EEG Many Pipelines - Code Mechanics

Sebastian Speer[1], Antonio Schettino[2,3], & Ana Martinovici[4]

[1] Social Brain Lab, Netherlands Institute for Neuroscience, Amsterdam, The Netherlands

[2] Erasmus Research Services, Erasmus University Rotterdam, Rotterdam, The Netherlands

[3] Institute for Globally Distributed Open Research and Education (IGDORE), Sweden

[4] Rotterdam School or Management, Erasmus University Rotterdam, Rotterdam, The Netherlands

Author Note

Authorship order was randomly determined via the `sample` function in *R*.

SS preprocessed the data and performed time-frequency analysis (research questions 2b, 2c, 3b, 4b). **AS** preprocessed the data and performed ERP analysis (research questions 1, 2a, 3a, 4a). **AM** was responsible for project management, GitHub repository, and reproducibility. **SS**, **AS**, and **AM** wrote the report.

Correspondence concerning this article should be addressed to Ana Martinovici, Burgemeester Oudlaan 50, 3062 PA Rotterdam, Netherlands. E-mail: martinovici@rsm.nl

Abstract

One or two sentences providing a **basic introduction** to the field, comprehensible to a scientist in any discipline.

Two to three sentences of **more detailed background**, comprehensible to scientists in related disciplines.

One sentence clearly stating the **general problem** being addressed by this particular study.

One sentence summarizing the main result (with the words “**here we show**” or their equivalent).

Two or three sentences explaining what the **main result** reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge.

One or two sentences to put the results into a more **general context**.

Two or three sentences to provide a **broader perspective**, readily comprehensible to a scientist in any discipline.

Keywords: EEG Many Pipelines, scene categorization, vision, EEG, ERP, time-frequency analysis, Bayesian multilevel linear regression, TFCE

Word count: X

EEG Many Pipelines - Code Mechanics

1 Introduction

Explain how we got the data and instructions.

2 Methods

2.1 Preprocessing - TFR

EEG data were filtered with a low cutoff filter of 1 Hz to remove slow drifts, a high cutoff filter at 40 Hz to attenuate high frequency power. Subsequently, bad and noisy channels were detected using several different approaches as implemented in the PREP pipeline (Bigdely-Shamlo, Mullen, Kothe, Su, & Robbins, 2015). First, by means of correlation, we checked how well a given channel is correlated with all other channels (categorized as bad at r , 0.4); second, we checked by using the robust z-score deviation aggregates per channel (categorized as bad at z . 5); third, by using the robust z-score estimates of high-frequency noise per channel (categorized as bad at z . 5); and finally, we checked by using the random sample consensus (RANSAC) channel correlations, which is the correlation for each channel with itself across the original data versus the RANSAC predicted data (categorized as bad at r , 0.75) as implemented in the PREP pipeline (Bigdely-Shamlo et al., 2015). After detection, these channels were removed from the data and subsequently interpolated (i.e., estimated from surrounding channels). Interpolation was performed using the spherical spline method (Perrin, Pernier, Bertrand, & Echallier, 1989) as implemented in MNE-Python, which projects the sensor locations onto a unit sphere and interpolates the signal at the channels identified as bad on the signals for the good channels. Afterwards, the EEG data were re-referenced to the average signal across channels. Next, ocular artifacts were removed by performing an independent component analysis on the data and then correlating the resulting components with the EOG channels to see which of the components represented the ocular artifacts. The component that correlated the highest with the EOG channels was then removed from the EEG data.

2.1.1 Cutoffs.

Implemented in file: /code_mechanics/scripts/TFR_preproc_step1.py, lines 30:32.

```
    # filter cutoffs
    cutoff_low = 1 # adapated to TFR analysis
    cutoff_high = 40
```

