary text

[7]:

```
# data variables from the table for clarity
n_trials = event_table["All"]["All"]
n_standards = event_table.loc["standard"]["All"]
n_targets = event_table.loc["target"]["All"]

# a bit of data validation
assert n_standards + n_targets == event_table["All"]["All"]

# compute the proportion ... a derived value
p_targets = n_targets / (n_standards + n_targets)
n_trials, n_standards, n_targets, p_targets
```

[7]:

```
(239, 201, 38, 0.1589958158995816)
```

[8]:

```
# embed data into formatted LaTex via the variables

arbitrary_text = f"""
% These two paragraphs are generated when the analysis is run

The essential feature of reproducible report generation is linking
data from the analysis with the text of the report. Style conventions
like APA 6\\textsuperscript{{th}}, 7\\textsuperscript{{th}} and others are
strict and varied which means the only general solution is a mechanism
for linking the analysis data and results to arbitrary text formatted
arbitrarily.  This is an old problem, solved long ago by string formatting
functions, e.g., \mintinline{{c}}{{sprintf()}} in C, which reappears in
various forms in scripting languages like R, MATLAB, and Python where the
f-string function (Python 3.6+) streamlines mixing text and variables.

To illustrate, the same Jupyter notebook that runs the analysis also
generates a text file containing the entire contents of the preceding
paragraph and this one, including the following sentence that describes
the number of trials in each experimental condition.
```

```
%%
%% In the next sentence, the Python f-string formatter embeds variables
%% computed during the analysis directly into the generated text which
%% typeset to APA 6th style specifications.
%%
After screening  artifacts, the proportion of target trials in the data
analyzed was {p_targets:0.3f} ({{\it N}} = {n_trials} trials, {n_standards}
standards, {n_targets} targets).
%%
This narrative description formats the quantitative results in APA 6th style
while the values are filled in by the same variables used to compute them. This
technique can be used to generate reproducible descriptions of an
entire results sections or portions thereof.
"""

# show (optional)
print(arbitrary_text)

# write the text to a file for import into the manuscript
with open("generated/arbitrary_text.tex", "w") as fh:
    fh.write(arbitrary_text)
```

```
% These two paragraphs are generated when the analysis is run

The essential feature of reproducible report generation is linking
data from the analysis with the text of the report. Style conventions
like APA 6\textsuperscript{th}, 7\textsuperscript{th} and others are
strict and varied which means the only general solution is a mechanism
for linking the analysis data and results to arbitrary text formatted
arbitrarily.  This is an old problem, solved long ago by string formatting
functions, e.g., \mintinline{c}{sprintf()} in C, which reappears in
various forms in scripting languages like R, MATLAB, and Python where the
f-string function (Python 3.6+) streamlines mixing text and variables.

To illustrate, the same Jupyter notebook that runs the analysis also
generates a text file containing the entire contents of the preceding
paragraph and this one, including the following sentence that describes
the number of trials in each experimental condition.
%%
%% In the next sentence, the Python f-string formatter embeds variables
%% computed during the analysis directly into the generated text which
%% typeset to APA 6th style specifications.
%%
After screening  artifacts, the proportion of target trials in the data
analyzed was 0.159 ({\it N} = 239 trials, 201
standards, 38 targets).
```

```
%%
This narrative description formats the quantitative results in APA 6th style
while the values are filled in by the same variables used to compute them. This
technique can be used to generate reproducible descriptions of an
entire results sections or portions thereof.
```

5 Example: Table 1

An easy LaTeX table with `pandas.DataFrame.to_latex()`

The output is not quite APA 6th style.

[9]:
```
# show
print(event_table.to_latex())

# save
event_table.to_latex('generated/p3_table1.tex')
```

```
\begin{tabular}{lrrr}
\toprule
tone &    hi &    lo &   All \\
stimulus &       &       &       \\
\midrule
standard &   107 &    94 &   201 \\
target   &    14 &    24 &    38 \\
All      &   121 &   118 &   239 \\
\bottomrule
\end{tabular}
```

6 Example: Table 2

An APA 6th style LaTeX table built with Python

Build the header, data rows and columns, footer strings, then write the LaTeX file.

[10]:
```
def df_to_tex(df):
    """format df values as a LaTeX string of rows x columns table data"""

    df_str = df.applymap(lambda x: f"{x}".capitalize()) # convert the data to␣
 ↪APA style text
    tex_cols = df_str.apply(lambda row: " & ".join(row), axis=1) # join the␣
 ↪columns with &
    tex_rows_cols = (r" \\ " + "\n").join(tex_cols)  # join the rows with \\
    return tex_rows_cols
```

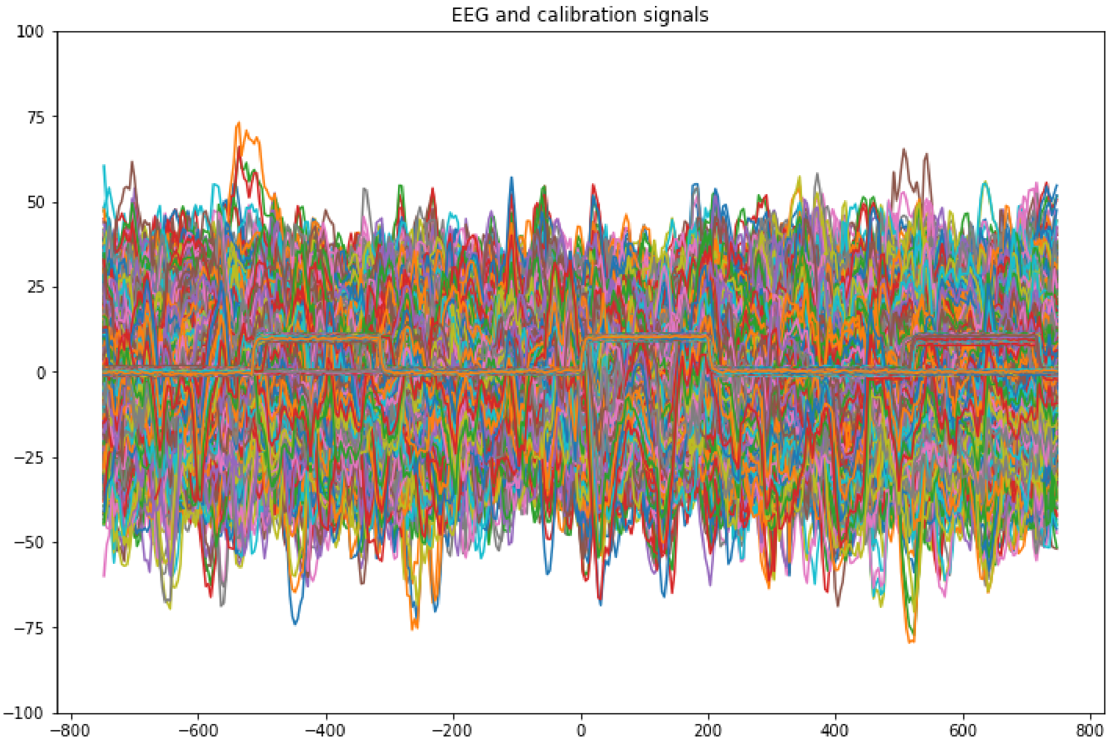
```
# 1. build the table header by hand thanks to APA style
table1_header = f"""
\\begin{{tabular}}{{llll}}
\\toprule
 &  \multicolumn{{2}}{{c}}{{Tone}} & \\\\
\\cmidrule{{2-3}}
 & {" & ".join([s.capitalize() for s in event_table.columns])} \\\\
\\midrule
"""

# 2. build the table rows and columns
table1_rows = df_to_tex(event_table.reset_index())

# 3. build table footer
table1_footer = "\\\\ \n\\bottomrule \n\end{tabular}"

# assemble the text
table1_tex = table1_header + table1_rows + table1_footer

# show
print(table1_tex)

# save for the manuscript
with open("generated/p3_table2.tex", "w") as fh:
    fh.write(table1_tex)
```

```
\begin{tabular}{llll}
\toprule
 &  \multicolumn{2}{c}{Tone} & \\
\cmidrule{2-3}
 & Hi & Lo & All \\
\midrule
Standard & 107 & 94 & 201 \\
Target & 14 & 24 & 38 \\
All & 121 & 118 & 239\\
\bottomrule
\end{tabular}
```

6.1 EEG data preview

[11]:

```
f_eeg, ax = plt.subplots(figsize=(12, 8))
ax.set_title("EEG and calibration signals")
ax.set_ylim(-100, 100)
times = p3_df.time_ms.unique()
for epoch_id, epoch in p3_df.groupby("epoch_id"):
    ax.plot(times, epoch[EEG_COLUMNS])
```



6.2 Compute time-domain average ERPs

```
[12]: p3_erp = p3_df.groupby(["stimulus", "time_ms"]).mean()[EEG_COLUMNS]
      p3_std = p3_df.groupby(["stimulus", "time_ms"]).std()[EEG_COLUMNS]
      p3_n = p3_df.groupby(["stimulus", "time_ms"]).count()[EEG_COLUMNS]  # n's␣
       ↪differs by condition after data QC

      for df in [p3_erp, p3_std, p3_n]:
          df.columns.name = "channel"
```

6.3 Example Figure: P300 midline ERP plots with Psychological Science matlab style sheets

https://www.psychologicalscience.org/publications/aps-figure-format-style-guidelines

2020-08-11

(emphasis in bold added here)

Details:

Please note that yellow may not show up well, especially in line graphs.

In **all labels including the key(the first letter of each important word and of any word of at least 4 letters should be capitalized**.

Exception: Units of measure indicated in parentheses don't have the first letter capitalized, e.g., "Response Time (ms)."

Minus signs **NOT HYPHENS** should be used to indicate negative numbers or subtraction (a minus sign can be inserted by holding down the key on a computer keyboard while pressing 0, 1, 5, 0 on the number pad, in sequence).

Do not insert a box around a key or a figure.)

A graph should have two axes (ordinate and abscissa) only. Do not include extraneous axes. In mathematical expressions, there should be a single letter space before and after each operator: =, ×, +, ?, <, >, etc.

Exception: Do not insert spaces in subscripts or superscripts.

The **ordinate axis must be labeled to indicate the nature of the quantities referred to**. For example, if a graph shows response times (ordinate) in various conditions (abscissa), the ordinate must be labeled "Response Time," in addition to showing the numerical values.

Numerical values on the ordinate axis should be oriented horizontally. If a figure includes error bars, they must be explained in the caption. In the case of a bar graph, be sure that error bars are easily visible (e.g., a black error bar will be invisible in a data bar with a black or dark-gray fill).

Font style and size:

Labels and numbers in figures should be in **Helvetica Neue 57 Condensed roman font**. (If you do not have this font installed on your device, please use regular **Helvetica** or Arial font.)

Do not use boldface font unless it's intended to highlight something. In that case, the caption should explain what the boldface indicates.

Symbols referring to variables should be in Helvetica Neue 57 Condensed italic font. (If you do not have this font installed on your device, please use regular Helvetica or Arial font.) Otherwise, do not use italics.

Greek letters (e.g., regression coefficients) should not be in italics.

All **ordinate and abscissa** quantities, or any sublabel along the ordinate or abscissa, should be in **9-point** font.

