Workflow analysis and standardisation of `PICCA`

Ignasi Pérez i Ràfols
Grau en enginyeria informàtica
Desenvolupament web

Consultor: Gregorio Robles Martínez
Professor responsable de l'assignatura: Santi Caballé Llobet

Data lliurament: 12/06/2020

FITXA DEL TREBALL FINAL

Títol del treball:	Workflow analysis and standardisation of `PICCA`
Autor:	Ignasi Pérez i Ràfols
Nom del consultor/a:	Gregorio Robles Martínez
Nom del PRA:	Santi Caballé Llobet
Data de lliurament (mm/aaaa):	*06/2020*
Titulació o programa:	Grau en enginyeria informàtica
Àrea del Treball Final:	Desenvolupament web
Idioma del treball:	Anglès
Paraules clau:	Cosmology, Standardization, Computer software

Resum del Treball (màxim 250 paraules):

In the precision cosmology era, more and more data are collected and analyzed. As data (and the associated systematics) is better understood the analysis is revised and refined. In turn, the analysis codes get more and more complex. These programs are usually developed by astronomers and their development is often not planned but born out of necessity. An example of such codes is `picca`. This code is used mainly to measure correlation functions associated with the Lyα forest. This includes the 3D Lyα BAO analysis, and the 1D Power Spectrum analysis of the Lyα forest. In this work, we present an analysis of the code, documenting its workflow for the first time, and providing a high level of standardisation of the code (quantified using `pylint`). We also evaluate the efficiency of the automatic testing of `PICCA`. We conclude that a full revision of the data model is required, and that more extensive tests that are smaller and independent are needed.

Contents

1. Introduction

In the past few years, the cosmology has moved from the usage of small datasets (a few tens of objects) to the usage of hundreds of thousands of spectra. This huge statistical leap has changed the cosmological framework from a more qualitative cosmology, where a factor of two wasn't really important, to a quantitative cosmology. We have entered the so-called precision cosmology era. This has been possible thanks to surveys such as Sloan Digital Sky Survey (SDSS; York et al., 2000; Abazajian et al., 2009), the 2-degree Field Galaxy Redshift Survey (2dFGRS; Colless et al., 2001), the Wilkinson Microwave Anisotropy Probe (WMAP; Bennett et al., 2003), the 6-degree Field Galaxy Survey (6dFGS; Jones et al., 2004), the WiggleZ survey(Drinkwater et al., 2010), Planck (Planck Collaboration, 2011), the Baryon Oscillation Spectroscopic Survey (BOSS; Dawson et al., 2013, part of the SDSS-III survey Eisenstein et al. 2011), the extended BOSS (eBOSS; Dawson et al., 2016, part of SDSS-IV Blanton et al. 2017), or Gaia (Gaia Collaboration, 2016). These datasets are not only photometric catalogues, but often are comprised of hundreds of thousands of spectra, and will become even larger in the near future with the next generation of surveys: WEAVE-QSO (Pieri et al. 2016; as part of WEAVE, Dalton et al. 2016), DESI[1] (DESI Collaboration, 2016), Euclid (Laureijs et al., 2010), 4MOST (de Jong et al., 2016), WFIRST (Dressler et al., 2012; Green et al., 2012; Spergel et al., 2015), PSF (Takada et al., 2014; Chiba et al., 2016), ...

One of the big challenges has been (and still is) to constrain the expansion history of the Universe. Very high redhsift measurements using the Cosmic Microwave Background (CMB) are lead by the Planck Collaboration (see e.g. Planck Collaboration, 2016). At the other end, local measurements of the expansion history are performed using supernovae (e.g. Guy et al., 2010; Conley et al., 2011; Sullivan et al., 2011). Measurements from the Baryon Acoutsic OScillations (BAO) allow us to complete the picture by providing intermediate redshift measurements.

BAO are measured as an excess probability at scales of $\sim 100\mathrm{h}^{-1}\mathrm{Mpc}$ in the correlation function between tracers of the underlying dark matter distribution. Equivalently they can be measured in the wiggles of its Fourier Transform, the Power Spectrum. Based on the tracers they use, there are two distinct BAO analysis techniques. The first technique is called BAO clustering. This technique uses galaxies (Eisenstein et al., 2005; Cole et al., 2005; Percival et al., 2007, 2010; Blake et al., 2011; Beutler et al., 2011; Chuang & Wang, 2012; Padmanabhan et al., 2012; Mehta et al., 2012; Xu et al., 2013; Anderson et al., 2012, 2014b,a; Ross et al., 2015; Alam et al., 2017; Bautista et al., 2018), galaxy clusters (Hong et al., 2016) or, more recently, quasars (Ata et al., 2018; Laurent et al., 2016) as tracers and measures the correlation function by counting the number of pairs found at a given distance.

The second technique is called Lyman-α (Lyα) BAO analysis. This technique aims to measure BAO at slightly higher redshifts, where the observed number density of galaxies is lower. To account for this, we use the Intergalactic Medium (IGM) absorption as tracer of the underlying matter density. This absorption is detected in the spectra of background quasars. The main absorption used in this analysis is the one produced by the Lyα hydrogen transition (hence the name). We measure both the auto-correlation (Busca et al., 2013; Slosar et al., 2013; Kirkby et al., 2013; Delubac et al., 2015; Bautista et al., 2017; de Sainte Agathe et al., 2019), and the cross-correlation of Lyα with quasars (Font-Ribera et al., 2014; du Mas des

[1]Note that observations have already begun

Bourboux et al., 2017; Blomqvist et al., 2019). The same technique has been used to measure BAO using the cross-correlation of the triply-ionized carbon absorption absorption with quasars (Blomqvist et al., 2018), of the singly-ionized magnesium absorption with galaxies (du Mas des Bourboux et al., 2019), and to measure other properties of the tracers themselves by cross-correlating the Lyα absorption and Damped Lyα Absorbers (DLAs; Font-Ribera et al., 2012; Pérez-Ràfols et al., 2018a,b).

Prior to the measurement of the Lyα-quasars cross-correlation from the twelfth Data Release (DR12) of BOSS (du Mas des Bourboux et al., 2017), multiple codes written in multiple languages (mostly `C++` and `Java`) were used to perfrom this analysis. As the astronomical comunity began transitioning to `Python`, so did the Lyα working group of BOSS/eBOSS. At this point we developed a Package for Igm Cosmological-Correlations Analyses (`PICCA`[2]). This code was then adopted as the standard package for the Lyα analysis and is currently being maintained and furhter developed by the Lyα working group of DESI.

`PICCA` has proven to be very useful for the progressively more refined Lyα BAO analysis. However, as the code has becomes more and more complex, it is becoming increasingly difficult for more junior developers to contribute to its development. Additionally, the code was originally designed to deal with BOSS data, but is now being used with data from the DESI collaboration. We have patched the code so that it can run with the new data format, but this is not a desired long term strategy.

Therefore, a re-engineering of `PICCA` would be desirable. We need to achieve a deeper understanding of the code if we want to adapt it to the new volumes and formats of data. However, with codes having reached the complexity of `PICCA`, specially if there are multiple developers, this is by no means an easy task. To make this task more accessible it is interesting to have the code homogenized. For this purpose Guido van Rossum et al. developed a series of recommendations on coding style for python (van Rossum et al., 2001 (accessed June 12, 2020). Very similar recommendations are also provided by Google[3], who provide as well an automatic formatter called `yapf`[4].

The purpose of this work, then, is to analyze the current level of standardisation of `PICCA`, and improve it, making it more accessible to future developers. This is a necessary initial step in the re-engineering of `PICCA` that will be necessary to address the future challenges explained above. We note that currently there is no publication explaining the behaviour of the code to newcomers, and we plan this work to serve as a reference in that sense. An analysis oriented towards the optimization of this code would also be desirable, but we leave it for future studies.

This work is organized as follows. We start by analysing the workflow of the code in Section 2. We then explain the standardisation of the code in Section 3, and we analyze the tests performed to ensure that the implemented changes do not affect the behaviour of the code in Section 4. Finally we summarize our conclusions in Section 5.

2. Workflow analysis of PICCA

We start our analysis of `PICCA` by studying the workflow of the different parts of the code. Because this workflow is rather complex, we will focus on the workflow for the two main

[2]`https://github.com/igmhub/picca`

[3]`http://google.github.io/styleguide/pyguide.html`

[4]`https://github.com/google/yapf/`

analysis, and then we will say a few words of alternative workflows. Hereafter, we refer as main 3D analysis (or simply 3D analysis) as the analysis including the combined measurements of the Lyα auto-correlation and its cross-correlations with quasars, and as main 1D analysis (or simply 1D analysis) as the analysis related to the one-dimensional Power Spectrum of the Lyα forest. These analysis are described in detail in du Mas des Bourboux et al. (202-0) and Chabanier et al. (2019) respectively, but we summarize them here.

There are four main steps to the main 3D analysis. In the first step we extract the Lyα mean transmission fluctuation field, hereafter the delta field. Then we compute the Lyα auto-correlation and the cross-correlation with quasars. These two steps involve the computation of the correlation function, the distortion matrix, and the covariance matrix (we will come back to this in Sections 2.2 and 2.3, and can be run in parallel. It also involves formatting the results in such a way that the fitter will understand. The last step concerns fitting the theoretical model to the measured correlation functions to extract cosmological constraints, and will not be explored in this work as a new version fitter is being constructed (Andrei Cuceu, private communication). This is shown graphically in Figure 1, where we also signal the scripts of PICCA used in each step.

We see that there are some elements decoupled from the main 3D workflow analysis. They are optional additional steps that can be added to the main analysis. We will describe each of this elements, as well as the main steps for the analysis, in the alternative workflow sections below (Section 2.5).

The main 1D analysis has basically two steps. In the first step, we again extract the delta field (as it is done for the main 3D analysis). Then we compute the 1D Power Spectrum. This analysis is inherently different from the main 3D analysis. There, we do not correlate delta field pixels with other pixels in the same line of sight. Here, we only correlate delta field pixels with other pixels in the same line of sight (hence 1D). Because of this, we do not require the computation of the distortion matrix. This is shown graphically in Figure 2, where we also signal the scripts of PICCA used in each step.

2.1 Compute deltas

We start by describing the computation of the delta field. This is a common step in the 3D and 1D analysis, though there are a few steps required only for the 1D analysis. The workflow for this step is shown graphically in Figure 3. Overall it can be summarised in these steps: read the calibration data, read the data, masking the data, computing the quasar continuum iteratively, computing the delta stacks, and saving the results. The difference between the workflow in the 1D and the 3D analysis lies basically in the reading of the data. This step is marked with a filled light green rectangle and is rather complex so it is further expanded in Figure 4. As a part of masking the data, DLAs are masked. The DLA masking step is signaled with a filled red box and is further expanded in Figure 5.

2.2 Compute auto-correlation

The auto-correlation is computed in the main 3D analysis. As we can see schematically in Figure 1 there are several steps to achieve this. The first two things we need to compute are the auto-correlation and the distortion matrix. They can be run in parallel and use as inputs the delta files produced previously (see Section 2.1). This produces the correlation function

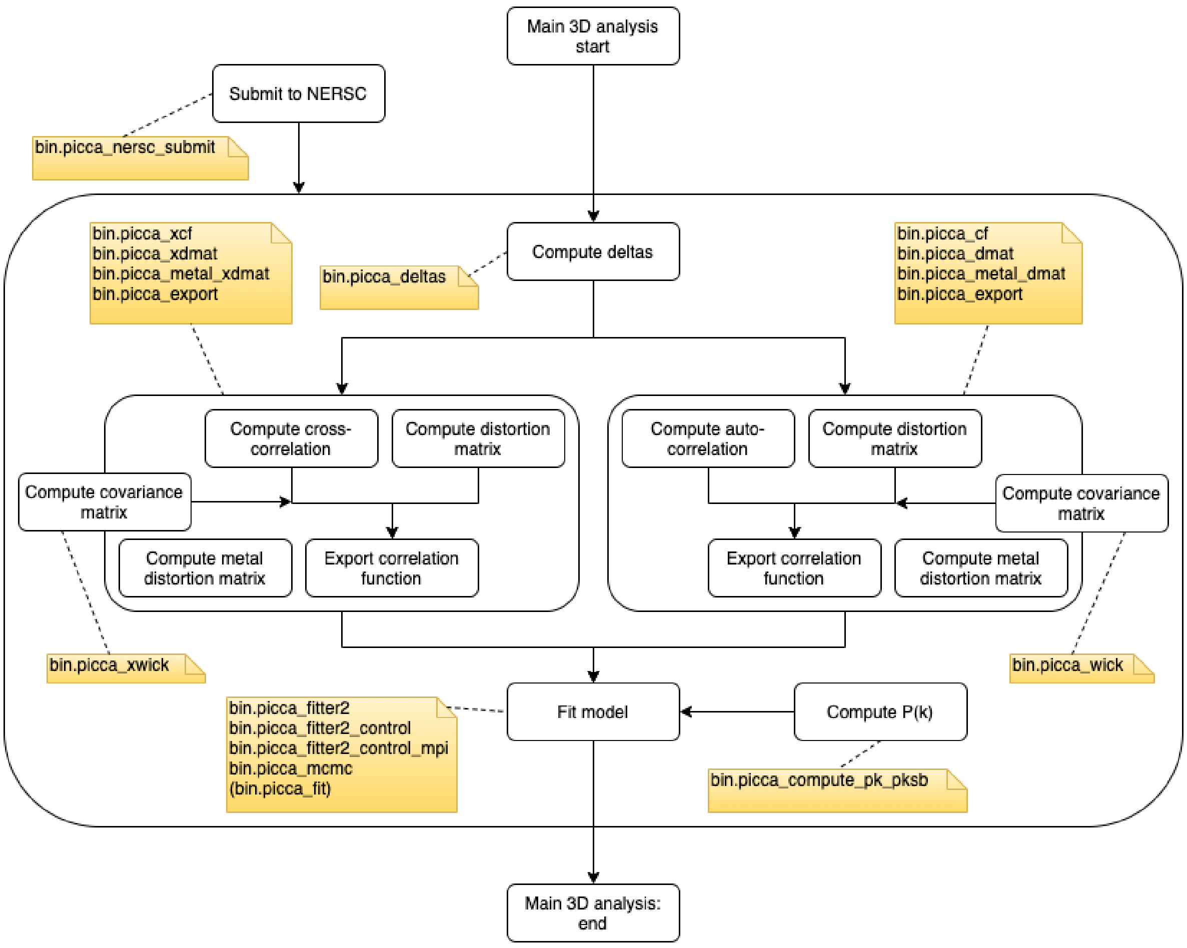

Figure 1: Workflow for the main 3D analysis. It can be split into four phases: computation of the delta field, computation of the auto-correlation, computation of the cross-correlation and fitting. The workflow for each step (except for the fitting step) is further detailed in Figures 3, 6, 10 respectively. The workflow for optional steps outside the main analysis are further detailed in Section 2.5. Yellow boxes indicate the scripts of PICCA used in each step

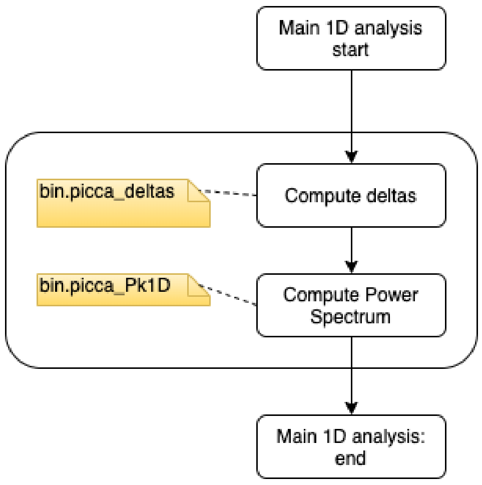

Figure 2: Workflow for the main 1D analysis. It can be split into two phases: computation of the delta field, and computation of the 1D Power Spectrum. The workflow for each step is further detailed in Figures 3, and 13 respectively. Yellow boxes indicate the scripts of `PICCA` used in each step

and the distortion matrix in a series of healpixs, and their workflow diagrams are shown in Figures 6 and 7. As seen in these figures, and despite the names of the steps here, these files can also be used to compute the cross-correlation between two different delta fields produced by different absorption lines or by the same absorption line but at different wavelengths.

They are then combined into the final correlation function measurement in the step called export correlation function. This step is also executed in the computation of the cross-correlation (see Section 2.3). The workflow for this step is given in Figure 8.

In parallel to the export of the correlation function, we can compute the distortion matrix for metal absorption. The workflow for this step is similar to that of the distortion matrix computation (though a bit more complex since multiple distortion matrices can be computed at the same time) and is given in Figure 9.

2.3 Compute cross-correlation

The cross-correlation is computed in the main 3D analysis, and the workflow is parallel to the computation of the auto-correlation (see Section 2.2). As we can see schematically in Figure 1 there are several steps to achieve this. The first two things we need to compute are the cross-correlation and the distortion matrix. They can be run in parallel and use as inputs the delta files produced previously (see Section 2.1). This produces the correlation function and the distortion matrix in a series of healpixs, and their workflow diagrams are shown in Figures 10 and 11.

They are then combined into the final correlation function measurement in the step called export correlation function. This step is also executed in the computation of the auto-correlation (see Section 2.2). The workflow for this step is given in Figure 8.

In parallel to the export of the correlation function, we can compute the distortion matrix for metal absorption. The workflow for this step is similar to that of the distortion matrix computation and is given in Figure 12.