2.1.2 Bad Channels at 0.4.

Implemented in file: /code_mechanics/scripts/TFR_preproc_step1.py, lines 279:298.

```
        # detect noisy channels based on:
        # - missing signal (NaN)
        # - flat signal
        # - deviation
        # - HF noise (frequency components > 50 Hz considerably higher than median cha
        # - correlation
        #   - bad correlation window if maximum correlation with another channel is be
        #   - a channel is "bad-by-correlation" if its fraction of bad correlation win
        # - low signal-to-noise ratio (too much high-frequency noise + low channel cor
        # - dropout (long time windows with completely flat signal, default fraction t
        # - RANSAC (random sample consensus approach): predict signal based on signals
        #   - split recording into non-overlapping time windows (default: 5 seconds)
        #   - correlate each channel's RANSAC-predicted signal with its actual signal
        #   - bad window if predicted vs. actual signal correlation below correlation
        #   - bad channel if its fraction of bad RANSAC windows is above threshold (de
        # for more info, see https://pyprep.readthedocs.io/en/latest/generated/pyprep..
```

```
    nd.find_all_bads(
        ransac = True,
        channel_wise = False
        )
```

2.1.3 Questions for Sebastian.

- Where do you check the “robust z-score deviation aggregates per channel (categorized as bad at z . 5)”?
- Is the “do not apply 1 Hz high-pass filter” comment accurate? If so, isn’t that what happens at lines 242:247?

Implemented in file: /code_mechanics/scripts/TFR_preproc_step1.py, lines 241:247.

```
    # filter data
    raw = raw.filter( # apply high-pass filter
        l_freq = cutoff_low,
        h_freq = None).filter( # apply low-pass filter
            l_freq = None,
            h_freq = cutoff_high
            )
```

Implemented in file: /code_mechanics/scripts/TFR_preproc_step1.py, lines 269:277.

```
    # detect noisy channels
    nd = NoisyChannels(
        raw,
        # do not apply 1 Hz high-pass filter before bad channel detection:
        # the data have already been high-pass filtered at 0.1 Hz
```

```
            # and we don't want to miss bad channels with slow drifts
            do_detrend = False,
            random_state = project_seed # RNG seed
            )
```

2.2 Epoching

The EEG data was then subsampled by a factor of four (i.e., from 512 Hz to 128 Hz and segmented into 800 ms epochs, with 300 ms before stimulus onset (onset of the scene) until 500 ms after stimulus onset. The epochs were baseline corrected using the 300 ms preceding the stimulus onset. The resulting epochs were then subjected to *Autoreject*, an automated artifact detection algorithm based on machine-learning classifiers, and cross-validation to estimate the optimal peak-to-peak threshold (Jas, Engemann, Bekhti, Raimondo, & Gramfort, 2017). This algorithm was implemented to remove artifacts not identified by previous preprocessing steps, and depending on the number of bad sensors for a given trial, either repairs the trial based on interpolation or excludes it from further analysis. The preprocessed data were then submitted to a Morlet wavelet analysis to transform the data into the time-frequency domain with 18 log-scaled frequency bins ranging from 4 to 40 Hz to have higher sensitivity in lower frequency ranges such as the theta band. To optimize both spectral and temporal resolution, the number of cycles to include in the sliding time window were defined by dividing each individual frequency by two. After transforming the data to the time-frequency domain, the data were decimated by a factor of two (sampling every second time point) to increase computational efficiency.

2.2.1 downsample and segmentation. Implemented in file: /code_mechanics/scripts/TFR_preproc_step1.py, lines 413:427.

```
        # create epochs (all conditions)
        # NOTE: epochs are subsampled (512 Hz --> 128 Hz),
        # to lower the chances of Type II error in subsequent statistical analyses
        epochs = mne.Epochs(
            raw,
            events, # events
            tmin = begin_epoch, # start epoch
            tmax = end_epoch, #end epoch
            baseline = (begin_epoch, 0), # time window for baseline correction
            picks = None, # include all channels
            preload = True,
            decim = 4, # subsample data by a factor of 4, i.e., 128 Hz (for more info,
            detrend = None, # do not detrend before baseline correction
            reject_by_annotation = False # do not reject based on annotations
            )
```

3 Analysis

3.1 RQ1

The first hypothesis we were asked to test was the following:

There is an effect of scene category (i.e., a difference between images showing man-made vs. natural environments) on the amplitude of the N1 component, i.e. the first major negative EEG voltage deflection.

To address this question, we fit a Bayesian multilevel linear model on N1 amplitude values, with *condition* (2 levels: *man-made*, *natural*) as *constant* (a.k.a. fixed) effect and participant and trial as *varying* (a.k.a. random) effects. We allowed intercepts and slopes to vary as a function of participant and trials, to model general and condition-specific inter-individual differences. As a likelihood function, we chose a Gaussian distribution.