All **main ordinate and abscissa labels** should be in **10-point** font.

The **title** header (at the top of a figure), if there is one, should be in **12-point** font.

Keys should be in **9-point** font.

This includes the height of boxes illustrating fills in a bar graph and symbols used to differentiate lines in a line graph.

Whenever possible, the **key should be placed toward the top of a graph** (i.e., toward the top inside the graph or above the graph, as space allows).

Symbols (e.g., squares, diamonds) plotted in a graph should be no smaller than the corresponding symbols in the key.

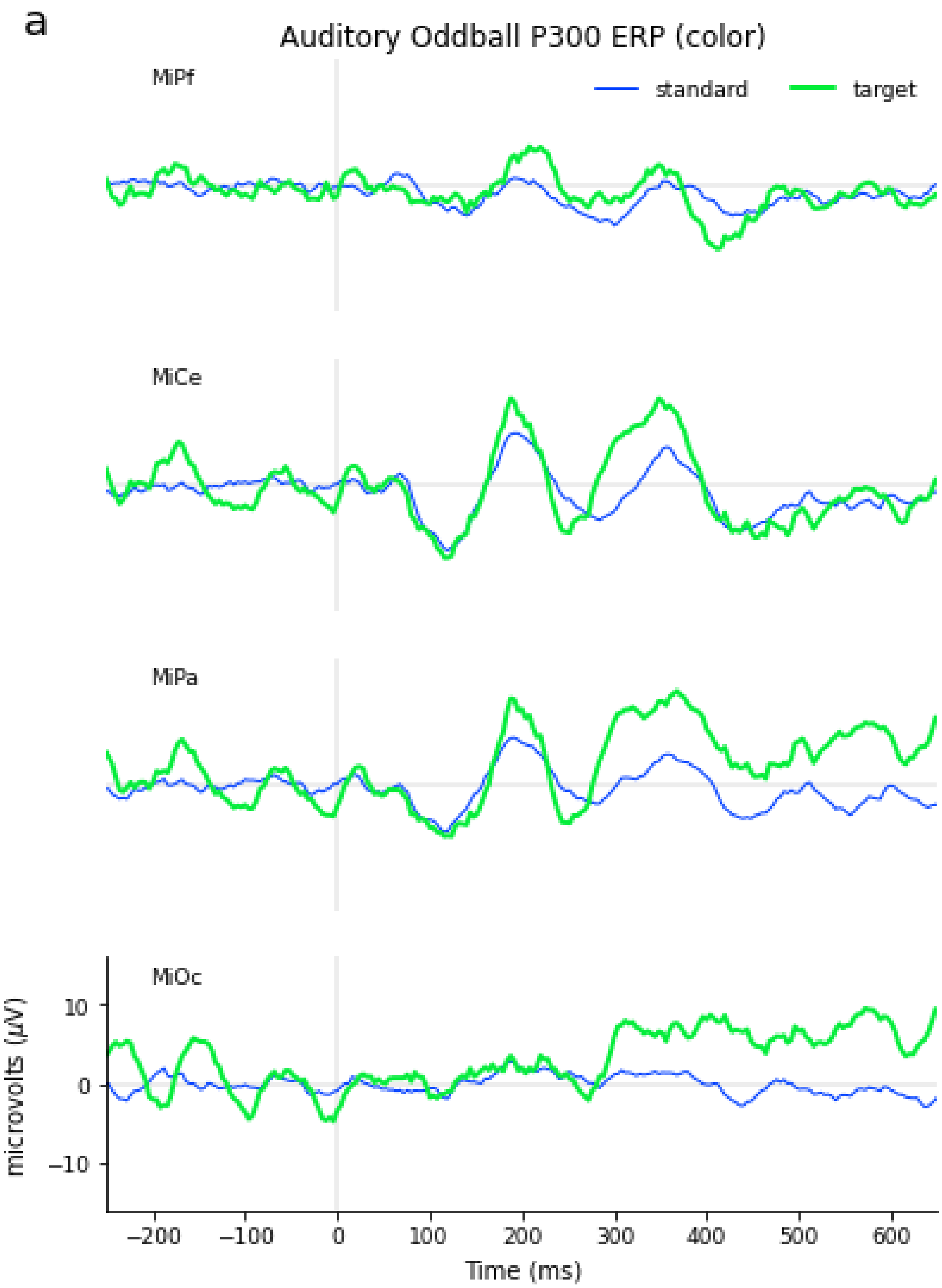
Panel labels (a, b, c, etc.) should be in 18-point font, lowercase, positioned to the upper left of the corresponding panels. They should not be followed by periods or surrounded by parentheses.

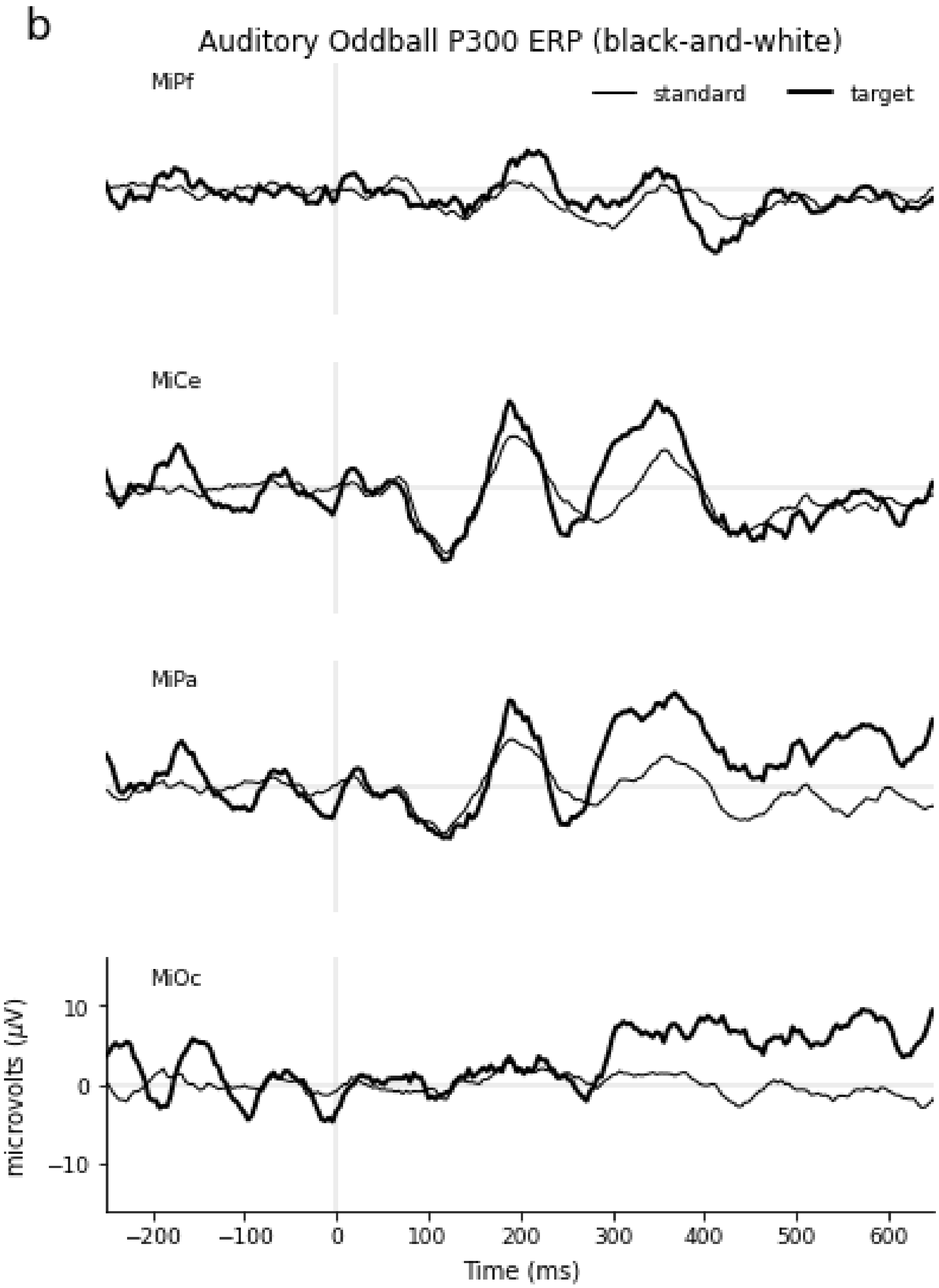
All other text in graphs (e.g., a label for a graphed line or symbol) should be in **9-point** font.

[13]:

```
# seaborn bright
colors = ['#003FFF', '#03ED3A', '#E8000B', '#8A2BE2'] # , '#FFC400', '#00D7FF']

n_colors = len(colors)

psych_sci_fig = {
    # set matplotlib style paramaters to Psych Science specs
    "font.sans-serif": ["Arial", "Helvetica", "DejaVu Sans"],
    "font.size": 18,   # default size for panel label
    "axes.labelsize": 10, # X, Y axis labels
    "axes.titlesize": 12,  # axes title
    "xtick.labelsize": 9,
    "ytick.labelsize": 9,
    "legend.fontsize": 9,
    "legend.frameon": False,
    "lines.linewidth": 2,
    "lines.markersize": 8,

    # set other aesthetics to taste
    "lines.color": "lightgray",
    "lines.solid_capstyle": "round",
    "lines.dash_capstyle": "round",
    "lines.dashdot_pattern": [6.4, 1.6, 1.0, 1.6],
    "lines.dashed_pattern": [4.0, 5.0],
    "lines.dotted_pattern": [0.01, 2.5],

    "axes.spines.top": False,
    "axes.spines.right": False,
    "axes.spines.bottom": False,
    "axes.spines.left": False,
    "axes.prop_cycle": (
        cycler(lw=["1", "2", "3", "3.5"])
        + cycler(ls=["-", "-", "-", "--"])
    )
}

# this cycles colors from our colorbrewer palette
cco = (cycler(color=colors))

# this "cycles" all black lines
cbw = cycler(color=["k"] * len(colors))
```

```
# Figures work in color or black-and-white
panels = {
    "a": {"subtitle": "color", "lines": cco},
    "b": {"subtitle": "black-and-white", "lines": cbw}
}


n_chan = len(EEG_MIDLINE)


for fig_n, (panel, design) in enumerate(panels.items()):
    with plt.style.context(psych_sci_fig):

        # update panel style with line colors
        plt.rcParams["axes.prop_cycle"] = (
            plt.rcParams["axes.prop_cycle"]
            + design["lines"]
        )

        # new figure
        f_ep, axs = plt.subplots(n_chan, 1, figsize=(6, 2 * n_chan),␣
 ↪sharex=True, sharey=True)

        for axi, chan in enumerate(EEG_MIDLINE):

            ax = axs[axi]

            # zero-lines
            ax.axvline(0, alpha=0.4)
            ax.axhline(0, alpha=0.4)
            ax.text(0.05, 0.9, s=chan, transform=ax.transAxes, fontsize=9)

            # ERP waveforms, line styles from the style sheet
            for stim, erp in p3_erp.query("stimulus != 'cal'").
 ↪groupby(["stimulus"]):
                erp = erp.reset_index()
                time = erp.time_ms.unique()
                ax.plot(time, erp[chan], label=stim)

            # panel label and title
            if axi == 0:
                ax.text(-0.1, 1.1, s=f"{panel}", transform=ax.transAxes)
                ax.set_title(f"Auditory Oddball P300 ERP␣
 ↪({design['subtitle']})")
                ax.legend(loc="upper right", ncol=2)
            ax.set(xlim=(-250, 650))
            ax.set(ylim=(-16, 16))

            # style the axes
            if axi == n_chan - 1:

                ax.set_xlabel("Time (ms)")
                ax.spines["left"].set_visible(True)
                ax.spines["bottom"].set_visible(True)

                ax.set_ylabel(r"microvolts ($\mu\mathrm{V}$)")
            else:
                ax.tick_params(bottom=False, labelbottom=False)
                ax.tick_params(left=False, labelleft=False)


        f_ep.tight_layout()
        f_ep.savefig(f"generated/p3_midline_plot{fig_n+1}.pdf")
```


a
Auditory Oddball P300 ERP (color)
MiPf
standard
target
MiCe
MiPa
MiOc
microvolts (μV)
10
0
−10
−200
−100
0
100
200
300
400
500
600
Time (ms)