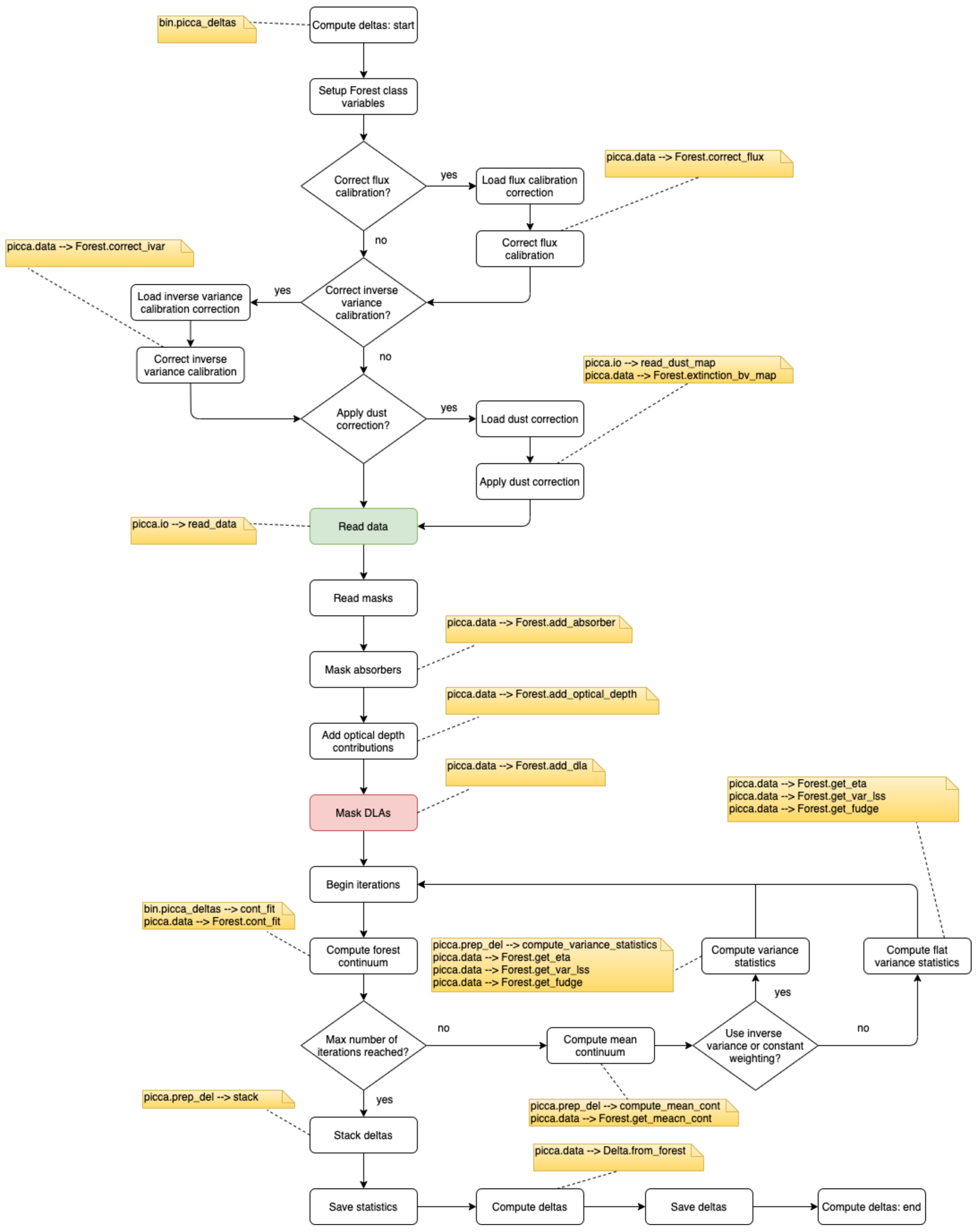
bin.picca_deltas
Compute deltas: start
Setup Forest class variables
Correct flux calibration?
yes
Load flux calibration correction
picca.data --> Forest.correct_flux
Correct flux calibration
no
picca.data --> Forest.correct_ivar
Load inverse variance calibration correction
yes
Correct inverse variance calibration?
Correct inverse variance calibration
no
picca.io --> read_dust_map
picca.data --> Forest.extinction_bv_map
Apply dust correction?
yes
Load dust correction
Apply dust correction
picca.io --> read_data
Read data
Read masks
picca.data --> Forest.add_absorber
Mask absorbers
picca.data --> Forest.add_optical_depth
Add optical depth contributions
picca.data --> Forest.add_dla
Mask DLAs
picca.data --> Forest.get_eta
picca.data --> Forest.get_var_lss
picca.data --> Forest.get_fudge
Begin iterations
bin.picca_deltas --> cont_fit
picca.data --> Forest.cont_fit
Compute forest continuum
picca.prep_del --> compute_variance_statistics
picca.data --> Forest.get_eta
picca.data --> Forest.get_var_lss
picca.data --> Forest.get_fudge
Compute variance statistics
Compute flat variance statistics
yes
Max number of iterations reached?
no
Compute mean continuum
Use inverse variance or constant weighting?
no
picca.prep_del --> stack
yes
Stack deltas
picca.prep_del --> compute_mean_cont
picca.data --> Forest.get_meacn_cont
picca.data --> Delta.from_forest
Save statistics
Compute deltas
Save deltas
Compute deltas: end

Figure 3: Workflow for the computation of the delta field. This phase is used both in the main 3D analysis (Figure 1) and in the main 1D analysis (Figure 2). The step "Read data" marked with a filled light green rectangle is further expanded in Figure 4, and is where the 1D and the 3D analysis differ. The step "Mask DLA" marked with a filled light red rectangle is further expanded in Figure 5. Yellow boxes indicate the scripts/functions of PICCA used in each step, and steps without explicit labelling occur in the main function.

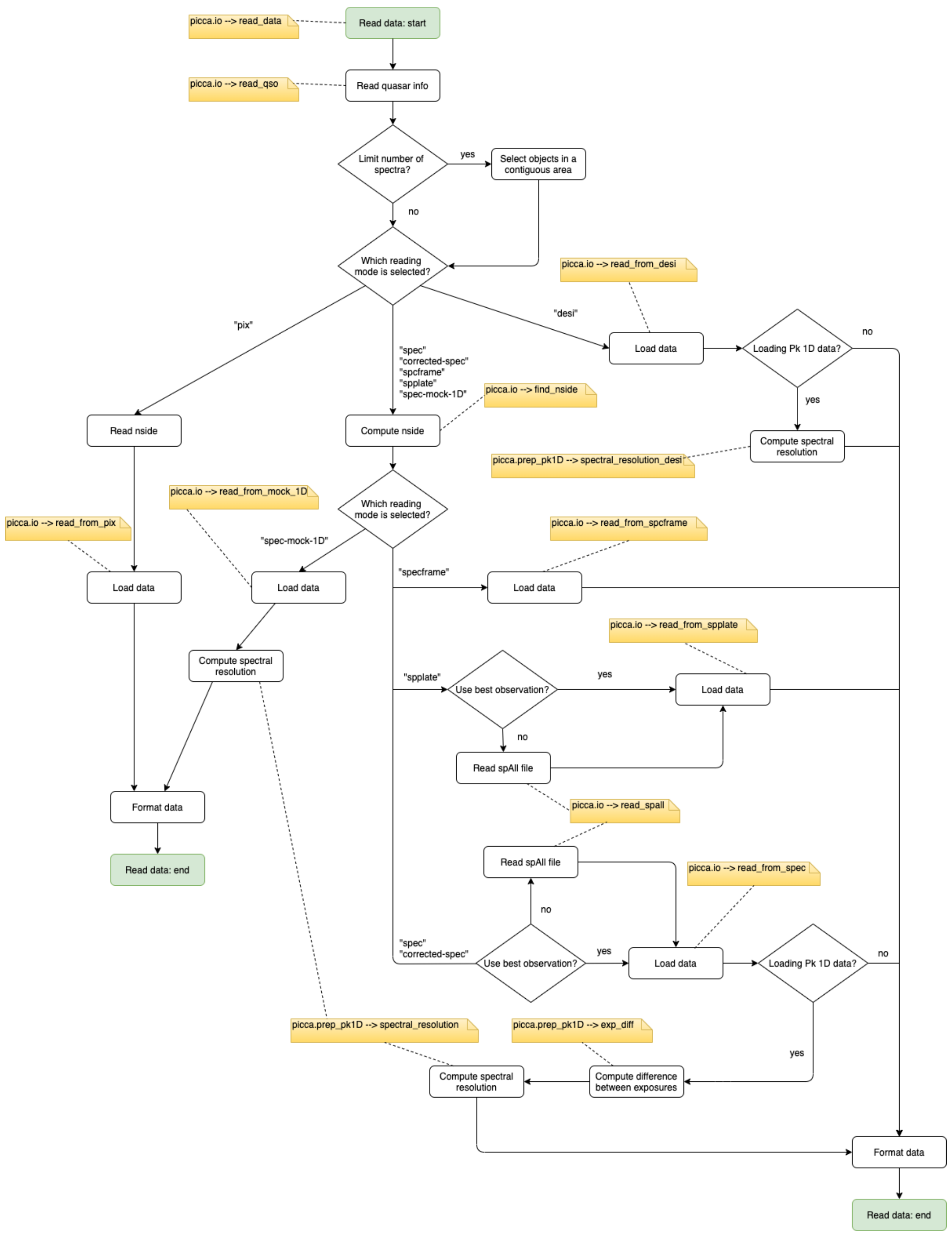

Figure 4: Workflow for the data reading. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

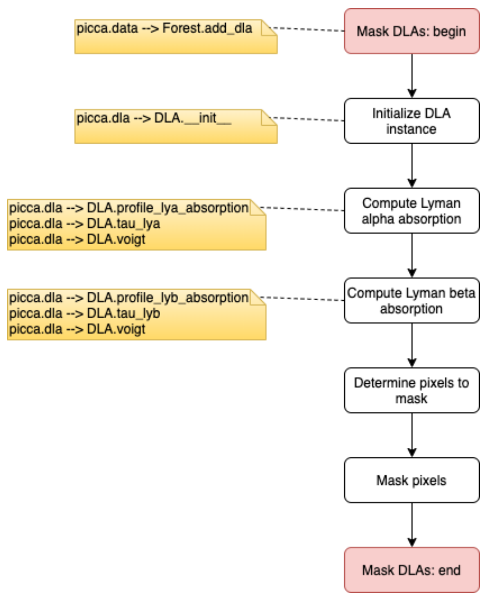
picca.data --> Forest.add_dla
Mask DLAs: begin
picca.dla --> DLA.__init__
Initialize DLA instance
picca.dla --> DLA.profile_lya_absorption
picca.dla --> DLA.tau_lya
picca.dla --> DLA.voigt
Compute Lyman alpha absorption
picca.dla --> DLA.profile_lyb_absorption
picca.dla --> DLA.tau_lyb
picca.dla --> DLA.voigt
Compute Lyman beta absorption
Determine pixels to mask
Mask pixels
Mask DLAs: end

Figure 5: Workflow for the DLA masking. Yellow boxes indicate the scripts of `PICCA` used in each step, and steps without explicit labelling occur in the main function.

2.4 Compute 1D power spectrum

The 1D Power Spectrum is computed in the main 1D analysis. The workflow for this step given in Figure 13. We can see that the workflow here is substantially different from that of the computation of the auto- or cross-correlations. In this case we work in Fourier space (as opposed to working in real space), and the error treatment is completely different. However, even if the analysis is this different, here we also use as inputs the delta files produced previously (see Section 2.1).

2.5 Alternative workflows

2.5.1 Submit to NERSC

To simplify the main 3D analysis, there is a prepared script that allows the user to submit all the steps directly. This script assumed that the code is run at NERSC[5]. The workflow for this script is given in Figure 14.

2.5.2 Correlation function from catalogues of objects

The main analysis is tested against systematic errors using mock realisations of the survey. These mocks are simulated data that mimic the real observations to the best of our abilities (see e.g. Farr et al., 2020). To make these mocks realistic, they need to model the right level of clustering not only for the Lyα forest, but also for the quasars and other modelled collapsed structures (like the High Column Density Absorbers). To test that these objects

[5]National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231

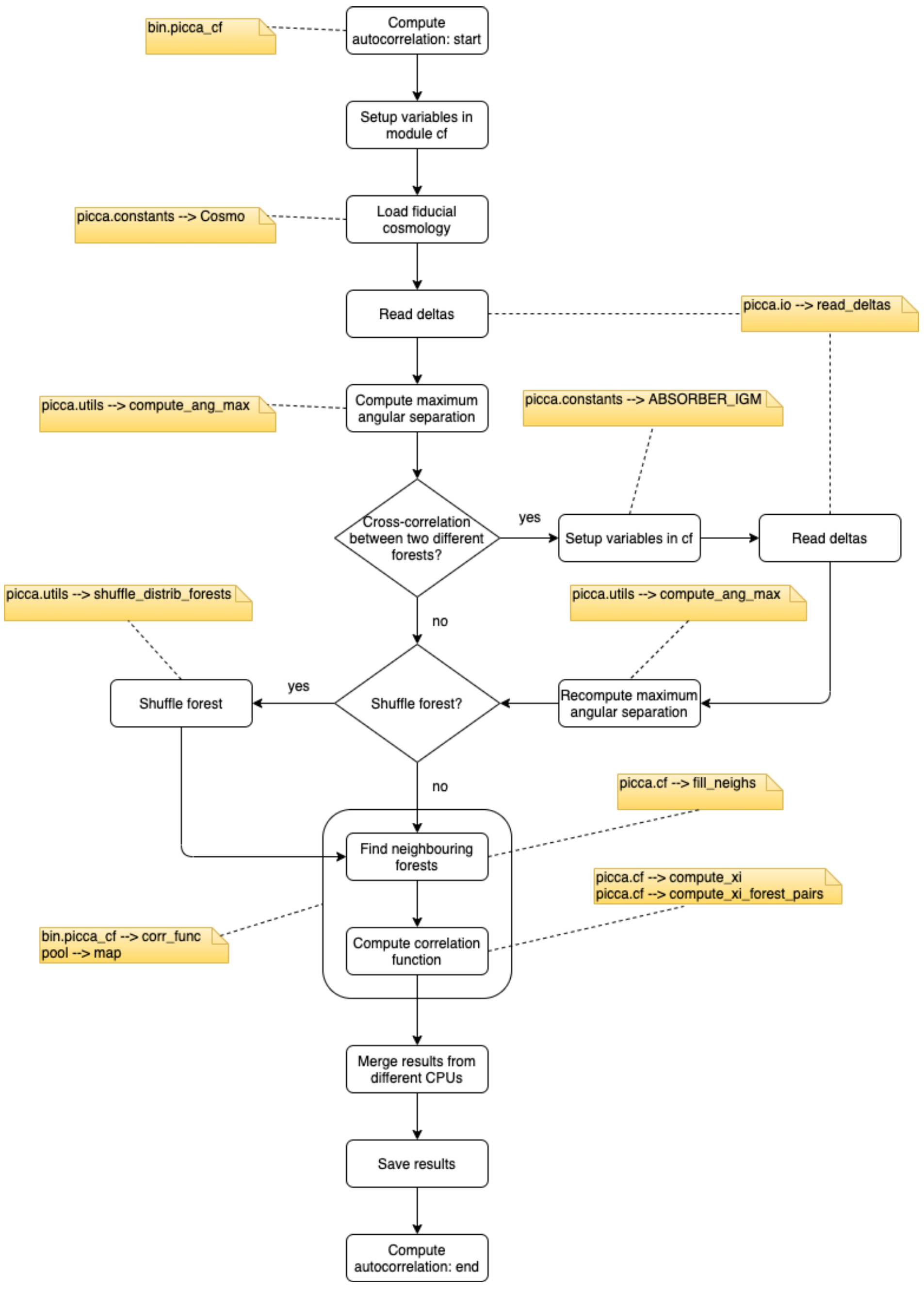
bin.picca_cf
Compute autocorrelation: start
Setup variables in module cf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.utils --> compute_ang_max
Compute maximum angular separation
picca.constants --> ABSORBER_IGM
Cross-correlation between two different forests?
yes
Setup variables in cf
Read deltas
no
picca.utils --> shuffle_distrib_forests
picca.utils --> compute_ang_max
Shuffle forest
yes
Shuffle forest?
Recompute maximum angular separation
no
picca.cf --> fill_neighs
Find neighbouring forests
picca.cf --> compute_xi
picca.cf --> compute_xi_forest_pairs
bin.picca_cf --> corr_func
pool --> map
Compute correlation function
Merge results from different CPUs
Save results
Compute autocorrelation: end

Figure 6: Workflow for the computation of the auto-correlation function. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