An important aspect of Bayesian analysis is the choice of priors (e.g., Natarajan & Kass, 2000). Given the susceptibility of the electrophysiological signal to inter-individual differences (e.g., skull thickness, skin conductance, or hair), we decided to base our priors on the current data by visually inspecting the grand average ERP waveform, i.e., collapsed across *man-made* and *natural* condition. This procedure follows the same logic of *collapsed localizers*, useful when prior research does not allow to reliably identify scalp electrodes and time windows related to ERP components of interest (e.g., Luck & Gaspelin, 2017, p. 150). For the main analysis, we placed **informative priors** on the *intercept* – a normal distribution with mean $\mu = 4$ and standard deviation $\sigma = 2$: $Normal(4, 2)$ – and β coefficient, $Normal(0, 1)$. To assess whether our chosen informative prior would bias parameter estimates (and, consequently, the interpretation of the results; Depaoli and van de Schoot (2017)), we ran the same multilevel linear model with **weakly informative** priors (*intercept*: $Normal(4, 4)$; β: $Normal(0, 4)$) and **uninformative** priors (*intercept*: $Normal(4, 10)$; β: $Normal(0, 10)$). We anticipated that the choice of prior would have negligible effects on the posterior distributions, because the influence of the prior washes out with a large amount of data (Edwards, Lindman, & Savage, 1963).

Since we had no prior knowledge regarding the standard deviation of participant and trials, we placed a **weakly informative prior** on these varying effects: a *t*-distribution with degrees of freedom $\nu = 3$, location $\mu = 0$, and scale $\sigma = 2$, $Student(3, 0, 2)$.

Models were fitted in *R* using the `brms` package (Bürkner, 2018), which employs the probabilistic programming language *Stan* (Carpenter et al., 2017) to implement a Markov chain Monte Carlo (MCMC) algorithm (i.e., No-U-Turn sampler; Homan and Gelman (2014)) to estimate posterior distributions of the parameters of interest. Four MCMC chains with 4000 iterations (2000 warm-up) and no thinning were run to estimate parameters in each of the fitted models. Model convergence was assessed as follows: (*i*) visual inspection of trace plots, rank plots, and graphical posterior predictive checks (Gabry, Simpson, Vehtari, Betancourt, & Gelman, 2019); (*ii*) Gelman-Rubin $\hat{R}$ statistic (Gelman et al., 2013) – comparing the between-chains variability to the within-chain variability – between 1 and 1.05 (see also Nalborczyk, Batailler, Lœvenbruck, Vilain, & Bürkner, 2019).

Posterior distributions of the model parameters were summarized using the mean and 95% credible interval (CI). Differences between conditions were calculated by computing the difference between posterior distributions of the respective conditions.

Statistical inference was performed using the **HDI + ROPE** decision rule (Kruschke, 2018): values were accepted or rejected against a null hypothesis considering a small effect as practically equivalent to zero (Region of Practical Equivalence; *ROPE*). To mitigate the inevitable subjectivity intrinsic in arbitrarily choosing the range of negligible values, we explored a range of plausible ROPEs, from $\pm 0.05\mu V$ to $\pm 0.5\mu V$ in steps of $0.01\mu V$. If the percentage of the posterior differences within the full ROPE was smaller than 5%, the null hypothesis was rejected.

3.2 RQ2

To test whether there are effects of image novelty (RQ2; i.e., between images shown for the first time/new vs. repeated/old images) we conducted a multilevel analysis contrasting the EEG data from trials with old images against trials with new images. To test for differences in theta power at fronto-central channels we focused on the frequency range from 4-8 Hz and all frontocentral channels (FC1, FCz, FC2). At the first level (i.e., the participant level), we computed the averaged time-frequency maps for each of the two conditions. We then tested the resulting averaged maps at the second level for significant group effects, using a paired-sample *t*-test. We used cluster-based permutation testing as a stringent control for multiple comparisons (Maris & Oostenveld, 2007). Specifically, for every sample across the three channels, we quantified the experimental effect by a *t* value.