7 Plot ERP scalp distribution and decorations

- box highlight an interval with `ax.axvspan(from, to, ...)`

- add uncertainty intervals around y +/- u with `ax.fill_between(x, y1=y + u, y2=y-u, ...)`
- highlight a cond1 vs. cond2 effect in an interval with `ax.fill_between(x, y1=cond1, y2=cond2, where, ...)`

[14]:

```
# more styling for bare axes ...
head_trace_style = {
    "xtick.bottom": False,
    "xtick.labelbottom": False,
    "ytick.left": False,
    "ytick.labelleft": False,
    "axes.prop_cycle": cco,
    "font.size": 9,
}


# semi-topographic locations
MPL_32_HEAD = {
    'w': .15,
    'h': .1,
    'chanlocs': {
        'cal': (0.0625, 0.2),
        'lle': (0.25, 0.85),
        'rle': (0.625, 0.85),
        'lhz': (0.0625, 0.85),
        'rhz': (0.8125, 0.85),
        'MiPf': (0.4375, 0.725),
        'MiCe': (0.4375, 0.425),
        'MiPa': (0.4375, 0.275),
        'MiOc': (0.4375, 0.125),
        'LLPf': (0.1875, 0.725),
        'RLPf': (0.6875, 0.725),
        'LMPf': (0.3125, 0.65),
        'RMPf': (0.5625, 0.65),
        'LLFr': (0.0625, 0.5),
        'RLFr': (0.8125, 0.5),
        'LMFr': (0.3125, 0.5),
        'RMFr': (0.5625, 0.5),
        'LDFr': (0.1875, 0.575),
        'RDFr': (0.6875, 0.575),
        'LDCe': (0.1875, 0.425),
        'RDCe': (0.6875, 0.425),
        'LLTe': (0.0625, 0.35),
        'RLTe': (0.8125, 0.35),
        'LMCe': (0.3125, 0.35),
        'RMCe': (0.5625, 0.35),
        'LMOc': (0.3125, 0.2),
        'RMOc': (0.5625, 0.2),
        'LDPa': (0.1875, 0.275),
        'RDPa': (0.6875, 0.275),
        'LLOc': (0.1875, 0.125),
        'RLOc': (0.6875, 0.125),
        'A2': (0.8125, 0.2)
    }
}

MPL_MIDLINE = {
    'w': .75,
    'h': .2,
    'chanlocs': {
        'MiPf': (0.1, 0.7),
        'MiCe': (0.1, 0.5),
        'MiPa': (0.1, 0.3),
        'MiOc': (0.1, 0.1),
        'cal': (0.1, 0.1),
    }
}
```

7.1 Define the decorations

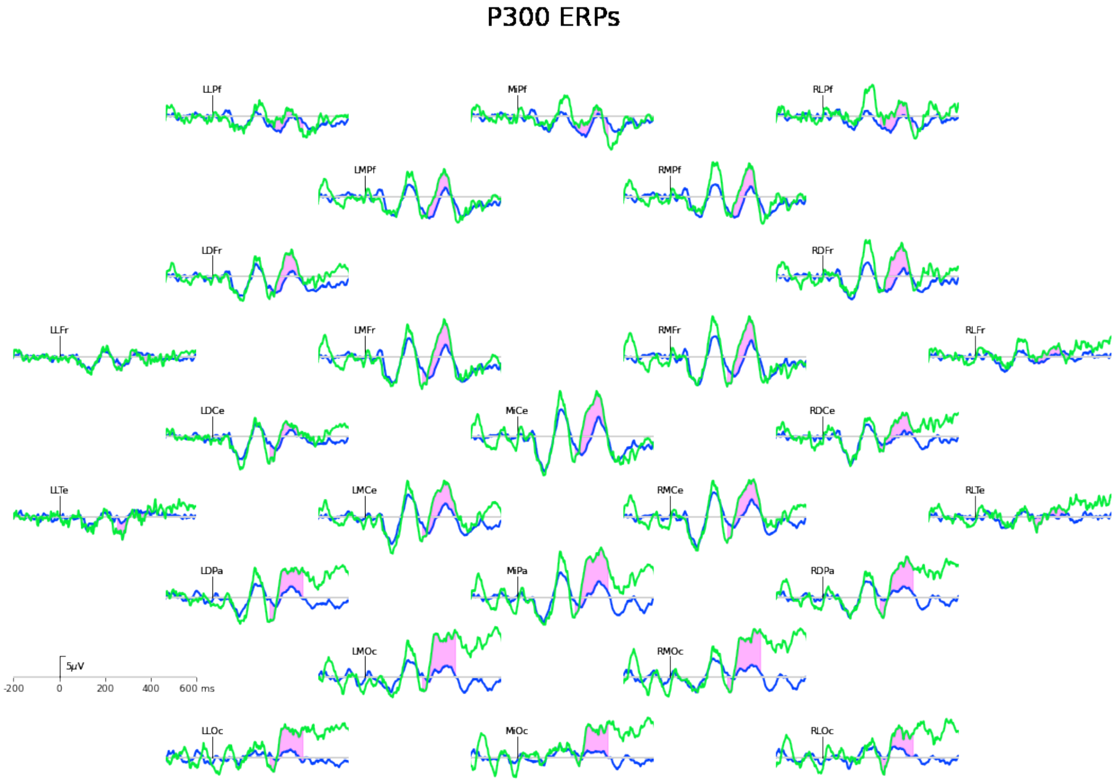
[15]:

```
# timeline, ticks, and labels
tmin, tmax = -200, 600
timeline_ticks = [-200, 0, 200, 400, 600]
timeline_ticklabels = [-200, 0, 200, 400, "600 ms"]



# cal bar in x, y data units
cal_bar_time = 0   # ms
cal_bar_min = 0   # uV
cal_bar_max = 5   # uV
cal_tick_width = 25   # ms

# cal bar line aesthetics
cal_bar_kws = {"color": "black", "lw": 1}

# cal bar label kwargs
cal_bar_label = {
    "x": cal_bar_time + cal_tick_width,
    "y": cal_bar_max / 2.0 ,
    "s": f"{cal_bar_max}" + r"$\mu\mathrm{V}$",
    "ha": "left",
    "va": "center",
}
```

```
# channel label kwargs, label text is per channel
chan_label = {
    "x": cal_bar_time,
    "y": cal_bar_max ,
    "ha": "center",
    "va": "bottom",
}

# ------------------------------
# montage and figure proportions

# chans = EEG_MIDLINE + ["cal"]
# chan_layout = MPL_MIDLINE

chans = EEG_COLUMNS + ["cal"]
chan_layout = MPL_32_HEAD
figsize = (16, 14)

# conditions to plot, add "cal" for fun
plot_stim = ["standard", "target"]
```

[16]:

```
# plot it
with plt.style.context([psych_sci_fig, head_trace_style]):

    fig, axs = plt.subplots(len(chans), figsize=figsize, sharey=True,␣
 ↪sharex=True)

    # proportions
    chan_width = chan_layout["w"]   # .2
    chan_height = chan_layout["h"]  # .1

    for axi, chan in enumerate(chans):

        # axis
        ax = axs[axi]
        ax.patch.set_alpha(0.0)  # see through
        ax.set_xlim(tmin, tmax)


        # lower left corner for this channel
        x0, y0 = chan_layout["chanlocs"][chan]

        # locate this channel
        bbox = mpl.transforms.Bbox([[x0, y0], [x0 + chan_width, y0 +␣
 ↪chan_height]])
        ax.set_position(bbox)
        # ERP waveforms, line styles from the style sheet
        for stim, erp in p3_erp.query("stimulus in @plot_stim").
↪groupby(["stimulus"]):

            # all axes get timeline, vertical cal bar
            ax.axhline(0, color='lightgray')
            ax.plot(
                [0, 0],
                [cal_bar_min, cal_bar_max],
                **cal_bar_kws
            )

            # --------------------------------------
            # special handling for cal and timeline
            if chan == "cal":
                ax.spines["bottom"].set_position(("data", 0))
                ax.set_xticks(timeline_ticks)
                ax.set_xticklabels(timeline_ticklabels)
                ax.tick_params(bottom=True, labelbottom=True)
                ax.plot(
                    [cal_bar_time, cal_tick_width],
                    [cal_bar_max, cal_bar_max],
                    **cal_bar_kws
                )
                ax.text(**cal_bar_label)
                continue

            # --------------------------
            # ERP label and traces
            ax.text(s=chan, **chan_label)
            erp = erp.reset_index()
            time = erp.time_ms.unique()
            ax.plot(time, erp[chan], label=stim)

            # Example: highlight P300 effect
            if stim == 'target':
                # pick one condition, fill to the other
                y2 = p3_erp.query("stimulus=='standard'")[chan]
                when = (time >= 250) & (time < 400) # highlight interval
                ax.fill_between(
                    time,
                    y1=erp[chan],
                    y2=y2,
                    where=when,
                    color="magenta",
                    alpha=.3
                )
        # set the title on the way out, ax doesn't matter, position is in fig␣
↪coords.
        ax.text(x=.45, y=.85, s="P300 ERPs", size=24, transform=fig.transFigure)
```



[ ]:

```
```

8 Compute mean P300 ERP

[17]:

```
# compute mean amplitude 250 - 450 ms, standards, targets, and␣
↪difference=target - standard
p300_amp = (
    p3_erp.query(
        "stimulus != 'cal' and time_ms >= 250 and time_ms < 450"
    ).groupby("stimulus")[EEG_COLUMNS]
    .mean()#
    .T
    .unstack()
    .to_frame()
)
```

```
p300_amp.columns = ["amplitude"]

# compute the P300 effect: target - standard
p300_amp_diff = p300_amp.unstack(0).apply(lambda row : row[1] - row[0], axis=1).
 ↪to_frame()
p300_amp_diff.columns = ["amplitude"]
p300_amp_diff["stimulus"] = "difference"
p300_amp_diff = p300_amp_diff.reset_index().set_index(["stimulus", "channel"])

p300_amp = pd.concat([p300_amp, p300_amp_diff])
p300_amp
```