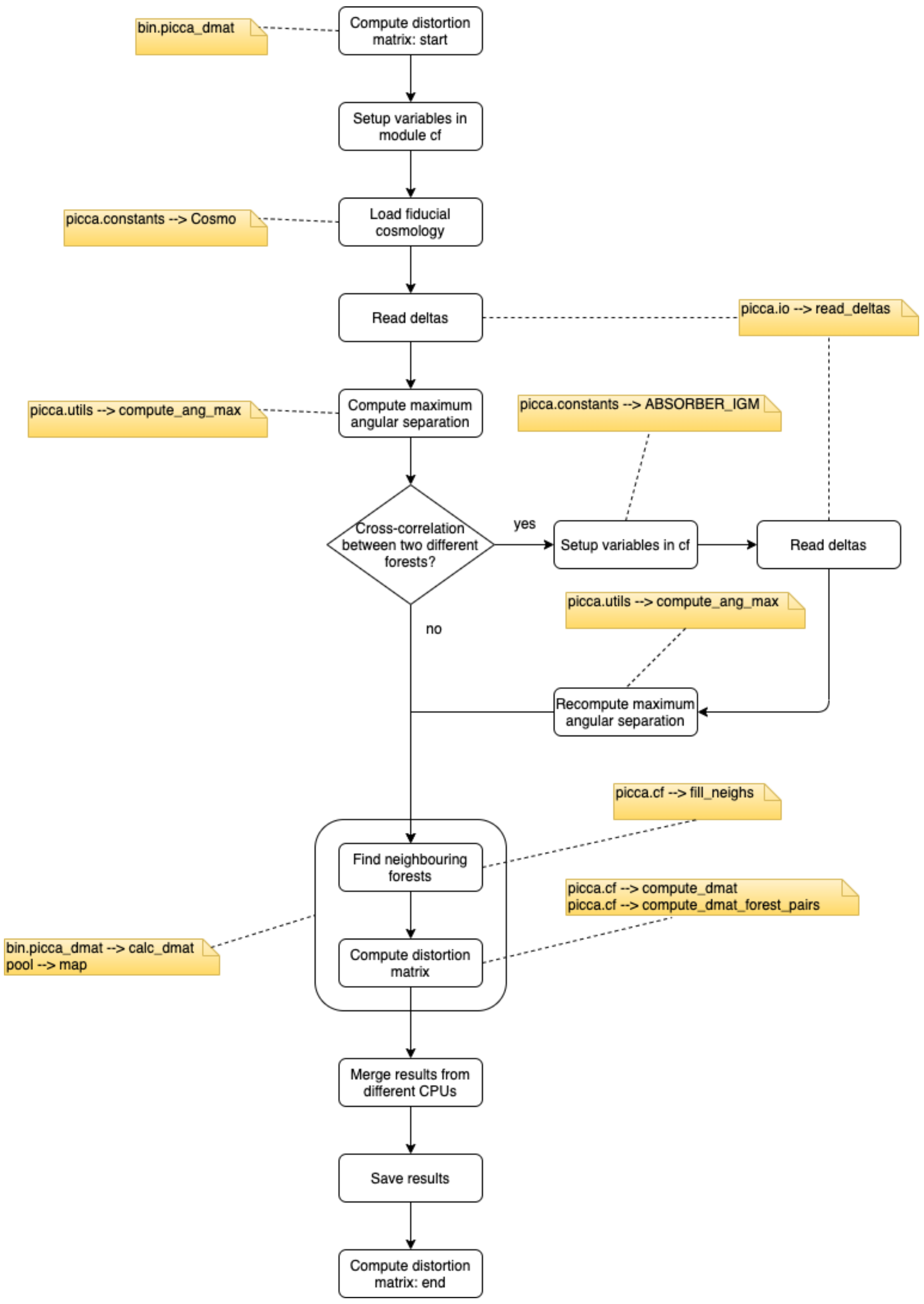
bin.picca_dmat
Compute distortion matrix: start
Setup variables in module cf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.utils --> compute_ang_max
Compute maximum angular separation
picca.constants --> ABSORBER_IGM
Cross-correlation between two different forests?
yes
Setup variables in cf
Read deltas
picca.utils --> compute_ang_max
no
Recompute maximum angular separation
picca.cf --> fill_neighs
Find neighbouring forests
picca.cf --> compute_dmat
picca.cf --> compute_dmat_forest_pairs
bin.picca_dmat --> calc_dmat
pool --> map
Compute distortion matrix
Merge results from different CPUs
Save results
Compute distortion matrix: end

Figure 7: Workflow for the computation of the distortion matrix for the auto-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

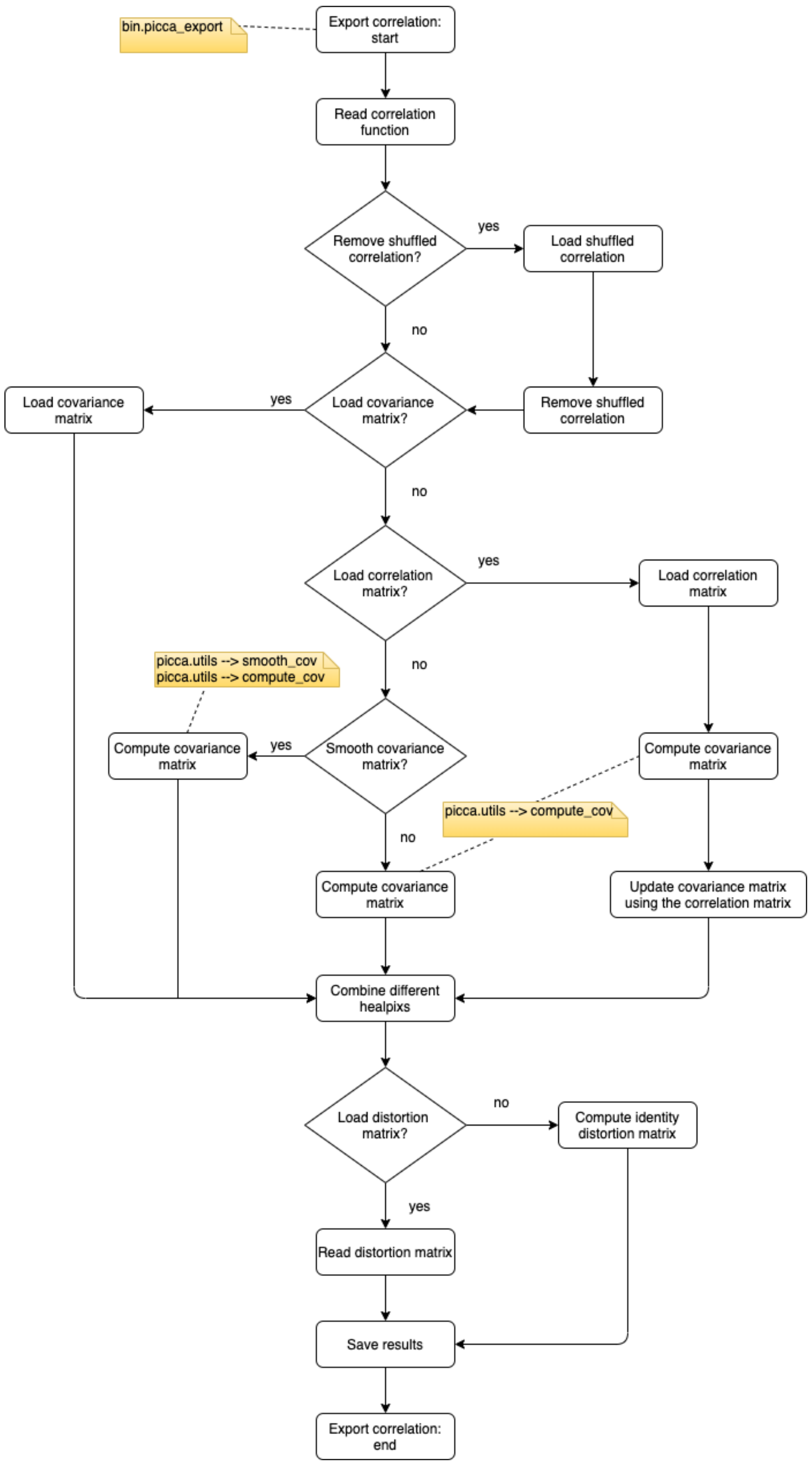
bin.picca_export
Export correlation: start
Read correlation function
Remove shuffled correlation?
yes
Load shuffled correlation
no
Remove shuffled correlation
Load covariance matrix?
yes
Load covariance matrix
no
Load correlation matrix?
yes
Load correlation matrix
no
picca.utils --> smooth_cov
picca.utils --> compute_cov
Smooth covariance matrix?
yes
Compute covariance matrix
Compute covariance matrix
picca.utils --> compute_cov
no
Compute covariance matrix
Update covariance matrix using the correlation matrix
Combine different healpixs
Load distortion matrix?
no
Compute identity distortion matrix
yes
Read distortion matrix
Save results
Export correlation: end

Figure 8: Workflow for the export of the correlation function (both auto- and cross-correlation). Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

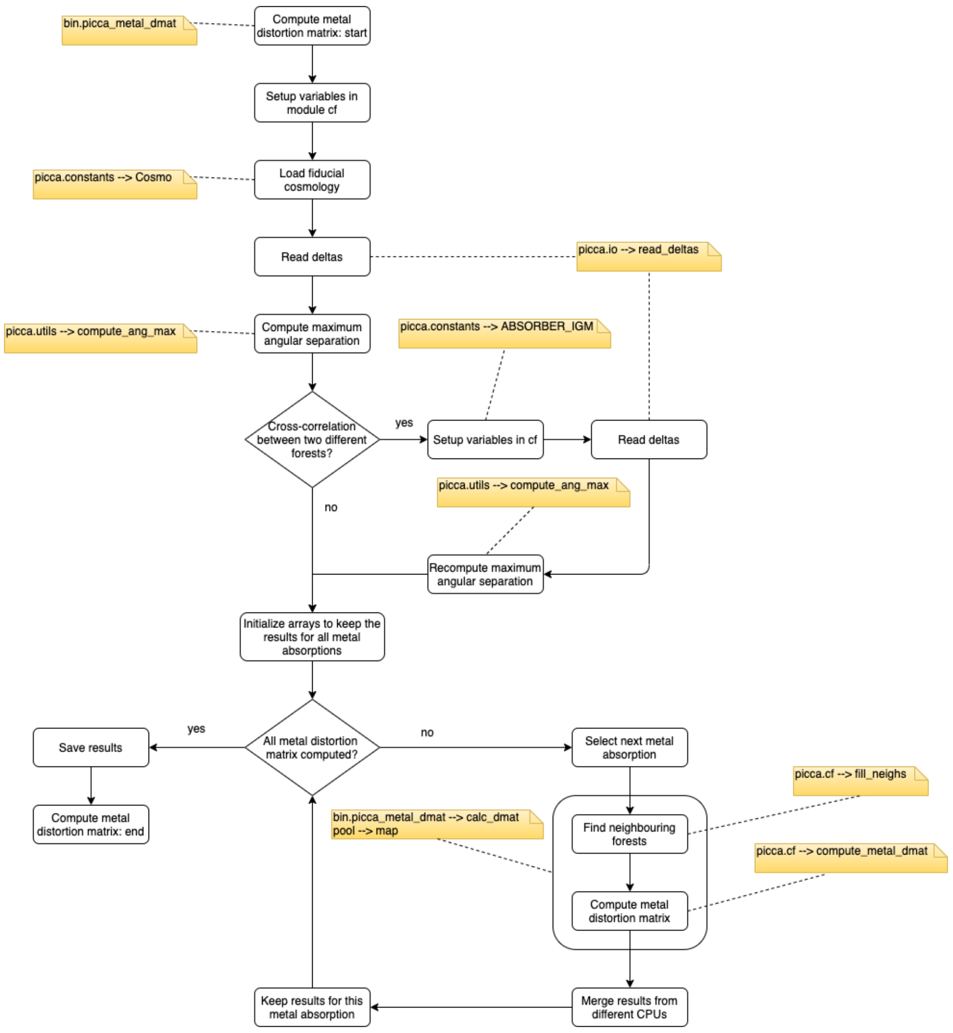
bin.picca_metal_dmat
Compute metal distortion matrix: start
Setup variables in module cf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.utils --> compute_ang_max
Compute maximum angular separation
picca.constants --> ABSORBER_IGM
Cross-correlation between two different forests?
yes
Setup variables in cf
Read deltas
picca.utils --> compute_ang_max
no
Recompute maximum angular separation
Initialize arrays to keep the results for all metal absorptions
Save results
yes
All metal distortion matrix computed?
no
Select next metal absorption
picca.cf --> fill_neighs
Compute metal distortion matrix: end
bin.picca_metal_dmat --> calc_dmat
pool --> map
Find neighbouring forests
picca.cf --> compute_metal_dmat
Compute metal distortion matrix
Keep results for this metal absorption
Merge results from different CPUs

Figure 9: Workflow for the computation of the distortion matrix for metal absorption and for the auto-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

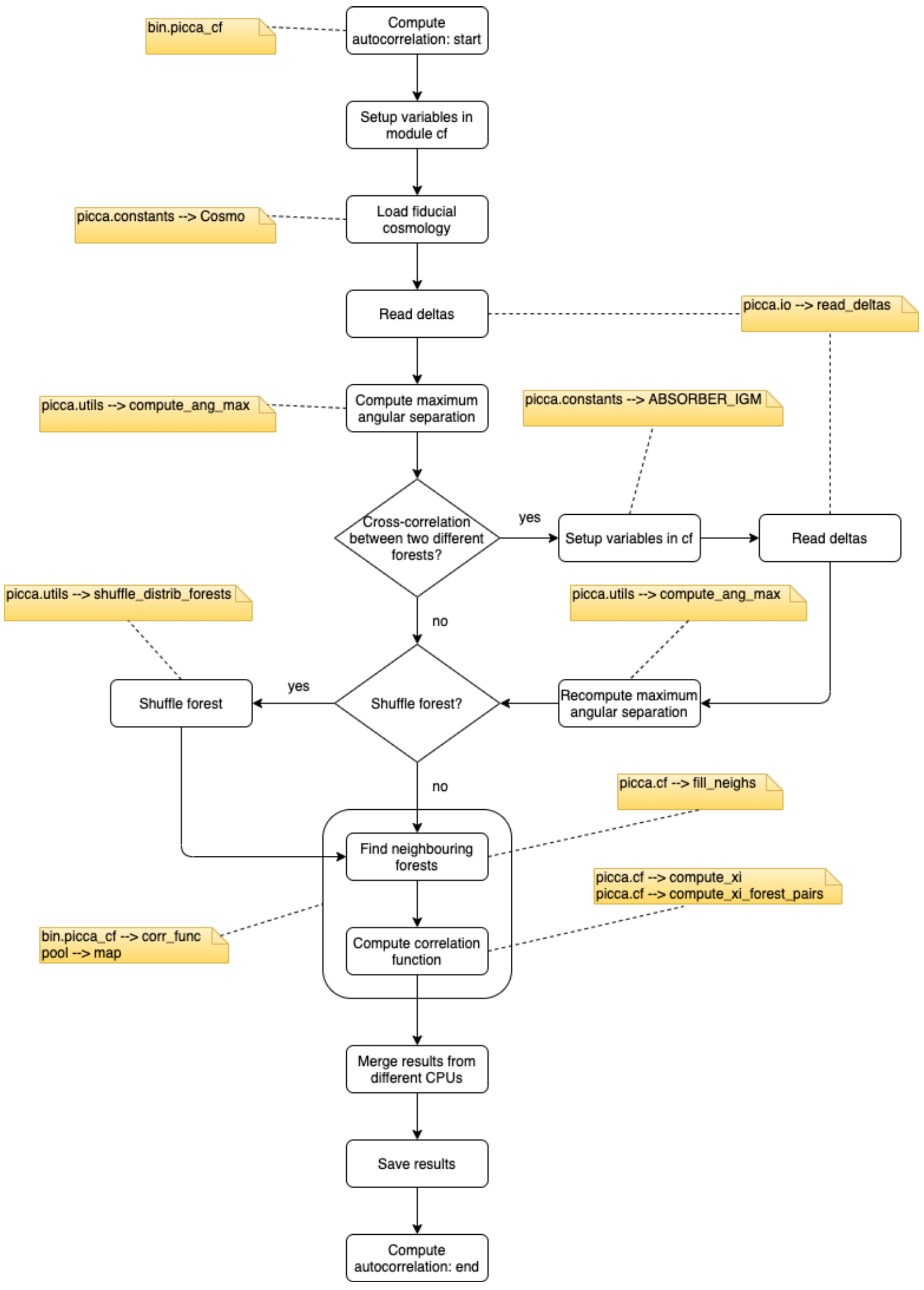
bin.picca_cf
Compute autocorrelation: start
Setup variables in module cf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.utils --> compute_ang_max
Compute maximum angular separation
picca.constants --> ABSORBER_IGM
Cross-correlation between two different forests?
yes
Setup variables in cf
Read deltas
no
picca.utils --> shuffle_distrib_forests
picca.utils --> compute_ang_max
Shuffle forest
yes
Shuffle forest?
Recompute maximum angular separation
no
picca.cf --> fill_neighs
Find neighbouring forests
picca.cf --> compute_xi
picca.cf --> compute_xi_forest_pairs
bin.picca_cf --> corr_func
pool --> map
Compute correlation function
Merge results from different CPUs
Save results
Compute autocorrelation: end

Figure 10: Workflow for the computation of the cross-correlation function. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

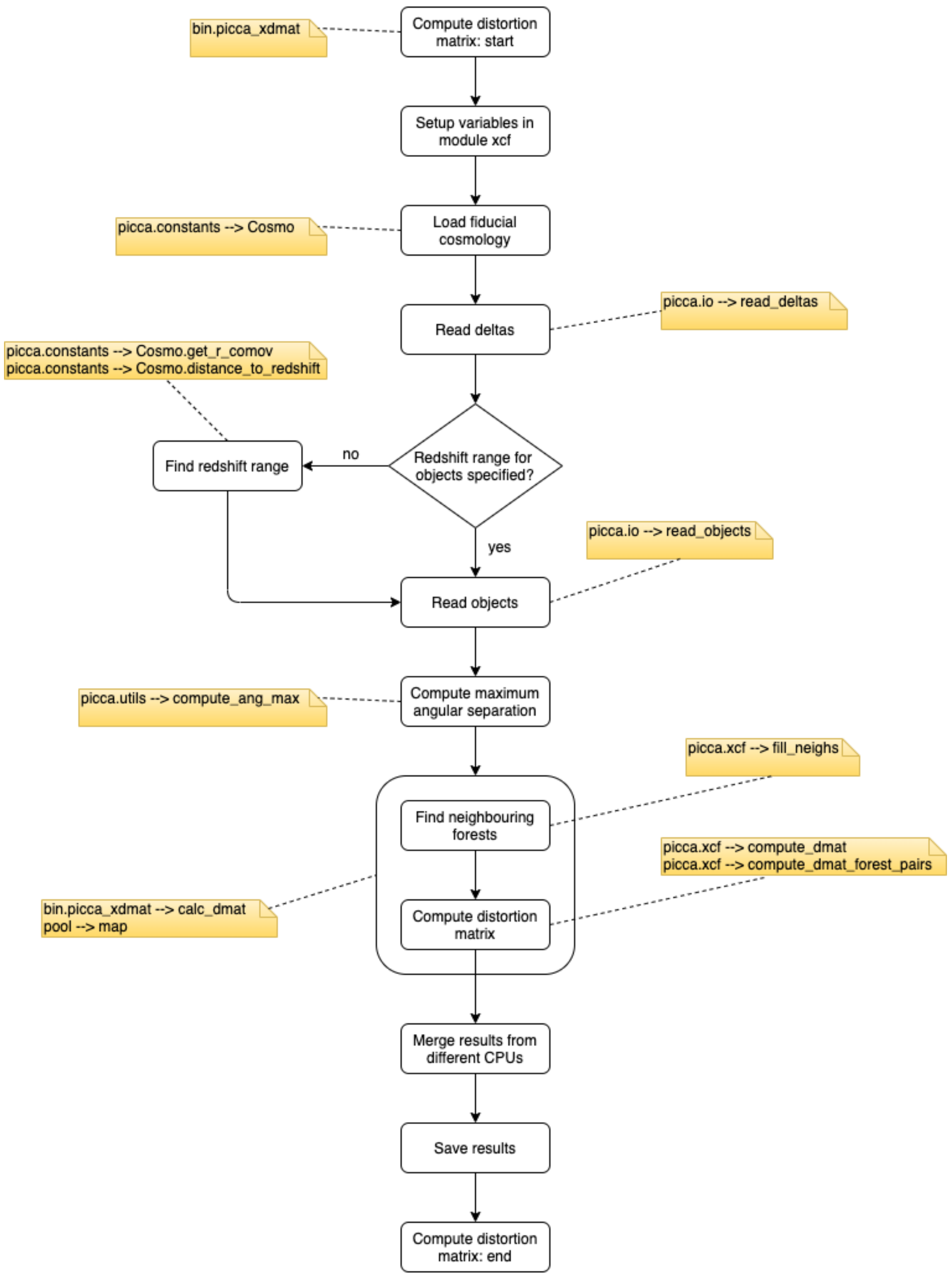
bin.picca_xdmat
Compute distortion matrix: start
Setup variables in module xcf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.constants --> Cosmo.get_r_comov
picca.constants --> Cosmo.distance_to_redshift
Find redshift range
no
Redshift range for objects specified?
yes
picca.io --> read_objects
Read objects
picca.utils --> compute_ang_max
Compute maximum angular separation
picca.xcf --> fill_neighs
Find neighbouring forests
picca.xcf --> compute_dmat
picca.xcf --> compute_dmat_forest_pairs
bin.picca_xdmat --> calc_dmat
pool --> map
Compute distortion matrix
Merge results from different CPUs
Save results
Compute distortion matrix: end