Selection of samples for inclusion in a cluster was implemented using threshold-free cluster enhancement (TFCE) (Smith & Nichols, 2009). TFCE eliminates the free parameter initial threshold value that determines which points are included in clustering by approximating a continuous integration across possible threshold values with a standard Riemann sum. We subsequently clustered selected samples in connected sets based on temporal and spectral adjacency, and we computed cluster-level statistics by taking the sum of the t values within every cluster. Subsequently, we performed permutation testing using the Monte Carlo method to compute the posterior significance probability of our observed effect(Maris & Oostenveld, 2007). This analysis results in a cluster of adjacent data points across time, frequencies, and channels, which significantly differs in activity between old and new images. To test for differences in alpha power at posterior channels we focused on the frequency range from 8-13 Hz and all posterior channels (P7, P5, P3, P1, P2, P4, P6).

3.3 RQ3

The same analysis approach as described for RQ2 was implemented for RQ3. Specifically, to test whether there are effects of successful recognition of old images on spectral power, at any frequencies, at any channel, at any time, we contrasted time-frequency decomposed EEG data from trials containing old images that were correctly recognized as old with old images incorrectly recognized as new. Here we included all frequencies, timepoints, and channels in the analysis. The same thresholding procedure and permutation testing approach as above was used.

3.4 RQ4

To test whether there are effects of subsequent memory, we conducted exactly the same analysis as described in RQ3, with the only difference that here we contrasted trial containing images that will be successfully remembered vs. forgotten on a subsequent repetition.

4 Results

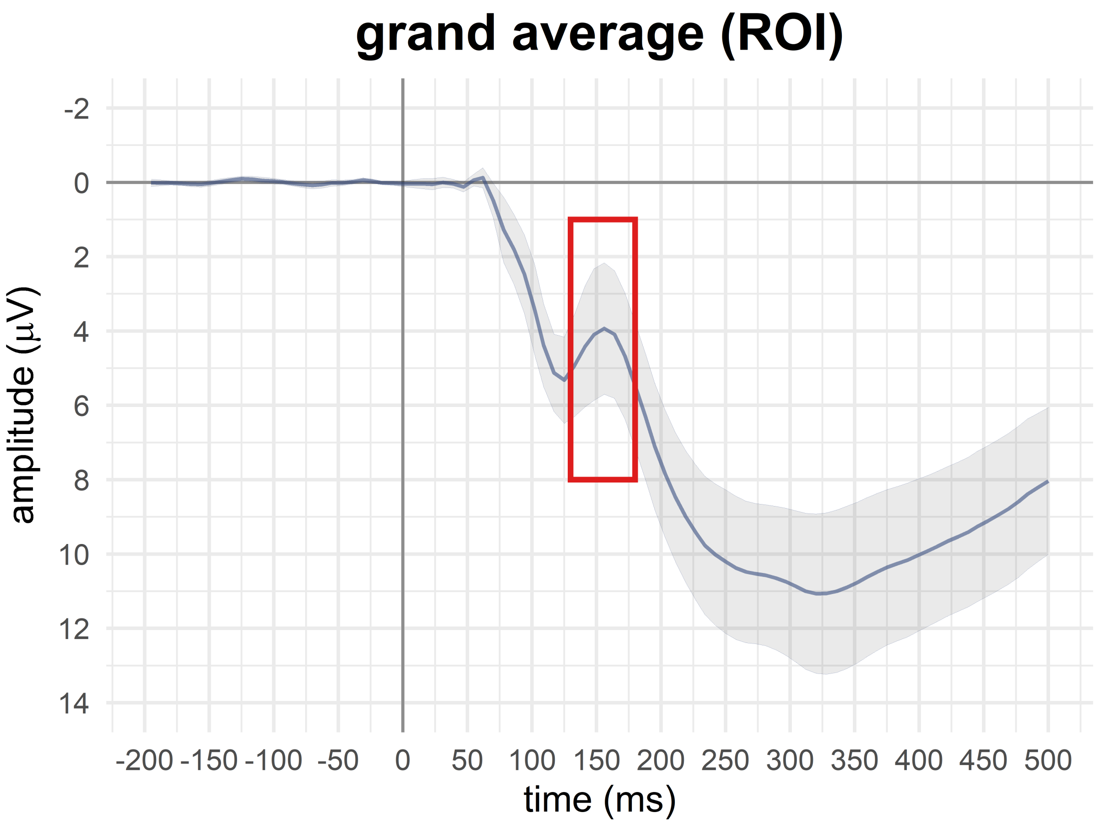

Figure 1. My caption.