[17]:

```
                     amplitude
stimulus   channel
standard   MiPf      -2.322935
           LLPf      -1.924552
           LLFr      -0.239627
           LLTe       0.564894
           LLOc       0.213780
…                         …
difference LMCe       3.214108
           MiPa       5.914676
           RMCe       2.656957
           RMFr       1.931009
           MiCe       3.280381

[78 rows x 1 columns]
```

8.1 Merge P300 mean amplitude with electrode locations

[18]:

```
p300_amp_locs = (
    p300_amp.reset_index("stimulus")
    .merge(
        SPH_CART_LOCS[["channel", "x_lambert", "y_lambert"]],
        on="channel"
    )
)
p300_amp_locs
```

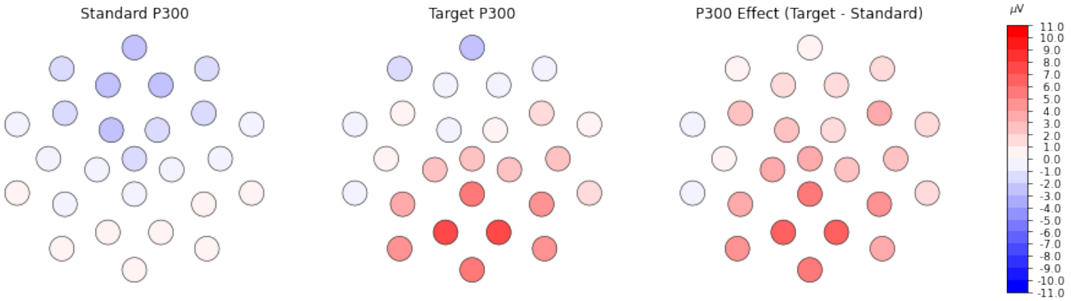
[18]:

```
   channel    stimulus  amplitude     x_lambert  y_lambert
0     MiPf    standard  -2.322935  6.123234e-17   1.000000
1     MiPf      target  -2.309737  6.123234e-17   1.000000
2     MiPf  difference   0.013199  6.123234e-17   1.000000
3     LLPf    standard  -1.924552 -5.877853e-01   0.809017
4     LLPf      target  -1.445431 -5.877853e-01   0.809017
..     …          …          …           …          …
73    RMFr      target   0.246611  1.869914e-01   0.257372
```

```
74    RMFr  difference   1.931009  1.869914e-01   0.257372
75    MiCe    standard  -1.254791  0.000000e+00   0.000000
76    MiCe      target   2.025590  0.000000e+00   0.000000
77    MiCe  difference   3.280381  0.000000e+00   0.000000

[78 rows x 5 columns]
```

[19]:

```
head_plot_style = {
    "axes.xmargin": 0.1,
    "axes.ymargin": 0.1,
    "axes.spines.left": False,
    "axes.spines.bottom": False,
    "xtick.color": "none",
    "ytick.color": "none",
    "lines.markersize": 20

}


# set up the color mapping
lower, upper = -11, 11
n_shades = 10  # for each color

n_colors = (2 * n_shades) + 2
bounds = np.linspace(lower, upper, n_colors + 1)
bwr_norm = mpl.colors.BoundaryNorm(bounds, n_colors)

# get blue-white-red divergent colormap
bwr_cmap = mpl.cm.get_cmap('bwr', n_colors)




with plt.style.context([psych_sci_fig, head_plot_style]):
    fig, axs = plt.subplots(1, 3, figsize=(14, 4),)


    stimulus = ["standard", "target", "difference"]
    for axi, stim in enumerate(stimulus):
        data = p300_amp_locs.query("stimulus == @stim")
        ax = axs[axi]
        if stim == "difference":
            ax.set_title("P300 Effect (Target - Standard)")
        else:
            ax.set_title(f"{stim.capitalize()} P300")
        p = ax.scatter(
            data["x_lambert"],
            data["y_lambert"],
            c=data["amplitude"],
            marker="o",
            cmap = bwr_cmap,
            norm=bwr_norm,
            lw=.5,
            edgecolor='k'
        )
        ax.set_aspect(0.9)

    # colorbar
    axins = axs[-1].inset_axes([1.2, 0, .075, 1])
    cb = fig.colorbar(
        p,
        cax=axins,
        ticks=bounds,
    )
    cb.ax.tick_params(axis="y", color='k')
    cb.ax.set_yticklabels(bounds, color='k')
    #cb.ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter("{x:5.1f}"))
    cb.ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter("{x:5.1f}"))
    cb.ax.text(
        x=0.5,
        y=1.05,
        s=r"$\mu\mathrm{V}$",
        fontsize=9,
        transform=cb.ax.transAxes,
        ha="center"
    )
    fig.savefig("generated/p3_head_plot3.pdf", format="pdf",␣
↪bbox_inches="tight")
```