Figure 11: Workflow for the computation of the distortion matrix for the cross-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

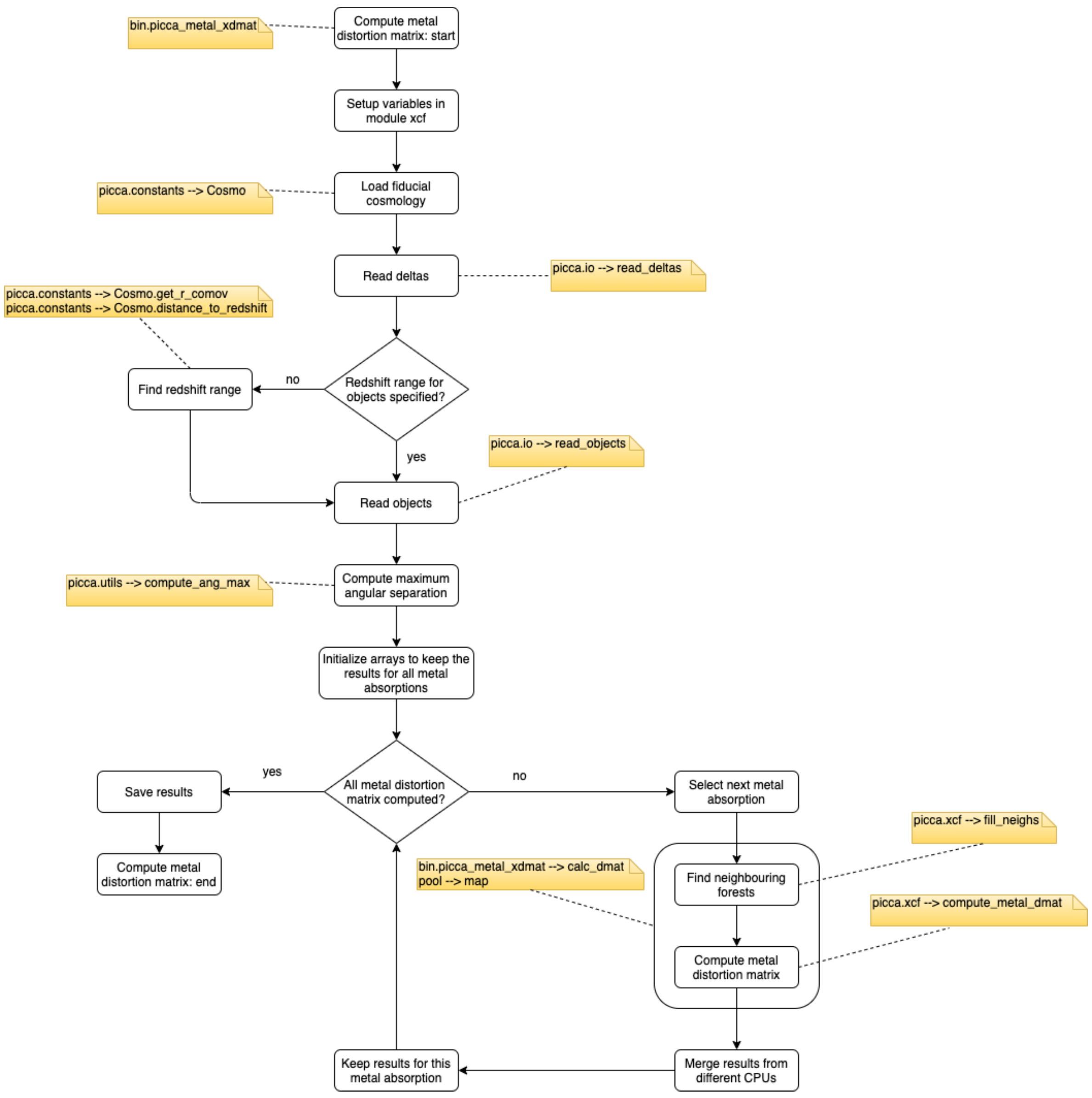
bin.picca_metal_xdmat
Compute metal distortion matrix: start
Setup variables in module xcf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.constants --> Cosmo.get_r_comov
picca.constants --> Cosmo.distance_to_redshift
Find redshift range
no
Redshift range for objects specified?
yes
picca.io --> read_objects
Read objects
picca.utils --> compute_ang_max
Compute maximum angular separation
Initialize arrays to keep the results for all metal absorptions
Save results
yes
All metal distortion matrix computed?
no
Select next metal absorption
picca.xcf --> fill_neighs
Compute metal distortion matrix: end
bin.picca_metal_xdmat --> calc_dmat
pool --> map
Find neighbouring forests
picca.xcf --> compute_metal_dmat
Compute metal distortion matrix
Keep results for this metal absorption
Merge results from different CPUs

Figure 12: Workflow for the computation of the distortion matrix for metal absorption and for the cross-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

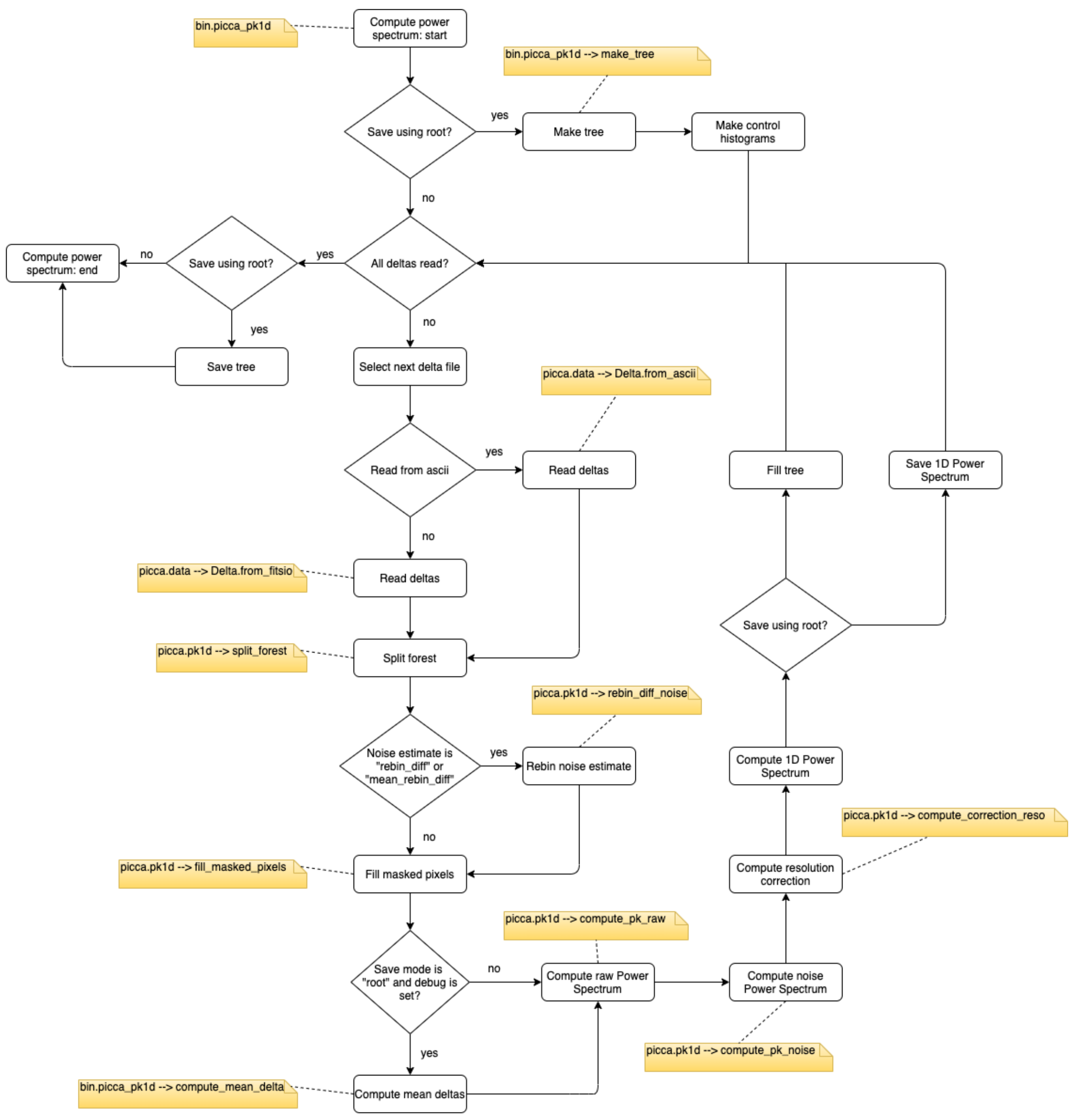
bin.picca_pk1d
Compute power spectrum: start
bin.picca_pk1d --> make_tree
Save using root?
yes
Make tree
Make control histograms
no
Compute power spectrum: end
no
Save using root?
yes
All deltas read?
yes
no
Save tree
Select next delta file
picca.data --> Delta.from_ascii
Read from ascii
yes
Read deltas
Fill tree
Save 1D Power Spectrum
no
picca.data --> Delta.from_fitsio
Read deltas
Save using root?
picca.pk1d --> split_forest
Split forest
picca.pk1d --> rebin_diff_noise
Noise estimate is "rebin_diff" or "mean_rebin_diff"
yes
Rebin noise estimate
Compute 1D Power Spectrum
picca.pk1d --> compute_correction_reso
no
picca.pk1d --> fill_masked_pixels
Fill masked pixels
Compute resolution correction
picca.pk1d --> compute_pk_raw
Save mode is "root" and debug is set?
no
Compute raw Power Spectrum
Compute noise Power Spectrum
yes
picca.pk1d --> compute_pk_noise
bin.picca_pk1d --> compute_mean_delta
Compute mean deltas

Figure 13: Workflow for the computation of the 1D Power Spectrum. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

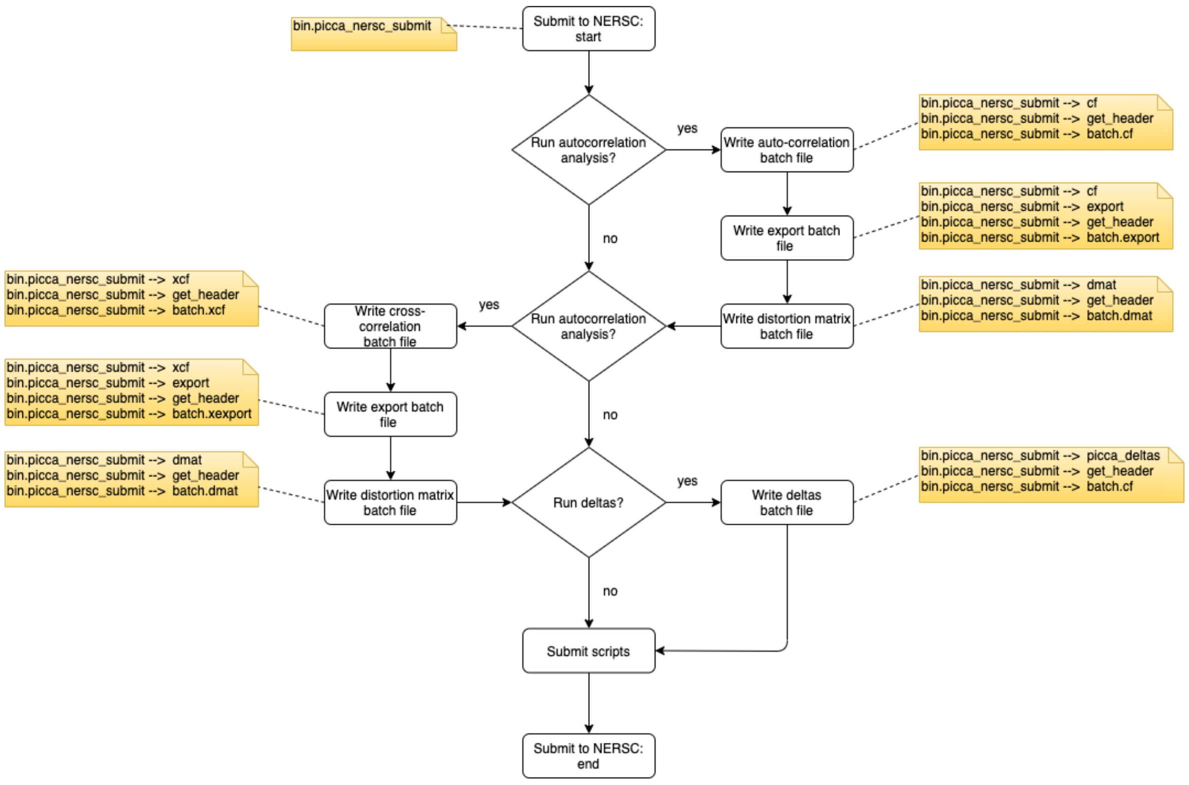
bin.picca_nersc_submit
Submit to NERSC: start
Run autocorrelation analysis?
yes
Write auto-correlation batch file
bin.picca_nersc_submit --> cf
bin.picca_nersc_submit --> get_header
bin.picca_nersc_submit --> batch.cf
no
Write export batch file
bin.picca_nersc_submit --> cf
bin.picca_nersc_submit --> export
bin.picca_nersc_submit --> get_header
bin.picca_nersc_submit --> batch.export
Write distortion matrix batch file
bin.picca_nersc_submit --> dmat
bin.picca_nersc_submit --> get_header
bin.picca_nersc_submit --> batch.dmat
Run autocorrelation analysis?
yes
Write cross-correlation batch file
bin.picca_nersc_submit --> xcf
bin.picca_nersc_submit --> get_header
bin.picca_nersc_submit --> batch.xcf
Write export batch file
bin.picca_nersc_submit --> xcf
bin.picca_nersc_submit --> export
bin.picca_nersc_submit --> get_header
bin.picca_nersc_submit --> batch.xexport
Write distortion matrix batch file
bin.picca_nersc_submit --> dmat
bin.picca_nersc_submit --> get_header
bin.picca_nersc_submit --> batch.dmat
no
Run deltas?
yes
Write deltas batch file
bin.picca_nersc_submit --> picca_deltas
bin.picca_nersc_submit --> get_header
bin.picca_nersc_submit --> batch.cf
no
Submit scripts
Submit to NERSC: end

Figure 14: Workflow for the submission of the main 3D analysis to NERSC. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

have (roughly) the right level of clustering, we can to compute their correlation function. This implies using the BAO galaxy clustering technique. `PICCA` contains a script for such an analysis. This analysis can be broken in two steps. First, we compute the correlation function for the different healpixs. This step needs to be run 3 times: once to correlate data with data, once to correlate data with a random catalogue, and once to correlate the random catalogue with the random catalogue. Then we combine them into the final correlation function in the export correlation function step. The workflow for the first and second steps is given in Figures 15 and 16.

2.5.3 Angular correlation functions

When computing the auto- and the cross-correlations (see Sections 2.2 and 2.3) one of the main steps is the conversion from angles and redshifts to distances using a fiducial cosmological model. Apart from the main analysis, `PICCA` also offers the possibility to compute the correlation function directly from the angles. The workflow for the computation of the angular auto-correlation is given in Figure 17, and for the cross-correlation in Figure 18. We can see that the workflows are very similar than those of the auto- and cross-correlations (Figures 6 and 10, respectively)

2.5.4 1D correlation functions

As we have seen in Sections 2.2 and 2.3 one of the steps to compute the correlation function is the computation of the metal distortion matrix. This step is necessary because we cannot distinguish the absorption from hydrogen from the absorption from other metals at a different distance. To determine which metals are contaminating our hydrogen absorption, we can compute the 1-dimensional correlation functions. The signal from the contaminant metals will be present as additional peaks in the 1D correlation function. Naturally, `PICCA` allows the user to compute such correlations. The workflow for the computation of the angular auto-correlation is given in Figure 19, and for the cross-correlation in Figure 20.