No significant clusters were identified for any of the research questions (at $\alpha = 0.05$).

5 Discussion

ADD DISCUSSION HERE

6 Software

EEG preprocessing was carried out using `MNE-Python` (Gramfort, 2013) in *Python* *v*3.9.7 (Van Rossum & Drake, 2009) and *Spyder IDE* *v*5.1.5 (Raybaut, 2009). Analysis, visualization, and report generation were carried out in *R* *v*4.1.3 (R Core Team, 2022) and *RStudio IDE* *v*2022.02.1+461 (RStudio Team, 2020). We used the following *R* packages:

- **data wrangling and analysis**: `here` *v*1.0.1 (Müller, 2020), `Rmisc` *v*1.5 (Hope, 2022), `tidyverse` *v*1.3.1 (Wickham et al., 2019) – in particular `tibble` *v*3.1.6 (Müller & Wickham, 2022), `tidyr` *v*1.2.0 (Wickham & Girlich, 2022), `readr` *v*2.1.2 (Wickham, Hester, & Bryan, 2022), `dplyr` *v*1.0.9 (Wickham, François, Henry, & Müller, 2022) –, `brms` *v*2.17.0 (Bürkner, 2018), `eegUtils` *v*0.7.0 (Craddock, 2022), `emmeans` *v*1.7.3 (Lenth, 2022), `bayestestR` *v*0.11.5.1 (Makowski, Ben-Shachar, & Lüdecke, 2019)
- **visualization**: `ggplot2` *v*3.3.6 (Wickham, 2016), `eegUtils` *v*0.7.0 (Craddock, 2022), `bayesplot` *v*1.9.0 (Gabry & Mahr, 2022), `viridis` *v*0.6.2 (Garnier et al., 2021-04-11, 2021-04), `tidybayes` *v*3.0.2 (Kay, 2022), `patchwork` *v*1.1.1 (Pedersen, 2020)
- **report generation**: `knitr` *v*1.39 (Xie, 2022), `rmarkdown` *v*2.14 (Allaire et al., 2022), `papaja` *v*0.1.0.9999 (Aust & Barth, 2022)

7 References

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., … Iannone, R. (2022). *Rmarkdown: Dynamic documents for R.*

Aust, F., & Barth, M. (2022). *Papaja: Prepare reproducible APA journal articles with R Markdown.*

Bigdely-Shamlo, N., Mullen, T., Kothe, C., Su, K.-M., & Robbins, K. A. (2015). The PREP pipeline: Standardized preprocessing for large-scale EEG analysis. *Frontiers in Neuroinformatics*, *9*. https://doi.org/10.3389/fninf.2015.00016

Bürkner, P.-C. (2018). Advanced bayesian multilevel modeling with the R package brms. *The R Journal*, *10*(1), 395. https://doi.org/10.32614/rj-2018-017

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., … Riddell, A. (2017). *Stan*: A probabilistic programming language. *Journal of Statistical Software*, *76*(1). https://doi.org/10.18637/jss.v076.i01

Craddock, M. (2022). *eegUtils: Utilities for electroencephalographic (EEG) analysis.*

Depaoli, S., & van de Schoot, R. (2017). Improving transparency and replication in bayesian statistics: The WAMBS-Checklist. *Psychological Methods*, *22*(2), 240–261. https://doi.org/10.1037/met0000065

Edwards, W., Lindman, H., & Savage, L. J. (1963). Bayesian statistical inference for psychological research. *Psychological Review*, *70*(3), 193–242. https://doi.org/10.1037/h0044139

Gabry, J., & Mahr, T. (2022). *Bayesplot: Plotting for bayesian models.*

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., & Gelman, A. (2019). Visualization in bayesian workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *182*(2), 389–402. https://doi.org/10.1111/rssa.12378

Garnier, S., Ross, N., BoB Rudis, Filipovic-Pierucci, A., Galili, T., Timelyportfolio, … JJ Chen. (2021-04-11, 2021-04). *Sjmgarnier/viridis: Viridis 0.6.0 (pre-CRAN release).* Zenodo. https://doi.org/10.5281/ZENODO.4679424

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC. https://doi.org/10.1201/b16018

Gramfort, A. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, *7*. https://doi.org/10.3389/fnins.2013.00267

Homan, M. D., & Gelman, A. (2014). The no-U-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *The Journal of Machine Learning Research*, *15*(1), 1593–1623.