Source: `research_report.tex`

This is the LaTex for the main report.

```latex
% for PsychSci APA6 TeXLive 2020 use this with biber/biblatex + styel=apa6
% figure note is not supported, put it in the caption
\documentclass[helv,10pt,man,floatsintext]{apa6}  %% man <-> jou <-> doc
\usepackage{csquotes}
\usepackage[backend=biber,style=apa6]{biblatex}
\addbibresource{apa_ms.bib}

% if you like line numbers ...
\usepackage{lineno}
%\linenumbers

\usepackage[american]{babel}
% \usepackage[utf8x]{inputenc}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage{multirow}
\usepackage{multicol}
\usepackage{xcolor}


% for tracking changes
\usepackage[draft]{changes}
\definecolor{skyblue2}{rgb}{.203, .395, .640}
\definecolor{orange2}{rgb}{.957, .473, .000}
\definecolor{plum2}{rgb}{.457, .313, .480}
\definechangesauthor[name=TPU, color=skyblue2]{TPU}
\definechangesauthor[name=ABC, color=orange2]{ABC}
\definechangesauthor[name=XYZ, color=plum2]{XYZ}


% to include one or more pages of multipage pdfs
\usepackage{pdfpages}

% for cross-references back to the main doc
% use \zref{} and \zlabel{} instead of latex native \ref{} and \label{}
\usepackage[xr, user, titleref]{zref}
\zexternaldocument{apa_si}  % other .tex file to cross reference

% to help control location of figures and tables
% \usepackage{float}

% highlight computer source code
\definecolor{bgc}{rgb}{.96,.96,.96}
\usepackage{minted}
\setminted[latex]{
  xleftmargin=0.5in,
  xrightmargin=0.5in,
  style=bw,
  frame=none, % lines,
  bgcolor=bgc,
  fontsize=\footnotesize,
  linenos
}

% for clickable URL links in pdfs
\usepackage{hyperref}
\hypersetup{
    colorlinks=true,
    citecolor=blue,
    linkcolor=blue,
    filecolor=blue,
    urlcolor=blue,
}

% use this to prevent LaTeX errors when urls break across pages
%% \hypersetup{draft}

\title{Analyzing open science data in style: A reproducible repodicible research report report}

\shorttitle{Reproducibl reproducible research report}

\author{Thomas P. Urbach}
\leftheader{Urbach}

\affiliation{
  Cognitive Science Department \\
  University of California, San Diego \\
  \today
}

\abstract{When the culmination of research is a research report, the
  culmination of reproducible research must be a reproducible
  report. To accomplish this, three problems must be solved: 1) the
  results of the reproducible data analysis must be incorporated into
  the narrative text, tables, and figures of the document; 2) the
  document must comply with the byzantine typographical requirements
  of professional publication style guides and their idiosyncratic
  modifications by various publishers; 3) the different parts and
  pieces of the report (manuscript, supplementary information,
  figures, tables, captions) must be reproducible digital objects in
  whatever specific document and image file format is required by the
  online platforms for submission to the journal and production by the
  publisher.  This report describes and demonstrates a flexible and
  generalizable approach that combines freely available open source
  data analysis and document preparation software tools to solve these
  three problems. The report itself is reproducibly generated by the
  approach it describes and demonstrates with real-world examples for
  psychologists: the manuscript is formatted in American Psychological
  Association style and digital objects are generated as required for
  the online submission and production platforms used by
  {\it Proceedings of the National Academy of Sciences}. The source
  code is available and may be downloaded from the Open Science
  Foundation archive or cloned from the GitHub repository and may be
  freely modified and used for other purposes under the Creative
  Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC
  BY-NC-SA 4.0) license. This reproducible report, together with the
  source code that reproduced it, comprise a complete self-contained
  tutorial, demonstration, and template for general use.  }

\begin{document}
\maketitle

\section{Introduction}

For any research project, after all the work of experimental design,
implementation, and data acquisition are in place, and the data
analysis is complete, there still remains the task of preparing and
publishing the peer-reviewed research report with a clear and accurate
presentation of the results through the text, tables, and
figures. However, the ``research report'' is an abstraction; in
practice it takes various forms on its trajectory from the authors'
desks to dissemination as a journal article in print and online in
digital form(s). For the authors, there all the usual chores of
document preparation: Writing the narrative text with qualitative and
quantiative analysis results, creating high-resolution graphics for
figures, preparing tables of data and results, adding and deleting
citations and bibliographic references, embedding links to URLs, and
aligning cross-references to elements within or across documents,
e.g., to the separate online supplementary information. During
preparation and revision the report is in flux and must be editable
with changes to the text tracked across versions.  For pre-print
archives and (re-)submission to peer-reviewed journals the text and
graphics are composited into a usually un-editable but easily
transmissible and viewable digital snapshot, e.g., typically Portable
Document Format (PDF). Finally, for journal and book publishers, the
process is unwound and the report must be comprised of separate
editable text and ``camera ready'' high-resolution graphics suitable
for production in digital form for online viewing and print
form. Throughout these transformations for publication, the report
must also satisfy specific style requirements and for psychologists
this often means a variation of the 6\textsuperscript{th} Edition of
the Publication Manual of the American Psychological
Association~(\cite{APAStyle6th}). Or maybe the 7th Edition. In short,
as a research report evolves from inception to DOI, it must sometimes
change and other times freeze in various highly specific forms and
digital file formts as it passes through different hands with
different requirements.

When the goal of reproducible research is fully embraced, the
``research report'' must also be reproducible throughout these stages
of preparation, revision, submission, and production. This requires
solving three problems: 1) the results of the reproducible data
analysis must be incorporated into the narrative text, tables, and
figures of the document; 2) the document must comply with the
byzantine typographical requirements of professional publication style
guides and their idiosyncratic modifications by various publishers; 3)
the different parts and pieces of the report (manuscript,
supplementary information, figures, tables, captions) must be
reproducible digital objects in whatever specific document and image
file format is required by the vagaries of an online journal
```

```latex
submission platform and then subsequently by a different online
production platform. Solutions to each of these problems individually
abound, the challenge is to combine them reproducibly. For instance,
reproducible data analyses are becoming commonplace though the use of
scientific computing platforms and open source scripting languages
like Python and R encaspulated in virtual environments (conda,
virtualenv) and containers (Docker, singularity). However the
technology for solving the data analysis problem is decoupled from the
strict typesetting requirements of different publication styles. On
the other hand, mature document preparation software like Microsoft
Word and \LaTeX{} provide the fine-grained control of formatting
necessary to comply with idiosyncratic style guidelines. However,
typing or copy-pasting the results decouples the report from the
analysis.  The results of the analysis may be reproducible when the
analysis is revised by co-authors or reviewers, but the results do not
propagate to all the digital objects that comprise the parts and
pieces of the report for (re-)submission and production.

This self-reproducing tutorial describes and demonstrates one approach
to solving all three problems at once using mature freely available
open-source computer software, a working knowledge of
\LaTeX{}~(\cite{latexproject}), and no more knowledge of computer
programming than is already required to implement the reproducible
data analysis it reports. The tutorial includes a sample reproducible
data analysis pipeline with open-acces data but focuses mainly on the
reproducible report per se, i.e., solutions to the second and third
problem needed to bridge the gap between the end of the reproducible
data analysis and the DOI of the peer-reviewed publication in an
academic journal. In addition to programmatically combining the data
analysis results with the narrative text, tables, and figures of the
report, the complete \LaTeX{} source code listings in the
Supplementary Materials provide working examples of some features
generally useful for manuscript preparation: tracking changes across
revisions, preparing camera ready graphics, automating
cross-references within and between documents, formatting and masking
the citations and bibliography, generating Portable Document Files,
compositing documents and pieces of documents in text and PDF file
formats, and preparing an author's manuscript for distribution while a
published article is embargoed.  The Supplementary Information
provides instructions for installing the open source software required
to reproduce the data analysis and this report. The complete source
code for the data analysis and report generation is publicly available
and may be downloaded from the Open Science Foundation archive or
cloned from the GitHub repository under a Creative Commons CC BY 4.0
license~\cite{ccby4.0} and used as a template and freely modified for
other purposes with appropriate attribution.

\section{Method}

\begin{figure}[ht]
\caption{Generating a reproducible APA 6th style research report: 1)
  Executing the reproducible data analysis code generates the complete
  results which appear as-is in the Supplementary
  Information. Selected results to be reported in the manuscript are
  exported to separate files as minimally styled narrative text and
  tables, and PDF graphics. 2. The graphics exported by the analysis
  are converted to camera ready APA-style figure graphic PDFs for the
  manuscript. 3. The Supporting information \LaTeX{} file is typeset
  as a document PDF which includes the complete analysis source,
  results, graphics, and document source. 4. The \LaTeX{} manuscript
  is typeset as a docuent PDF which includes the results text
  generated by the data analysis, the camera ready PDF figures, and
  bibliography.}  \zlabel{ms:report_generation}
\includegraphics[width=.95\textwidth]{images/report_generation.png}

\end{figure}


This appropach to generating reproducible research reports requires
the four main components, outlined schematically in
Figure~\zref{ms:report_generation}. While the approach is flexible and
generalizable, the specific examples are selected for researchers in
Psychology and demonstrate how to satisfy all the requirements (except
word count) for submitting and publishing a research report in the
journal, {\it Psychological Science}. Accordingly the manuscript is
structured with a Cover Page, Abstract, Introduction, Method, Results,
and Discussion~\parencite{APSStructStyle} and formatted according to
the APA 6th edition style~\parencite{APAStyle6th}. The approach here
is readily adapted to the APA 7\textsuperscript{th} Edition with a
change of the document
class~({\href{https://www.overleaf.com/project/5f3053af0af0dc00016f191b}{apa7}})
and minor modifications to the text described in Supporting
Information. The approach generalizes to other publication styles for
which \LaTeX{} style files have been defined.  A convenienty inventory
is collected here:
\href{https://www.overleaf.com/latex/templates/tagged/academic-journal}{Overleaf.com
  Templates\textemdash Academic Journal}. Many styles are community
contributions, for instance,
\href{https://www.overleaf.com/latex/templates/tagged/arxiv}{arXiv,
  bioRxiv}.
%
A number of journals and publishers provide official styles, such as
%
\href{https://www.overleaf.com/latex/templates/tagged/npg}{Nature},
\href{https://www.overleaf.com/latex/templates/tagged/pnas}{Proceedings of the National Academy of Sciences},
\href{https://www.overleaf.com/latex/templates/tagged/elife-official}{eLife}
%
and publishers
%
\href{https://www.overleaf.com/latex/templates/tagged/cup-official}{Cambridge University Press},
\href{https://www.overleaf.com/latex/templates/oup-general-template/fqkhysbcbpwv}{Oxford University Press},
\href{https://www.overleaf.com/latex/templates/tagged/springer}{Springer}
%
including
%
\href{
  https://www.overleaf.com/latex/templates/a-demonstration-of-the-latex2e-class-file-for-sage-publications/jcdy
}{
  SAGE
},
```

```latex
%
the publisher of {\em Psychological Science}.
%
The Supporting Information for this report provides installation
instructions for the necessary software and complete source code
listings for the analyses, documents, and figures which are freely
available under the CC-BY-4.0 license and may serve as templates for
a range of research projects in the psychological sciences.


\subsection{Data analysis pipeline: \mintinline{bash}{apa_analysis.ipynb}}

For demonstration purposes, a toy reproducible data analysis pipeline
is implemented in a Jupyter notebook running a Python
kernel~(\cite{kluEtAl2016}).  The pipleline (down)loads and
transforms a sample EEG dataset~(\cite{Urbach2020z}), computes summary
measures, and generates figures and text output. The particulars are
incidental, the data may as well be response times and the analysis
could be implemented in R, MATLAB, or any language that can format
numerical values as strings, write string variables to a text file,
and export as PDF, EPS, PNG, JPEG (or a format programmatically
converatble to one of these). This PDF is used for vector graphics and
PNG for raster graphics in this report since these have proved
reliable and both support transparency; EPS and JPEG also work if
these are required by the publisher.


\subsection{Preparing camera ready figures with \LaTeX{} and Ti{\it k}Z}

Ideally, graphic images generated by an analysis pipeline will be in
final ``camera ready'' form but this is not always practical or
possible.  A figure may require annotations, e.g., math notation, not
supported by the figure generator and a multipanel figure may need to
combine images from different sources. To demonstrate how this may be
done programmatically for reproducibility, three of the ``rough'' plot
graphics generated by the analyis pipeline are reconfigured, annotated
and converted into two camera-ready APA-style manuscript figures
(Figure~\zref{ms:multipanel} and Figure~\zref{ms:tikzfig}) using \LaTeX{} and
the Ti{\it k}Z graphic library without additional software or manual
editing.


\subsection{Manuscript: \mintinline{bash}{apa_ms.tex}}

LaTeX{} is a form of markup language where
the document text is intermingled with short typesetting
instructions. For instance, {\it this phrase is typeset in italics},
and the instruction looks like this:
%
\mintinline{latex}{{\it this phrase is typeset in italics}}.
%
Mathematical symbols and more complex equations are very
well-supported and set in the same way, e.g., partial eta squared
($\eta_p^2$) is set like so: \mintinline{latex}{$\eta_p^2$}.  Other
instructions are more general. For instance, the manuscript document
```

```latex
begins with this,
\mintinline{latex}{\documentclass[man,helv,10pt,draftall,floatsintext]{apa6}},
that says to typeset the document as a manuscript, in Helvetica 10
point font with a draft watermark on all pages, formatted to the APA
6th Edition style except that tables and figures should be placed near
where they appear in the text (``floatsintext'') rather than collected
at the end. This style, including the deviation from the APA 6th table
and figure position, corresponds to the submission guidlines for
Psychological Science~\parencite{PsychSciSubmissions2020}.  Like all
\LaTeX{} files, the main manuscript file is a plain text document and
thus virus-free, portable, viewable, and editable with any text
editor, although one that supports LaTeX syntax highlighting
on-the-fly syntax error checking is strongly recommended.

\subsubsection{Supplementary Information: \mintinline{bash}{apa_si.tex}}

Supplementary Information is as much a part of the report as the
manuscript and must be likewise reproducible. For demonstration here,
the Supplementary Information is comprised of a separate \LaTeX{}
file.  It provides instructions for downloading this report from
public repositories and installing the software to reproduce it. It
also includes source code listings of the Makefile used to reproduce
portions or all of the analysis, source code and output of the entire
executed analysis Jupyter notebook and listings of all the \LaTeX{}
files used to generate the report, figures, and supporting
information, which includes the self-reflexive listing of the
Supporting Information listing itself.


\subsection{Reproducing the report: \mintinline{bash}{Makefile}}

The \mintinline{bash}{make} program is a widely used command line
utility for managing the execution of a interdependent computer code
in complex programming projects, where changes in one file may might
impact some but not all other files. Reproducible data analysis and
report generation is similar in that, e.g., generating the
camera-ready figure PDFs depends on the rough plots generated by the
analysis which in turn depends on executing the analysis. The make
utility provides a useful mechanism for expressing the
interdependencies and comparmentalizing the project as work
progresses, e.g., \mintinline{bash}{make analysis} or
\mintinline{bash}{make fig2} or \mintinline{bash}{make ms} while
\mintinline{bash}{make all} ensures that all the components execute in
the correct order to completely reproduce the analysis and generate
all the files and documents for the figures, manuscript, supporting
information. Here is a summary of the make file components for
generating this report, execution times are for a high performance
workstation.

\begin{description}

\item [\mintinline{bash}{make analysis} (45 s)] Reproduce the data analysis by
  executing all the computer code in the analysis notebook start to
  finish. This has four side effects:

```

```
\begin{enumerate}
  \item The data analysis computations are executed and the results captured
    as standard output and plots in the Jupyter notebook cells.
  \item Results to be included in the manuscript as narrative text and
    tables are embeded in text strings, minimally formatted to APA
    style with \LaTeX{}, and exported as separate text files (.tex).
  \item Plots to be included in the manuscript figures are exported as
    PDF graphics.
  \item After execution is complete, a snapshot of the complete
    notebook\textemdash text, computer code, and results captured in
    the output cells\textemdash is exported to a PDF file. The PDF is
    included in its entirety in the Supplementary Information.
\end{enumerate}

\item [\mintinline{bash}{make fig1} (1 s)] Run
  \mintinline{latex}{pdflatex fig1.tex} to convert two rough plot
  graphics as generated by the analysis pipleine into the camera-ready
  Figure~\zref{ms:multipanel} graphic shown in the manuscript.

\item [\mintinline{bash}{make fig2} (1 s)] Run \mintinline{latex}{pdflatex fig2.tex}
  to convert the rough plot graphic generated by the analysis
  pipleine into the camera-ready Figure~\zref{ms:tikzfig} graphic shown in the
  manuscript.

\item [\mintinline{bash}{make figs} (47 s)] Execute the analysis to generate the rough PDF graphic
  output files then make fig1 and fig2 as above.

\item [\mintinline{bash}{make ms} (9 s)] Run \mintinline{bash}{pdflatex apa_ms.tex}
  to generate the manuscript PDF.

\item [\mintinline{bash}{make si} (4 s)] Run \mintinline{bash}{pdflatex apa_si.tex}
   to generate the Supporting Information PDF.

\item [\mintinline{bash}{make all}] Run make figs to execute the
  analysis and generated camera ready figures then make ms and si
  enough times to update and the cross-references between the
  manuscript and supplementary information.

\end{description}


\label{sec:results}

\section{Results}

The results are this report and the Supplementary Information. Both
are reproducibly reproduced using freely available open source
software, a working knowledge of \LaTeX{} and no more computer
programming than the Python used for the data analysis.  A few points
merit further discussion.

\section{Discussion}

\subsection{Linking data and arbitrary text}