2.5.5 Wick covariance matrix

In the main 3D analysis, the covariance matrix is computed using the subsampling technique when exporting the correlation function. This is an approximation to the full covariance matrix, the computation of which is very expensive. Nevertheless, the subsampling covariance matrix was tested against the full computation using the Wick expansion in Delubac et al. (2015). `PICCA` offers the option of computing the covariance matrix using the wick expansion. This is shown in Figure 1 as a possible extension of the main 3D analysis. Note that this extension makes use of the 1d correlation functions described in Section 2.5.4. We show the workflow for the computation of the Wick covariance matrices in Figures 21 and 22. Note that the computation of the Wick covariance matrices requires the measurement of the 1D correlation functions (see Section 2.5.4).

2.5.6 Export cross-covariance

The fitter offers the option to perform joined fits of several correlation functions. However, these fits are performed assuming that the different correlation functions are independent. To

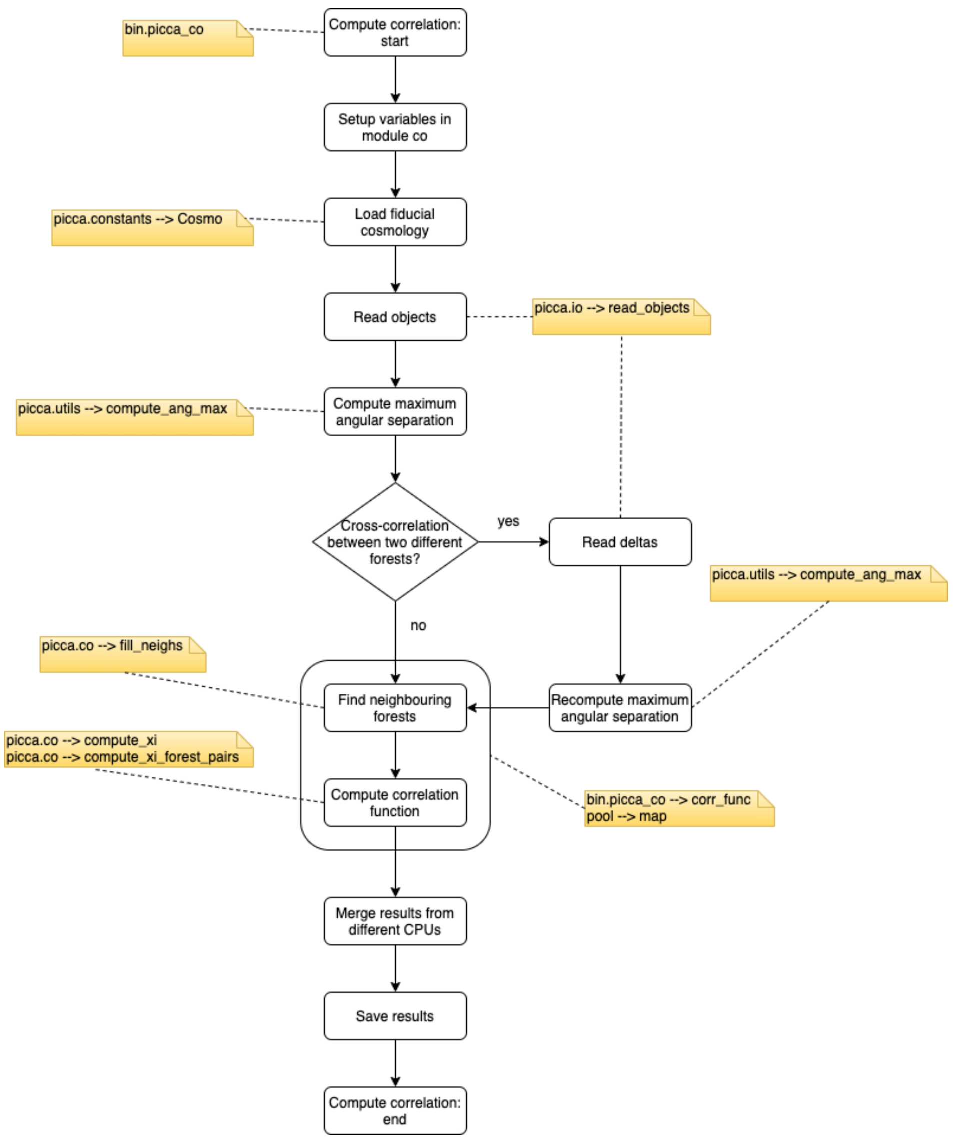
bin.picca_co
Compute correlation: start
Setup variables in module co
picca.constants --> Cosmo
Load fiducial cosmology
Read objects
picca.io --> read_objects
picca.utils --> compute_ang_max
Compute maximum angular separation
Cross-correlation between two different forests?
yes
Read deltas
picca.utils --> compute_ang_max
no
picca.co --> fill_neighs
Find neighbouring forests
Recompute maximum angular separation
picca.co --> compute_xi
picca.co --> compute_xi_forest_pairs
Compute correlation function
bin.picca_co --> corr_func
pool --> map
Merge results from different CPUs
Save results
Compute correlation: end

Figure 15: Workflow for the computation of the correlation between two object catalogues. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

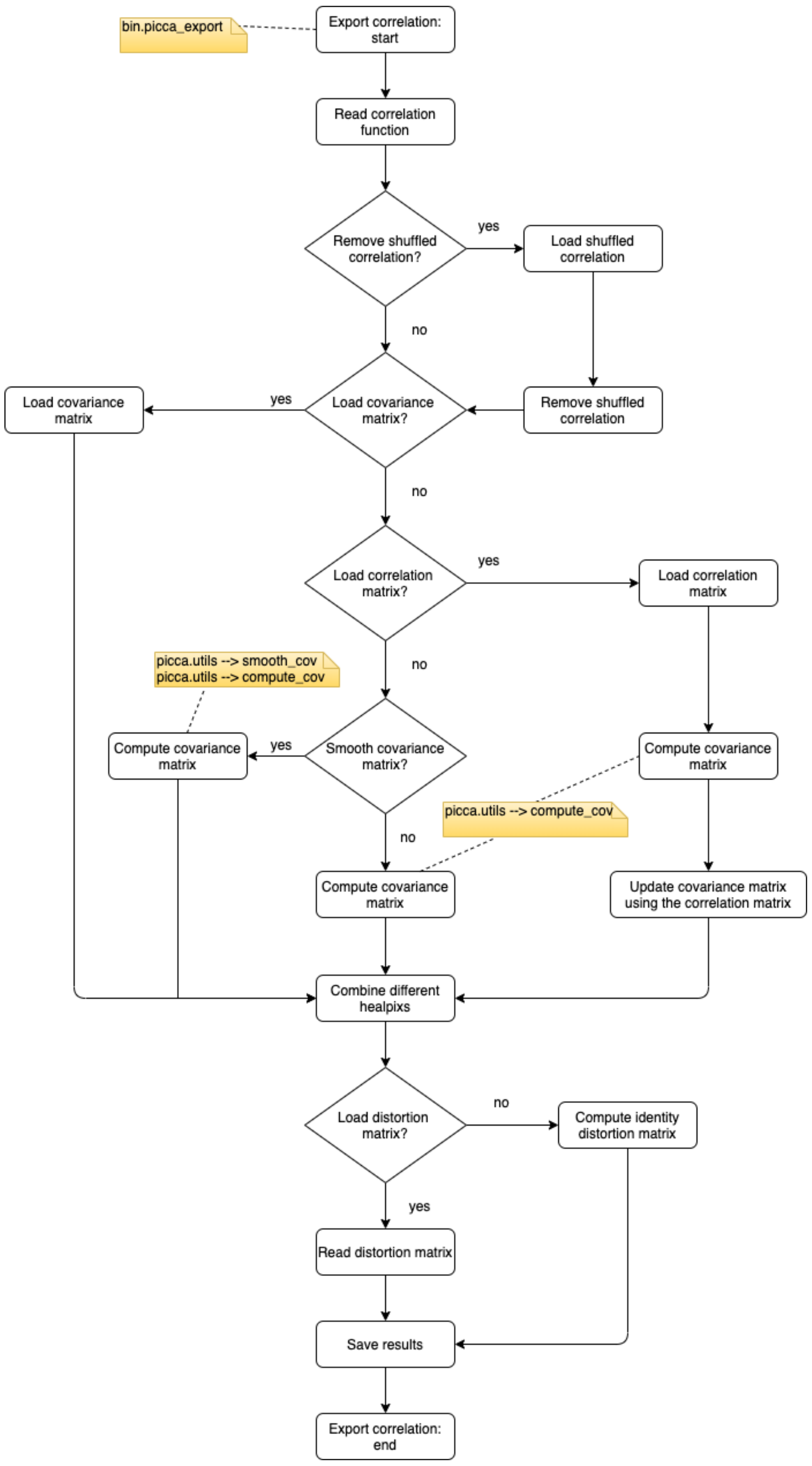
bin.picca_export
Export correlation: start
Read correlation function
Remove shuffled correlation?
yes
Load shuffled correlation
no
Load covariance matrix
yes
Load covariance matrix?
Remove shuffled correlation
no
Load correlation matrix?
yes
Load correlation matrix
picca.utils --> smooth_cov
picca.utils --> compute_cov
no
Compute covariance matrix
yes
Smooth covariance matrix?
Compute covariance matrix
picca.utils --> compute_cov
no
Compute covariance matrix
Update covariance matrix using the correlation matrix
Combine different healpixs
Load distortion matrix?
no
Compute identity distortion matrix
yes
Read distortion matrix
Save results
Export correlation: end

Figure 16: Workflow for the export of the correlation function between to object catalogues. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

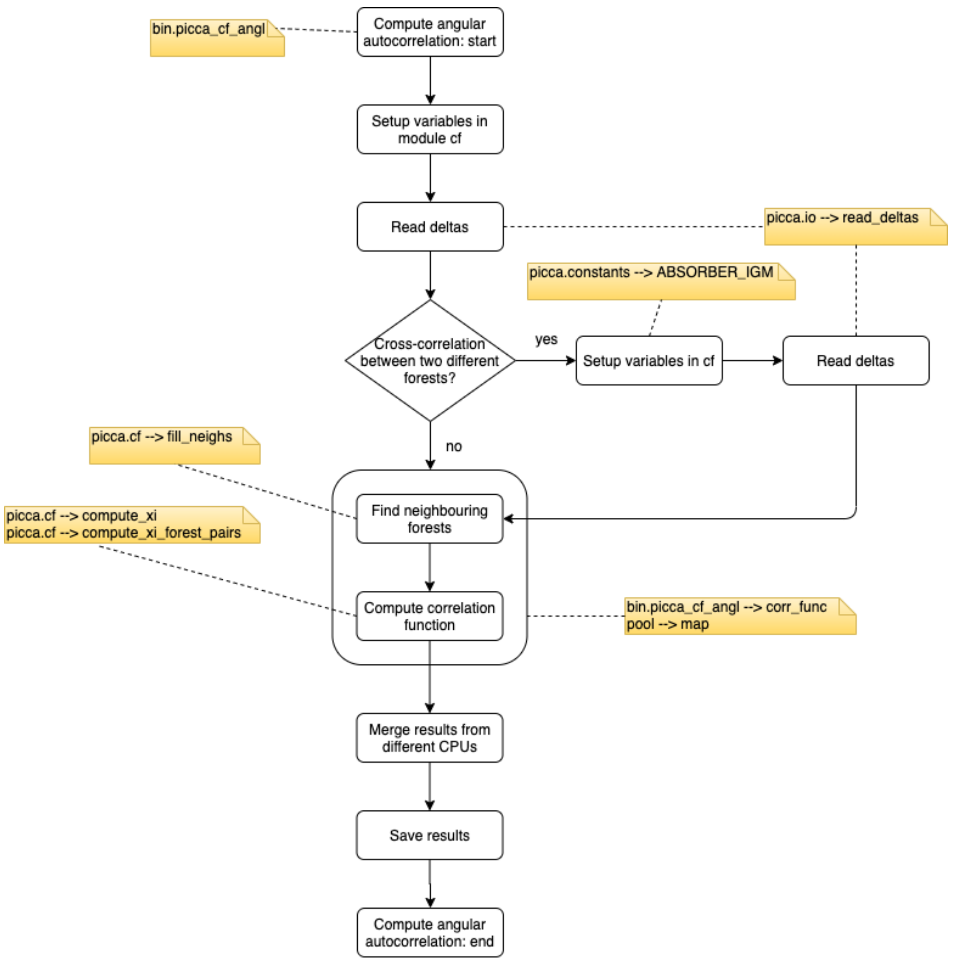
bin.picca_cf_angl
Compute angular autocorrelation: start
Setup variables in module cf
Read deltas
picca.io --> read_deltas
picca.constants --> ABSORBER_IGM
Cross-correlation between two different forests?
yes
Setup variables in cf
Read deltas
no
picca.cf --> fill_neighs
Find neighbouring forests
picca.cf --> compute_xi
picca.cf --> compute_xi_forest_pairs
Compute correlation function
bin.picca_cf_angl --> corr_func
pool --> map
Merge results from different CPUs
Save results
Compute angular autocorrelation: end

Figure 17: Workflow for the computation of the angular auto-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

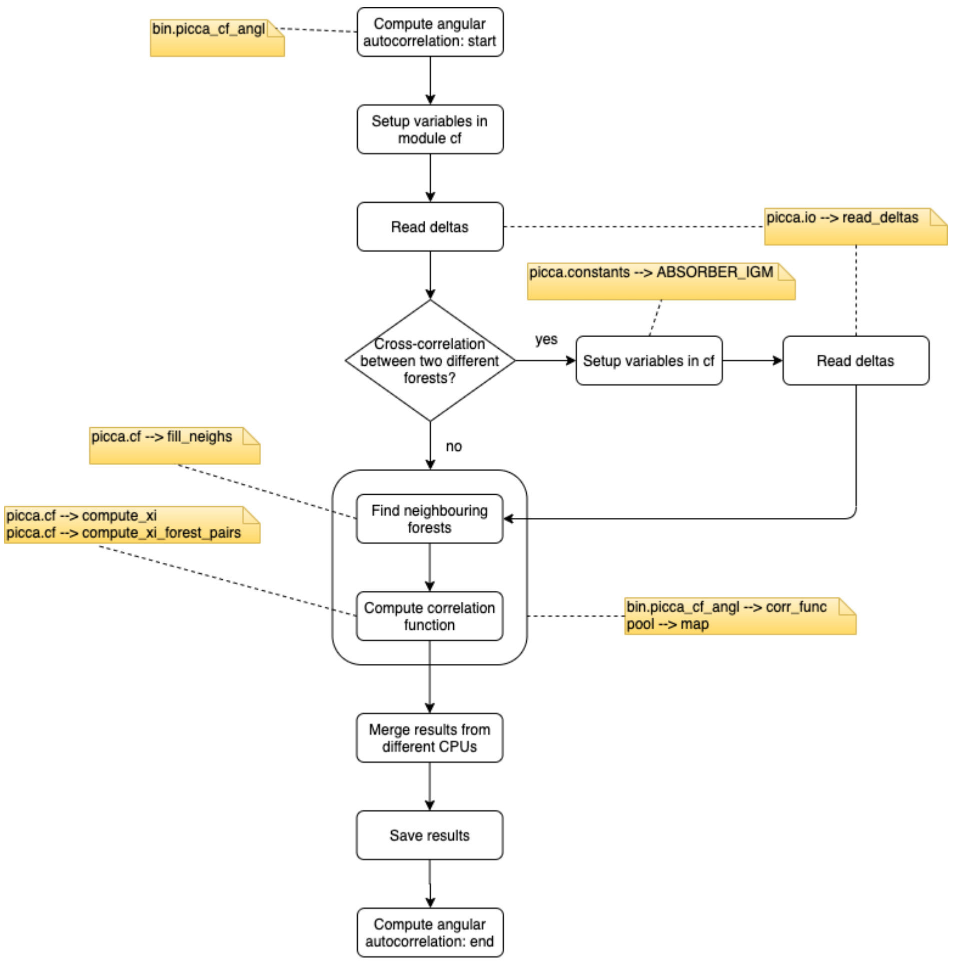
bin.picca_cf_angl
Compute angular autocorrelation: start
Setup variables in module cf
Read deltas
picca.io --> read_deltas
picca.constants --> ABSORBER_IGM
Cross-correlation between two different forests?
yes
Setup variables in cf
Read deltas
no
picca.cf --> fill_neighs
Find neighbouring forests
picca.cf --> compute_xi
picca.cf --> compute_xi_forest_pairs
Compute correlation function
bin.picca_cf_angl --> corr_func
pool --> map
Merge results from different CPUs
Save results
Compute angular autocorrelation: end

Figure 18: Workflow for the computation of the angular cross-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