Hope, R. M. (2022). *Rmisc: Ryan miscellaneous.*

Jas, M., Engemann, D. A., Bekhti, Y., Raimondo, F., & Gramfort, A. (2017). Autoreject: Automated artifact rejection for MEG and EEG data. *NeuroImage*, *159*, 417–429. https://doi.org/10.1016/j.neuroimage.2017.06.030

Kay, M. (2022). *Tidybayes: Tidy data and geoms for bayesian models.* https://doi.org/10.5281/zenodo.1308151

Kruschke, J. K. (2018). Rejecting or accepting parameter values in bayesian estimation. *Advances in Methods and Practices in Psychological Science*, *1*(2), 270–280. https://doi.org/10.1177/2515245918771304

Lenth, R. V. (2022). *Emmeans: Estimated marginal means, aka least-squares means.*

Luck, S. J., & Gaspelin, N. (2017). How to get statistically significant effects in any ERP experiment (and why you shouldn't). *Psychophysiology*, *54*(1), 146–157. https://doi.org/10.1111/psyp.12639

Makowski, D., Ben-Shachar, M. S., & Lüdecke, D. (2019). *bayestestR: Describing effects and their uncertainty, existence and significance within the bayesian framework.* *4*, 1541. https://doi.org/10.21105/joss.01541

Maris, E., & Oostenveld, R. (2007). Nonparametric statistical testing of EEG- and MEG-data. *Journal of Neuroscience Methods*, *164*(1), 177–190. https://doi.org/10.1016/j.jneumeth.2007.03.024

Müller, K. (2020). *Here: A simpler way to find your files.*

Müller, K., & Wickham, H. (2022). *Tibble: Simple data frames.*

Nalborczyk, L., Batailler, C., Lœvenbruck, H., Vilain, A., & Bürkner, P.-C. (2019). An introduction to bayesian multilevel models using brms: A case study of gender effects on vowel variability in standard indonesian. *Journal of Speech, Language, and Hearing Research*, *62*(5), 1225–1242. https://doi.org/10.1044/2018_jslhr-s-18-0006

Natarajan, R., & Kass, R. E. (2000). Reference bayesian methods for generalized linear mixed models. *Journal of the American Statistical Association*, *95*(449), 227–237. https://doi.org/10.1080/01621459.2000.10473916

Pedersen, T. L. (2020). *Patchwork: The composer of plots.*

Perrin, F., Pernier, J., Bertrand, O., & Echallier, J. F. (1989). Spherical splines for scalp potential and current density mapping. *Electroencephalography and Clinical Neurophysiology*, *72*(2), 184–187. https://doi.org/10.1016/0013-4694(89)90180-6

R Core Team. (2022). *R: A language and environment for statistical computing* [Manual]. Vienna, Austria.

Raybaut, P. (2009). Spyder-documentation. *Available Online at: Pythonhosted. Org.*

RStudio Team. (2020). *RStudio: Integrated development environment for R* [Manual]. Boston, MA.

Smith, S., & Nichols, T. (2009). Threshold-free cluster enhancement: Addressing problems of smoothing, threshold dependence and localisation in cluster inference. *NeuroImage*, *44*(1), 83–98. https://doi.org/10.1016/j.neuroimage.2008.03.061

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual.* Scotts Valley, CA: CreateSpace.

Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis.*

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., … Yutani, H. (2019). *Welcome to the tidyverse.* *4*, 1686. https://doi.org/10.21105/joss.01686

Wickham, H., François, R., Henry, L., & Müller, K. (2022). *Dplyr: A grammar of data* *manipulation.*

Wickham, H., & Girlich, M. (2022). *Tidyr: Tidy messy data.*

Wickham, H., Hester, J., & Bryan, J. (2022). *Readr: Read rectangular text data.*

Xie, Y. (2022). *Knitr: A general-purpose package for dynamic report generation in R.*