```

```latex
% This is the complete latex for this entire section
\input{generated/arbitrary_text.tex}


The listing below shows the minimally styled \LaTeX{} text generated
by the analysis pipeline. For illustration, it includes comments
(\%\%), narrative text with the data values filled in progrmmatically,
and \mintinline{latex}{{\it N }}, which italicizes the capital N
according to APA 6th style:

% this shows it as a source listting
\inputminted{latex}{generated/arbitrary_text.tex}


\subsection{Tables}

The ability to link data with arbitrary text is nowhere more valuable
than in preparing reproducible data tables styled to editorial
standards.  The primary challenges are the intricate requirements for
laying out headings and notes as illustrated by the following exerpts,
drawn from the 40 pages of APA Publication Manual 7th edition table
guidelines:

\begin{quote}
{\bf headings} Tables may include a variety of headings depending on
the nature and arrangement of the data. All tables should include
column headings, including a stub heading (heading for the leftmost
column). Some tables also include column spanners, decked heads, and
table spanners (see Section 7.12)

\ldots

{\bf notes:} Thee types of notes (general, specific, and probabiity)
appear below the table as needed to describe contents of the table
that cannot be understood from the table title or body alone \ldots
\end{quote}

\noindent
It is straightforward to reproducibly link table text to the analysis
data they tabulate. It is less straightforward, but still tractable to
do while also generating the three types of notes, four types of
headings and column spanners, and ``a border at the top and bottom of
the table, beneath column headings (including decked heads), and above
column spanners.''  (p. 205)

The tabular exhibit labeled Table~\zref{ms:table1} illustrates a
not-quite conforming tabular array of data. When the analysis runs,
the table is reproducibly generated as a \LaTeX{} .tex file with one
line of code \mintinline{python}{pandas.DataFrame.to_latex()}.
\footnote{
  For analyses scripted in R, the \mintinline{R}{xtable} library
  similarly generates \LaTeX{} format table from dataframes
  \url{https://cran.r-project.org/web/packages/xtable/index.html}.
}
The .tex file is imported into the manuscript the same way as the arbitrary
```

```latex
text file above.

\begin{table}[ht]
  \centering
  \caption{A non-APA Style data table and note generated
    as \LaTeX{} by calling \mintinline{python}{pandas.DataFrame.to_latex()}.} \zlabel{ms:table1}
  \begin{threeparttable}
    \input{generated/p3_table1.tex}
    \begin{tablenotes}[flushleft]
      Note: Python variables are conventionally lower case.
    \end{tablenotes}
  \end{threeparttable}
\end{table}

\noindent
This approach is simple and easy and well-suited for data tables
presented in supporting information where styling requirements are
typically less strict. When easily generated tables will not do, the
fall back is arbitrary text generation.  A few lines of Python code
and common string formatting methods suffice to generate the \LaTeX{}
required to format the table header, footer, notes and row data to APA
style. The following listing shows the programmatically generated
\LaTeX{}, the result is shown as Table~\zref{ms:table2}. The Python
source code to is Jupyter notebook in the~Supporting
Information.

\inputminted{latex}{generated/p3_table2.tex}


\begin{table}[ht]
  \centering
  \caption{An APA style data table and note generated as \LaTeX{} with
    a few lines of pure Python.}
  \zlabel{ms:table2}

  \centering
  \begin{threeparttable}
    \input{generated/p3_table2.tex}
    \begin{tablenotes}[para, flushleft]
      Note: APA Style capitalization.
    \end{tablenotes}
  \end{threeparttable}
\end{table}



\subsection{Figures}

Graphics figures in PNG, PDF, and JPEG can be included in a \LaTeX{}
document with the \mintinline{latex}{command}. Of these PDF seems to
be the most reliable for vector graphics (plots, line drawings,
charts, plots) and PNG for raster graphics.  Including figures is
straightforward, creating figures for a data analysis reproducibly is
another matter. In some case it may be possible to generate
camera-ready graphics from the data anlysis pipeline itself. Although
```

```
this takes some effort to fine tune at the outset when Reviewer 2
insists on some mid-stream revision that requires re-running the
analysis, the change propagates all the way through to the final
figures included in the report. However this is not always
possible. One recourse is to use an interactive vector graphics
manipulation programs like Inkscape to import the graphic and edit to
style but, like manually typing results into a data table, the results
may change but the representation of the results does not.

Since hand editing figures amounts to using a mouse to select a
sequence of drawing commands, it can be done programmatically with the
right vector graphics manipulation tools. In the LaTeX{} ecosystem, a
particularly powerful package for this is
\href{https://en.wikipedia.org/wiki/PGF/TikZ}{Ti{\it k}Z} and the
learning curve is correspondingly steep. However, for simple tasks
like laying out and annotating the figures, it is reasonably
straightforward. The tikz figure is a canvas with coordinates.
Graphics can be placed and aligned, and drawing elements like lines,
arrows, and shading added. Figure~\zref{ms:multipanel} and
Figure~\zref{ms:tikzfig} are worked examples of this approach and show
how to convert graphics generated by the data analysis into ``camera
ready'' figures to APAstyle specifications saved as separate PDF files
for upload to the publisher.  Figure~\zref{ms:multipanel} is a simple
example that lays out two graphics side by side and
Figure~\zref{ms:tikzfig} illustrates a more elaborate example that
selects portions of a single graphic, rearranges and resizes them and
adds additional graphic and text annotations. The \LaTeX{} and Ti{\it
  k}Z code for both figures is listed in the Supplemental Information.

\begin{figure}[ht]
  \caption{
    A complete multi-panel color figure generated
    reproducibly from the data to Psychological Science figure
    specifications. The figure is generated using the matplotlib package in
    Jupyter Notebook running a Python kernel. The code illustrates
    some useful Python idioms and matplotlib functionality including
    style sheets, the style context manager, how to lay out panels,
    add labels including with mathematical symbols, and export the figure as
     as a PDF graphic.
  }

  \zlabel{ms:multipanel}
  \centering
  \includegraphics[width=.95\textwidth]{apa_fig1.pdf}

\end{figure}



\begin{figure}[ht]
  \caption{Reproducible figure layout and annotation. Panel a shows
    the pdf as generated by the analysis script and a stock montage
    image. Panel b shows the ``camera ready'' figure output generated
    by post-processing the generated graphic with \LaTeX{} and the
    Ti{\it k}Z drawing library as part of the documentation generation
    pipeline. The data are the same as in Figure~\zref{ms:multipanel}
  }\zlabel{ms:tikzfig} \includegraphics[width=\textwidth]{apa_fig2.pdf}
\end{figure}


%% % Figure 2
%% \begin{figure*}[ht]
%% \centering
%% \includegraphics[width=0.9\textwidth]{fig2.pdf}

%% \caption{
%%   Simple resizing and clipping can be done in LaTeX{} by tuning the
%%   options for includegraphics. This is the same .pdf plot as
%%   in Figure~\ref{fig_1} resized with to 90\% of the width of the text.
%% }\zlabel{lp_filt}
%% \end{figure*}





\subsection{Citations, masked citations, and references}

In \LaTeX{} citations in the text are indicated by typing commands
like \mintinline{latex}{\cite{}} with the author, name, year,
parenthesis information for APA style are determined when the document
is typeset. Typing the citation commands amounts to
``cite-while-you-write''. LaTeX automatically generates a bibliography
in the APA style from the corresponding .bib file (bibliography
database) according to the citations that appear in the text.  There
lots of options for citation format, see the
\mintinline{latex}{biblatex} and \mintinline{latex}{apa6} docs for
reference. For instance, the \mintinline{latex}{\parencite} command
generates a formatted citation in parentheses
\parencite{Lamport1986}. The cite command generates one without
parentheses, as in~\cite{Lamport1986}. When manuscript submission
requires citation masking for blind review, the masked variants of the
citation commands, e.g., \mintinline{latex}{\maskparencite} can be
used: \maskparencite{Lamport1986}. The masked citations are indictaed
in bold when the manuscript is typeset normally and replaced with {\it
  (1 citation removed for masked review)} when typeset with the mask
option.

The .bib file is a text file with bibliography entries that have the
usual author, title, data, publisher, fields, and a great many others,
in a specific format.  There are several options for where to get the
.bib file. Scientific literature search engines, publisher websites
routinely export citations in .bib format which can be copy-pasted
instead of tediously typed. If a reference manager is already being
used, it may also be able to export its references to .bib format. And
there are a number of reference managers that are designed from the
ground up to use .bib. As of this writing, the open-source JabRef
seems to have emerged as pick of the litter, being fully featured
enough to support general use and working across platforms. BibDesk
is another option but only runs on OSX. If other options fail, the
```

```latex
entry can be typed.


\subsection{Cross references}

To cross-reference between elements like tables, figures, and sections
\LaTeX{} links them via \mintinline{latex}{\label}
\mintinline{latex}{\ref} pairs. However a more general approach is to
use the \href{https://ctan.org/pkg/zref}{zref package} which links
elements with \mintinline{latex}{\zlabel} \mintinline{latex}{\zref}
pairs that work across documents which the built-in version does
not. This is particularly useful for cross-referencing information in
the Supplementary Information from the main manuscript and vice
version. When there are two or more docs and a series of figures
and/or tables and/or document sections in each and have to add or
delete another, it is mighty handy to have the references everywhere
in both documents automagically update the numbering and page
locations. Here is an example cross reference a section in the
Supporting Information, if that section title changes so does this
reference:~\ztitleref{si:analysis_nb}.  To cross-reference between
.tex documents, both documents must be compiled and this may not be
possible in all online submission systems, even those that accept .tex
format documents. For instance, the PNAS online submission system
accepts latex for manuscripts but requires .pdf for supporting
information and does not accept uploads of the auxiliary files
required by zrefs in the main manuscript which means the submission
system cannot correctly compile .tex manuscripts with zrefs.

\subsection{Tracking changes}

Revisions to a document marked and tracked in a document in the same
way as other types of formatting. With the
\mintinline{latex}{\changes} package, authors indicate the type of
change or markup, e.g., add, delete, replace, highlight, and then
bracket the relevant text, like so:
\mintinline{latex}{\added[id=TPU]{Here is some new text}}.  When the
document is type typeset in draft mode:
(\mintinline{latex}{\usepackage[draft]{changes}}), the changes are
highlighted and tagged by author. For instance \added[id=TPU]{This
  text is marked by TPU as added} and \deleted[id=ABC]{this text is
  marked by ABC as deleted}. Furthermore, \highlight[id=TPU,
  comment={is this helpful?}]{this text is marked by TPU as
  highlighted} and \replaced[id=XYZ]{this is XYZ's replacement
  text}{this text was replaced}.

In draft mode, a list of the changes can be generated by inserting the
\mintinline{latex}{\listofchanges} command, typically at the beginning
or end, though shown here at the end of this section for illustration.
Collaborators can review the changes in the pdf and add make further
revisions to the .tex document. When the document is typeset for the
final version (\mintinline{latex}{\usepackage[final]{changes}}), the
changes are applied and remaining comments, markup, and annotations
stripped, similar to accepting tracked changes in a WSYSIG
document. The draft and final versions may both be useful when
resubmission of a document following revision requires both ``clean''
```

```latex
version with the changes made and a draft version marked up to
indicate where the revisions were made. For cases where there are two
versions of a .tex document and the changes are not explicitly marked
up inline, the command line utility program
\mintinline{bash}{latexdiff} can be used to automatically generate a
single pdf with the differences between the versions indicated as in
changes. Both of these features are best suited to marking revisions
and changes in the text of relative similar documents and are not
well-suited to track massive restructuring or revisions to figures and
tables. Here is the list of changes explicitly marked up in the
previous paragraph.

\listofchanges


\subsection{Compositing documents: files and file formats}

Various files and formats are required go submit and publish a
research report. These may include a main editable manuscript
(document), supporting information (document, data), figures (vector
and raster image graphics files), tables, and bibilographic
info. Journals and publishers have divergent interests (readability
for evaulation in review vs. production for print and digitial
formats) and (thus) different requirements for document
preparation. This is further complicated by open-access policies that
require authors to deposit a final pre-publication manuscript if the
publisher won't (but most do, eventually).  For submission to
Psychological Science for instance, the file formats are \LaTeX (.tex)
for editable text and Portable Document Format (.pdf) for graphics, a
vector format that is scalable without loss of resolution. To submit
the report to the journal for review the .tex and .pdf graphic files
composited into a single .pdf file and all files
uploaded~\cite{PsychSciSubmissions2020, PsychSciFigs2013}. Whereas the
journal submission portal requires the a single composited document
with text and graphics all in one, the publisher's portal requires the
separate editable text and graphics files, i.e., the .tex and graphics
.pdfs.

Working with \LaTeX{} simplifies some aspects of this by allowing
files in different digital formats to be included in documents in
various ways. As illustrated by linking results and abitrary text for
narrative descriptions and tables, separate files of \LaTeX{} can be
inserted directly into the document as if typed in place. This allows
the tables to be reproducibly prepared as separate files (as required
by some publishers) and also incorporated in exactly the same form in
the body of the manuscript (as also required by these publishers). The
same holds for the camera ready graphics for Figure~\zref{ms:multipanel} and
\zref{ms:tikzfig} which are also separate files included as-is in the
mansucript. Additionally the \mintinline{latex}{\includepdf} package,
allows all or selected pages of a multi-page PDF documents to be
included in a \LaTeX{} as demonstrated in by the Supplementary
Informatinon that includes the entire PDF of the fully executed data
analysis Jupyter Notebook.  Finally, the \mintinline{latex}{\minted}
package used extensively throughout this document will import the
contents of separte files into the \LaTeX{} document and also
```

```latex
highlight the code according to the syntax of the specfic language,
e.g., Python, R, \LaTeX{} which is of great value in documenting
scripted reproducible research pipelines. The Supplemental Information
demonstrates this by importing and highlighting all the \LaTeX{} files
used in the producition and reproduction of this tutorial report.

\subsection{Author manuscripts}

Whereas journals may require submission as a double spaced manuscript,
the published articles typeset single space in two columns with
figures and tables where they belong are generally easier to read.
Switching the \mintinline{latex}{documentclass} option from man
(manuscript) to jou (journal) typesets the document in a
more-nearly-journal-like format (Figure~\zref{ms:apa67_jou}), which
may be useful for distributing working drafts or post-publication
author manuscripts during a publisher's embargo period.

\begin{figure}
\caption{Example of typesetting this document with the jou option}
\zlabel{ms:apa67_jou}
\centering
\includegraphics[width=.65\textwidth]{images/apa67_jou.png}
\end{figure}


\section{Conclusion}

There are many ways to prepare a research report but far fewer to do
so reproducibly while at the same time satisfying the requirements of
publication styles and online platforms for submission and production.
This report illustrates one approach that does so while aligning well
with best practices in open science data analysis. Once a
reproducible analysis in place, the additional cost of the
reproducible report is acquiring a working knowledge of \LaTeX{} and
if necessary Ti{\it}Z.

\newpage
\printbibliography

\end{document}
```