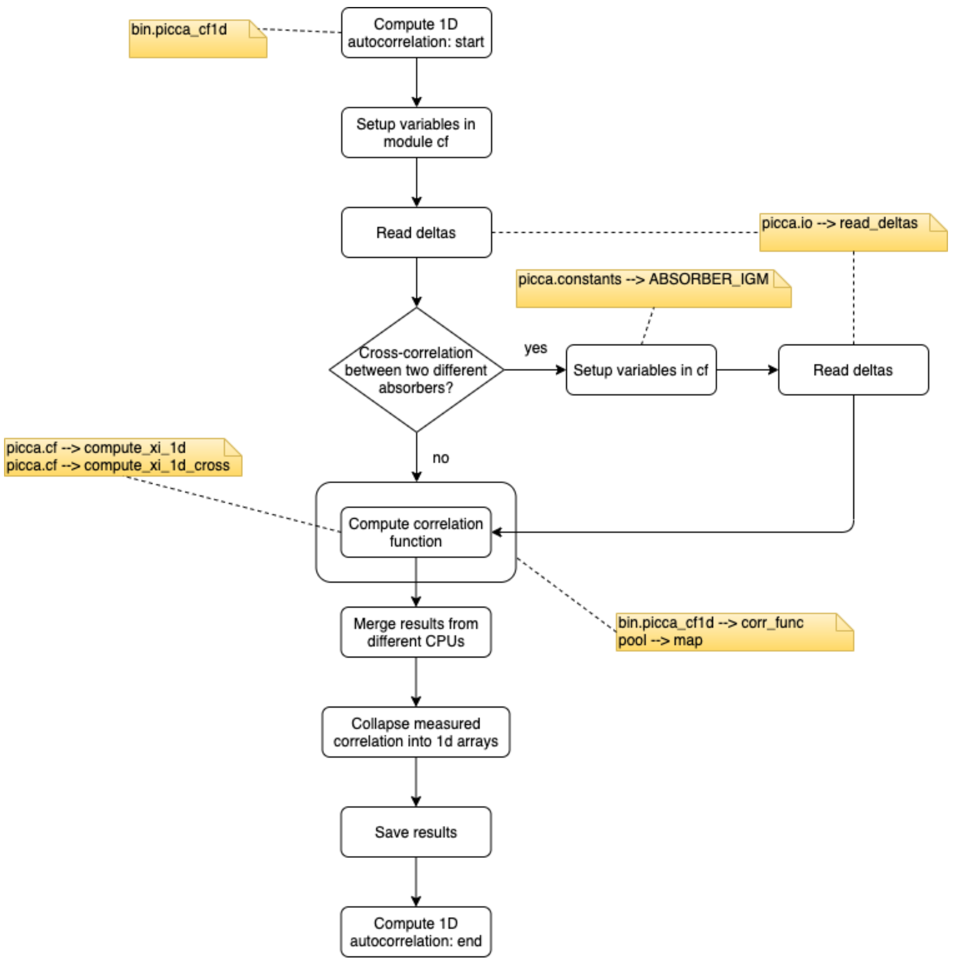
bin.picca_cf1d
Compute 1D autocorrelation: start
Setup variables in module cf
Read deltas
picca.io --> read_deltas
picca.constants --> ABSORBER_IGM
Cross-correlation between two different absorbers?
yes
Setup variables in cf
Read deltas
no
picca.cf --> compute_xi_1d
picca.cf --> compute_xi_1d_cross
Compute correlation function
bin.picca_cf1d --> corr_func
pool --> map
Merge results from different CPUs
Collapse measured correlation into 1d arrays
Save results
Compute 1D autocorrelation: end

Figure 19: Workflow for the computation of the 1D auto-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

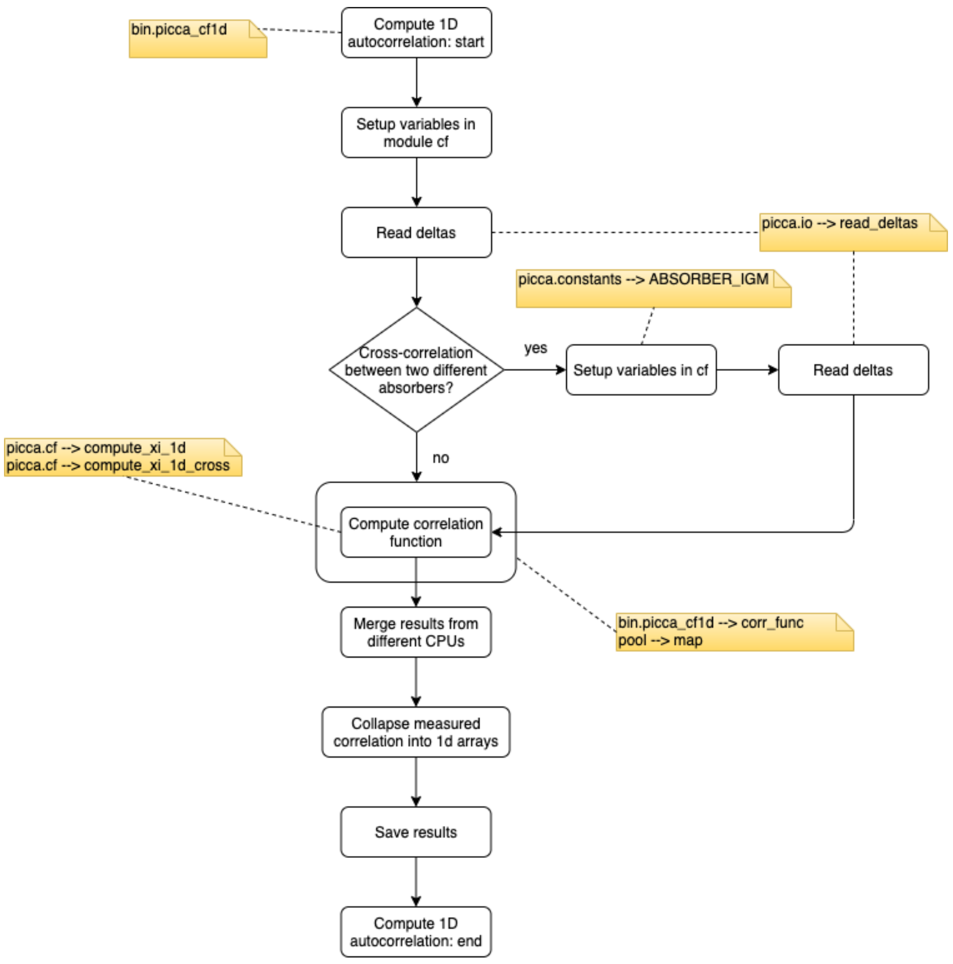
bin.picca_cf1d
Compute 1D autocorrelation: start
Setup variables in module cf
Read deltas
picca.io --> read_deltas
picca.constants --> ABSORBER_IGM
Cross-correlation between two different absorbers?
yes
Setup variables in cf
Read deltas
no
picca.cf --> compute_xi_1d
picca.cf --> compute_xi_1d_cross
Compute correlation function
bin.picca_cf1d --> corr_func
pool --> map
Merge results from different CPUs
Collapse measured correlation into 1d arrays
Save results
Compute 1D autocorrelation: end

Figure 20: Workflow for the computation of the 1D cross-correlation. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

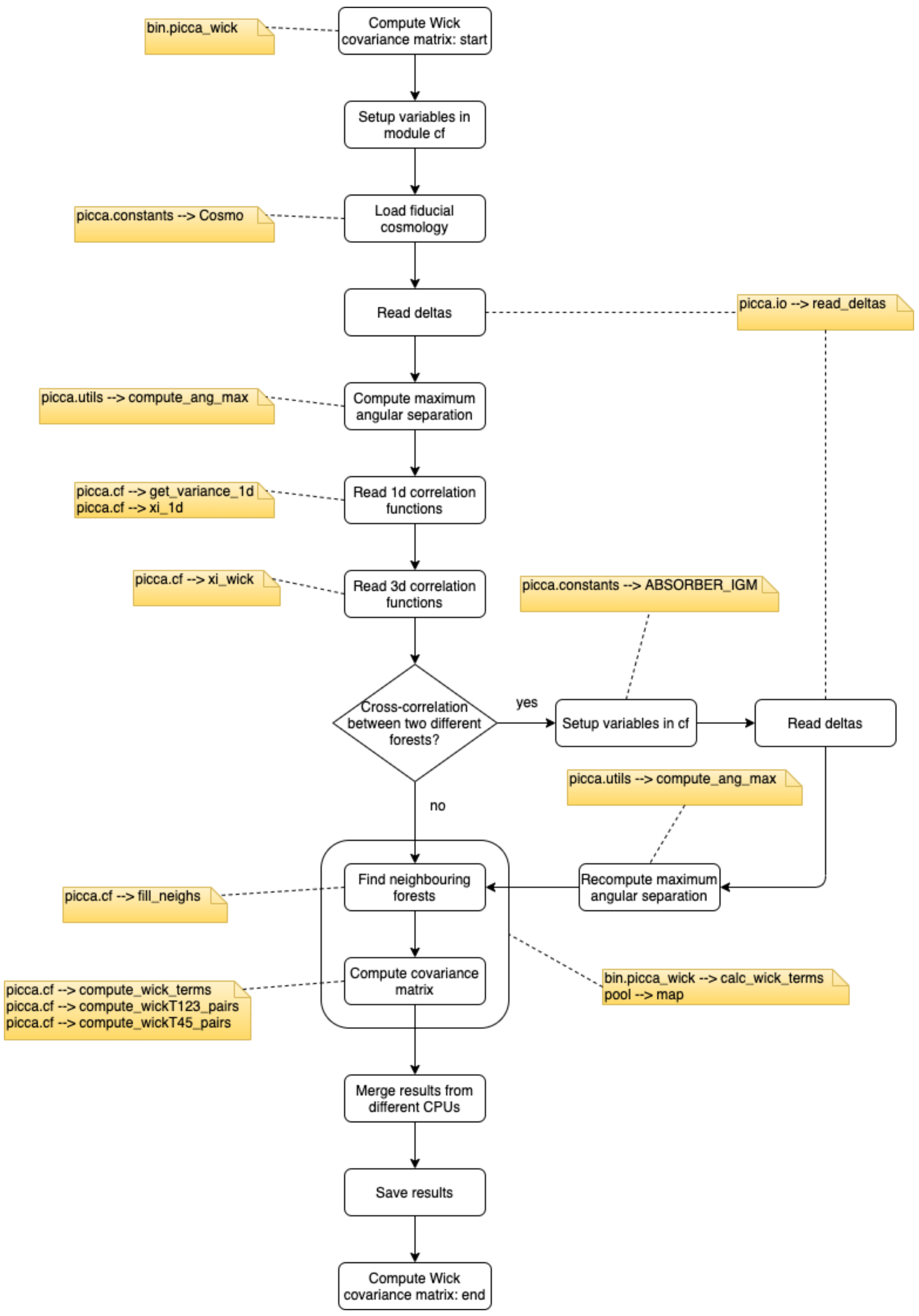
bin.picca_wick
Compute Wick covariance matrix: start
Setup variables in module cf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.utils --> compute_ang_max
Compute maximum angular separation
picca.cf --> get_variance_1d
picca.cf --> xi_1d
Read 1d correlation functions
picca.cf --> xi_wick
Read 3d correlation functions
picca.constants --> ABSORBER_IGM
Cross-correlation between two different forests?
yes
Setup variables in cf
Read deltas
picca.utils --> compute_ang_max
no
picca.cf --> fill_neighs
Find neighbouring forests
Recompute maximum angular separation
picca.cf --> compute_wick_terms
picca.cf --> compute_wickT123_pairs
picca.cf --> compute_wickT45_pairs
Compute covariance matrix
bin.picca_wick --> calc_wick_terms
pool --> map
Merge results from different CPUs
Save results
Compute Wick covariance matrix: end

Figure 21: Workflow for the computation of the Wick covariance matrix for the auto-correlation analysis. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

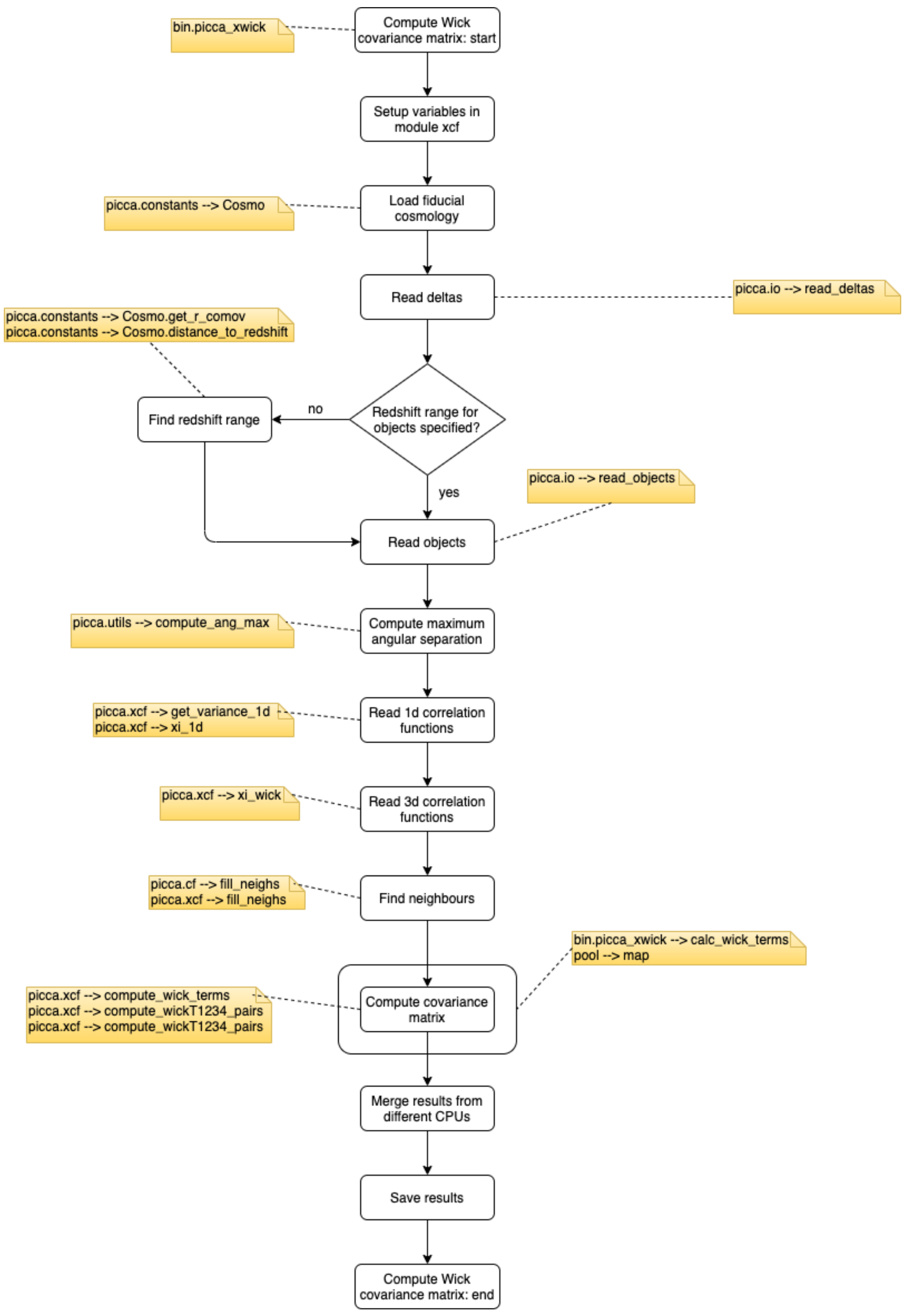
bin.picca_xwick
Compute Wick covariance matrix: start
Setup variables in module xcf
picca.constants --> Cosmo
Load fiducial cosmology
Read deltas
picca.io --> read_deltas
picca.constants --> Cosmo.get_r_comov
picca.constants --> Cosmo.distance_to_redshift
Find redshift range
no
Redshift range for objects specified?
yes
picca.io --> read_objects
Read objects
picca.utils --> compute_ang_max
Compute maximum angular separation
picca.xcf --> get_variance_1d
picca.xcf --> xi_1d
Read 1d correlation functions
picca.xcf --> xi_wick
Read 3d correlation functions
picca.cf --> fill_neighs
picca.xcf --> fill_neighs
Find neighbours
bin.picca_xwick --> calc_wick_terms
pool --> map
picca.xcf --> compute_wick_terms
picca.xcf --> compute_wickT1234_pairs
picca.xcf --> compute_wickT1234_pairs
Compute covariance matrix
Merge results from different CPUs
Save results
Compute Wick covariance matrix: end

Figure 22: Workflow for the computation of the Wick covariance matrix for the cross-correlation analysis. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

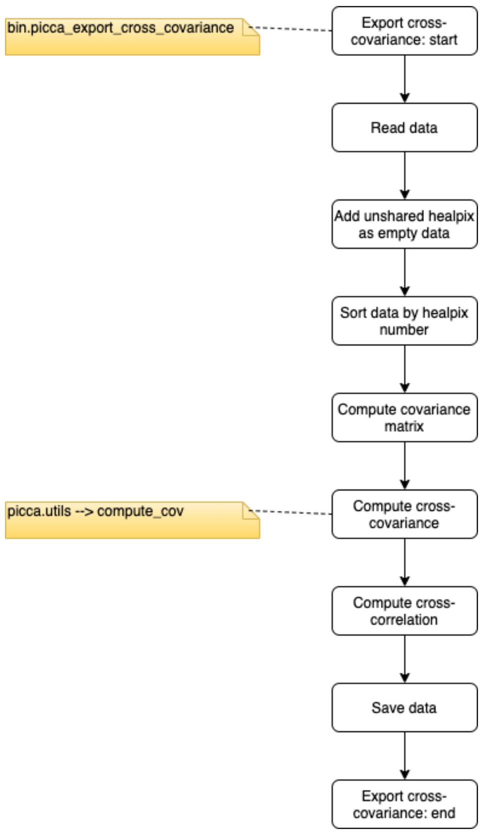
bin.picca_export_cross_covariance
Export cross-covariance: start
Read data
Add unshared healpix as empty data
Sort data by healpix number
Compute covariance matrix
picca.utils --> compute_cov
Compute cross-covariance
Compute cross-correlation
Save data
Export cross-covariance: end

Figure 23: Workflow for the computation of the cross-covariance between two correlation functions. Yellow boxes indicate the scripts of `PICCA` used in each step, and steps without explicit labelling occur in the main function.

measure the validity of this statement, it is necessary to measure the cross-covariance between the correlation functions, or the degree to which they are independent. `PICCA` allows the user to compute this cross-covariance between two samples. The workflow for this step is given in Figure 23.

2.5.7 Coadd correlation functions

As more is collected, it is envisage to compute the correlation function in different redshift bins. However, it might also be interesting to compute the full correlation functions, combining the information from all redshifts. Since this is a costly thing, a script was created in `PICCA` to coadd the correlation function from different redshift intervals. Its workflow is given in Figure 24. Note that this script also exports the resulting correlation function.