Source: `author_si.tex`

This is the LaTex for this Supporting Information, i.e., it is typesetting itself.

```latex
\documentclass[letter,doc,natbib,11pt]{apa7}  %% man <-> jou <-> doc

\usepackage[american]{babel}
\usepackage[utf8x]{inputenc}
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage[colorinlistoftodos]{todonotes}
\usepackage{xcolor}


% use this for URLs
\usepackage{hyperref}
\hypersetup{
    colorlinks=true,
    citecolor=blue,
    linkcolor=blue,
    filecolor=blue,
    urlcolor=blue,
}

% for cross references back to the main doc
% use \zref{} and \zlabel{} instead of latex native \ref{} and \label{}
\usepackage[xr, user, titleref]{zref}
\zexternaldocument{apa_ms}  % other .tex file to cross reference

% use this to include text files verbatim (not shown)
\usepackage{verbatim}

%use this package for highlighted source, e.g., research_report.tex
\usepackage{minted}
\setminted[latex]{
  frame=lines,
  bgcolor=bgc,
  fontsize=\footnotesize,
  linenos
}

% use this to include multipage pdf docs, e.g., conveted jupyter notebook, other docs
\usepackage{pdfpages}


\title{Supporting Information: Analyzing open science data in style: A reproducible repoducible research report
\shorttitle{Supporting Information: Reproducible reports with LaTeX{}}
\author{Thomas P. Urbach}
\affiliation{Kutas Lab \\ Cognitive Science Department \\ University of California, San Diego}

\begin{document}
\maketitle

\tableofcontents

```

```latex
\section{Summary}

Additional information can go here and be formated to APA 6th
guidelines or something else. The supporting information and main
document can use the same research\_report.bib file so the references
match. With {\tt \textbackslash usepackage[xr,user,titleref]\{zref\}},
you can cross-reference back and forth between documents \ldots the
main report and this SI. For instance here is a reference to
Figure~\zref{ms:multipanel} in the main document. The reference is via the
label, i.e., {\tt \textbackslash zlabel\{ms:multipanel\}} so if the figure
is moved to a different page or its number changes because of
additions or deletions, this reference by number will update
automatically. The following sections show the source files that
generated the plots, figures, manuscript and si .pdfs.


APA 7th docs \url{http://ctan.math.washington.edu/tex-archive/macros/latex/contrib/apa7/apa7.pdf}
APA 6th docs \url{http://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/biblatex-contrib/biblatex-apa




% For APA 7th TeXLive 2020 use change the figure captions from
%
%   \caption{First sentence. Rest of the caption.}
%
% to
%
%   \caption{First sentence.} \figurenote{Rest of caption.}
%
% and this preamble
%
% \documentclass[man,biblatex,10pt]{apa7}
% \usepackage{csquotes}
% \DeclareLanguageMapping{american}{american-apa}
% \usepackage[backend=biber,style=apa]{biblatex}



\section{System setup}

\subsection{Installing conda environments}

If you already use conda environments in a recent linux operating
system, you can install a minimal conda environment to run the
notebooks like so and follow the prompt (or omit -y to the end of the
command to install the packages without prompting).

\begin{minted}{bash}
conda create -n apa67_report pandas pyarrow matplotlib jupyter firefox -y
$ activate apa67_report
$ jupyter notebooks
\end{minted}

If you are not yet set up to use conda environments, you can follow
```

```
the instructions to download and install a minimal conda installer,
miniconda3
(\href{https://docs.conda.io/en/latest/miniconda.html}). This provides
just enough infrastructure to create a conda environemnt and install
packages as shown in the example above. If you want to create conda
environments and install packages faster, then install the `mamba`
conda package (\href{https://mamba.readthedocs.io/en/latest/}).

If you are not yet set up to use conda environments and don't want to
be then you are on your own. You can run pipeline\_1.ipynb if you have
numpy, pandas, matplotlib and jupyter.  You need the spudtr package to
run pipeline\_2.ipynb. Older versions are available via pip install,
but there is no assurance it is compatible with the versions of
packages you already have installed.

\subsection{Installing \LaTeX}

\subsubsection{Linux Installation via network}

You do not need to be root or admin to install TeX Live over the
networks and best practices are to install your copy in your
directory. That way you control the version and packages you
use. First read through the quick installation instructions
\href{https://www.tug.org/texlive/quickinstall.html}{here}. Then,
(summarizing from
\url{https://www.tug.org/texlive/acquire-netinstall.html}):

\begin{enumerate}

\item Download \url{install-tl-unx.tar.gz} to some scratch/working
  directory, unpack the archive, change to the new directory it
  made, i.e., \mbox{install\textendash tl\textendash YEARMONTHDAY} for
  whatever version, and run the installer.

  \begin{minted}{bash}
    $ tar -xf install-tl-unx.tar.gz
    $ cd install-tl-20200814
    $ perl install-tl
  \end{minted}

  Follow the prompts, make sure you are happy with and have write
  permissions in the default installation directory, and press ``i''
  to install.

\item Update your ~/.bashrc file with the path to the new TeX Live
  installation.

  \begin{minted}{bash}
    PATH=/home/turbach/texlive/2020/bin/x86_64-linux:$PATH
    INFOPATH=/home/turbach/2020/texmf-dist/doc/info:$INFOPATH
    MANPATH=/home/turbach/2020/texmf-dist/doc/man:$MANPATH
  \end{minted}

\end{enumerate}

```

```
That's it, you have a complete functioning installation of \LaTeX{}
with the latest packages, TeX Live 2020 as of this writing.

The installation probably has everything you need including the apa6
and apa7 styles used for this report.

If there is a new package or update you want and you want to manage
the TeX packages with the TeX Live GUI you also need to install
perl/tk. There is a conda package for this, you can install into any
compatible conda env.

  \begin{minted}{bash}
    $ conda activate some_general_purpose_env
    $ conda install perl-tk -c BioBuilds -y
  \end{minted}


\subsubsection{OSX Installation}

See instructions for MacTeX here: \url{https://www.tug.org/mactex/}

\subsubsection{Windows}


See Quick Install insructions
\href{https://www.tug.org/texlive/quickinstall.html}{here}

and Windows installer instructions
\href{https://www.tug.org/texlive/acquire-netinstall.html}{here}.


% The next two sections show the (converted-to-pdf)
% jupyter notebook for generating the figures, lateLaTeX{} .tex file for the main report and the jupyter
% notebook that generates the pdf plots for the filter figures.

%  ------------------------------------------------------------
% Jupyter notebook source
\newpage
\normalsize
\section{Source: author\_analysis.ipynb}\zlabel{si:analysis_nb}

The pdf of the notebook is generated by {\tt jupyter convert ... --to pdf}. The
LaTeX{} package {\tt pdfpages} is used to slurp it into the SI pdf.

\includepdf[pages={1-}]{apa_analysis}


% ------------------------------------------------------------
% research report LaTeX
\newpage
\section{Source: {\tt research\_report.tex}}\zlabel{apa_ms_tex}
This is the LaTex{} for the main report.

\definecolor{bgc}{rgb}{1.0,.96,1.0}
\inputminted{latex}{apa_ms.tex}
```

```latex


% ------------------------------------------------------------
% supporting information LaTeX
\newpage
\section{Source: {\tt author\_si.tex}}\zlabel{apa_si_tex}

This is the LaTex{} for this Supporting Information, i.e., it is
typesetting itself.

\inputminted{latex}{apa_si.tex}


% ------------------------------------------------------------
% Figure 1 LaTeX
\newpage
\section{Source: {\tt fig1.tex}}\zlabel{si:fig1_src}
This is basic LaTex{} template for a free-standing .tex file that pdflatex can turn
into a .pdf graphic for import or upload. It is just the graphic, no caption or numbering.

\inputminted{latex}{apa_fig1.tex}


% ------------------------------------------------------------
% Figure 2 LaTeX
\newpage
\section{Source: {\tt fig3.tex}}\zlabel{si:fig3_src}

This is the LaTex{} for the multipanel TikZ figure with fancy layout
and annotation stuff. Again, just for the pdf graphic, no caption.

\inputminted{latex}{apa_fig2.tex}

% ------------------------------------------------------------
% Makefile
\newpage
\section{Source: \mintinline{makefile}{Makefile}}\zlabel{si:makefile_src}
This is the Makefile used to build/rebuild the ms, si, figs indidually
and all the documents in one fell-swoop.

\inputminted{makefile}{Makefile}


% ------------------------------------------------------------
% bib
\newpage
\section{Source: {\tt research\_report.bib}}\zlabel{si:bib_src}
This is the .bib for citations and references, shared by the ms and this SI.

\inputminted{bibtex}{apa_ms.bib}

% Supporting Information References (if any)
\bibliography{research_report}

\end{document}
```