2.5.8 Plotting

Finally, `PICCA` has some code to help plotting the results. There are basically two pieces. For the main 1D analysis, we have the module `bin.picca_plot_pk1d`. This is a full script to produce the plots, and its workflow is given in Figure 25. On the other hand, for the main 3D analysis we have the module `picca.wedgize`. This is not a full script but a simple class

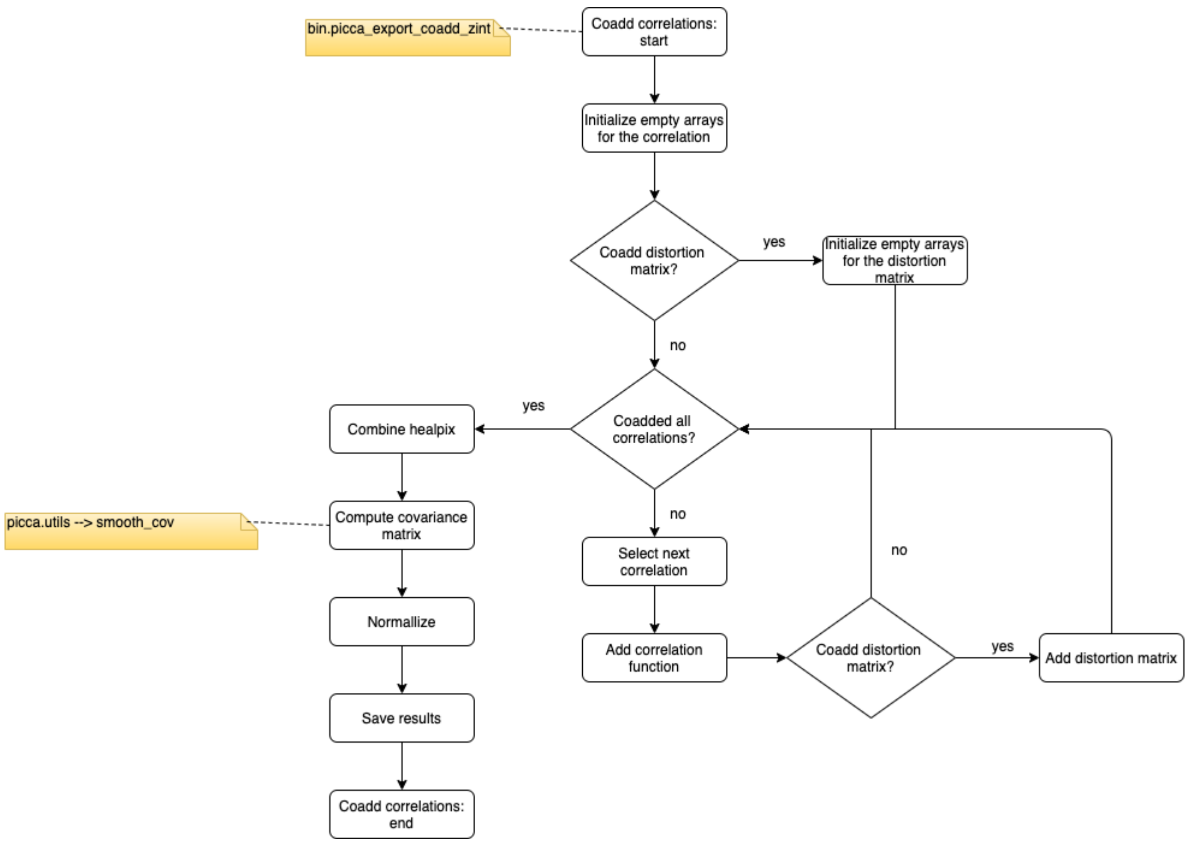
bin.picca_export_coadd_zint
Coadd correlations: start
Initialize empty arrays for the correlation
Coadd distortion matrix?
yes
Initialize empty arrays for the distortion matrix
no
Coadded all correlations?
yes
Combine healpix
picca.utils --> smooth_cov
Compute covariance matrix
Normallize
Save results
Coadd correlations: end
no
Select next correlation
Add correlation function
Coadd distortion matrix?
no
yes
Add distortion matrix

Figure 24: Workflow for the computation of the coaddition of several correlation functions. Yellow boxes indicate the scripts of PICCA used in each step, and steps without explicit labelling occur in the main function.

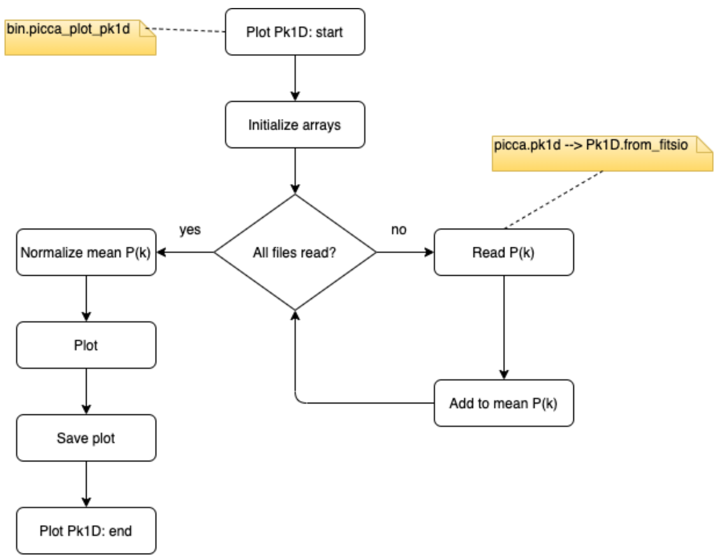

Figure 25: Workflow for the plotting of the 1D Power Spectrum. Yellow boxes indicate the scripts of `PICCA` used in each step, and steps without explicit labelling occur in the main function.

designed to help with the split of the correlation function in wedges according to the angular separation from the line of sight.

3. Standardisation of PICCA

In this section we perform an analysis of the current status of `PICCA`. `PICCA` uses a version control system based on `git`[6]. As such, we can freeze the version of `PICCA` by creating a new branch called `standardisation`. All analysis results will be based on this branch of `PICCA` unless otherwise stated. We use the Google Python Style Guide[7] as our standard for python. To ensure the format of the files we apply the `yapf` formatter with the option `--style google`

We perform an initial superficial analysis of the code to qualitatively assess possible defects on the code. We see that the docstrings are incomplete or missing for most of the functions and files. The variable naming clearly does not follow the coding standards, and variable names are obscure. This makes significantly impacts readability and testability of the code. To make this analysis more quantitative, we run `pylint`[8]. We obtain an initial score of -5.54/10 for the whole package. The output also reveals that the indentation of continued lines is not always correct. The level of standardisation of `PICCA` is, then, very low.

Similarly to the structure we followed in Section 2, we split the analysis into smaller chunks to make it more manageable. We start by analysing the parts constituting the main analysis, and we then analyze the alternative workflows. We note that since `PICCA` uses object oriented programming, it will not always be possible to completely disassociate the different blocs.

[6] `https://git-scm.com/`
[7] `http://google.github.io/styleguide/pyguide.html`
[8] `https://www.pylint.org/`

bin.picca_cf_ang	bin.picca_cf
bin.picca_cf1d	bin.picca_dmat
bin.picca_export_co	bin.picca_export_coadd_zint
bin.picca_export	bin.picca_fit
bin.picca_mcmc	bin.picca_metal_dmat
bin.picca_metal_xdmat	bin.picca_Pk1D
bin.picca_wick	bin.picca_xcf_angl
bin.picca_xcf	bin.picca_xdmat
bin.picca_xwick	picca.cf
picca.co	picca.data
picca.io	picca.Pk1D
picca.utils	picca.xcf
bin.picca_co	bin.picca_compute_fvoight
bin.picca_compute_pk_pksb	bin.picca_export_cross_covariance
bin.picca_plot_pk1d	bin.picca_xcf1d
picca.fitter.broadband_cross	picca.fitter.Chi2
picca.fitter.cosmo	picca.fitter.fftlog
picca.fitter.metals	picca.fitter.parameters
picca.fitter2.chi2	picca.fitter2.data
picca.fitter2.effective_bins	picca.fitter2.parser
picca.fitter2.xi	picca.test.test_cor

Table 1: Top bloc: modules affected by the change of function `print` from `picca.utils` to avoid redefining the built-in `print` function. Bottom bloc: modules that used the function `print` that have been modified to adopt the printing function defined in `picca.utils` for consistency in the package.

3.1 Compute deltas

The main file in this bloc is the module `bin.picca_deltas` but the workflow analysis (see Section 2.1) revealed that some other files are also involved in this bloc. These module are `picca.data`, `picca.dla`, `picca.io`, `picca.prep_del`, and `picca.prep_pk1d`. Additionally, some of the functions in these modules use the constants defined in `picca.constants`. In this bloc we standardised these 7 modules. We now discuss the non-trivial changes in more detail.

We first address the `redefined-builtin` error. The `print` function is redefined in the module `picca.utils`. Even though the new behaviour is similar to the original function, redefining built-in functions is a dangerous practice. Thus, We rename the redefined `print` function to `userprint`, and propagate the changes. At this point we also drop the import of the print function from `__future__`. Note that this makes the package not compatible with `Python2`. The modules affected by this change are tabulated in the top bloc in Table 1. We also note that other modules use the regular `print` function (see bottom block in Table 1). We homogenize the package by replacing it with its extended version in module `picca.utils`.

`pylint` also complains about member `mean_cont` from class `forest` not being callable. However, the fact that this variable is called does not raise any exceptions at run-time. We analyse the behaviour of the code and we observe that the member is a function, meaning

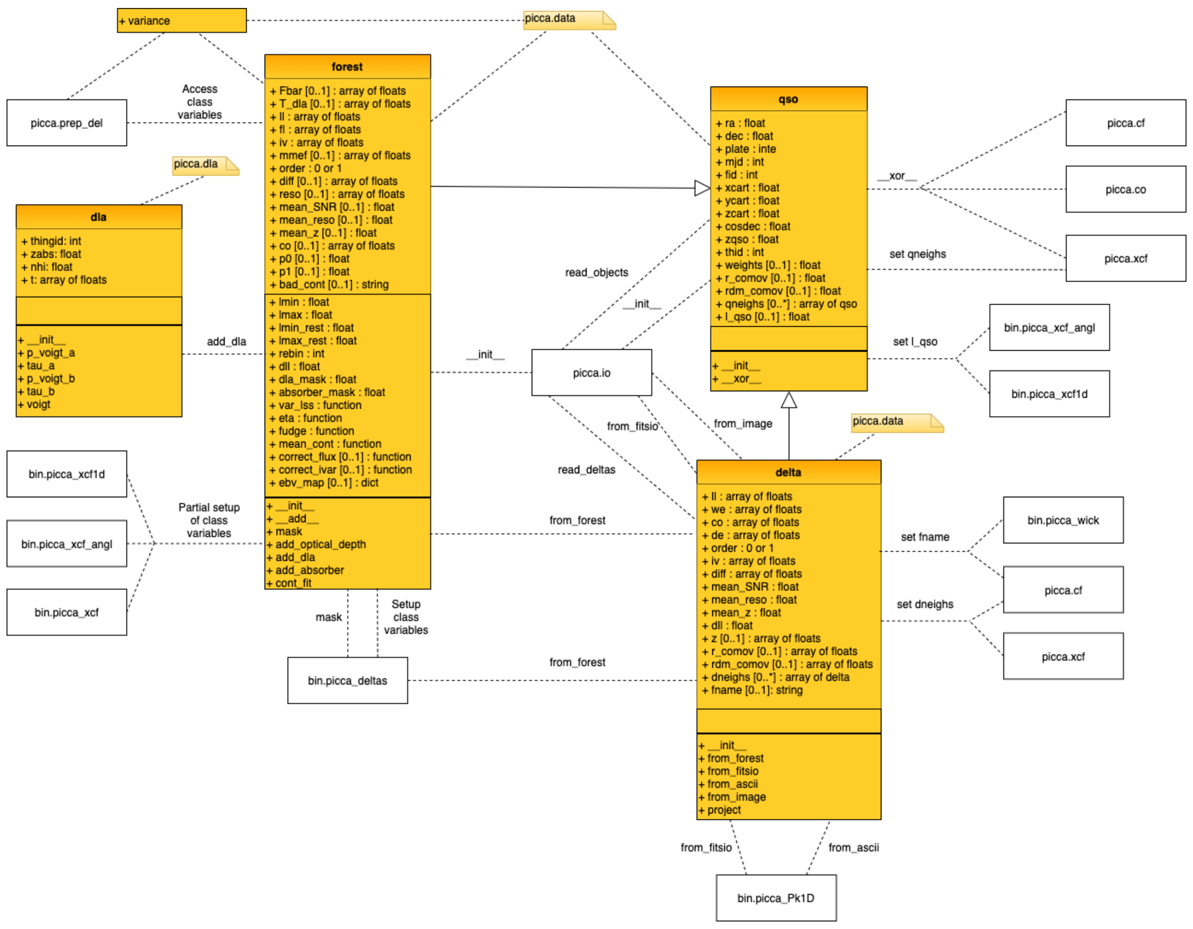

Figure 26: Original class diagram for class `forest`. Classes in the module are shown in filled boxes with three blocs: variables, class variables, and methods. Functions are shown as single-bloc filled boxes. A note (filled box with folded corner) specifies the file containing each function or class. Other modules interacting with this one are represented as unfilled boxes

it is indeed callable. We conclude that the complaint originates in a design defect of class `forest`. Figure 26 shows the original class diagram for the module `picca.data` (including the class `forest`) and the related entities. We can see that some members are functions and should, therefore, be declared as such. We revisit the class diagram and fix the design defect. The resulting class diagram is shown in Figure 27. We note that classes `forest`, and `delta` share many variables. A change on the inheritance scheme could probably benefit the data model, but this goes beyond the scope of the current work and we leave it for the future. We also note that several of the attributes are not declared in the definition of the classes, but are declared elsewhere, making it more difficult to follow the workflow. We included these declarations in the respective `__init__` methods.

Finally, we analyse all the variables present in the 7 modules of the bloc. For each variable, we state the original variable name, the modified name (if changed), the module it was identified in, other modules where it appears, its description, and the decision made regarding its name. Occasionally, there are also some additional comments with extra information. The entire list of variables is posted at `https://docs.google.com/spreadsheets/d/1PougvkVWN_`

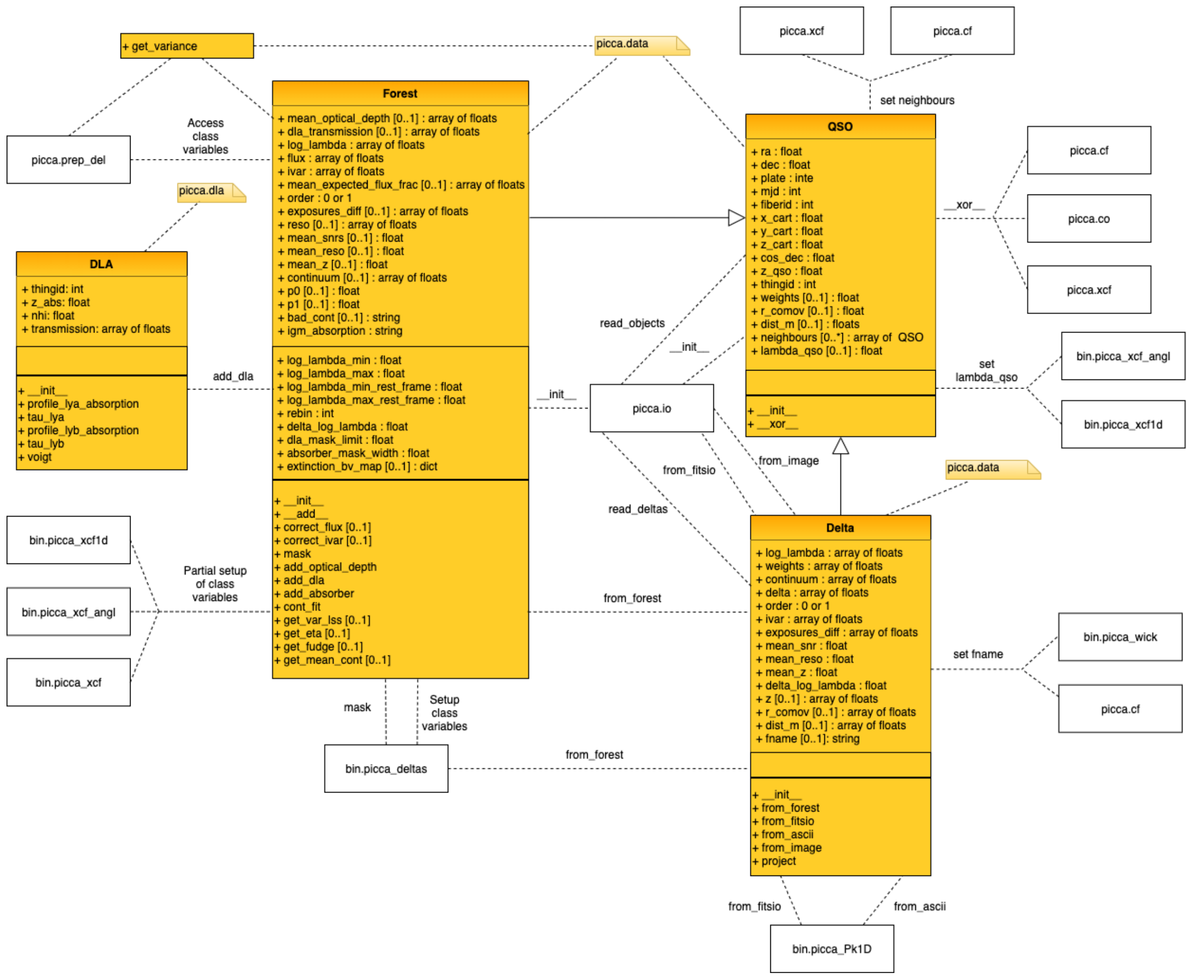

Figure 27: Same as Figure 26 but for the modified class diagram.

H5BubOCOs1ylkSOwnFSyDbOc3cvPEJXoc/edit?usp=sharing.

3.2 Compute auto-correlation

This bloc uses several scripts: `bin.picca_cf` to compute the auto-corrlation, `bin.picca_dmat` to compute the distortion matrix, `bin.picca_export` to combine the information from the different regions in the sky, and `bin.piccca_metal_dmat` to compute the metal distortion matrix. Additionally, the modules `picca.cf`, `picca.utils`, `picca.data`, `picca.io`, and `picca.constants` are involved in this bloc. The standardisation of some of these was already done in the previous bloc (see Section 3.1). Here we standardize the rest. We now discuss the non-trivial changes in more detail.

In module `picca.cf` there are two functions that are almost equal: `fill_neighs` and `fill_neighs_x_correlation`. The difference between those functions is that former is used when the two forests are the same, and the later is used when cross-correlating two forests. The function used is selected at run-time, and there is a control variable in the module to keep track of this option. As such, it is safe to merge both functions into a single one, simplifying the workflow of the code. Note that this change is already implemented in the workflow analysis presented in Section 2.2. This change affects the modules `bin.picca_cf_angl`, `bin.picca_cf`, `bin.picca_dmat`, `bin.picca_metal_dmat`, and `bin.picca_wick`.

We did not fully standardize the module `picca.cf`. The reason for this that this module uses several global variables (global to this module) that `pylint` identifies as constants. This causes a lot of confusion in the variable usage within the functions. We note that a similar thing happens with modules `picca.xcf` and `picca.co` (see Sections 3.3 and 3.5 respectively). The analysis of these three modules reveals a great similarity of these codes. This points to a possible inheritance class design that would make the code cleaner. However, such a change in the data model is outside the scope of the current work and is left for future analysis.

In module `picca.utils` there is a circular import that was prevented by importing inside the conflicting function. However, we detected that the conflicting function and a few others were file converters from format of origin to a format usable by `PICCA`. These functions are not used as part of any of the workflows (see Section 2), but are pre-steps that are sometimes required. As such, we decided to split the module, saving these functions in a new module called `picca.converters`. We note that we found a couple of bugs in the functions from `picca.converters`. They were not detected as those functions are not automatically tested (see Section 4).

Finally, we analyse all the variables present in the modules of the bloc. Their information is added to the list of variables generated in the previous bloc (see Section 3.1).

3.3 Compute cross-correlation

This bloc uses several scripts: `bin.picca_xcf` to compute the cross-corrlation, `bin.picca_xdmat` to compute the distortion matrix, `bin.picca_export` to combine the information from the different regions in the sky, and `bin.piccca_metal_xdmat` to compute the metal distortion matrix. Additionally, the modules `picca.xcf`, `picca.utils`, `picca.data`, `picca.io`, and `picca.constants` are involved in this bloc. The standardisation of most of these was already done in the previous bloc (see Sections 3.1 and 3.2). Here we standardize the rest. We now discuss the non-trivial changes in more detail.

Module `picca.xcf` is mostly equivalent to module `picca.cf`. As such we find again that there are two functions that are almost equal: `fill_neighs` and `fill_neighs_x_correlation`. Again, we merge both functions into a single one, simplifying the workflow of the code. Note that this change is already implemented in the workflow analysis presented in Section 2.3. This change affects the modules `bin.picca_xcf_angl`, `bin.picca_xcf`, `bin.picca_xdmat`, `bin.picca_metal_xdmat`, and `bin.picca_xwick`.

As with the module `picca.cf`, we did not fully standardize the module `picca.xcf`. This module also uses several global variables (global to this module) that `pylint` identifies as constants.

Finally, we analyse all the variables present in the modules of the bloc. Their information is added to the list of variables generated in the previous blocs (see Section 3.1).

3.4 Compute 1D power spectrum

This bloc uses the modules `bin.picca_Pk1D`, `picca.Pk1D`, `picca.data`, and `picca.utils`. At this stage only the first two are not standardised. Here we discuss the main changes that affect these two modules. Note that to ensure backwards compatibility we chose not to standardize the name of the module `bin.picca_Pk1D`.

In function `split_forest` in module `picca.Pk1D`, the usage of Lyα as absorption line is hard-coded. While for now only Lyα absorption is used in the 1D analysis, the rest of the code allows for other absorption lines to be used. Thus, this is a source of potential bugs in future analysis. To minimize this risk, we change the function to take the absorption line as a keyword argument. The same thing occurs within the main function in `bin.picca_pk1d`. However, in this case the variable was not used afterward. We suspect that this is because it was originally computed in `bin.picca_pk1d` but later moved to `picca.Pk1D`. Thus, we removed the variable.

Function `fill_masked_pixels` has an option to return the original arrays. This option is used to rename some of the arrays. We believe that this behaviour is misleading and we recommend it to be changed. However, such a change should be made upon revision of the data model, and we leave it for the future as this goes beyond the scope of this work.

Finally, we analyse all the variables present in the modules of the bloc. Their information is added to the list of variables generated in the previous blocs (see Section 3.1).

3.5 Alternative workflows

All the main modules have been already standardized. Only the scripts concerning the execution of these alternative workflows remain to be standardized. These are:

- `bin.picca_cf_angl`
- `bin.picca_cf1d`
- `bin.picca_co`
- `bin.picca_export_co`
- `bin.picca_export_coadd_zint`
- `bin.picca_export_cross_covariance`
- `bin.picca_nersc_submit`
- `bin.picca_plot_pk1d`
- `bin.picca_wick`
- `bin.picca_xcf_angl`

- `bin.picca_xcf1d`
- `bin.picca_xwick`
- `picca.co`

As with the modules `picca.cf` and `picca.xcf`, we did not fully standardize the module `picca.co`. This module also uses several global variables (global to this module) that `pylint` identifies as constants. We analyse all the variables present in the modules of the bloc. Their information is added to the list of variables generated in the previous blocs (see Section 3.1).

3.6 Final remarks

The initial `pylint` score for the level of standardization of `PICCA` was -5.54/10. After all the changes discussed above, we have reached a standardisation level of -1.08/10. This new score doesn't seem much higher than before. However this is to be expected since we decided not to standardize the files related to the fitters. This includes the libraries for `fitter` and `fitter2`, and the modules

- bin.picca_compute_pk_pksb.py
- bin.picca_fit.py
- bin.picca_fitter2.py
- bin.picca_fitter2_control.py
- bin.picca_fitter2_control_mpi.py
- bin.picca_mcmc.py
- bin.picca_pk2fits.py

Additionally, we didn't standardize neither the `setup` script, nor the `test` and `tutorials` modules. Once these modules are removed from the `pylint` run, then the `pylint` score is 9.07/10.

We can see on the final score that the code is still not fully standardized. While all the relevant files have been individually standardized, there is substantial code repetition between the files that lowers the final `pylint` score. This code repetition appears naturally due to the current data model for the code. This indicates that a more carefully designed data model will simplify the code and allow it to be fully standardized. However, the redesign of the data model is a project of its own and goes beyond the scope of this work. We leave this for the future.

4. Automatic testing

In this section we analyse the automatic tests (*unittest*) run to assess changes in the code. These tests are very useful when dealing with changes in the code that do not affect the theoretical performance, but rather the efficiency of the computation. An example of such changes are all the changes that we performed to standardize the code.

`PICCA` provides these tests in the module `picca.test.test_cor`. Note that the test module has not been standardized to ensure the stability of the tests. According to the `unittest` run, `PICCA` has only one test, but in reality there are several tests that are run under the cover of a larger test. These are the tests that are actually run:

- send_delta
- send_cf1d
- send_cf1d_cross
- send_cf_angl
- send_cf
- send_dmat
- send_metal_dmat
- send_wick
- send_export_cf
- send_cf_cross
- send_dmat_cross
- send_metal_dmat_cross
- send_export_cf_cross
- send_xcf_angl
- send_xcf
- send_xdmat
- send_metal_xdmat
- send_xwick
- send_export_xcf
- send_export_cross_covariance_cf_xcf
- send_co
- send_export_co
- send_fitter2
- send_delta_Pk1D
- send_Pk1D

Clearly, these tests cover most of `PICCA` scripts (see Section 2). However, a more detailed study of these tests reveals that only the standard options from these scripts are tested. This leaves a substantial portion of the code untested. From this, we conclude that even if the changes implemented in Section 3 do pass all these tests, there is no guarantee that the code is entirely bug-free for all possible uses.

This analysis of the test reveal also some other problems. Because all the test are encompassed in a single larger test, it is impossible to run only part of them. For example, improvements on the export of the correlation function should be testable independently of any changes in the computation of the distortion matrix.

Combined with the fact that there are missing tests, we conclude that the test module should be rethought from scratch. Separate test files should independently test the different steps in the analysis, and all possible options should be included. Ideally, this would be made in the context of a revision of the data model. To ensure the correctness of the new data model these improved tests will definitely be required. However, this is beyond the scope of the current work and is left for the future.

5. Summary and conclusions

We have carefully analyzed the code `PICCA` to help new developers gain a deeper understanding of the code. This is motivated by the fact that the code will be used for the analysis of data from the DESI collaboration, with further development of `PICCA` being a key element.

We have started the analysis by reviewing the workflow analysis of the code, excluding the elements linked with the fitters. The workflow plots produced here (see Section 2) will be very useful in further developing `PICCA`. We have also standardized the individual files (see

Section 3) and analyzed a total of 1,098 variables. The list of analyzed variables, along with their location, description, and the decisions made, is given in `https://docs.google.com/spreadsheets/d/1PougvkVWN_H5BubOCOs1ylkS0wnFSyDbOc3cvPEJXoc/edit?usp=sharing`. Finally, we have analysed the level of automatic testing of `PICCA` (see Section 4.

We conclude the following:

- The current level of standardisation is much higher than originally. The main remaining issue is the code duplication.
- Even though we have not analysed it in detail, it is clear that the level of standardisation of the fitters remains very low. However, we note that a new fitter is being developed.
- We have identified a need for a revision of the entire data model. There exists very similar classes that could be merged together, and there are some modules (e.g. `picca.cf` and `picca.xcf`) that could greatly benefit from a conversion to classes.
- The automatic testing of `PICCA` is enough to test against changes of the typical analysis choices, but we are vulnerable against the presence of bugs in less common analysis choices. The testing model should be revised to extend its coverage to all the functions and methods. Additionally, smaller, more independent tests should be designed.
- A task force should be designated to coordinate future development of `PICCA`. They should take care of ensuring the level of standardisation of the code, and avoid unnecessary complexity of the data model.

The current work presents substantial amounts of changes that are yet to be merged in the master branch of `PICCA`. A strategy to gradually implement all the changes is currently being discussed within the Lyα working group of DESI, and is expected to occur soon. Natural extensions of this work include the revision of the data model of `PICCA`, and the extension of the automatic testing to cover all functions and methods. These projects are also currently under discussion.

REFERENCES

Abazajian, K. N., Adelman-McCarthy, J. K., Agüeros, M. A., et al. 2009, ApJS, 182, 543

Alam, S., Zhu, H., Croft, R. A. C., et al. 2017, MNRAS, 470, 2822

Anderson, L., Aubourg, E., Bailey, S., et al. 2012, MNRAS, 427, 3435

Anderson, L., Aubourg, É., Bailey, S., et al. 2014a, MNRAS, 441, 24

Anderson, L., Aubourg, E., Bailey, S., et al. 2014b, MNRAS, 439, 83

Ata, M., Baumgarten, F., Bautista, J., et al. 2018, MNRAS, 473, 4773

Bautista, J. E., Busca, N. G., Guy, J., et al. 2017, A&A, 603, A12

Bautista, J. E., Vargas-Magaña, M., Dawson, K. S., et al. 2018, ApJ, 863, 110

Bennett, C. L., Hill, R. S., Hinshaw, G., et al. 2003, ApJS, 148, 97

Beutler, F., Blake, C., Colless, M., et al. 2011, MNRAS, 416, 3017

Blake, C., Davis, T., Poole, G. B., et al. 2011, MNRAS, 415, 2892

Blanton, M. R., Bershady, M. A., Abolfathi, B., et al. 2017, AJ, 154, 28

Blomqvist, M., Pieri, M. M., du Mas des Bourboux, H., et al. 2018, JCAP, 2018, 029

Blomqvist, M., du Mas des Bourboux, H., Busca, N. G., et al. 2019, A&A, 629, A86

Busca, N. G., Delubac, T., Rich, J., et al. 2013, A&A, 552, A96

Chabanier, S., Palanque-Delabrouille, N., Yèche, C., et al. 2019, JCAP, 2019, 017

Chiba, M., Cohen, J., & Wyse, R. F. G. 2016, in IAU Symposium, Vol. 317, The General Assembly of Galaxy Halos: Structure, Origin and Evolution, ed. A. Bragaglia, M. Arnaboldi, M. Rejkuba, & D. Romano, 280–281

Chuang, C.-H., & Wang, Y. 2012, MNRAS, 426, 226

Cole, S., Percival, W. J., Peacock, J. A., et al. 2005, MNRAS, 362, 505

Colless, M., Dalton, G., Maddox, S., et al. 2001, MNRAS, 328, 1039

Conley, A., Guy, J., Sullivan, M., et al. 2011, ApJS, 192, 1

Dalton, G., Trager, S., Abrams, D. C., et al. 2016, in Proc. SPIE, Vol. 9908, Ground-based and Airborne Instrumentation for Astronomy VI, 99081G

Dawson, K. S., Schlegel, D. J., Ahn, C. P., et al. 2013, AJ, 145, 10

Dawson, K. S., Kneib, J.-P., Percival, W. J., et al. 2016, AJ, 151, 44

de Jong, R. S., Barden, S. C., Bellido-Tirado, O., et al. 2016, in Proc. SPIE, Vol. 9908, Ground-based and Airborne Instrumentation for Astronomy VI, 99081O

de Sainte Agathe, V., Balland, C., du Mas des Bourboux, H., et al. 2019, A&A, 629, A85

Delubac, T., Bautista, J. E., Busca, N. G., et al. 2015, A&A, 574, A59

DESI Collaboration. 2016, arXiv e-prints, arXiv:1611.00036

Dressler, A., Spergel, D., Mountain, M., et al. 2012, arXiv e-prints, arXiv:1210.7809

Drinkwater, M. J., Jurek, R. J., Blake, C., et al. 2010, MNRAS, 401, 1429

du Mas des Bourboux, H., Le Goff, J.-M., Blomqvist, M., et al. 2017, A&A, 608, A130

du Mas des Bourboux, H., Dawson, K. S., Busca, N. G., et al. 2019, ApJ, 878, 47

du Mas des Bourboux, H., Rich, J., Font-Ribera, A., et al. 202-0, Submitted to ApJ

Eisenstein, D. J., Zehavi, I., Hogg, D. W., et al. 2005, ApJ, 633, 560

Eisenstein, D. J., Weinberg, D. H., Agol, E., et al. 2011, AJ, 142, 72

Farr, J., Font-Ribera, A., du Mas des Bourboux, H., et al. 2020, JCAP, 2020, 068

Font-Ribera, A., Miralda-Escudé, J., Arnau, E., et al. 2012, JCAP, 2012, 059

Font-Ribera, A., Kirkby, D., Busca, N., et al. 2014, JCAP, 2014, 027

Gaia Collaboration. 2016, A&A, 595, A1

Green, J., Schechter, P., Baltay, C., et al. 2012, arXiv e-prints, arXiv:1208.4012

Guy, J., Sullivan, M., Conley, A., et al. 2010, A&A, 523, A7

Hong, T., Han, J. L., & Wen, Z. L. 2016, ApJ, 826, 154

Jones, D. H., Saunders, W., Colless, M., et al. 2004, MNRAS, 355, 747

Kirkby, D., Margala, D., Slosar, A., et al. 2013, JCAP, 2013, 024

Laureijs, R. J., Duvet, L., Escudero Sanz, I., et al. 2010, in Proc. SPIE, Vol. 7731, Space Telescopes and Instrumentation 2010: Optical, Infrared, and Millimeter Wave, 77311H

Laurent, P., Le Goff, J.-M., Burtin, E., et al. 2016, JCAP, 2016, 060

Mehta, K. T., Cuesta, A. J., Xu, X., Eisenstein, D. J., & Padmanabhan, N. 2012, MNRAS, 427, 2168

Padmanabhan, N., Xu, X., Eisenstein, D. J., et al. 2012, MNRAS, 427, 2132

Percival, W. J., Cole, S., Eisenstein, D. J., et al. 2007, MNRAS, 381, 1053

Percival, W. J., Reid, B. A., Eisenstein, D. J., et al. 2010, MNRAS, 401, 2148

Pérez-Ràfols, I., Miralda-Escudé, J., Arinyo-i-Prats, A., Font-Ribera, A., & Mas-Ribas, L. 2018a, MNRAS, 480, 4702

Pérez-Ràfols, I., Font-Ribera, A., Miralda-Escudé, J., et al. 2018b, MNRAS, 473, 3019

Pieri, M. M., Bonoli, S., Chaves-Montero, J., et al. 2016, in SF2A-2016: Proceedings of the Annual meeting of the French Society of Astronomy and Astrophysics, 259–266

Planck Collaboration. 2011, A&A, 536, A1

—. 2016, A&A, 594, A13

Ross, A. J., Samushia, L., Howlett, C., et al. 2015, MNRAS, 449, 835

Slosar, A., Iršič, V., Kirkby, D., et al. 2013, JCAP, 2013, 026

Spergel, D., Gehrels, N., Baltay, C., et al. 2015, arXiv e-prints, arXiv:1503.03757

Sullivan, M., Guy, J., Conley, A., et al. 2011, ApJ, 737, 102

Takada, M., Ellis, R. S., Chiba, M., et al. 2014, PASJ, 66, R1

van Rossum, G., Warsaw, B., & Coghlan, N. 2001 (accessed June 12, 2020), PEP 8 – Style Guide for Python Code

Xu, X., Cuesta, A. J., Padmanabhan, N., Eisenstein, D. J., & McBride, C. K. 2013, MNRAS, 431, 2834

York, D. G., Adelman, J., Anderson, Jr., J. E., et al. 2000, AJ, 120, 1579