Source: fig1.tex

This is basic LaTex template for a free-standing .tex file that pdflatex can turn into a .pdf graphic for import or upload. It is just the graphic, no caption or numbering.

```
%% use this to make a free-standing pdf graphc instead of a paginated latex doc

% bare bones 2-panel figure, no annotations
\documentclass[border=0in]{standalone}
\usepackage{graphicx}
\begin{document}
\includegraphics[width=.45\textwidth]{generated/p3_midline_plot1.pdf}
\includegraphics[width=.45\textwidth]{generated/p3_midline_plot2.pdf}
\end{document}
```

Source: `fig3.tex`

This is the LaTex for the multipanel TikZ figure with fancy layout and annotation stuff. Again, just for the pdf graphic, no caption.

```
%% use this to make a free-standing pdf graphc instead of a paginated latex doc
\documentclass[border=0in]{standalone}

% dejavu san serif matches matplotlib default
\usepackage{dejavu}
\renewcommand*\familydefault{\sfdefault} % set base font to sans serif
\usepackage[T1]{fontenc}
\usepackage{amsmath}  % math symbols
%% \usepackage{pbox}
\usepackage{tikz}

\usetikzlibrary{arrows,shapes,backgrounds,shadows,fit,positioning,scopes, calc}

%% whitesmoke background
\definecolor{whitesmoke}{rgb}{.9607843137, .9607843137, .9607843137}

%% style general layout
%\tikzstyle{background rectangle} = [fill=whitesmoke]
\tikzstyle{background rectangle} = [fill=white]
\tikzstyle{every node} = [outer sep=0pt, inner sep=3pt]

%% define the plot label spec: #1=tag, #2=location, #3=text
\def\plabel[#1]#2#3{
  \node [left, scale=1.0] (#1) at (#2.north west) {#3};
}

\begin{document}
\begin{tikzpicture}[
    >=stealth,  %% shape of the annotation arrows
    show background rectangle,
    %% inner frame sep=2mm  % sep = bleed or 0 for tight background
]

  % ------------------------------------------------------------
  % Panel a  figure as generated
  \coordinate (axy) at (0, 0);
  \plabel[label-a]{axy}{a};
  \node [
    anchor=north west,
    rectangle,
    fill=whitesmoke
  ] (p3-head-pdf) at (label-a.north east){
    \includegraphics[height=1in]{generated/p3_head_plot3.pdf}
  };

  \node[
    xshift=.75in,
    rectangle,
    fill=whitesmoke
  ] (montage) at (p3-head-pdf.east){
    \includegraphics[height=1in]{images/TopHead.pdf}
  };



  % ------------------------------------------------------------
  % Panel b TikZ layout and annotations

  % crop top and bottom of generated pdf
  \newcommand{\tbtrim}{0.4in}
  \newcommand{\mathfontscale}{2}

  \coordinate [yshift=-0.5in] (bxy) at (p3-head-pdf.south west);
  \plabel[label-b]{bxy}{b};

  % P300 effect in a shadow box

  % frame + drop shadow
  \node (b-effect-box) [
    anchor=north west,
    draw=black!40,
    fill=white,
    rounded corners=4pt,
    drop shadow,
    minimum height=1.5in,
    minimum width=1.9in
  ]
  at (label-b.south east) {};

  % electrode scatter + colorbar
  \node (p3-effect) at (b-effect-box) {
    \includegraphics[
      clip,
      trim={8.1in, \tbtrim, 1.35in, \tbtrim},
      height=1.125in
    ]{generated/p3_head_plot3.pdf}
    \includegraphics[
      clip,
      trim={11.75in, 0, 0, 0},
      height=1.125in
    ]{generated/p3_head_plot3.pdf}
  };


  % montage head
  \node[
    xshift=.2in,
    yshift=-.2in,
    opacity=.25
  ] (montage) at (b-effect-box.north west){
    \includegraphics[height=.25in]{images/TopHead.pdf}
  };

  % Title
  \node [
    anchor=south,
    scale=.66
  ] (effect-label) at (b-effect-box.north) {
    P300 ERP effect (Target $-$ Standard)
  };

  % annotation text
  \node [
    anchor=north west,
    xshift=0.025in,
    yshift=0.05in,
    scale=.5
  ] (post-pointer) at (p3-effect.south) {
    posterior maximum
  };

  % annotation arrow
  \coordinate [xshift=-0.03in, yshift=-0.425in] (RDPa) at (p3-effect);
  \draw [->] (post-pointer.north west) -- (RDPa);


  % equals (=)
  \node [scale=\mathfontscale, anchor=west] (text-equals) at (b-effect-box.east){$=$};

  % P300 target
  \node [anchor=west] (b-target) at (text-equals.east){
    \includegraphics[
      clip,
      trim={4.25in, \tbtrim, 5.25in, \tbtrim},
      height=1in
    ]{generated/p3_head_plot3.pdf}


  };
  \node [anchor=south, scale=.66] (target-label) at (b-target.north) {
    Target
  };


  % minus (-)
  \node [scale=\mathfontscale, anchor=west] (text-minus) at (b-target.east){$-$};

  % P300 standard
  \node [anchor=west] (b-standard) at (text-minus.east){
    \includegraphics[
      clip,
      trim={0.5in, \tbtrim, 9.0in, \tbtrim},
      height=1in
    ]{generated/p3_head_plot3.pdf}
  };
  \node [anchor=south, scale=.66] (standard-label) at (b-standard.north) {
    Standard
  };

\end{tikzpicture}
```

```latex
\end{document}
```

Source: Makefile

This is the Makefile used to build/rebuild the ms, si, figs indidually and all the documents in one fell-swoop.

```
# TODO: for reproducibility check we are running in the right conda environment

# where to find the files
HOME_DIR = /home/turbach/TPU_Projects/demos/latex/apa_6th_example

# jupyter notebook figure generator ... slurp the actual research data
# and generate the pdf plots that will appear in the ms and si Figures
JUPYTER_CONVERT = jupyter nbconvert --ExecutePreprocessor.timeout=None --execute

export_env:
        conda list --explicit > environment.txt

# the minted syntax highlighting package insists on -shell-escape
ms:
        pdflatex -shell-escape apa_ms
        biber apa_ms
        pdflatex -shell-escape apa_ms
        pdflatex -shell-escape apa_ms


si:
        pdflatex -shell-escape apa_si
        biber apa_ms
        pdflatex -shell-escape apa_si
        pdflatex -shell-escape apa_si

bib:
        pdflatex -shell-escape apa_ms
        pdflatex -shell-escape apa_si
        biber apa_ms

# for long-running jobs use --ExecutePreprocessor.timeout=None
analysis: export_env
        jupyter nbconvert --execute --to pdf ./apa_analysis.ipynb

fig1:
        pdflatex apa_fig1.tex

fig2:
        pdflatex apa_fig2.tex
```

```
figs: analysis fig1 fig2


# remove intermediate latex files aux, log and stash backup files
# move
clean_aux:
        latexmk -c

# multiple passes to get the zref cross-document cross references right
all: figs bib si ms si ms si


# build everything then wipe the intermediate stuff
for_upload: all clean_aux
```

Source: `research_report.bib`

This is the .bib for citations and references, shared by the ms and this SI.

```
@book{APAStyle6th,
  author =         {{American Psychological Association}},
  title =         {Publication Manual of the American Psychological
                  Association},
  edition =         {6th},
  publisher =         {American Psychological Association},
  pages =         272,
  year =         2010,
  type =         {Book}
}

@misc{APSStructStyle,
  title =         {Manuscript Structure, Style, and Content Guidelines},
  publisher =         {Association for Psychological Science},
  url =
                  {https://www.psychologicalscience.org/publications/ms-structure-guideline
  urldate =         {2020-08-11},
  type =         {Web Page}
}

@ARTICLE{Lamport1986,
  author =         {L[eslie] A. Lamport},
  title =         {The Gnats and Gnus Document Preparation System},
  journal =         {G-Animal's Journal},
  year =         1986,
  volume =         41,
  number =         7,
  pages =         "73+",
  month =         jul,
}

@misc{PsychSciFigs2013,
  title =         {{APS} Figure Format and Style Guidelines},
  publisher =         {Association for Psychological Science},
  month =         10,
  url =
                  {https://www.psychologicalscience.org/publications/aps-figure-format-styl
  urldate =         {2020-08-11},
  year =         2013,
  type =         {Web Page}
}
```

```
@article{PsychSciSubmissions2020,
  author =         {},
  title =         {Psychological Science 2020 Submission Guidelines},
  publisher =         {Association for Psychological Science},
  urldate =         {2020-08-11},
  url-modified = {2020-07-13},
  url =
                  {https://www.psychologicalscience.org/publications/psychological_science/
  year =         2020,
  type =         {Web Page}
}

@misc{Urbach2020z,
  author =         {Urbach, T.~P.},
  title =         {eeg-workshops/mkpy\_data\_examples/data [data set]},
  DOI =                 {10.5281/zenodo.4099632},
  year =         2020,
  month =         11,
}

@misc{ccby4.0,
  title =         {Creative Commons
                  Attribution-NonCommercial-ShareAlike 4.0
                  International (CC BY-NC-SA 4.0) [software license]},
  url =          {https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode},
}

@incollection{kluEtAl2016,
  title =         {Jupyter Notebooks-a publishing format for
                  reproducible computational workflows.},
  author =         {Kluyver, T. and Ragan-Kelley, B. and P{\'e}rez,
                  F. and Granger, B.~E. and Bussonnier, M. and
                  Frederic, J. and Kelley, K. and Hamrick, J.~B. and
                  Grout, J. and Corlay, S. and others},
  booktitle =         {Positioning and Power in Academic Publishing:
                  Players, Agents and Agendas},
  editor =         {Loizides, F. and Schmidt, B.},
  volume =         2016,
  year =         2016,
  doi =
                  {https://doi.org/10.3233/10.3233/978-1-61499-649-1-87}
}

@misc{latexproject,
  title =         {\LaTeX{} --- A docoument preparation system
```

```
                  [software]},
  author =        {{\LaTeX{} developers}},
  url =                   {https://www.latex-project.org/},
}
```